Support Vector Machines

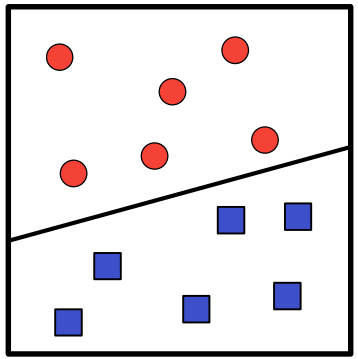
Separating data with the **largest margin**

Faculty of Mathematics and Computer Science, University of Bucharest
and
Sparktech Software

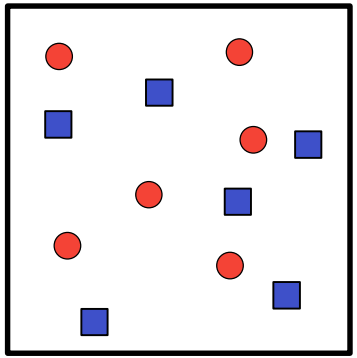
Academic Year 2018/2019, 1st Semester

Linear Separability

- Two sets of points are **linearly separable**, if there is at least one **hyperplane** which completely separates them.
- An **n-dimensional hyperplane** is a flat *n-1* dimensional subset of the space.
  - A *1d* hyperplane is a **point.**
  - A *2d* hyperplane is a **line.**
  - A *3d* hyperplane is a **plane.**



Separable



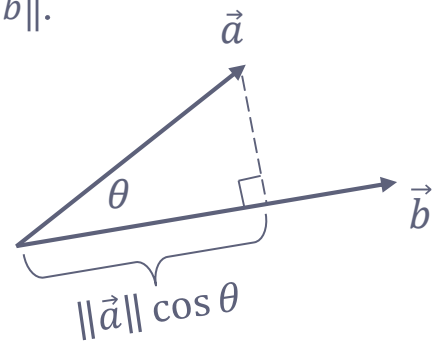
Non-separable

Refresher – Dot Product

- An operation which takes two vectors and returns a **scalar** value.
- Also called the *scalar product* or *inner product*.

$$\vec{a} \cdot \vec{b} = \langle \vec{a}, \vec{b} \rangle = \|a\| \|b\| \cos\theta = \sum_i a_i b_i$$

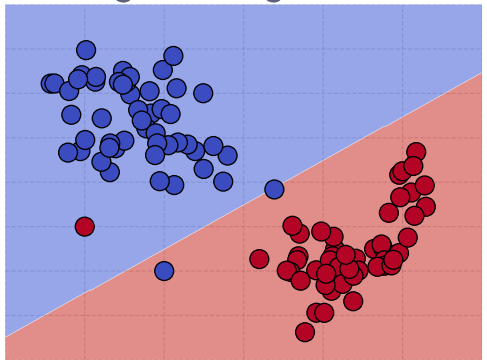
- It can be interpreted as a **similarity measure** between vectors.
- If we write it as $(\|\vec{a}\| \cos\theta)\|\vec{b}\|$, it can be viewed as the **length of the projection** of $\vec{a}$ on $\vec{b}$, measured in units of length $\|\vec{b}\|$.



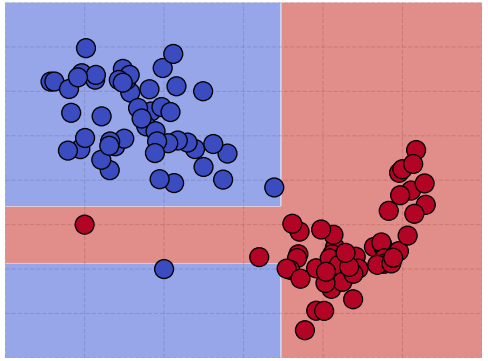
Decision Boundaries

- Classifications means learning a **good decision boundary**.
- Different algorithms have different *allowed hypotheses* and, therefore, different types of decision boundaries.

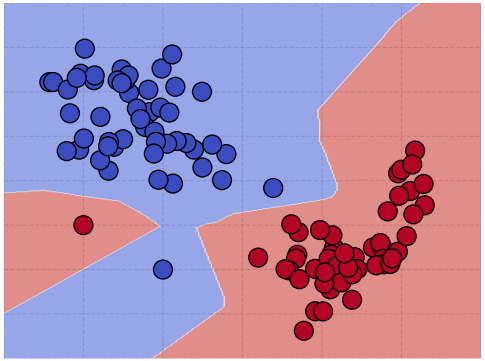
Logistic Regression



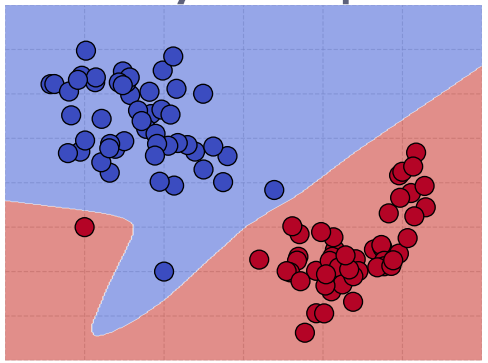
Decision Tree



KNN

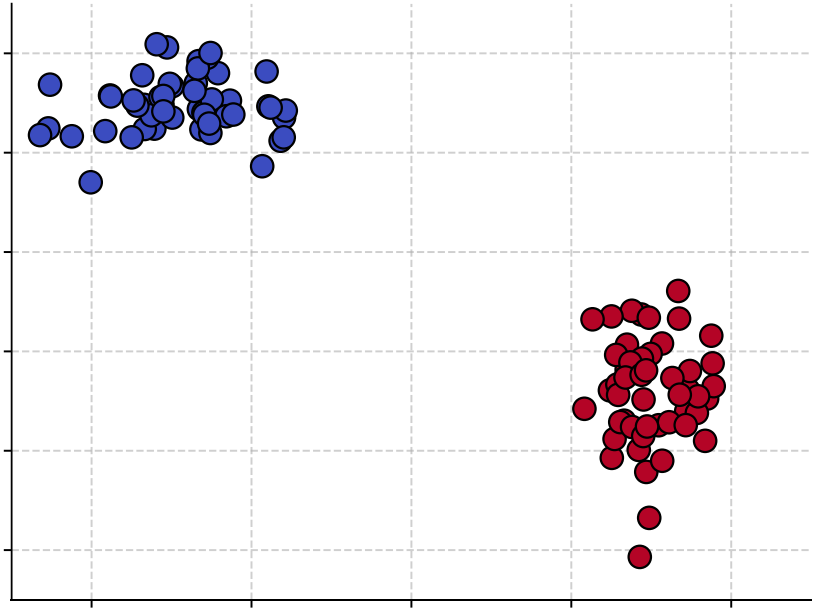


Multilayer Perceptron



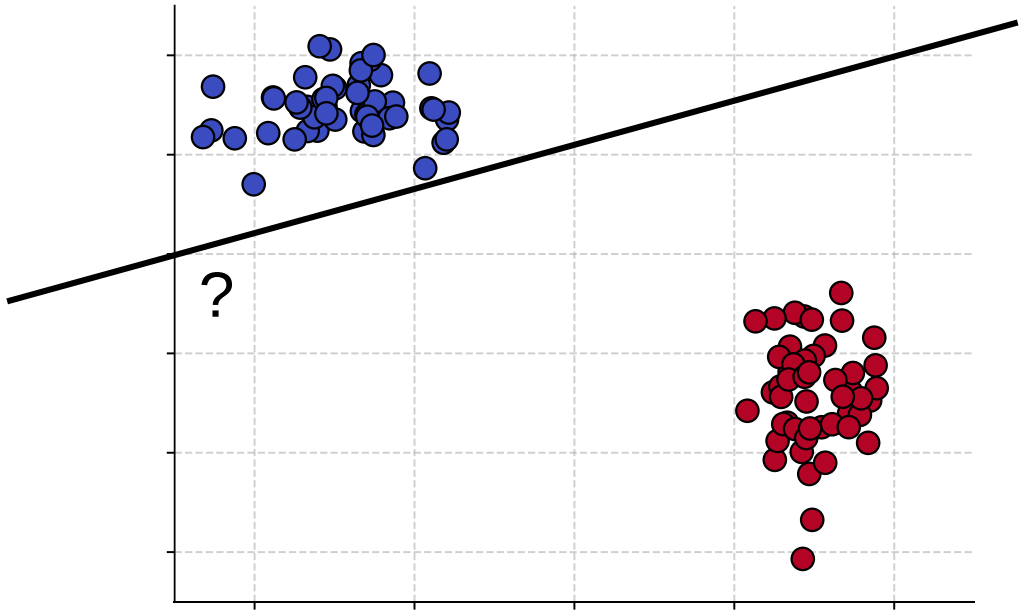
Decision Boundaries

- How would you draw a straight line through these two sets of points?



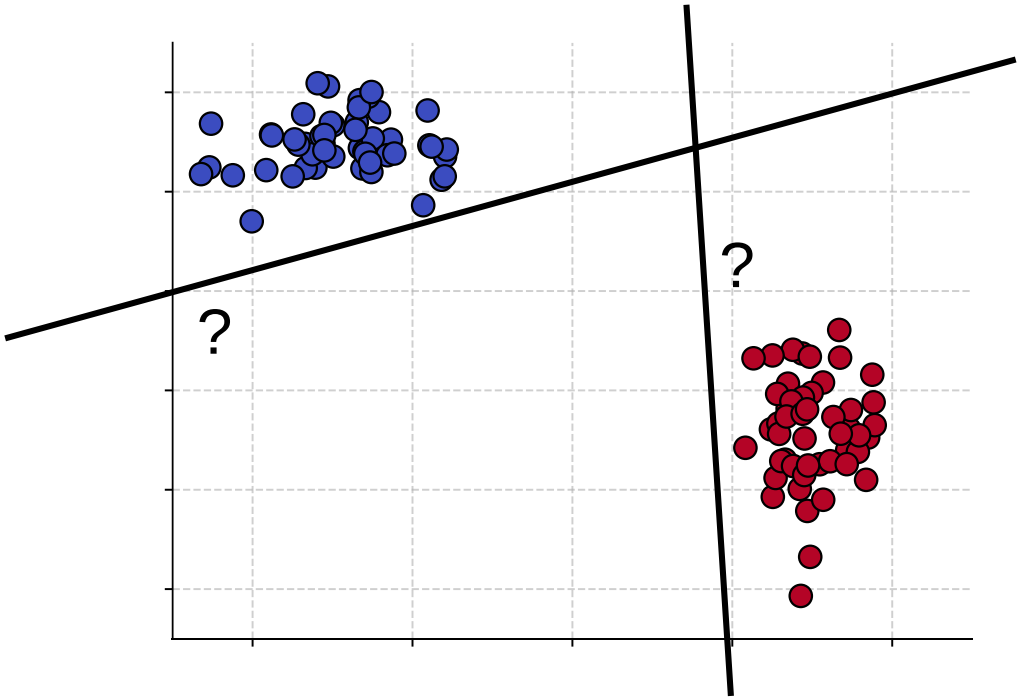
Decision Boundaries

- How would you draw a straight line through these two sets of points?



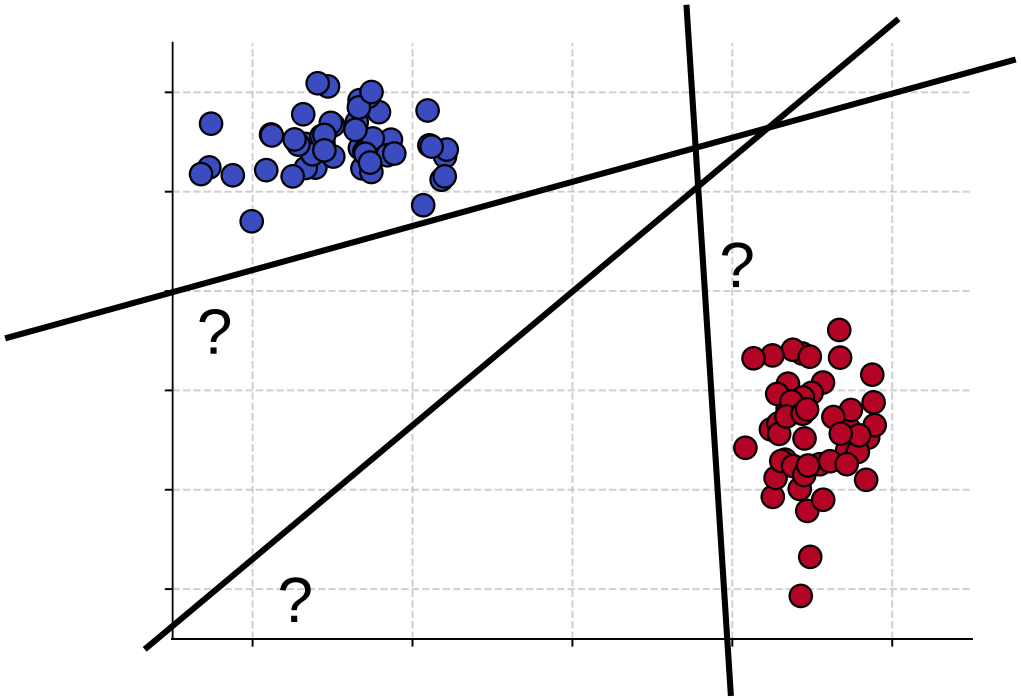
Decision Boundaries

- How would you draw a straight line through these two sets of points?



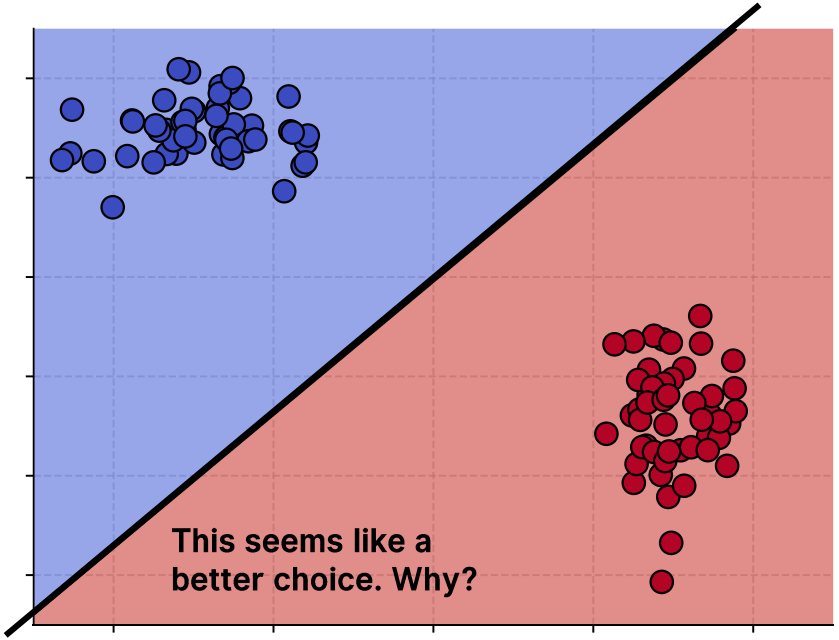
Decision Boundaries

- How would you draw a straight line through these two sets of points?



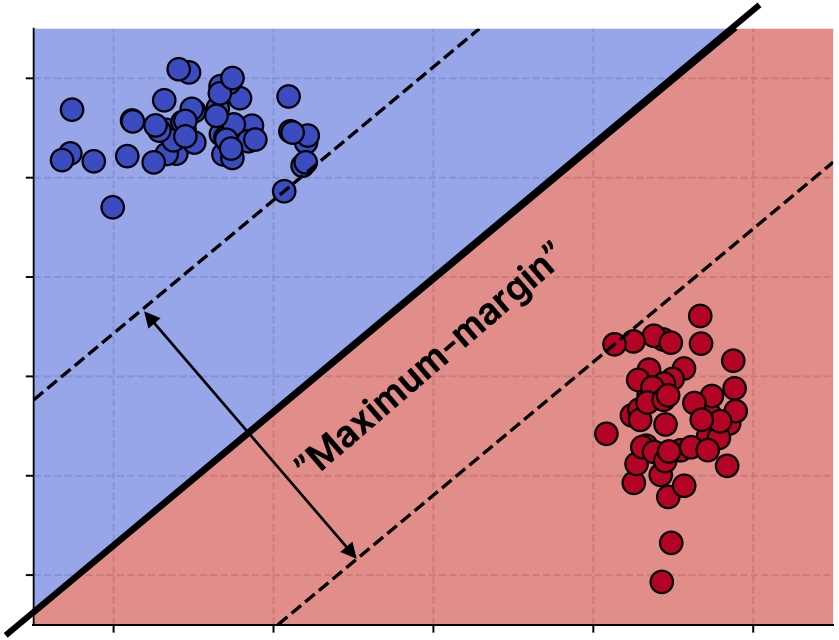
Decision Boundaries

- How would you draw a straight line through these two sets of points?



Decision Boundaries

- How would you draw a straight line through these two sets of points?



SVM Definition

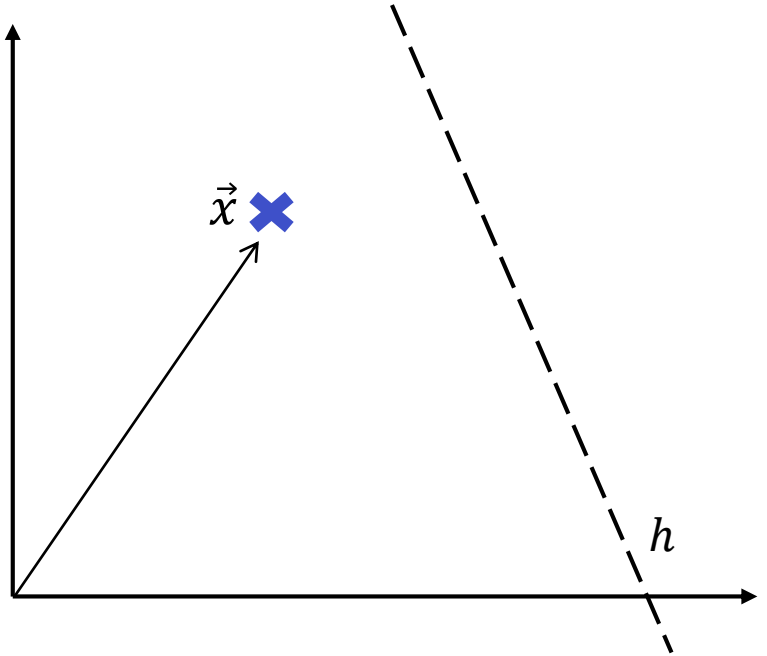
- A **Support Vector Machine** (**SVM**) is a *non-probabilistic binary linear classifier*.
  - **Classifier** – A supervised learning method which predicts a categorical class.
  - **Linear** – The decision boundary is an n-dimensional hyperplane.
  - **Binary** – It can learn to predict one of two classes (a “+” class and a ”-” class).
  - **Non-probabilistic** – The output of its *decision function* is not bounded, so it cannot be interpreted as probability.

There are methods of adapting an SVM to be used for **regression** problems and for making it **non-linear**, **multiclass** and/or **probabilistic**.

- An SVM tries to find a *separating* hyperplane, which is as far away from all training points at the same time (a **maximum-margin hyperplane**)
  - A point is classified as “+” or “-”, depending on which part of the hyperplane it lies.

Decision Rule

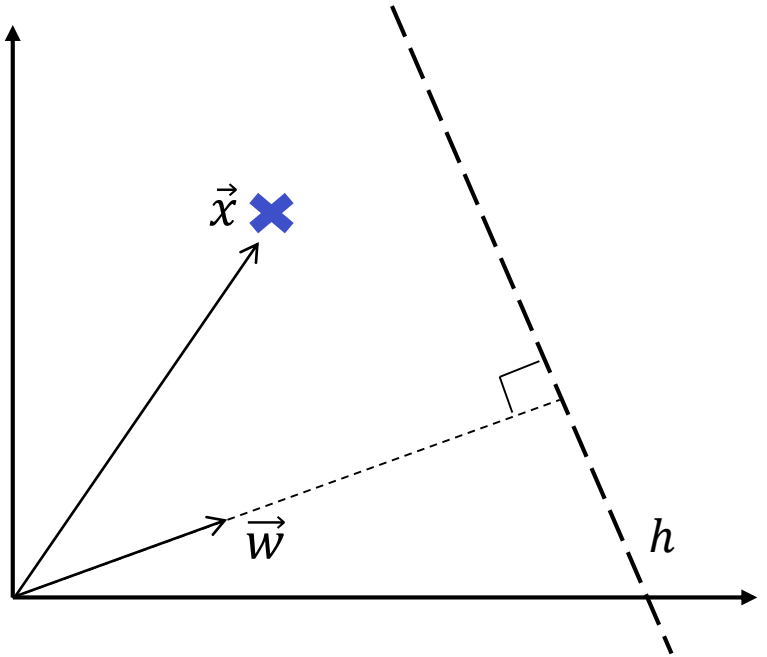
- How do we **decide** if a point $\vec{x}$ is on some side of a hyperplane h?



Decision Rule

- How do we **decide** if a point $\vec{x}$ is on some side of a hyperplane h?

Let $\vec{w} \perp h$



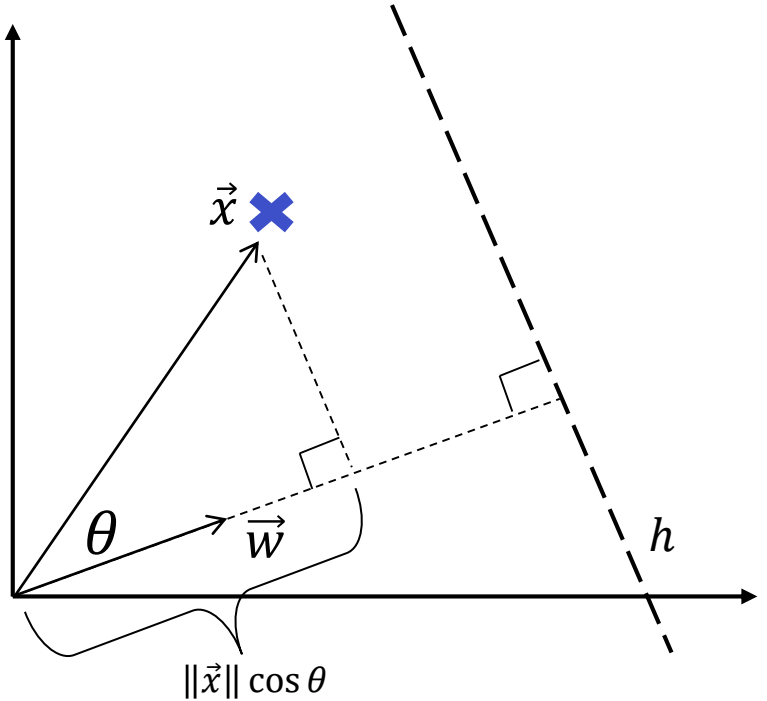
Decision Rule

- How do we **decide** if a point $\vec{x}$ is on some side of a hyperplane h?

Let $\vec{w} \perp h$

$$\langle \vec{x}, \vec{w} \rangle = (\|\vec{x}\| \cos\theta)\|\vec{w}\|$$

is proportional to the projection of $\vec{x}$ on $\vec{w}$



Decision Rule

- How do we **decide** if a point $\vec{x}$ is on some side of a hyperplane h?

Let $\vec{w} \perp h$

$\langle \vec{x}, \vec{w} \rangle = (\|\vec{x}\| \cos\theta)\|\vec{w}\|$

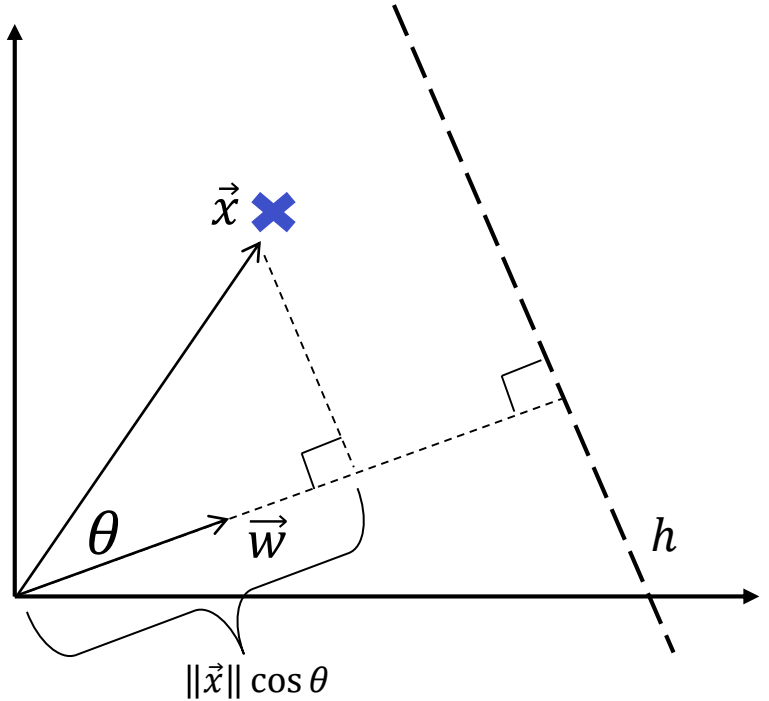
is proportional to the projection of $\vec{x}$ on $\vec{w}$

$\Longrightarrow$

$\exists c \in \mathbb{R}$ such that

if $\langle \vec{x}, \vec{w} \rangle \geq c$ then $\vec{x}$ is on the right side of h

(i.e. $\vec{x}$ is a "+" sample)



Decision Rule

- How do we **decide** if a point $\vec{x}$ is on some side of a hyperplane h?

Let $\vec{w} \perp h$

$\langle \vec{x}, \vec{w} \rangle = (\|\vec{x}\| \cos\theta)\|\vec{w}\|$

is proportional to the projection of $\vec{x}$ on $\vec{w}$

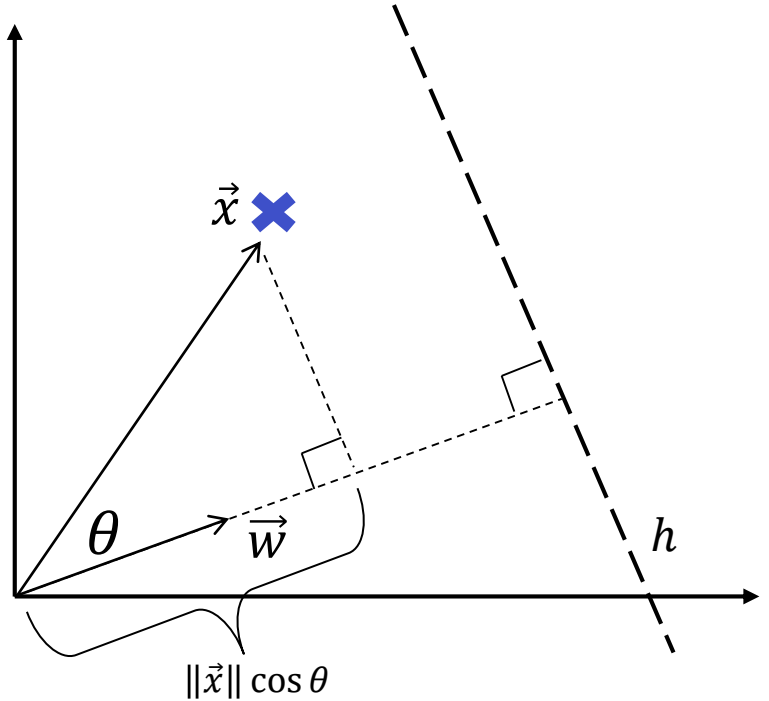
$\Longrightarrow$

$\exists c \in \mathbb{R}$ such that

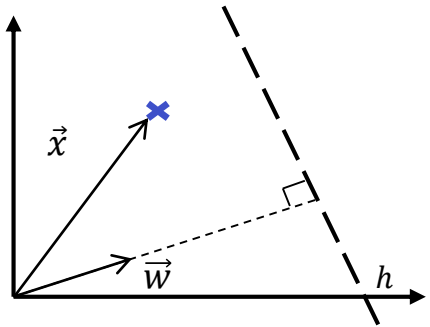
if $\langle \vec{x}, \vec{w} \rangle \geq c$ then $\vec{x}$ is on the right side of h

(i.e. $\vec{x}$ is a "+" sample)

Let $b = -c$



Decision Rule



$\langle \vec{x}, \vec{w} \rangle + b > 0 \Rightarrow \vec{x}$ is on the right side of h

$\langle \vec{x}, \vec{w} \rangle + b < 0 \Rightarrow \vec{x}$ is on the left side of h

$\langle \vec{x}, \vec{w} \rangle + b = 0 \Rightarrow \vec{x}$ is exactly on h

$|\langle \vec{x}, \vec{w} \rangle + b|$ is proportional to the distance from h

In fact, we can say that h is determined by $\vec{w}$ and b

- SVM decision rule:

$\vec{x}$ is a "+" sample if $\langle \vec{x}, \vec{w} \rangle + b \geq 0$

$\vec{x}$ is a "-" sample otherwise

- Which $\vec{w}$? Which b?

Learning a “good” separating hyperplane

- Training set: $E = \{(\vec{x}^{(1)}, y^{(1)}), (\vec{x}^{(2)}, y^{(2)}), \dots, (\vec{x}^{(m)}, y^{(m)})\},\ \vec{x}^{(i)} \in \mathbb{R}^n,\ \mathrm{y}^{(i)} \in \{+1, -1\}$
- We want to separate the ”+” and ”-” training samples:

Learning a “good” separating hyperplane

- Training set: $E = \{(\vec{x}^{(1)}, y^{(1)}), (\vec{x}^{(2)}, y^{(2)}), \dots, (\vec{x}^{(m)}, y^{(m)})\},\ \vec{x}^{(i)} \in \mathbb{R}^n,\ y^{(i)} \in \{+1, -1\}$
- We want to separate the ”+” and ”-” training samples:

$$\langle \overrightarrow{x_+}, \vec{w} \rangle + b \geq 0 \qquad \forall \overrightarrow{x_+} \in \{\vec{x}^{(i)} | y^{(i)} = +1\}$$
$$\langle \overrightarrow{x_-}, \vec{w} \rangle + b < 0 \qquad \forall \overrightarrow{x_-} \in \{\vec{x}^{(i)} | y^{(i)} = -1\}$$

Learning a “good” separating hyperplane

- Training set: $E = \{(\vec{x}^{(1)}, y^{(1)}), (\vec{x}^{(2)}, y^{(2)}), \dots, (\vec{x}^{(m)}, y^{(m)})\},\ \vec{x}^{(i)} \in \mathbb{R}^n,\ \mathrm{y}^{(i)} \in \{+1, -1\}$
- We want to separate the ”+” and ”-” training samples:

$$\langle \overrightarrow{x_+}, \vec{w} \rangle + b \geq 0 \qquad \forall \overrightarrow{x_+} \in \{\vec{x}^{(i)} | y^{(i)} = +1\}$$
$$\langle \overrightarrow{x_-}, \vec{w} \rangle + b < 0 \qquad \forall \overrightarrow{x_-} \in \{\vec{x}^{(i)} | y^{(i)} = -1\}$$

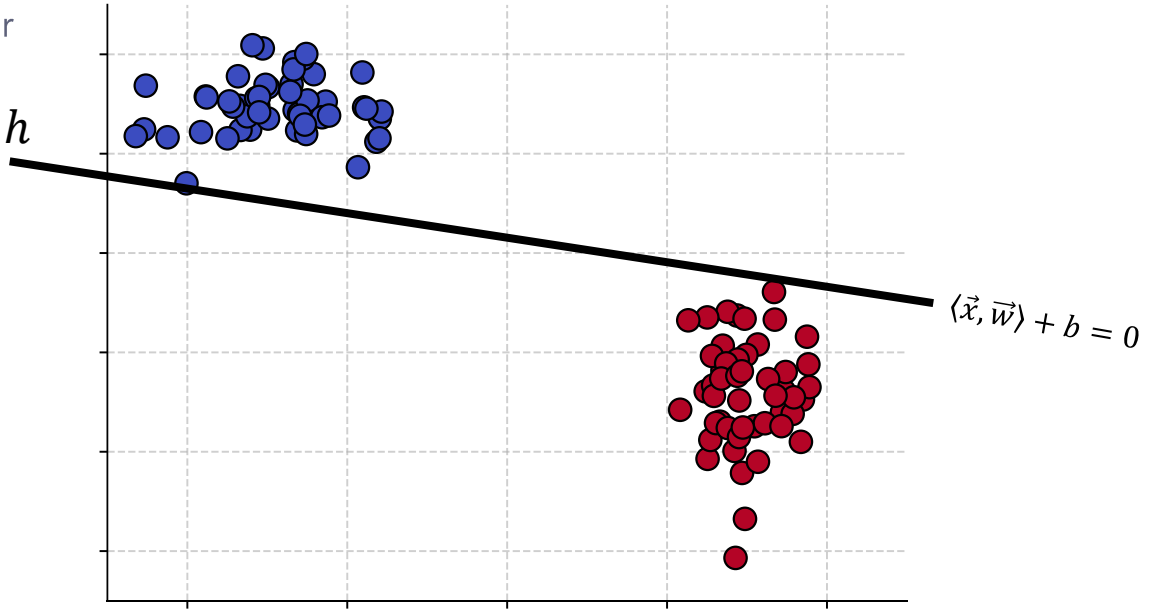
Are these conditions good enough?

Learning a "good" separating hyperplane

$$\langle \overrightarrow{x_+}, \vec{w} \rangle + b \geq 0 \qquad \forall \overrightarrow{x_+} \in \left\{ \vec{x}^{(i)} \middle| y^{(i)} = +1 \right\}$$
$$\langle \overrightarrow{x_-}, \vec{w} \rangle + b < 0 \qquad \forall \overrightarrow{x_-} \in \left\{ \vec{x}^{(i)} \middle| y^{(i)} = -1 \right\}$$

This hyperplane satisfies our conditions, but it has *absolutely no margin*.

We want it to have *some margin*, which we can later maximize.



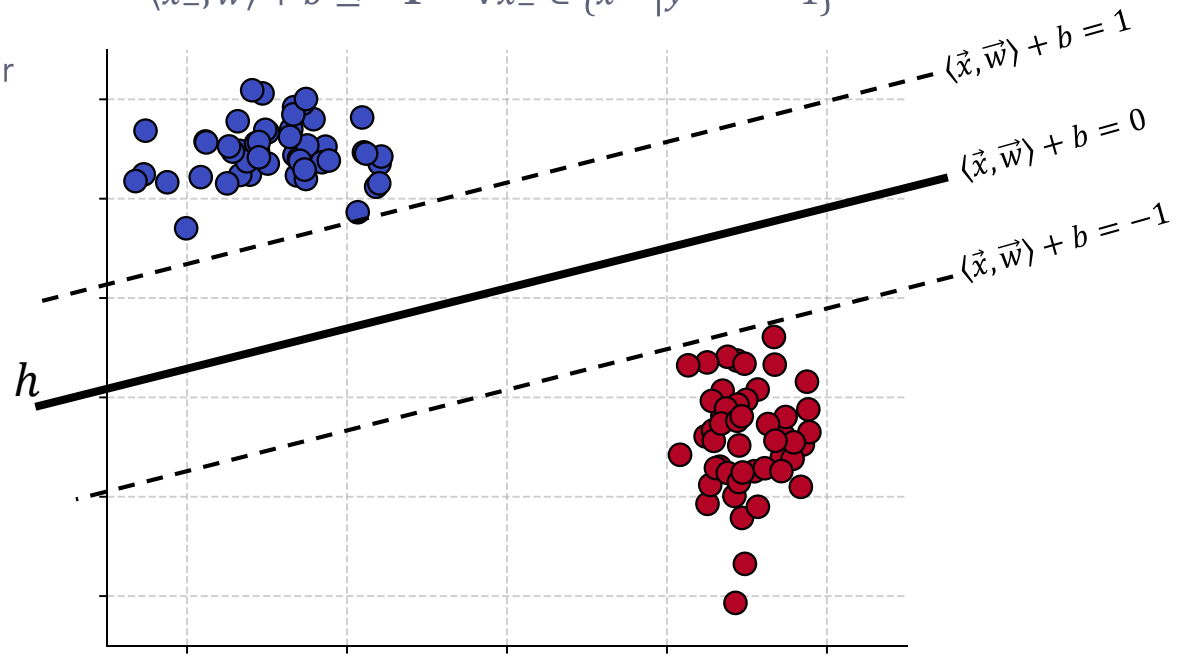
Learning a "good" separating hyperplane

$$\langle \overrightarrow{x_+}, \vec{w} \rangle + b \geq 1 \qquad \forall \overrightarrow{x_+} \in \{\vec{x}^{(i)} | y^{(i)} = +1\}$$
$$\langle \overrightarrow{x_-}, \vec{w} \rangle + b \leq -1 \qquad \forall \overrightarrow{x_-} \in \{\vec{x}^{(i)} | y^{(i)} = -1\}$$

This hyperplane satisfies our conditions, but it has *absolutely no margin*.

We want it to have *some margin*, which we can later maximize.

We modify the conditions such that we force a **gap** around the *separating hyperplane* in which no training examples lie.



Learning a “good” separating hyperplane

- Training set: $E = \{(\vec{x}^{(1)}, y^{(1)}), (\vec{x}^{(2)}, y^{(2)}), \dots, (\vec{x}^{(m)}, y^{(m)})\},\ \vec{x}^{(i)} \in \mathbb{R}^n,\ y^{(i)} \in \{+1, -1\}$
- We want to separate the ”+” and ”-” training samples and have **some margin between the classes:**

$$\langle \overrightarrow{x_+}, \vec{w} \rangle + b \geq \mathbf{1} \qquad \forall \overrightarrow{x_+} \in \{\vec{x}^{(i)} | y^{(i)} = +1\}$$
$$\langle \overrightarrow{x_-}, \vec{w} \rangle + b \leq -\mathbf{1} \qquad \forall \overrightarrow{x_-} \in \{\vec{x}^{(i)} | y^{(i)} = -1\}$$

Learning a “good” separating hyperplane

- Training set: $E = \{(\vec{x}^{(1)}, y^{(1)}), (\vec{x}^{(2)}, y^{(2)}), \dots, (\vec{x}^{(m)}, y^{(m)})\},\ \vec{x}^{(i)} \in \mathbb{R}^n,\ y^{(i)} \in \{+1, -1\}$
- We want to separate the ”+” and ”-” training samples and have **some margin between the classes:**

$$\langle \overrightarrow{x_+}, \vec{w} \rangle + b \geq \mathbf{1} \qquad \forall \overrightarrow{x_+} \in \{\vec{x}^{(i)} | y^{(i)} = +1\}$$
$$\langle \overrightarrow{x_-}, \vec{w} \rangle + b \leq -\mathbf{1} \qquad \forall \overrightarrow{x_-} \in \{\vec{x}^{(i)} | y^{(i)} = -1\}$$

- Combining the two inequations:

$$y^{(i)}\left(\langle \vec{x}^{(i)}, \vec{w} \rangle + b\right) - 1 \geq 0$$

Learning a “good” separating hyperplane

- Training set: $E = \{(\vec{x}^{(1)}, y^{(1)}), (\vec{x}^{(2)}, y^{(2)}), \dots, (\vec{x}^{(m)}, y^{(m)})\},\ \vec{x}^{(i)} \in \mathbb{R}^n,\ y^{(i)} \in \{+1, -1\}$
- We want to separate the ”+” and ”-” training samples and have **some margin between the classes:**

$$\langle \overrightarrow{x_+}, \vec{w} \rangle + b \geq \mathbf{1} \qquad \forall \overrightarrow{x_+} \in \{\vec{x}^{(i)} | y^{(i)} = +1\}$$
$$\langle \overrightarrow{x_-}, \vec{w} \rangle + b \leq -\mathbf{1} \qquad \forall \overrightarrow{x_-} \in \{\vec{x}^{(i)} | y^{(i)} = -1\}$$

- Combining the two inequations:

$$y^{(i)}\left(\langle \vec{x}^{(i)}, \vec{w} \rangle + b\right) - 1 \geq 0$$

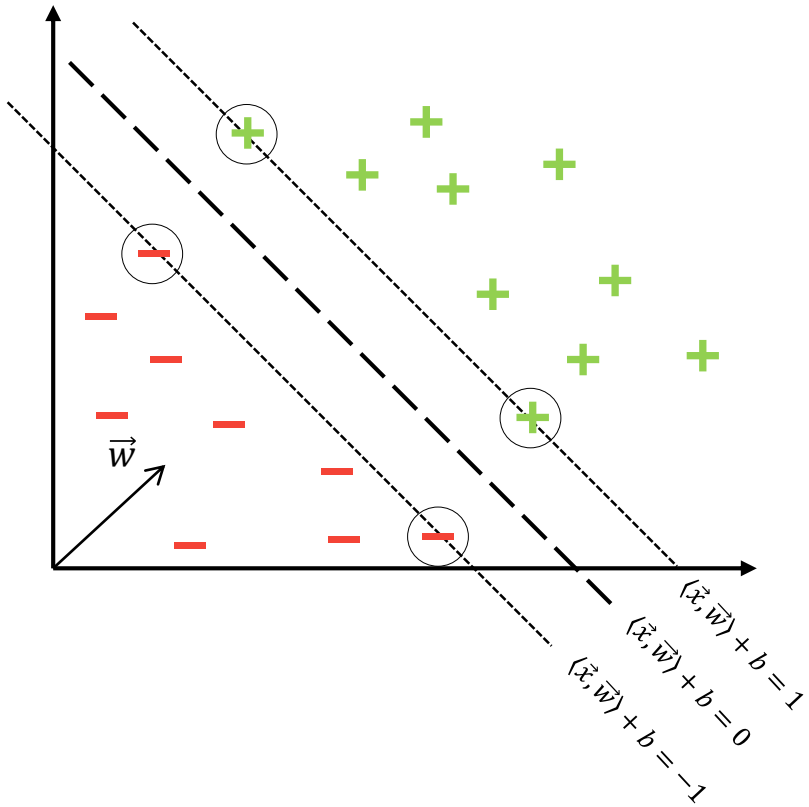
- For examples $(\vec{x}^{(i)}, y^{(i)})$ which lie exactly on the edges of the gap:

$$y^{(i)}\left(\langle \vec{x}^{(i)}, \vec{w} \rangle + b\right) - 1 = 0$$

Learning a "good" separating hyperplane

- For training examples $(\vec{x}^{(i)}, y^{(i)})$, which lie exactly on the edges of the gap:

$$y^{(i)}\left(\langle \vec{x}^{(i)}, \vec{w} \rangle + b\right) - 1 = 0$$

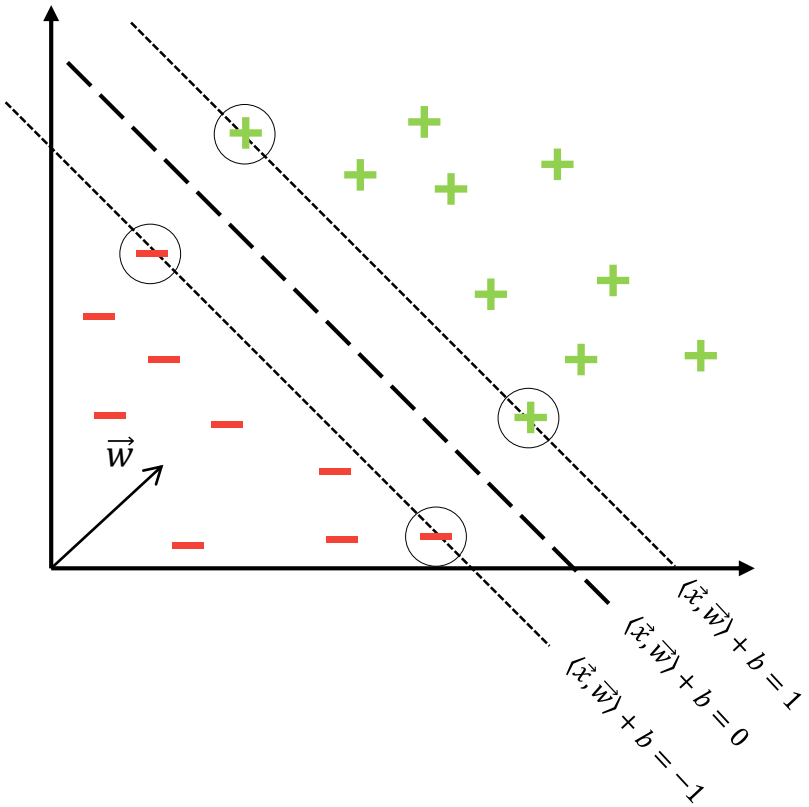


Learning a "good" separating hyperplane

- For training examples $\left(\vec{x}^{(i)}, y^{(i)}\right)$, which lie exactly on the edges of the gap:

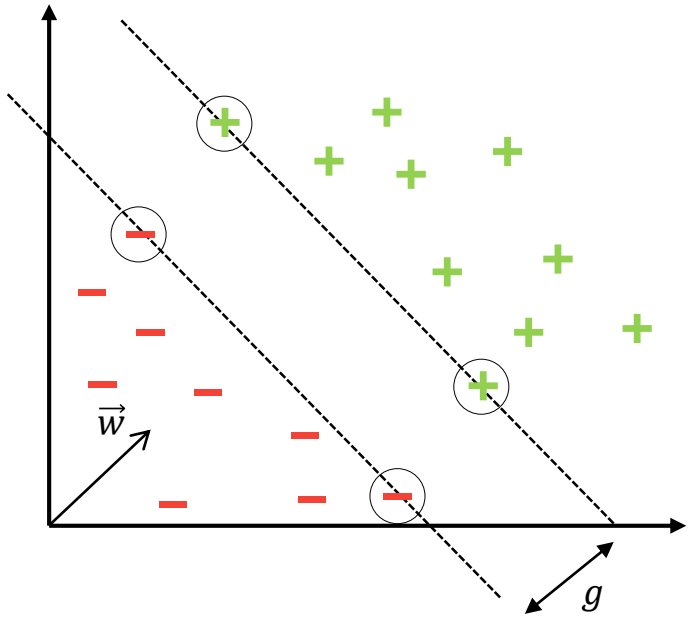
$$\boldsymbol{y^{(i)}\left(\left\langle \vec{x}^{(i)}, \vec{w} \right\rangle + b\right) - 1 = 0}$$

- We call these examples "**Support Vectors**"



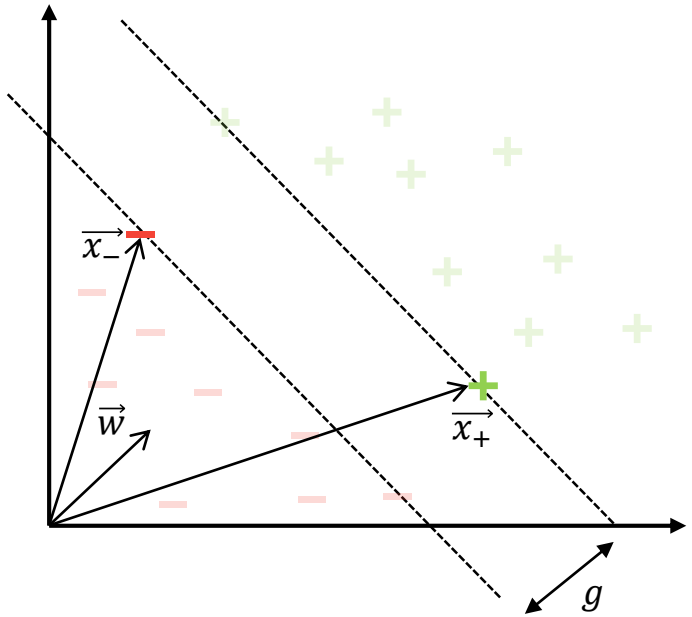
Making the margin “wide”

- How do we express the width of the gap?



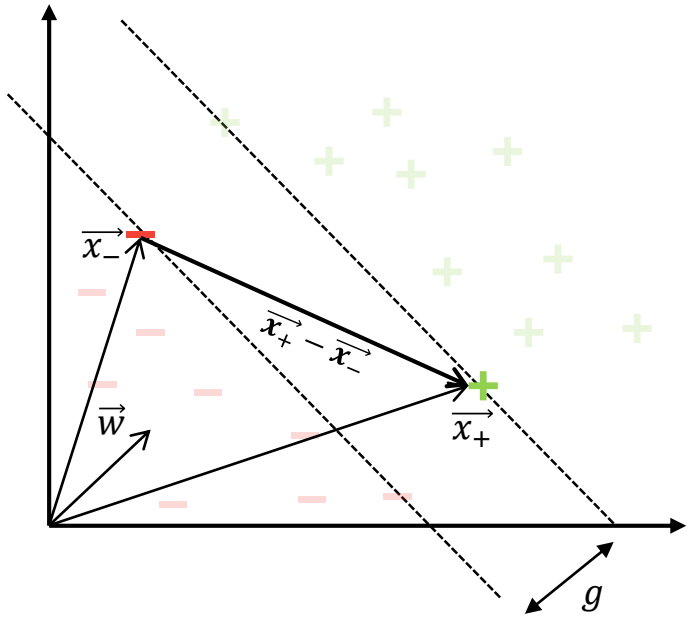
Making the margin “wide”

- How do we express the width of the gap?



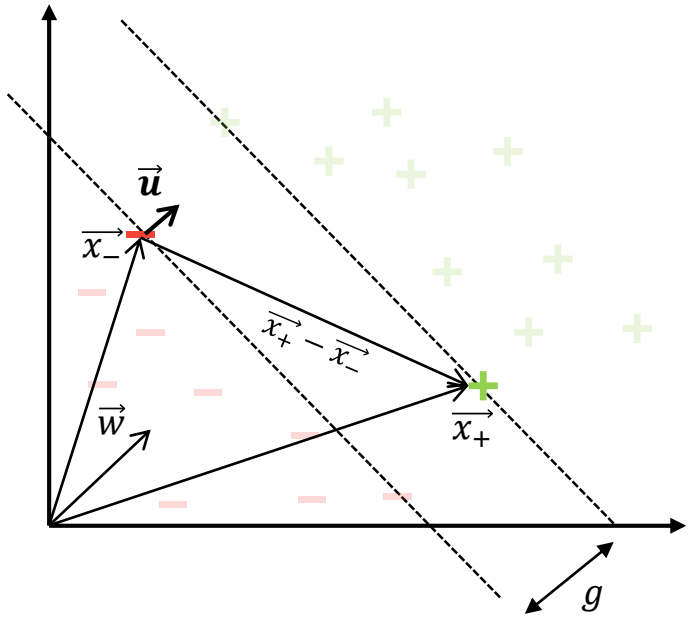
Making the margin “wide”

- How do we express the width of the gap?
- $(\overrightarrow{x_+} - \overrightarrow{x_-})$ is a vector from one side of the gap to the other.



Making the margin “wide”

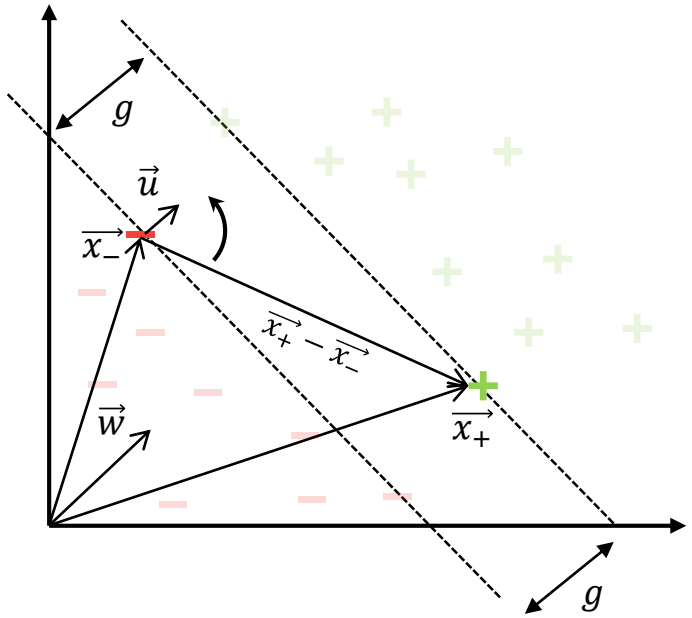
- How do we express the width of the gap?
- $(\overrightarrow{x_+} - \overrightarrow{x_-})$ is a vector from one side of the gap to the other.
- Let $\vec{u} \perp h$ be a unit vector



Making the margin “wide”

- How do we express the width of the gap?
- $(\overrightarrow{x_+} - \overrightarrow{x_-})$ is a vector from one side of the gap to the other.
- Let $\vec{u} \perp h$ be a unit vector

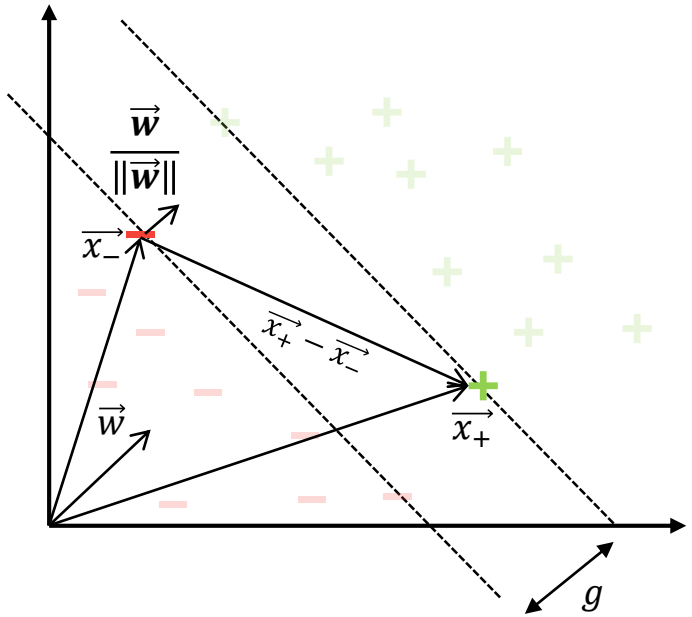
$$g = (\overrightarrow{x_+} - \overrightarrow{x_-}) \cdot \vec{u}$$



Making the margin “wide”

- How do we express the width of the gap?
- $(\overrightarrow{x_+} - \overrightarrow{x_-})$ is a vector from one side of the gap to the other.
- Let $\vec{u} \perp h$ be a unit vector

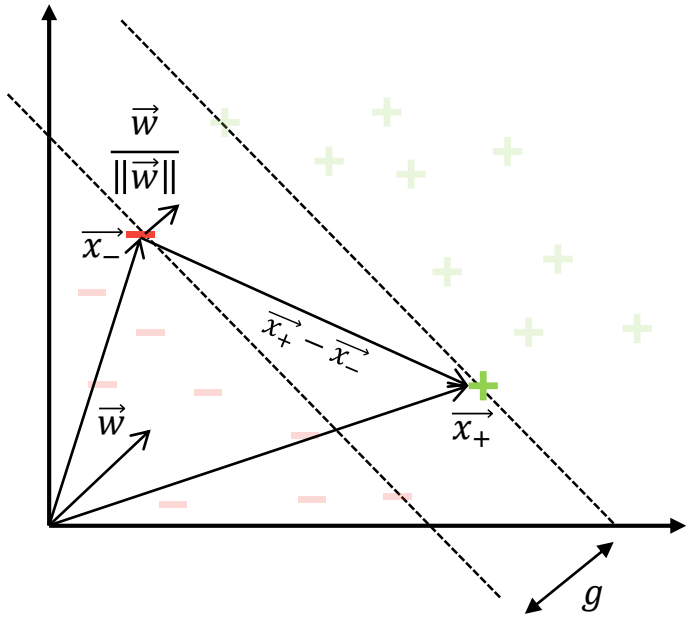
$$g = (\overrightarrow{x_+} - \overrightarrow{x_-}) \cdot \frac{\vec{w}}{\|\vec{w}\|}$$



Making the margin “wide”

- How do we express the width of the gap?
- $(\overrightarrow{x_+} - \overrightarrow{x_-})$ is a vector from one side of the gap to the other.
- Let $\vec{u} \perp h$ be a unit vector

$$g = \frac{\langle \overrightarrow{x_+}, \vec{w} \rangle - \langle \overrightarrow{x_-}, \vec{w} \rangle}{\|\vec{w}\|}$$

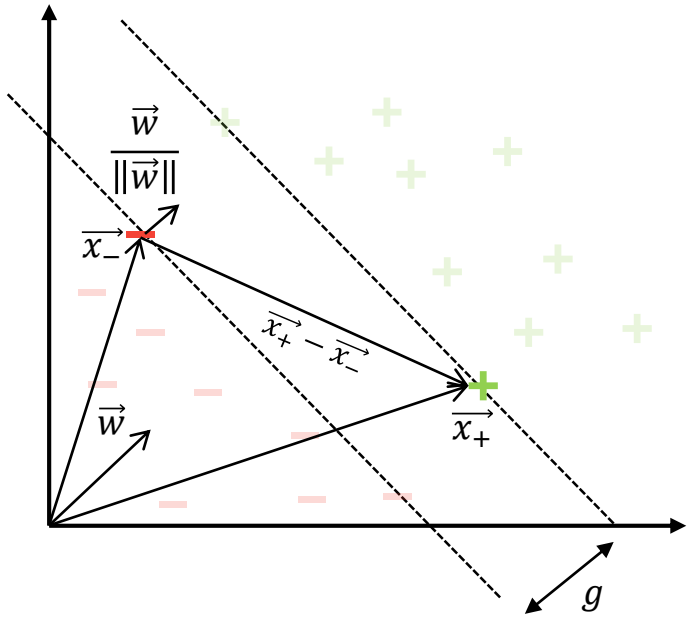


Making the margin “wide”

- How do we express the width of the gap?
- $(\overrightarrow{x_+} - \overrightarrow{x_-})$ is a vector from one side of the gap to the other.
- Let $\vec{u} \perp h$ be a unit vector

$$g = \frac{\langle \overrightarrow{x_+}, \vec{w} \rangle - \langle \overrightarrow{x_-}, \vec{w} \rangle}{\|\vec{w}\|}$$

- But $y(\langle \vec{x}, \vec{w} \rangle + b) - 1 = 0$ for $\vec{x}$ on the edge:



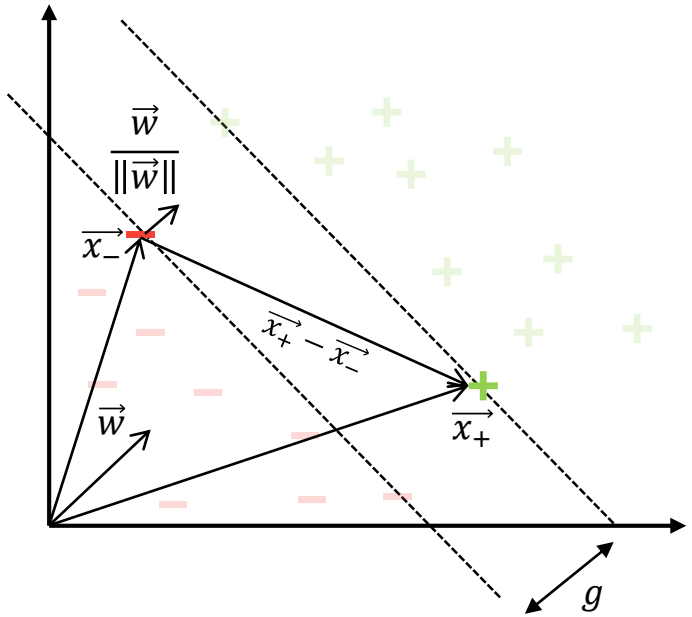
Making the margin “wide”

- How do we express the width of the gap?
- $(\overrightarrow{x_+} - \overrightarrow{x_-})$ is a vector from one side of the gap to the other.
- Let $\vec{u} \perp h$ be a unit vector

$$g = \frac{\langle \overrightarrow{x_+}, \vec{w} \rangle - \langle \overrightarrow{x_-}, \vec{w} \rangle}{\|\vec{w}\|}$$

- But $y(\langle \vec{x}, \vec{w} \rangle + b) - 1 = 0$ for $\vec{x}$ on the edge:

$$g = \frac{(1 - b) - (-1 - b)}{\|\vec{w}\|}$$



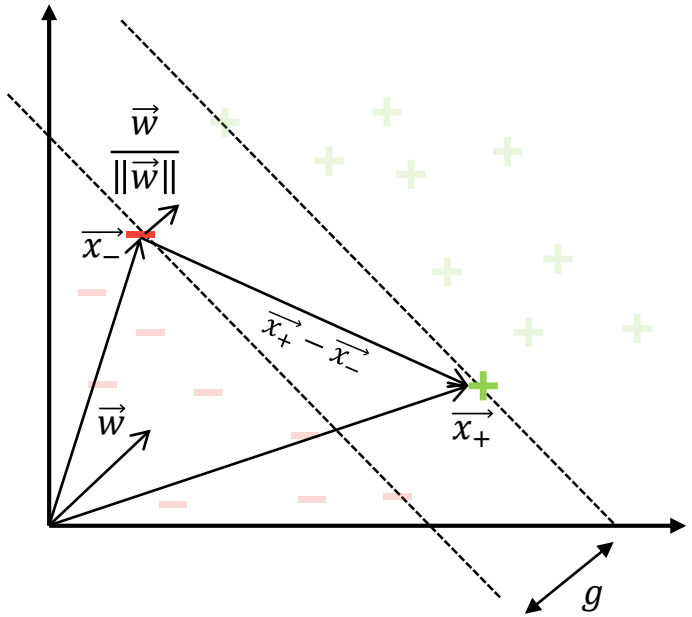
Making the margin “wide”

- How do we express the width of the gap?
- $(\overrightarrow{x_+} - \overrightarrow{x_-})$ is a vector from one side of the gap to the other.
- Let $\vec{u} \perp h$ be a unit vector

$$g = \frac{\langle \overrightarrow{x_+}, \vec{w} \rangle - \langle \overrightarrow{x_-}, \vec{w} \rangle}{\|\vec{w}\|}$$

- But $y(\langle \vec{x}, \vec{w} \rangle + b) - 1 = 0$ for $\vec{x}$ on the edge:

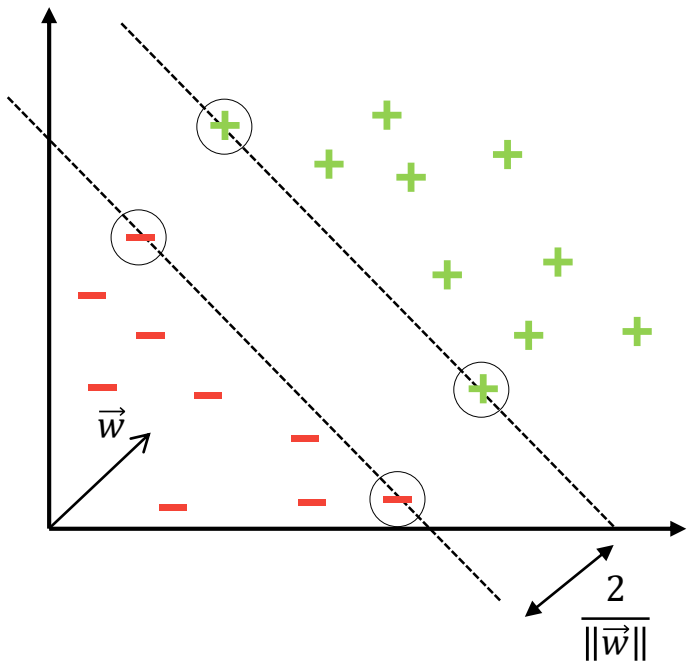
$$\boldsymbol{g = \frac{2}{\|\vec{w}\|}}$$



Making the margin “wide”

- How do we express the width of the gap?

$$g = \frac{2}{\|\vec{w}\|}$$



Making the margin "wide"

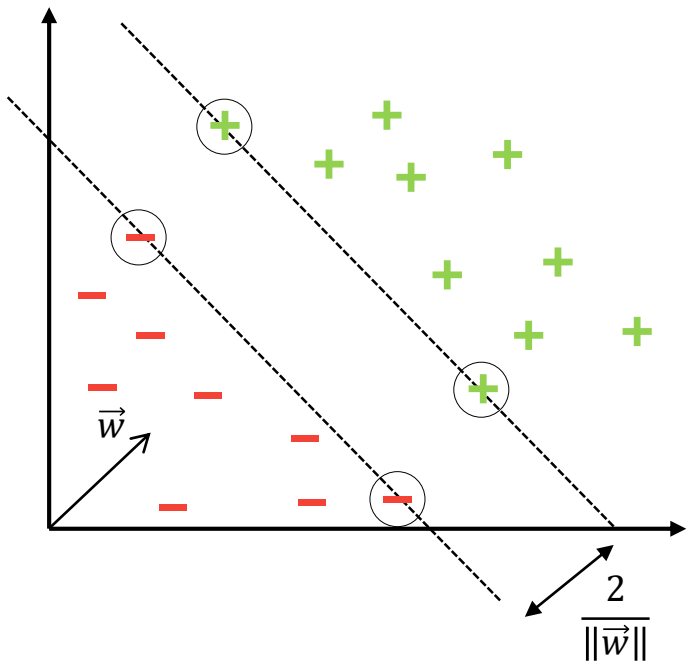
- How do we express the width of the gap?

$$g = \frac{2}{\|\vec{w}\|}$$

- We want to *maximize the gap:*

$$\text{maximize} \frac{2}{\|\vec{w}\|} \Rightarrow \text{minimize } \|\vec{w}\| \Rightarrow$$

$$\textbf{minimize} \frac{\|\vec{w}\|^2}{2}$$



What we have so far

SVM Primal Form

- The decision rule is:

$$\vec{x} \text{ is a “+” sample if } \langle \vec{x}, \vec{w} \rangle + \mathbf{b} \geq \mathbf{0}$$

- In order to obtain $\vec{w}$ and b we need to:

$$\text{minimize } \frac{\|\vec{w}\|^2}{2}$$

$$\text{subject to } y^{(i)}\left(\langle \vec{x}^{(i)}, \vec{w} \rangle + b\right) - 1 \geq 0$$

SVM Dual Form

Lagrange multipliers

- Method of finding the extremum of a function subject to constrains.
- We want to **minimize** $\frac{\|\vec{w}\|^2}{2}$ **subject to** $y^{(i)}\left(\langle \vec{x}^{(i)}, \vec{w} \rangle + b\right) - 1 \geq 0$

Lagrange multipliers

- Method of finding the extremum of a function subject to constrains.
- We want to **minimize** $\frac{\|\vec{w}\|^2}{2}$ **subject to** $y^{(i)}\left(\langle \vec{x}^{(i)}, \vec{w} \rangle + b\right) - 1 \geq 0$
- We define a new function and find its minimum instead:

$$L(\vec{w}, b, \alpha) = \frac{\|\vec{w}\|^2}{2} - \sum_i \alpha_i \left[y^{(i)}\left(\langle \vec{x}^{(i)}, \vec{w} \rangle + b\right) - 1 \right] \qquad \alpha_i \geq 0$$

α_i are called **Lagrange multipliers**.

Lagrange multipliers

- Method of finding the extremum of a function subject to constrains.
- We want to **minimize** $\frac{\|\vec{w}\|^2}{2}$ **subject to** $y^{(i)}\left(\langle \vec{x}^{(i)}, \vec{w} \rangle + b\right) - 1 \geq 0$
- We define a new function and find its minimum instead:

$$L(\vec{w}, b, \alpha) = \frac{\|\vec{w}\|^2}{2} - \sum_i \alpha_i \left[y^{(i)}\left(\langle \vec{x}^{(i)}, \vec{w} \rangle + b\right) - 1 \right] \qquad \alpha_i \geq 0$$

$$\frac{\partial L}{\partial \vec{w}} = \vec{w} - \sum_i \alpha_i y^{(i)} \vec{x}^{(i)} = 0 \Longrightarrow \boldsymbol{\vec{w} = \sum_i \alpha_i y^{(i)} \vec{x}^{(i)}}$$

$$\frac{\partial L}{\partial b} = -\sum_i \alpha_i y^{(i)} = 0 \Longrightarrow \boldsymbol{\sum_i \alpha_i y^{(i)} = 0}$$

α_i are called **Lagrange multipliers.**

Lagrange multipliers

- Method of finding the extremum of a function subject to constrains.
- We want to **minimize** $\frac{\|\vec{w}\|^2}{2}$ **subject to** $y^{(i)}\left(\langle \vec{x}^{(i)}, \vec{w} \rangle + b\right) - 1 \geq 0$
- We define a new function and find its minimum instead:

$$L(\vec{w}, b, \alpha) = \frac{\|\vec{w}\|^2}{2} - \sum_i \alpha_i \left[y^{(i)}\left(\langle \vec{x}^{(i)}, \vec{w} \rangle + b\right) - 1 \right] \qquad \alpha_i \geq 0$$

$$\frac{\partial L}{\partial \vec{w}} = \vec{w} - \sum_i \alpha_i y^{(i)} \vec{x}^{(i)} = 0 \Longrightarrow \vec{w} = \sum_i \boldsymbol{\alpha_i y^{(i)} \vec{x}^{(i)}}$$

$$\frac{\partial L}{\partial b} = -\sum_i \alpha_i y^{(i)} = 0 \Longrightarrow \sum_i \boldsymbol{\alpha_i y^{(i)}} = \mathbf{0}$$

α_i are called **Lagrange multipliers.**

Lagrange multipliers

$$L = \frac{\|\vec{w}\|^2}{2} - \sum_i \alpha_i \left[y^{(i)} \left(\langle \vec{x}^{(i)}, \vec{w} \rangle + b \right) - 1 \right]$$

$$\vec{w} = \sum_i \alpha_i y^{(i)} \vec{x}^{(i)}$$

$$\sum_i \alpha_i y^{(i)} = 0$$

Lagrange multipliers

$$\left.\begin{aligned} L &= \frac{\|\vec{w}\|^2}{2} - \sum_i \alpha_i \left[y^{(i)} \left(\langle \vec{x}^{(i)}, \vec{w} \rangle + b \right) - 1 \right] \\ \vec{w} &= \sum_i \alpha_i y^{(i)} \vec{x}^{(i)} \\ \sum_i \alpha_i y^{(i)} &= 0 \end{aligned}\right\} L = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y^{(i)} y^{(j)} \langle \vec{x}^{(i)}, \vec{x}^{(j)} \rangle$$

Lagrange multipliers

$$L = \frac{\|\vec{w}\|^2}{2} - \sum_i \alpha_i \left[y^{(i)} \left(\langle \vec{x}^{(i)}, \vec{w} \rangle + b \right) - 1 \right]$$
$$\vec{w} = \sum_i \alpha_i y^{(i)} \vec{x}^{(i)}$$
$$\sum_i \alpha_i y^{(i)} = 0$$

$$\Rightarrow L = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y^{(i)} y^{(j)} \langle \vec{x}^{(i)}, \vec{x}^{(j)} \rangle$$

- The optimization only depends on the **dot product of pairs of training samples**.

Lagrange multipliers

$$\left.\begin{aligned} L &= \frac{\|\vec{w}\|^2}{2} - \sum_i \alpha_i \left[y^{(i)} \left(\langle \vec{x}^{(i)}, \vec{w} \rangle + b \right) - 1 \right] \\ \vec{w} &= \sum_i \alpha_i y^{(i)} \vec{x}^{(i)} \\ \sum_i \alpha_i y^{(i)} &= 0 \end{aligned}\right\} \quad L = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y^{(i)} y^{(j)} \langle \vec{x}^{(i)}, \vec{x}^{(j)} \rangle$$

- The optimization only depends on the **dot product of pairs of training samples**.
- Our decision rule was: $\vec{x}$ is a "+" sample if $\langle \vec{x}, \vec{w} \rangle + b \geq 0$

  now it becomes: $\sum_i \alpha_i y^{(i)} \langle \vec{x}^{(i)}, \vec{x} \rangle + b \geq 0$

Lagrange multipliers

$$\left.\begin{aligned} L &= \frac{\|\vec{w}\|^2}{2} - \sum_i \alpha_i \left[y^{(i)} \left(\langle \vec{x}^{(i)}, \vec{w} \rangle + b \right) - 1 \right] \\ \vec{w} &= \sum_i \alpha_i y^{(i)} \vec{x}^{(i)} \\ \sum_i \alpha_i y^{(i)} &= 0 \end{aligned} \right\} \quad L = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y^{(i)} y^{(j)} \langle \vec{x}^{(i)}, \vec{x}^{(j)} \rangle$$

- The optimization only depends on the **dot product of pairs of training samples**.
- Our decision rule was: $\vec{x}$ is a "+" sample if $\langle \vec{x}, \vec{w} \rangle + b \geq 0$

  now it becomes: $\sum_i \alpha_i y^{(i)} \langle \vec{x}^{(i)}, \vec{x} \rangle + b \geq 0$
- The decision also only depends on the **dot product.**

Lagrange multipliers

$$L = \frac{\|\vec{w}\|^2}{2} - \sum_i \alpha_i \left[y^{(i)} \left(\langle \vec{x}^{(i)}, \vec{w} \rangle + b \right) - 1 \right]$$

$$\vec{w} = \sum_i \alpha_i y^{(i)} \vec{x}^{(i)}$$

$$\sum_i \alpha_i y^{(i)} = 0$$

$$\Rightarrow \quad L = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y^{(i)} y^{(j)} \langle \vec{x}^{(i)}, \vec{x}^{(j)} \rangle$$

α_i will be 0 for all samples which are not support vectors

- The optimization only depends on the **dot product of pairs of training samples**.
- Our decision rule was: $\vec{x}$ is a "+" sample if $\langle \vec{x}, \vec{w} \rangle + b \geq 0$

  now it becomes: $\sum_i \alpha_i y^{(i)} \langle \vec{x}^{(i)}, \vec{x} \rangle + b \geq 0$
- The decision also only depends on the **dot product.**

Lagrange multipliers

$$L = \frac{\|\vec{w}\|^2}{2} - \sum_i \alpha_i \left[y^{(i)} \left(\langle \vec{x}^{(i)}, \vec{w} \rangle + b \right) - 1 \right]$$

$$\vec{w} = \sum_i \alpha_i y^{(i)} \vec{x}^{(i)}$$

$$\sum_i \alpha_i y^{(i)} = 0$$

$$\Rightarrow \quad L = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y^{(i)} y^{(j)} \langle \vec{x}^{(i)}, \vec{x}^{(j)} \rangle$$

α_i will be 0 for all samples which are not support vectors

- The optimization only depends on the **dot product of pairs of training samples**.
- Our decision rule was: $\vec{x}$ is a "+" sample if $\langle \vec{x}, \vec{w} \rangle + b \geq 0$

  now it becomes: $\sum_i \alpha_i y^{(i)} \langle \vec{x}^{(i)}, \vec{x} \rangle + b \geq 0$
- The decision also only depends on the **dot product.**

b can be computed from $\sum_i \alpha_i y^{(i)} \langle \vec{x}^{(i)}, \vec{x}_+ \rangle + b = 1$ for some positive support vector $\vec{x}_+$

SVM Dual Form

- The decision rule is:

$$\vec{x} \text{ is a “+” sample if } \sum_i \alpha_i y^{(i)} \langle \vec{x}^{(i)}, \vec{x} \rangle + b \geq 0$$

- In order to obtain α_i and b we need to:

$$\text{minimize } \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y^{(i)} y^{(j)} \langle \vec{x}^{(i)}, \vec{x}^{(j)} \rangle$$
$$\text{subject to } \alpha_i \geq 0 \; \forall i$$

Primal vs. Dual

	Decision Rule	Optimization Problem
Primal Form	$\langle \vec{x}, \vec{w} \rangle + b \geq 0$	$\text{minimize } \frac{\|\vec{w}\|^2}{2}$ $\text{subject to } y^{(i)}\left(\langle \vec{x}^{(i)}, \vec{w} \rangle + b\right) - 1 \geq 0$
Dual Form	$\sum_i \alpha_i y^{(i)} \langle \vec{x}^{(i)}, \vec{x} \rangle + b \geq 0$	$\text{minimize } \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y^{(i)} y^{(j)} \langle \vec{x}^{(i)}, \vec{x}^{(j)} \rangle$ $\text{subject to } \alpha_i \geq 0$

- Primal has as many parameters as there are features (n).
- Dual has as many parameters as there are data points (m).

Not counting the bias term b.

- It is more efficient to optimize the primal if $m \gg n$ (more samples then features).
- When $n \approx m$ it is slightly easier to optimize the dual (more efficient techniques).
- But the true power of the dual is that the optimization and decision are expressed only in terms of **dot product with training samples**.

Usually data is not linearly separable

- All previous computations started from the assumption that the data points were *linearly separable*.
  - i.e. The hyperplane which separates the “+” and “-” classes exists.
- How can an SVM deal with *non-separable data*?

“Hard-margin” **linear SVM**

Usually data is not linearly separable

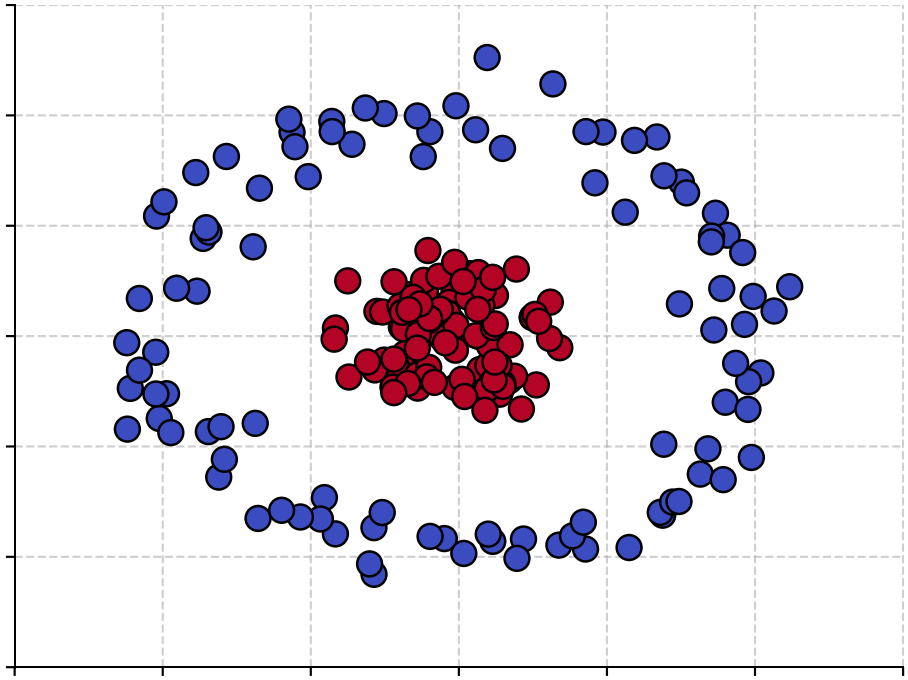
- All previous computations started from the assumption that the data points were *linearly separable*.
  - i.e. The hyperplane which separates the “+” and “-” classes exists.

“Hard-margin” linear SVM

- How can an SVM deal with *non-separable data*?
- **“Kernel trick”** – Map the data in an *implicit higher-dimensional* space where it is linearly separable.
- **“Soft-margin SVM”** – Allow some mistakes during training, with a cost.

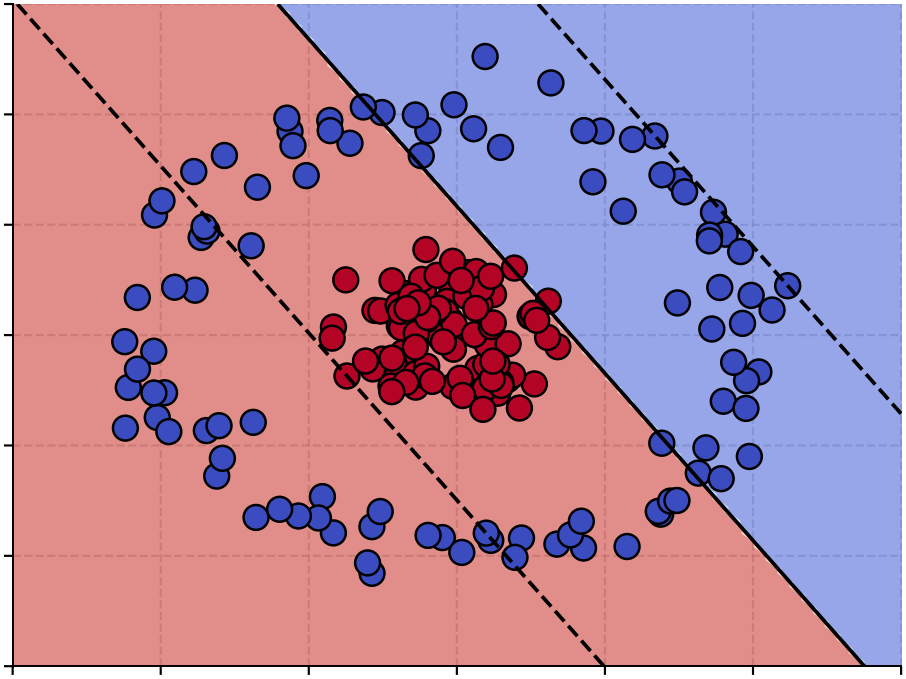
The Kernel Trick

Non-linear data



Non-linear data

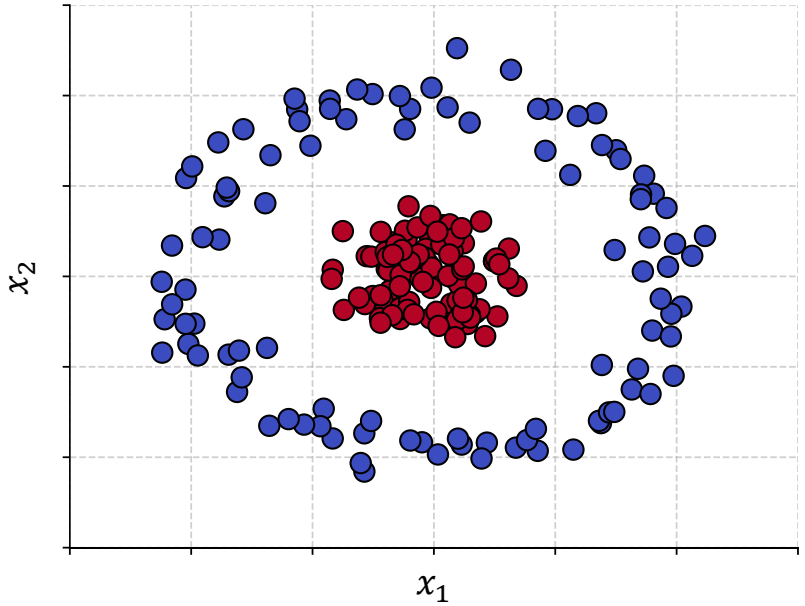
- A *hard-margin SVM* doesn't perform too well.



Non-linear data

$\vec{x} \in \mathbb{R}, \vec{x} = (x_1, x_2)$

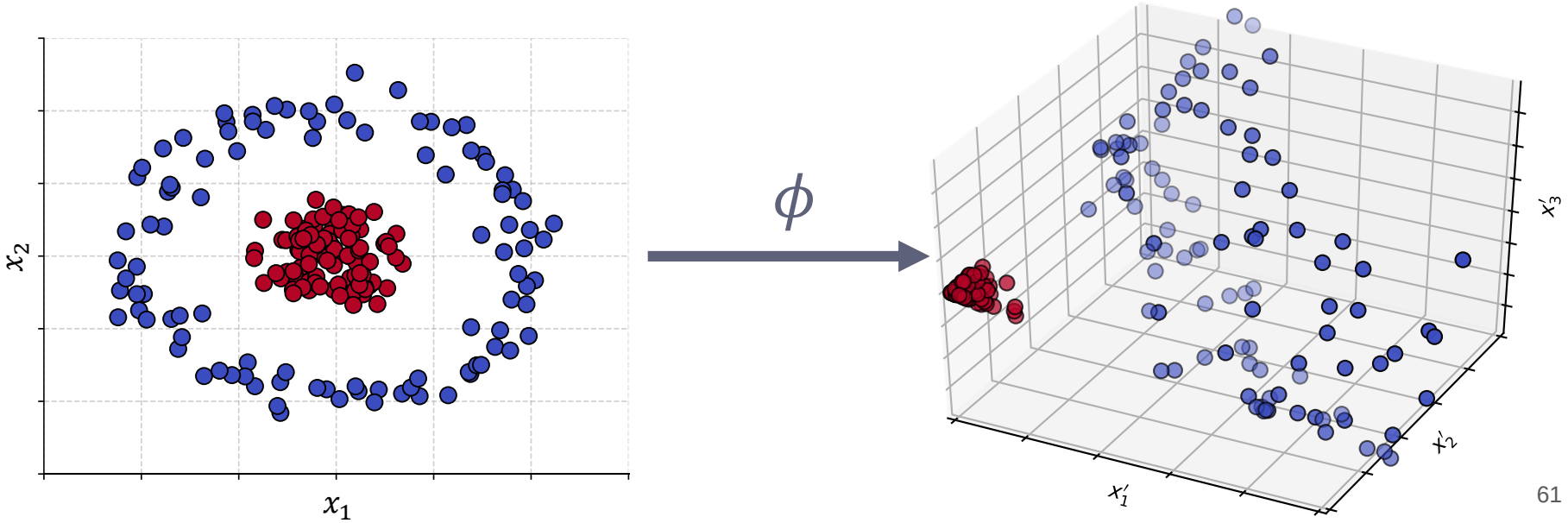
$\phi: \mathbb{R}^2 \longrightarrow \mathbb{R}^3, \phi\big((x_1, x_2)\big) = (x_1', x_2', x_3') = \big(x_1^2, x_2^2, x_1 x_2 \sqrt{2}\big)$



Non-linear data

$\vec{x} \in \mathbb{R}, \vec{x} = (x_1, x_2)$

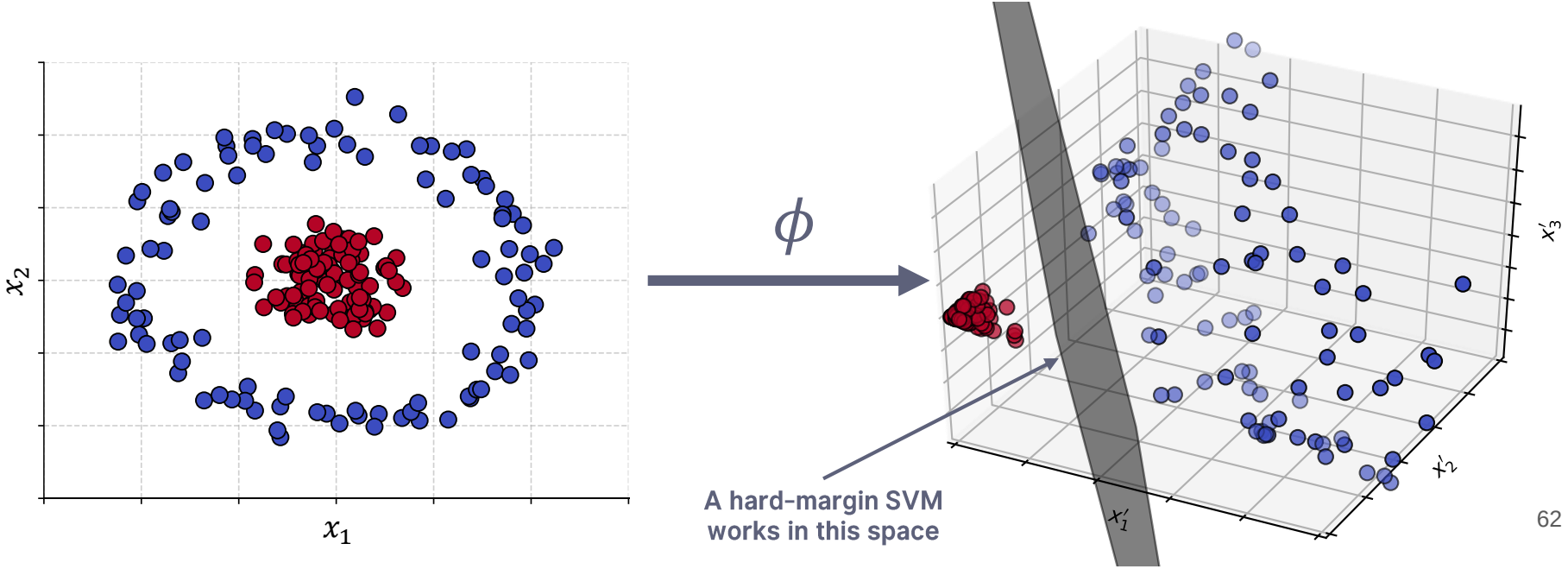
$\phi: \mathbb{R}^2 \longrightarrow \mathbb{R}^3, \phi\big((x_1, x_2)\big) = (x_1', x_2', x_3') = \big(x_1^2, x_2^2, x_1 x_2 \sqrt{2}\big)$



Non-linear data

$\vec{x} \in \mathbb{R}, \vec{x} = (x_1, x_2)$

$\phi: \mathbb{R}^2 \longrightarrow \mathbb{R}^3, \phi\big((x_1, x_2)\big) = (x_1', x_2', x_3') = \big(x_1^2, x_2^2, x_1 x_2 \sqrt{2}\big)$



So what is the “trick”?

- Remember that the optimization and decision rule only depend on the dot product:

$$L = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y^{(i)} y^{(j)} \left\langle \vec{x}^{(i)}, \vec{x}^{(j)} \right\rangle$$

$$\sum_i \alpha_i y^{(i)} \left\langle \vec{x}^{(i)}, \vec{x} \right\rangle + b \geq 0 \Rightarrow \vec{x} \text{ is a } +$$

- So all we need to do is a suitable transformation ϕ and use:

$$L = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y^{(i)} y^{(j)} \left\langle \phi\left(\vec{x}^{(i)}\right), \phi\left(\vec{x}^{(j)}\right) \right\rangle$$

$$\sum_i \alpha_i y^{(i)} \left\langle \phi\left(\vec{x}^{(i)}\right), \phi(\vec{x}) \right\rangle + b \geq 0 \Rightarrow \vec{x} \text{ is a } +$$

?

So what is the “trick”?

- Remember that the optimization and decision rule only depend on the dot product:

$$L = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y^{(i)} y^{(j)} \left\langle \vec{x}^{(i)}, \vec{x}^{(j)} \right\rangle$$

$$\sum_i \alpha_i y^{(i)} \left\langle \vec{x}^{(i)}, \vec{x} \right\rangle + b \geq 0 \Rightarrow \vec{x} \text{ is a } +$$

- So all we need to do is a suitable transformation ϕ and use:

$$L = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y^{(i)} y^{(j)} \left\langle \phi\left(\vec{x}^{(i)}\right), \phi\left(\vec{x}^{(j)}\right) \right\rangle$$

$$\sum_i \alpha_i y^{(i)} \left\langle \phi\left(\vec{x}^{(i)}\right), \phi(\vec{x}) \right\rangle + b \geq 0 \Rightarrow \vec{x} \text{ is a } +$$

? No

So what is the “trick”?

$$\vec{x}, \vec{y} \in \mathbb{R}^2, \phi(\vec{x}) = \overrightarrow{x'}, \phi(\vec{y}) = \overrightarrow{y'}$$

$$\langle \vec{x}, \vec{y} \rangle = \langle (x_1, x_2\ ), (y_1, y_2\ ) \rangle = x_1 y_1 + x_2 y_2$$

So what is the “trick”?

$$\vec{x}, \vec{y} \in \mathbb{R}^2, \phi(\vec{x}) = \overrightarrow{x'}, \phi(\vec{y}) = \overrightarrow{y'}$$

$$\langle \vec{x}, \vec{y} \rangle = \langle (x_1, x_2\ ), (y_1, y_2\ ) \rangle = x_1 y_1 + x_2 y_2$$

$$\langle \overrightarrow{x'}, \overrightarrow{y'} \rangle = \langle (x_1', x_2', x_3'), (y_1', y_2', y_3') \rangle = \langle (x_1^2, x_2^2, x_1 x_2 \sqrt{2}), (y_1^2, y_2^2, y_1 y_2 \sqrt{2}) \rangle$$

So what is the “trick”?

$$\vec{x}, \vec{y} \in \mathbb{R}^2, \phi(\vec{x}) = \overrightarrow{x'}, \phi(\vec{y}) = \overrightarrow{y'}$$

$$\langle \vec{x}, \vec{y} \rangle = \langle (x_1, x_2\ ), (y_1, y_2\ ) \rangle = x_1 y_1 + x_2 y_2$$

$$\left\langle \overrightarrow{x'}, \overrightarrow{y'} \right\rangle = \langle (x_1', x_2', x_3'), (y_1', y_2', y_3') \rangle = \left\langle \left(x_1^2, x_2^2, x_1 x_2 \sqrt{2}\right), \left(y_1^2, y_2^2, y_1 y_2 \sqrt{2}\right) \right\rangle =$$
$$= x_1^2 y_1^2 + x_2^2 y_2^2 + 2 x_1 x_2 y_1 y_2$$

So what is the “trick”?

$$\vec{x}, \vec{y} \in \mathbb{R}^2, \phi(\vec{x}) = \overrightarrow{x'}, \phi(\vec{y}) = \overrightarrow{y'}$$

$$\langle \vec{x}, \vec{y} \rangle = \langle (x_1, x_2\ ), (y_1, y_2\ ) \rangle = x_1 y_1 + x_2 y_2$$

$$\langle \overrightarrow{x'}, \overrightarrow{y'} \rangle = \langle (x_1', x_2', x_3'), (y_1', y_2', y_3') \rangle = \langle (x_1^2, x_2^2, x_1 x_2 \sqrt{2}), (y_1^2, y_2^2, y_1 y_2 \sqrt{2}) \rangle =$$
$$= x_1^2 y_1^2 + x_2^2 y_2^2 + 2 x_1 x_2 y_1 y_2 = (x_1 y_1 + x_2 y_2)^2$$

So what is the “trick”?

$$\vec{x}, \vec{y} \in \mathbb{R}^2, \phi(\vec{x}) = \overrightarrow{x'}, \phi(\vec{y}) = \overrightarrow{y'}$$

$$\langle \vec{x}, \vec{y} \rangle = \langle (x_1, x_2\ ), (y_1, y_2\ ) \rangle = x_1 y_1 + x_2 y_2$$

$$\left\langle \overrightarrow{x'}, \overrightarrow{y'} \right\rangle = \langle (x_1', x_2', x_3'), (y_1', y_2', y_3') \rangle = \left\langle \left(x_1^2, x_2^2, x_1 x_2 \sqrt{2}\right), \left(y_1^2, y_2^2, y_1 y_2 \sqrt{2}\right) \right\rangle =$$
$$= x_1^2 y_1^2 + x_2^2 y_2^2 + 2 x_1 x_2 y_1 y_2 = (x_1 y_1 + x_2 y_2)^2 = \langle \vec{x}, \vec{y} \rangle^2$$

So what is the “trick”?

$$\vec{x}, \vec{y} \in \mathbb{R}^2, \phi(\vec{x}) = \overrightarrow{x'}, \phi(\vec{y}) = \overrightarrow{y'}$$

$$\langle \vec{x}, \vec{y} \rangle = \langle (x_1, x_2\ ), (y_1, y_2\ ) \rangle = x_1 y_1 + x_2 y_2$$

$$\langle \overrightarrow{x'}, \overrightarrow{y'} \rangle = \langle (x_1', x_2', x_3'), (y_1', y_2', y_3') \rangle = \langle (x_1^2, x_2^2, x_1 x_2 \sqrt{2}), (y_1^2, y_2^2, y_1 y_2 \sqrt{2}) \rangle =$$
$$= x_1^2 y_1^2 + x_2^2 y_2^2 + 2 x_1 x_2 y_1 y_2 = (x_1 y_1 + x_2 y_2)^2 = \langle \vec{x}, \vec{y} \rangle^2$$

The dot product in the transformed space is the square of the dot product in the original space.

So what is the "trick"?

$$\vec{x}, \vec{y} \in \mathbb{R}^2, \phi(\vec{x}) = \overrightarrow{x'}, \phi(\vec{y}) = \overrightarrow{y'}$$

$$\langle \vec{x}, \vec{y} \rangle = \langle (x_1, x_2\ ), (y_1, y_2\ ) \rangle = x_1 y_1 + x_2 y_2$$

$$\langle \overrightarrow{x'}, \overrightarrow{y'} \rangle = \langle (x_1', x_2', x_3'), (y_1', y_2', y_3') \rangle = \langle (x_1^2, x_2^2, x_1 x_2 \sqrt{2}), (y_1^2, y_2^2, y_1 y_2 \sqrt{2}) \rangle =$$
$$= x_1^2 y_1^2 + x_2^2 y_2^2 + 2 x_1 x_2 y_1 y_2 = (x_1 y_1 + x_2 y_2)^2 = \langle \vec{x}, \vec{y} \rangle^2$$

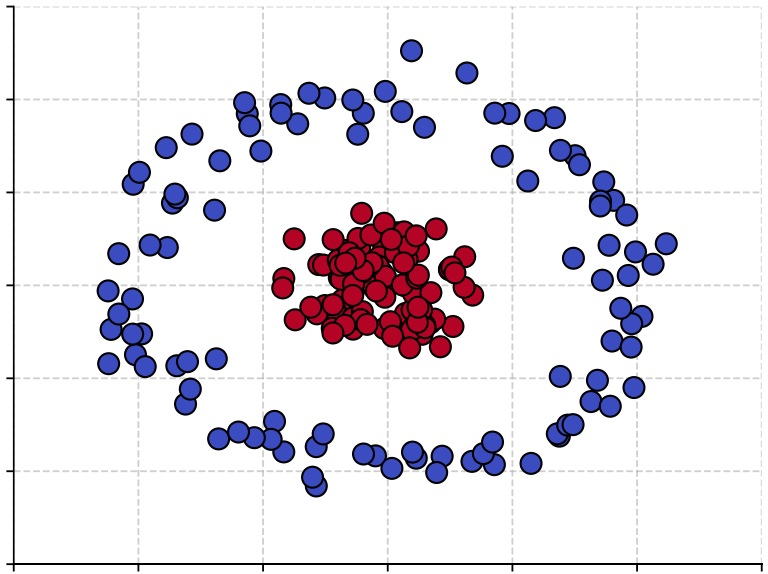
The dot product in the transformed space is the square of the dot product in the original space.

Let $K: \mathbb{R}^2 \times \mathbb{R}^2 \longrightarrow \mathbb{R}, K(\vec{x}, \vec{y}) = \langle \vec{x}, \vec{y} \rangle^2 = \langle \phi(\vec{x}), \phi(\vec{y}) \rangle$

The function K is defined in the original space, but computes the dot product in the transformed space!

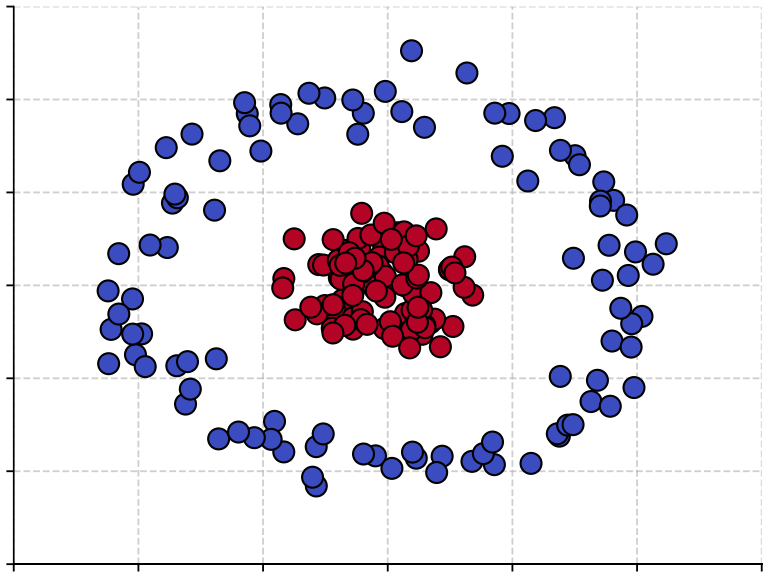
So what is the “trick”?

- Again, the optimization and decision only depend on the dot product.



So what is the “trick”?

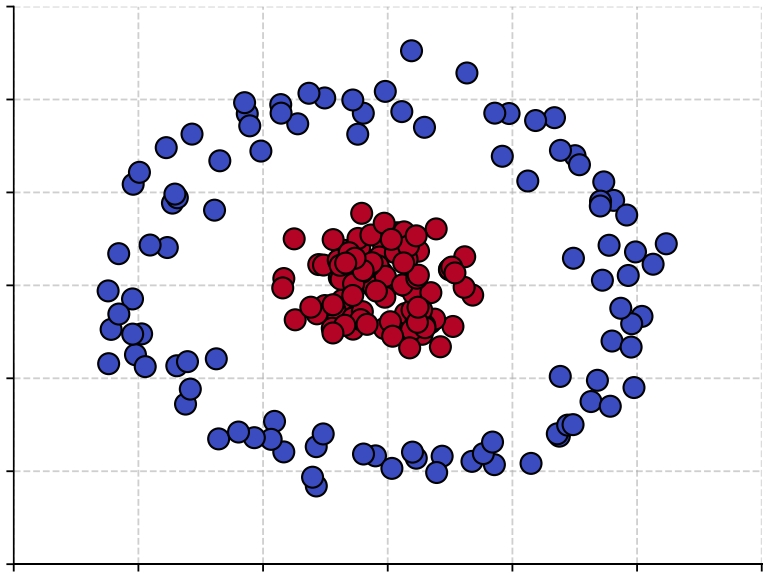
- Again, the optimization and decision only depend on the dot product.
- How do we compute the dot product in the high-dimensional space, without explicitly mapping each point?



So what is the “trick”?

- Again, the optimization and decision only depend on the dot product.
- How do we compute the dot product in the high-dimensional space, without explicitly mapping each point?

$$K(\vec{x}, \vec{y}) = \langle \vec{x}, \vec{y} \rangle^2 = \langle \phi(\vec{x}), \phi(\vec{y}) \rangle$$



So what is the “trick”?

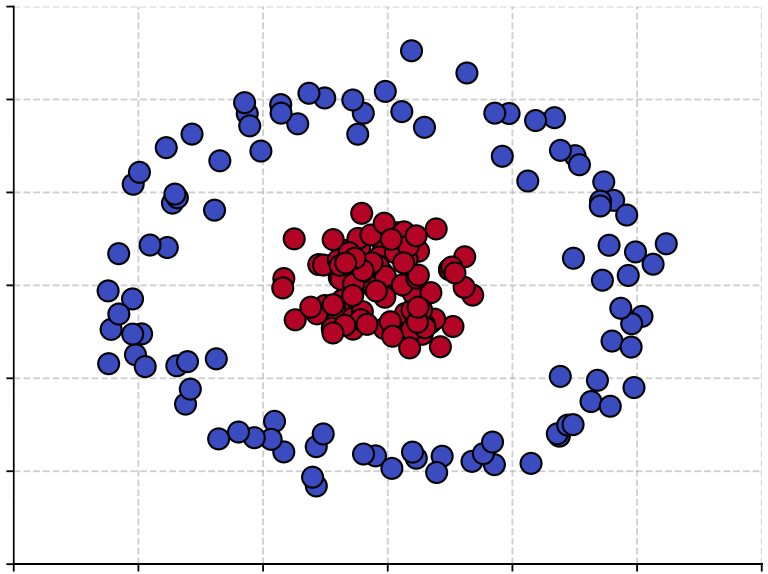
- Again, the optimization and decision only depend on the dot product.
- How do we compute the dot product in the high-dimensional space, without explicitly mapping each point?

$$K(\vec{x}, \vec{y}) = \langle \vec{x}, \vec{y} \rangle^2 = \langle \phi(\vec{x}), \phi(\vec{y}) \rangle$$

- All we need to do is use:

$$L = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y^{(i)} y^{(j)} K\left(\vec{x}^{(i)}, \vec{x}^{(j)}\right)$$

$$\sum_i \alpha_i y^{(i)} K\left(\vec{x}^{(i)}, \vec{x}\right) + b \geq 0 \Rightarrow \vec{x} \text{ is a } +$$



So what is the “trick”?

- Again, the optimization and decision only depend on the dot product.
- How do we compute the dot product in the high-dimensional space, without explicitly mapping each point?

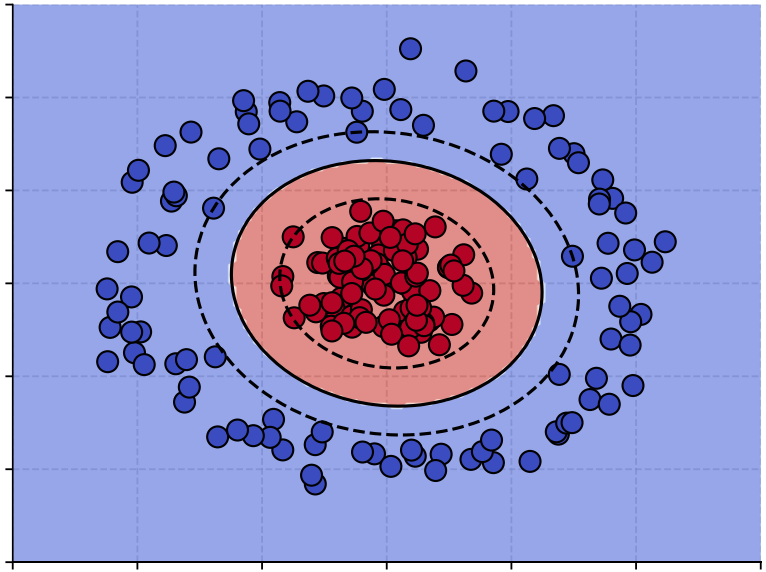
$$K(\vec{x}, \vec{y}) = \langle \vec{x}, \vec{y} \rangle^2 = \langle \phi(\vec{x}), \phi(\vec{y}) \rangle$$

- All we need to do is use:

$$L = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y^{(i)} y^{(j)} K\left(\vec{x}^{(i)}, \vec{x}^{(j)}\right)$$

$$\sum_i \alpha_i y^{(i)} K\left(\vec{x}^{(i)}, \vec{x}\right) + b \geq 0 \Rightarrow \vec{x} \text{ is a } +$$

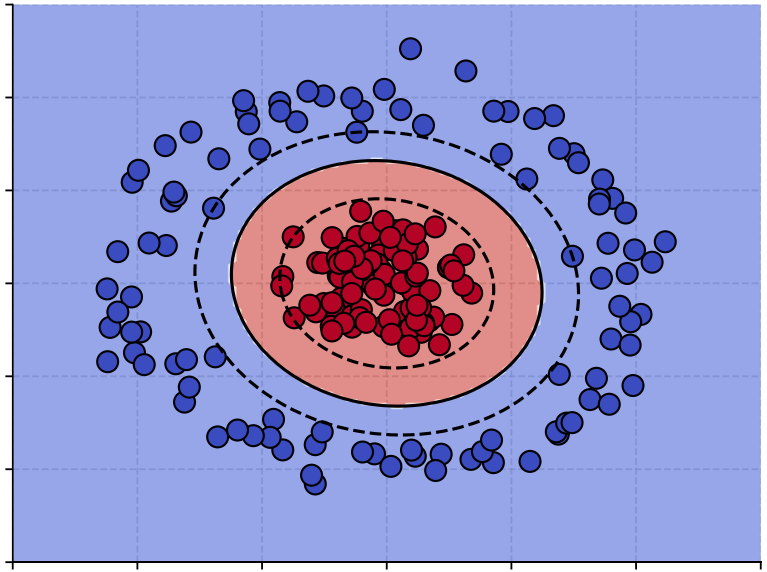
The decision boundary is no longer linear in the original space.



So what is the “trick”?

- The **kernel trick** means replacing the dot product with a *kernel function*, which creates an *implicit* high-dimensional feature space.
- Any algorithm which uses this approach is called a **kernel method**
- The kernel function can be viewed as a *similarity function* between data points.

The decision boundary is no longer linear in the original space.



SVM Dual Form with Kernel

- The decision rule is:

$$\vec{x} \text{ is a “+” sample if } \sum_i \alpha_i y^{(i)} K\left(\vec{x}^{(i)}, \vec{x}\right) + b \geq 0$$

- In order to obtain α_i and b we need to:

$$\text{minimize} \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y^{(i)} y^{(j)} K\left(\vec{x}^{(i)}, \vec{x}^{(j)}\right)$$
$$\text{subject to } \alpha_i \geq 0 \; \forall i$$

Most common kernel functions

Linear kernel:	$K(\vec{x}, \vec{y}) = \langle \vec{x}, \vec{y} \rangle$
Polynomial kernel:	$K(\vec{x}, \vec{y}) = (\langle \vec{x}, \vec{y} \rangle + c)^d$
Radial basis function (RBF) kernel:	$K(\vec{x}, \vec{y}) = e^{-\frac{\|\vec{x}-\vec{y}\|^2}{2\sigma^2}}$
Sigmoid kernel:	$K(\vec{x}, \vec{y}) = \tanh(\gamma \langle \vec{x}, \vec{y} \rangle + c)$

Most common kernel functions

Linear kernel:	$K(\vec{x}, \vec{y}) = \langle \vec{x}, \vec{y} \rangle$
Polynomial kernel:	$K(\vec{x}, \vec{y}) = (\langle \vec{x}, \vec{y} \rangle + c)^d$
Radial basis function (RBF) kernel:	$K(\vec{x}, \vec{y}) = e^{-\frac{\|\vec{x}-\vec{y}\|^2}{2\sigma^2}}$
Sigmoid kernel:	$K(\vec{x}, \vec{y}) = \tanh(\gamma \langle \vec{x}, \vec{y} \rangle + c)$

Kernel functions need to be symmetric and *positive semidefinite* (a.k.a. *Mercer's condition*), otherwise the existence of the underlying mapping ϕ is not guaranteed.

SVM Forms Recap

	Decision Rule	Optimization Problem
Primal Form	$\langle \vec{x}, \vec{w} \rangle + b \geq 0$	$\text{minimize } \frac{\|\vec{w}\|^2}{2}$ $\text{subject to } y^{(i)}\left(\langle \vec{x}^{(i)}, \vec{w} \rangle + b\right) - 1 \geq 0$
Dual Form	$\sum_i \alpha_i y^{(i)} \langle \vec{x}^{(i)}, \vec{x} \rangle + b \geq 0$	$\text{minimize} \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y^{(i)} y^{(j)} \langle \vec{x}^{(i)}, \vec{x}^{(j)} \rangle$ $\text{subject to } \alpha_i \geq 0$
Dual Form with Kernel	$\sum_i \alpha_i y^{(i)} K\left(\vec{x}^{(i)}, \vec{x}\right) + b \geq 0$	$\text{minimize} \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y^{(i)} y^{(j)} K\left(\vec{x}^{(i)}, \vec{x}^{(j)}\right)$ $\text{subject to } \alpha_i \geq 0$

There is no “primal form with kernel” because it is not expressed only in terms of dot product.

SVM Forms Recap

	Decision Rule	Optimization Problem
Primal Form	$\langle \vec{x}, \vec{w} \rangle + b \geq 0$	$\text{minimize } \frac{\|\vec{w}\|^2}{2}$ $\text{subject to } y^{(i)}\left(\langle \vec{x}^{(i)}, \vec{w} \rangle + b\right) - 1 \geq 0$
Dual Form *(with Kernel)*	$\sum_i \alpha_i y^{(i)} K\left(\vec{x}^{(i)}, \vec{x}\right) + b \geq 0$	$\text{minimize} \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y^{(i)} y^{(j)} K\left(\vec{x}^{(i)}, \vec{x}^{(j)}\right)$ $\text{subject to } \alpha_i \geq 0$

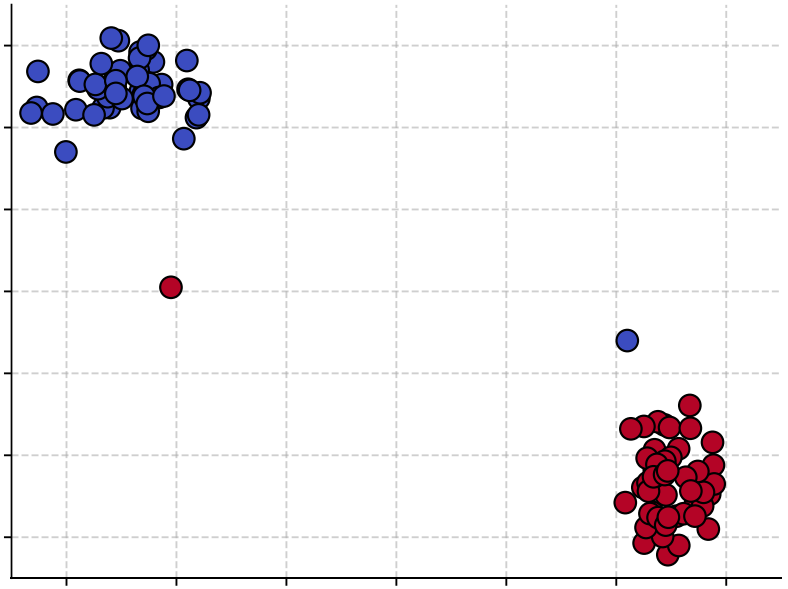
There is no “primal form with kernel” because it is not expressed only in terms of dot product.

The “no kernel dual form” can really be considered “dual form with linear kernel”.

Soft-margin SVM

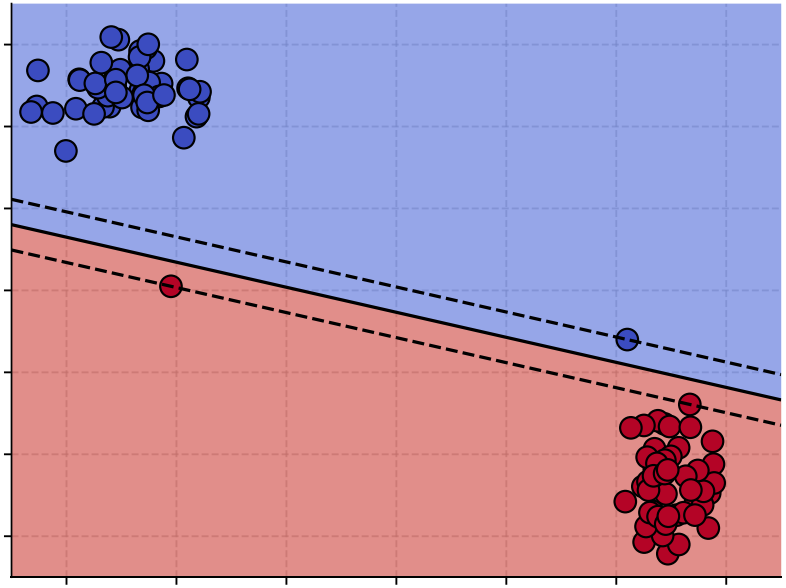
Soft-margin SVM

- What happens if we train a *hard-margin* SVM on this dataset?



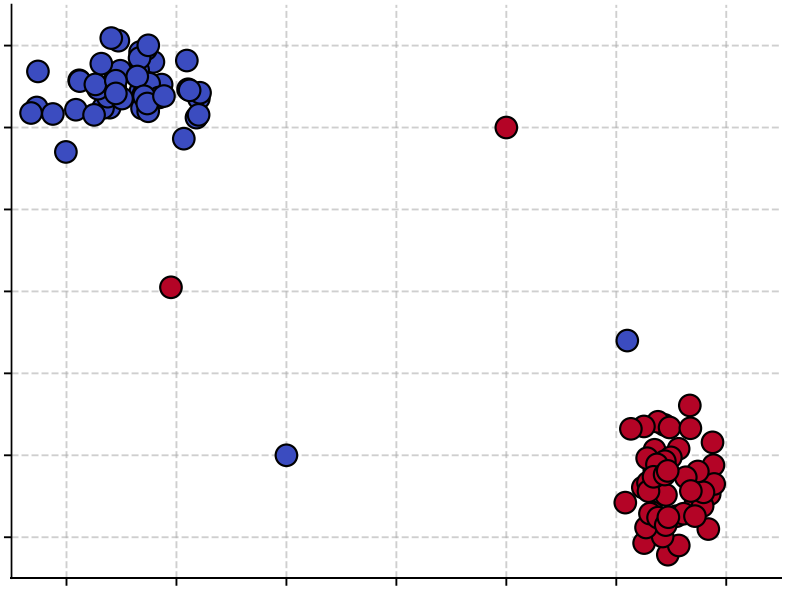
Soft-margin SVM

- What happens if we train a *hard-margin* SVM on this dataset?
- It **succeeds** at separating the training samples, but with a **very tight margin.**



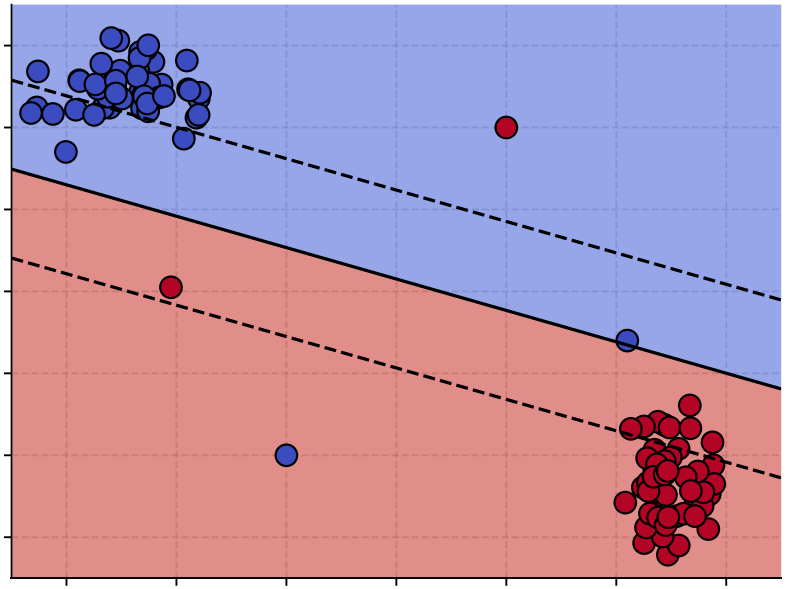
Soft-margin SVM

- What happens if we train a *hard-margin* SVM on this dataset?



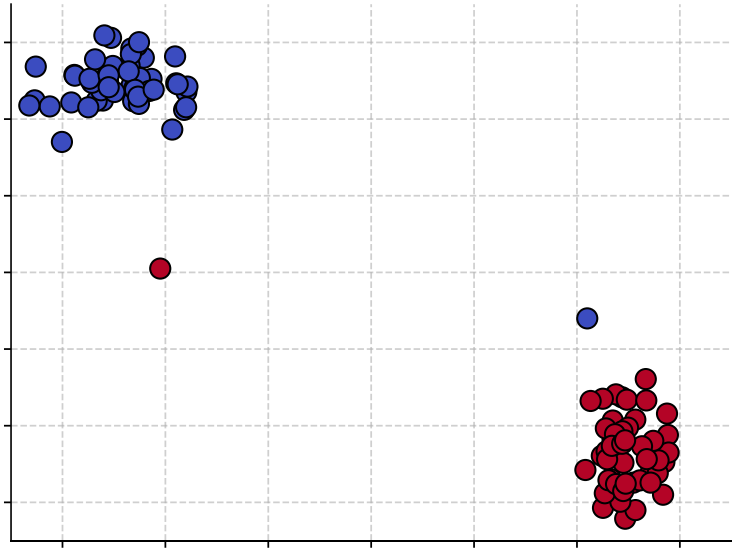
Soft-margin SVM

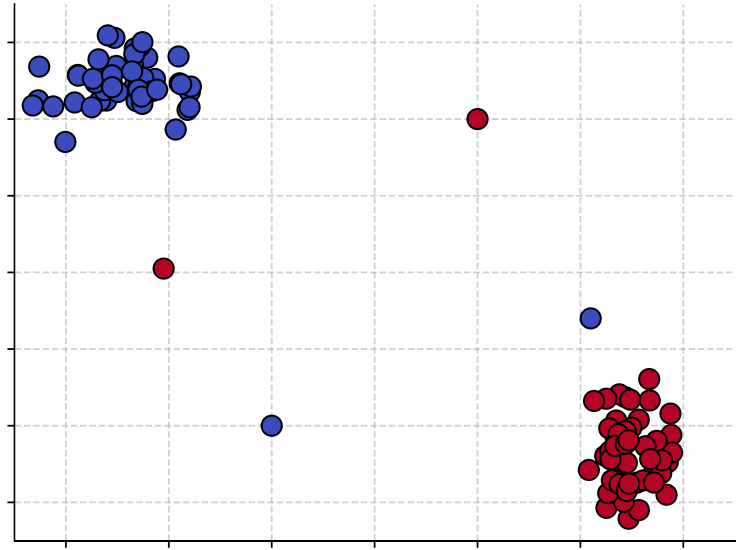
- What happens if we train a *hard-margin* SVM on this dataset?
- It **fails**, because the points are not linearly separable.
- However, it find the largest margin with the **fewest training mistakes**.
  - It there is no limit on the number of iterations, it might never stop!



Soft-margin SVM

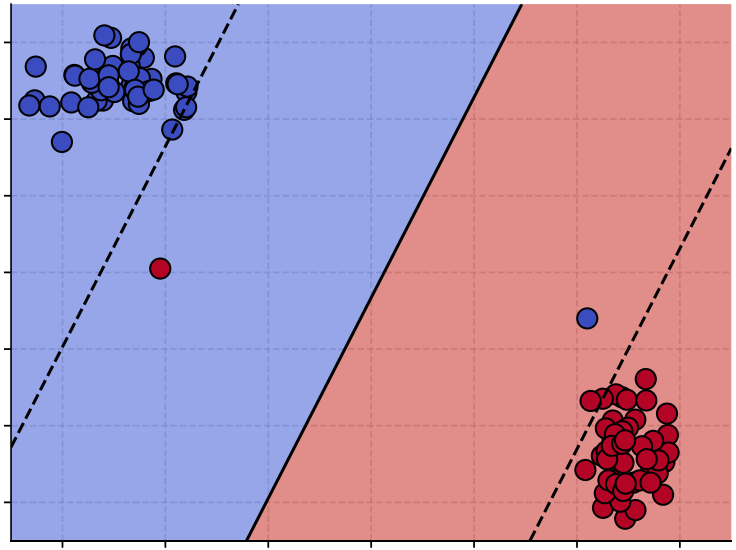
- What would be the more “reasonable” solutions?

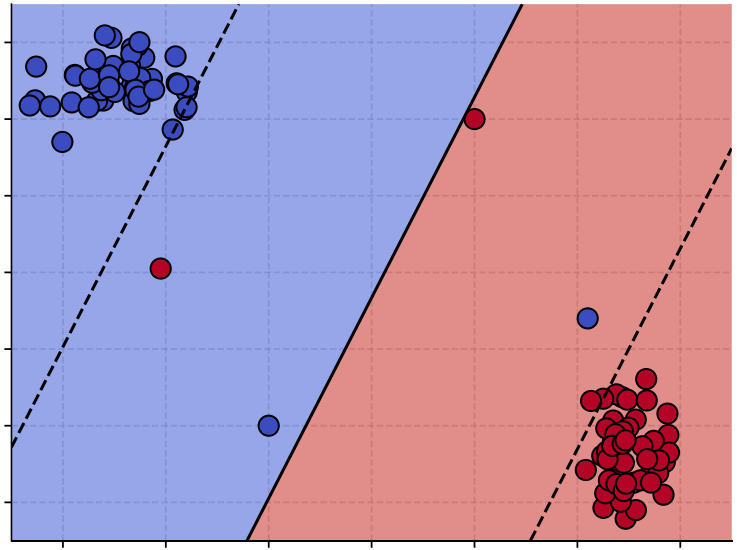




Soft-margin SVM

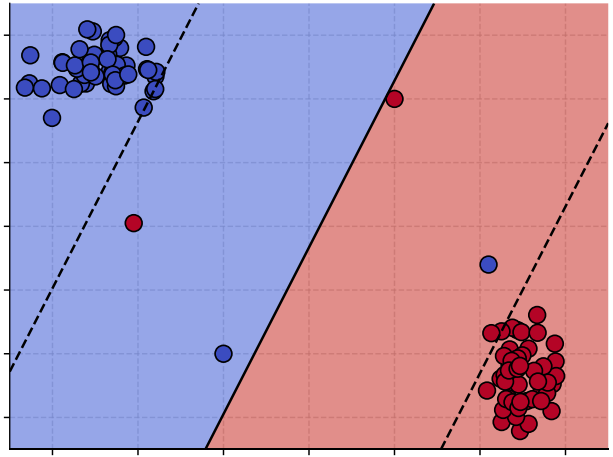
- What would be the more “reasonable” solutions?





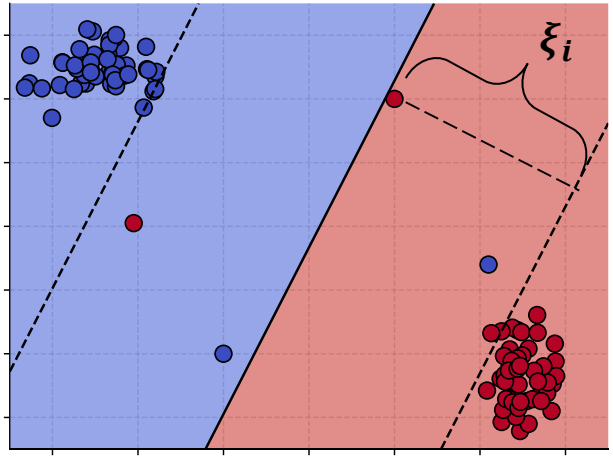
Soft-margin SVM

- *Soft-margin SVM*: Allow some training samples to be misclassified, at a cost.



Soft-margin SVM

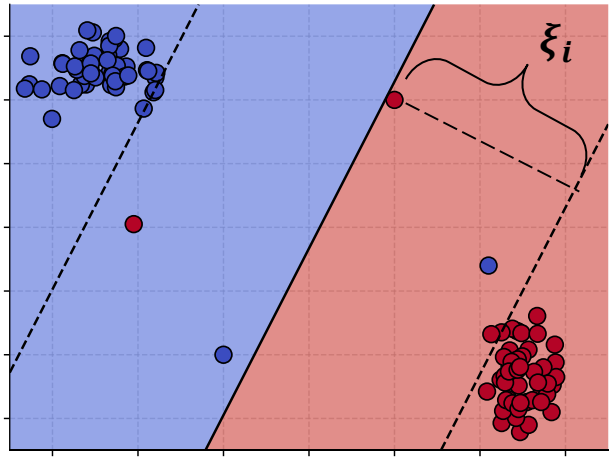
- *Soft-margin SVM*: Allow some training samples to be misclassified, at a cost.
- We introduce "slack variables" $\xi_i \geq 0$ (How much example $\vec{x}^{(i)}$ is allowed to cross the edge).



Soft-margin SVM

- *Soft-margin SVM*: Allow some training samples to be misclassified, at a cost.
- We introduce "slack variables" $\xi_i \geq 0$ (How much example $\vec{x}^{(i)}$ is allowed to cross the edge).

$y^{(i)}\left(\langle \vec{x}^{(i)}, \vec{w} \rangle + b\right) - 1 \geq 0$ becomes $y^{(i)}\left(\langle \vec{x}^{(i)}, \vec{w} \rangle + b\right) - 1 \geq -\xi_i$



Soft-margin SVM

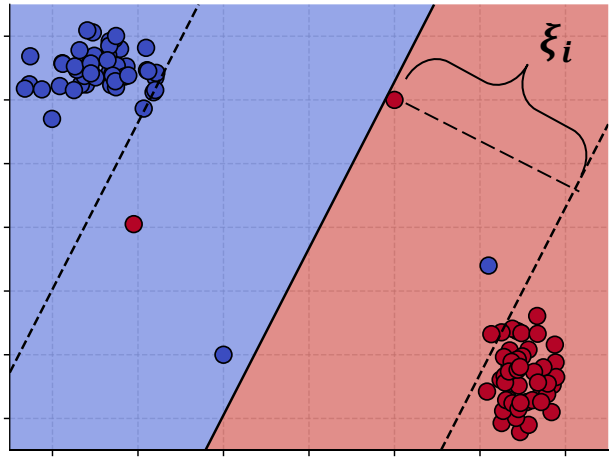
- *Soft-margin SVM*: Allow some training samples to be misclassified, at a cost.
- We introduce "slack variables" $\xi_i \geq 0$ (How much example $\vec{x}^{(i)}$ is allowed to cross the edge).

$y^{(i)}\left(\langle \vec{x}^{(i)}, \vec{w} \rangle + b\right) - 1 \geq 0$ becomes $y^{(i)}\left(\langle \vec{x}^{(i)}, \vec{w} \rangle + b\right) - 1 \geq -\xi_i$

- The optimization problem becomes:

$$\textbf{minimize} \frac{\|\vec{\mathbf{w}}\|^2}{2} + \mathbf{C}\sum_i \boldsymbol{\xi}_\mathbf{i}$$

$$\textbf{subject to } \boldsymbol{y^{(i)}}\left(\langle \vec{\boldsymbol{x}}^{(\boldsymbol{i})}, \vec{\boldsymbol{w}} \rangle + \boldsymbol{b}\right) \geq \mathbf{1} - \boldsymbol{\xi}_\mathbf{i}$$



Soft-margin SVM

- *Soft-margin SVM*: Allow some training samples to be misclassified, at a cost.
- We introduce "slack variables" $\xi_i \geq 0$ (How much example $\vec{x}^{(i)}$ is allowed to cross the edge).

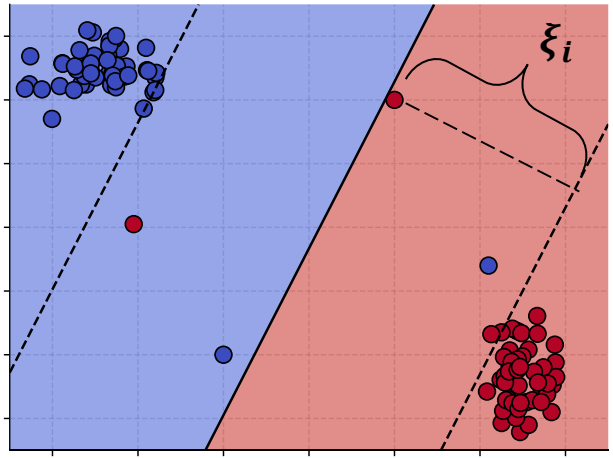
$y^{(i)}\left(\langle \vec{x}^{(i)}, \vec{w} \rangle + b\right) - 1 \geq 0$ becomes $y^{(i)}\left(\langle \vec{x}^{(i)}, \vec{w} \rangle + b\right) - 1 \geq -\xi_i$

- The optimization problem becomes:

$$\textbf{minimize} \frac{\|\vec{\mathbf{w}}\|^2}{2} + \mathbf{C}\sum_i \boldsymbol{\xi}_\mathbf{i}$$

$$\textbf{subject to } \boldsymbol{y^{(i)}\left(\langle \vec{x}^{(i)}, \vec{w} \rangle + b\right) \geq 1 - \xi_i}$$

- Tradeoff between making the margin wide and allowing training mistakes.
- *C* controls the weight of a mistake.



Hinge Loss

$$\text{minimize} \quad \left(\frac{\|\vec{w}\|^2}{2} + C \sum_i \xi_i \right)$$

$$\text{subject to} \quad y^{(i)} \left(\langle \vec{x}^{(i)}, \vec{w} \rangle + b \right) \geq 1 - \xi_i$$

$$\xi_i \geq 0$$

Hinge Loss

$$\left.\begin{array}{ll} \text{minimize} & \left(\frac{\|\vec{w}\|^2}{2} + C\sum_i \xi_i\right) \\ \text{subject to} & y^{(i)}\left(\langle \vec{x}^{(i)}, \vec{w}\rangle + b\right) \geq 1 - \xi_i \\ & \xi_i \geq 0 \end{array}\right\} \Rightarrow \quad \xi_i = \max\left(0, 1 - y^{(i)}\left(\langle \vec{x}^{(i)}, \vec{w}\rangle + b\right)\right)$$

Hinge Loss

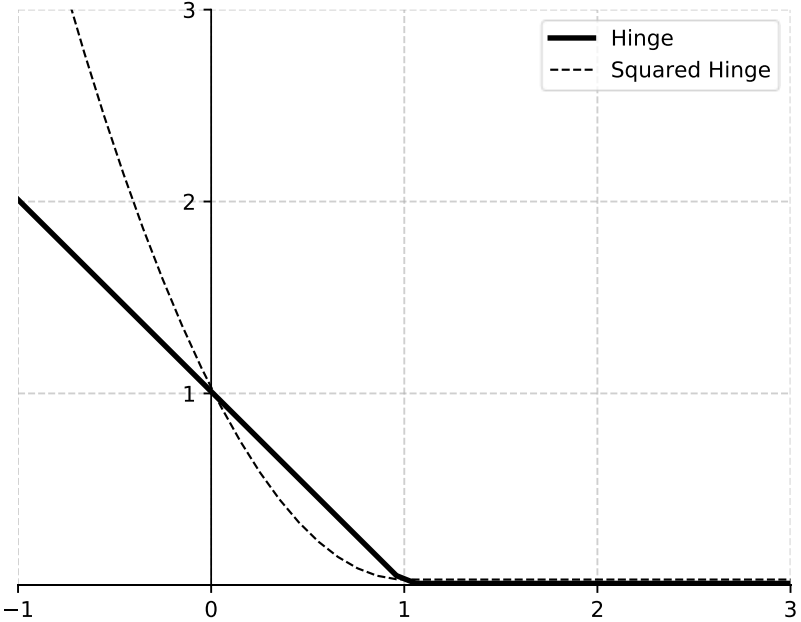
minimize $\left(\frac{\|\vec{w}\|^2}{2} + C\sum_i \max\left(0, 1 - y^{(i)}\left(\langle\vec{x}^{(i)}, \vec{w}\rangle + b\right)\right)\right)$

Hinge Loss

minimize $\left(\frac{\|\vec{w}\|^2}{2} + C\sum_i \max\left(0, 1 - y^{(i)}\left(\langle\vec{x}^{(i)}, \vec{w}\rangle + b\right)\right)\right)$

$$\mathcal{L}(\vec{x}) \stackrel{\text{def}}{=} \max\left(0, 1 - y(\langle\vec{x}, \vec{w}\rangle + b)\right)$$

Hinge Loss



Hinge Loss

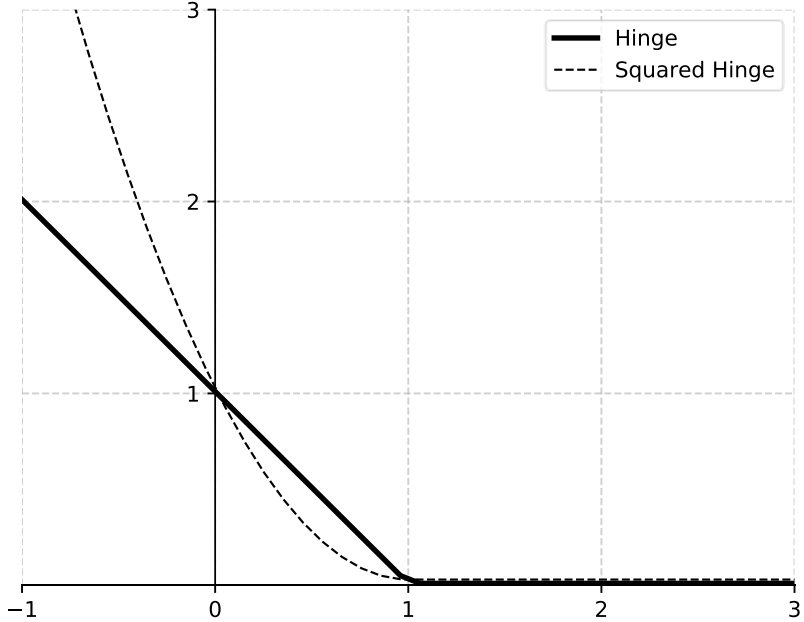
$$\text{minimize} \quad \left(\frac{\|\vec{w}\|^2}{2} + C \sum_i \max\left(0, 1 - y^{(i)}\left(\langle \vec{x}^{(i)}, \vec{w} \rangle + b\right)\right)\right)$$

$$\mathcal{L}(\vec{x}) \stackrel{\text{def}}{=} \max\left(0, 1 - y(\langle \vec{x}, \vec{w} \rangle + b)\right)$$

Hinge Loss

$$\Longrightarrow \text{minimize} \left(C \sum_i \mathcal{L}\left(\vec{x}^{(i)}\right) + \frac{\|\vec{w}\|^2}{2} \right)$$

What the soft-margin SVM is actually doing **is *minimizing the hinge loss*** with ***Tikhonov regularization***.



Soft-margin SVM Primal Form

- Decision rule:

$$\vec{x} \text{ is a “+” sample if } \langle \vec{x}, \vec{w} \rangle + \mathbf{b} \geq \mathbf{0}$$

- In order to obtain $\vec{w}$ and b we need to:

$$\text{minimize} \frac{\|\vec{w}\|^2}{2} + \mathbf{C} \sum_i \boldsymbol{\xi}_\mathbf{i}$$
$$\text{subject to } y^{(i)}\left(\langle \vec{x}^{(i)}, \vec{w} \rangle + b\right) \geq 1 - \xi_i$$

or

$$\text{minimize} \left(\frac{\|\vec{w}\|^2}{2} + C \sum_i \max\left(0, 1 - y^{(i)}\left(\langle \vec{x}^{(i)}, \vec{w} \rangle + b\right)\right)\right)$$

Soft-margin SVM Dual Form

- The decision rule is:

$$\vec{x} \text{ is a "+" sample if } \sum_i \alpha_i y^{(i)} \langle \vec{x}^{(i)}, \vec{x} \rangle + b \geq 0$$

- In order to obtain α_i and b we need to:

$$\text{minimize } \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y^{(i)} y^{(j)} \langle \vec{x}^{(i)}, \vec{x}^{(j)} \rangle$$

$$\text{subject to } 0 \leq \alpha_i \leq C \; \forall i$$

The only difference from the hard-margin is the upper-bound on the Lagrange multipliers.

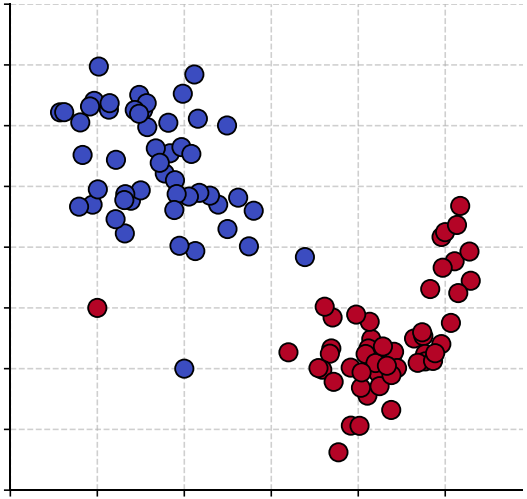
All SVM Forms

		Decision Rule	Optimization Problem
Hard-margin	Primal Form	$\langle \vec{x}, \vec{w} \rangle + b \geq 0$	$\text{minimize } \frac{\|\vec{w}\|^2}{2}$ $\text{subject to } y^{(i)}\left(\langle \vec{x}^{(i)}, \vec{w} \rangle + b\right) - 1 \geq 0$
	Dual Form	$\sum_i \alpha_i y^{(i)} \langle \vec{x}^{(i)}, \vec{x} \rangle + b \geq 0$	$\sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y^{(i)} y^{(j)} K\left(\vec{x}^{(i)}, \vec{x}^{(j)}\right)$ $\text{subject to } \alpha_i \geq 0$
Soft-margin	Primal Form	$\langle \vec{x}, \vec{w} \rangle + b \geq 0$	$\text{minimize} \frac{\|\vec{w}\|^2}{2} + C \sum_i \xi_i$ $\text{subject to } y^{(i)}\left(\langle \vec{x}^{(i)}, \vec{w} \rangle + b\right) \geq 1 - \xi_i$
	Dual Form	$\sum_i \alpha_i y^{(i)} \langle \vec{x}^{(i)}, \vec{x} \rangle + b \geq 0$	$\sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y^{(i)} y^{(j)} K\left(\vec{x}^{(i)}, \vec{x}^{(j)}\right)$ $\text{subject to } 0 \leq \alpha_i \leq C$

Predicting Probabilities with SVMs

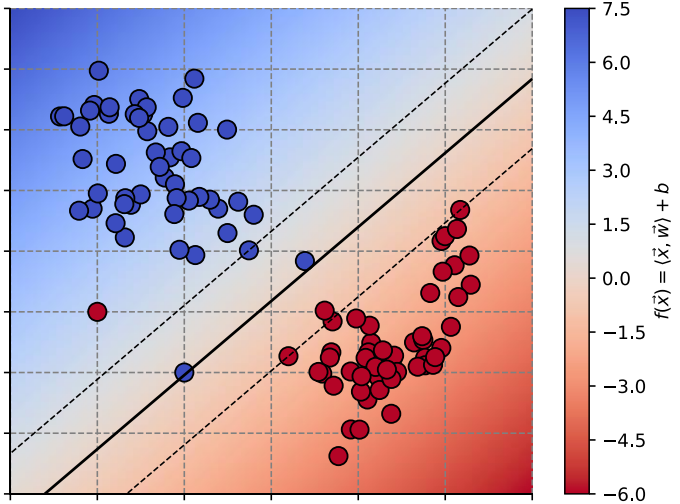
Predicting probabilities with SVMs

- Having a model which can predict class probabilities is sometimes very useful.



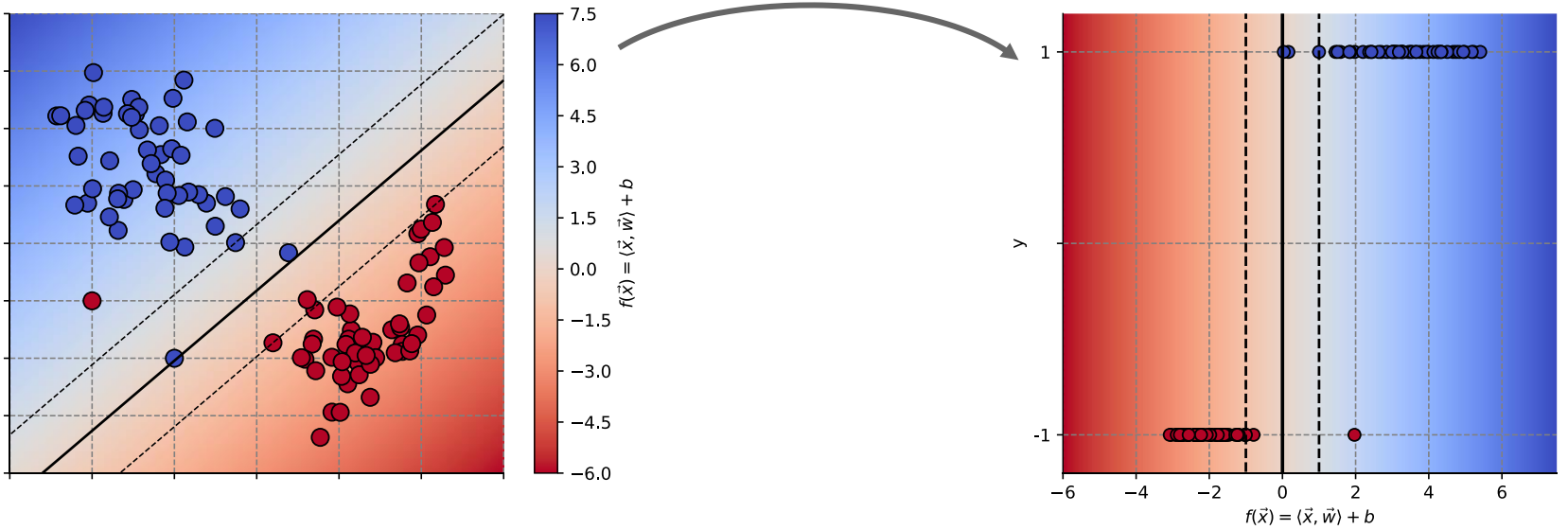
Predicting probabilities with SVMs

- Having a model which can predict class probabilities is sometimes very useful.
- The SVM's decision function $f(\vec{x}) = \langle \vec{x}, \vec{w} \rangle + b$ provides an unbounded value that is not a probability.



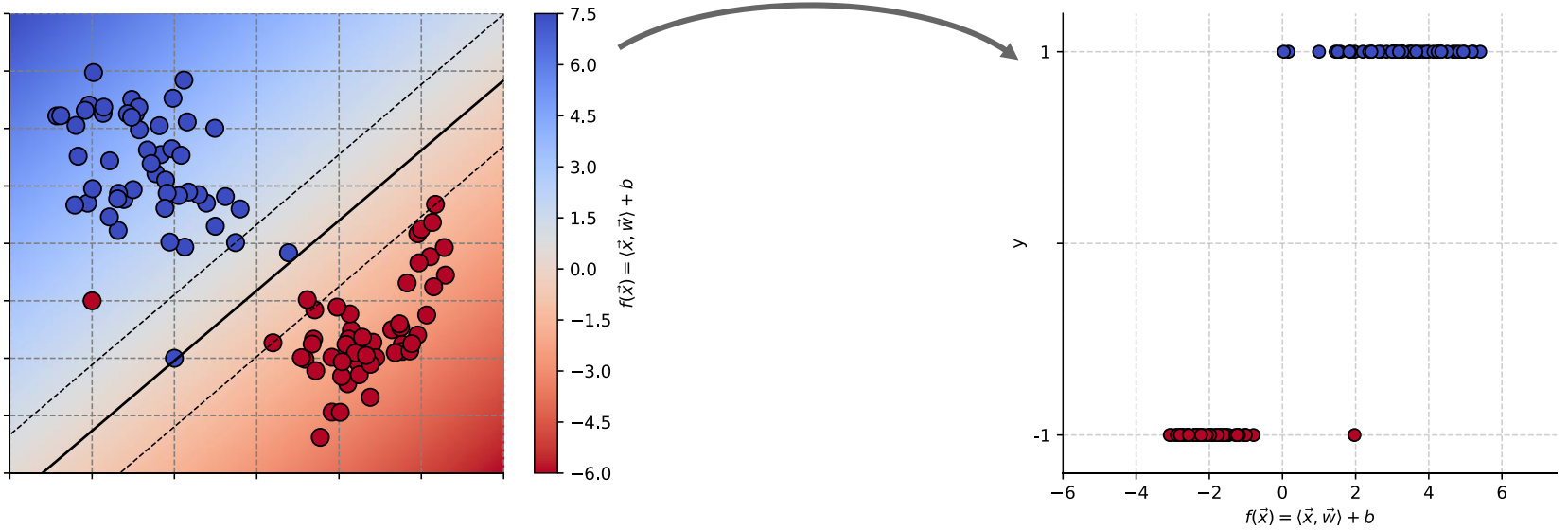
Predicting probabilities with SVMs

- Having a model which can predict class probabilities is sometimes very useful.
- The SVM's decision function $f(\vec{x}) = \langle \vec{x}, \vec{w} \rangle + b$ provides an unbounded value that is not a probability.



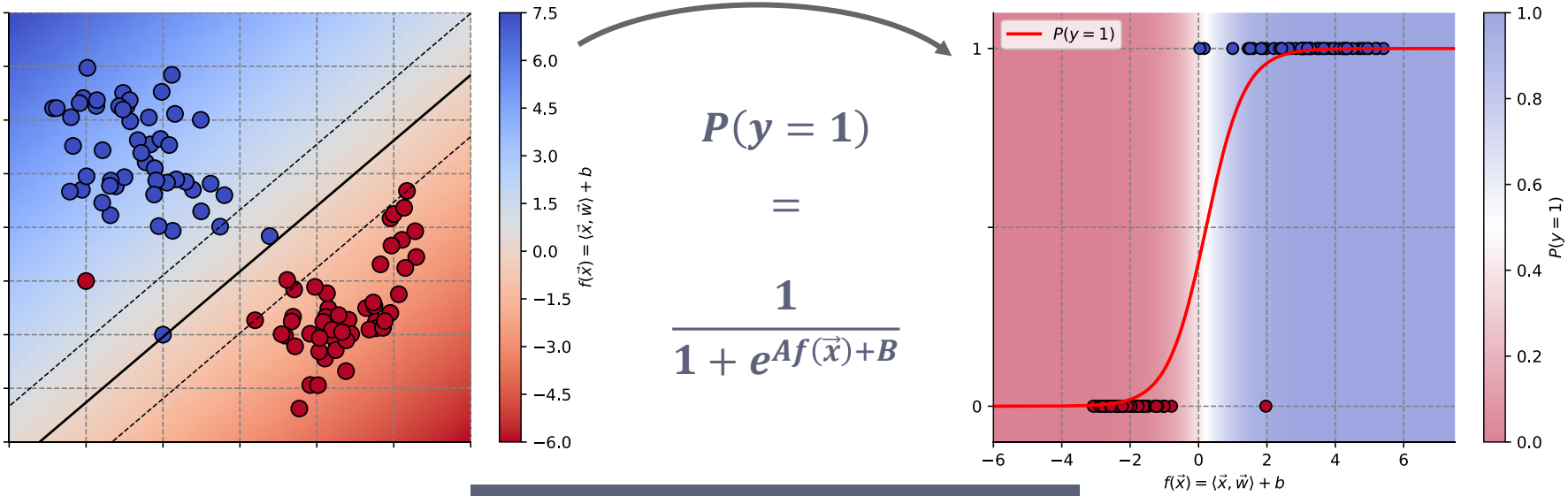
Predicting probabilities with SVMs

- Having a model which can predict class probabilities is sometimes very useful.
- The SVM's decision function $f(\vec{x}) = \langle \vec{x}, \vec{w} \rangle + b$ provides an unbounded value that is not a probability.



Predicting probabilities with SVMs

- Having a model which can predict class probabilities is sometimes very useful.
- The SVM's decision function $f(\vec{x}) = \langle \vec{x}, \vec{w} \rangle + b$ provides an unbounded value that is not a probability.

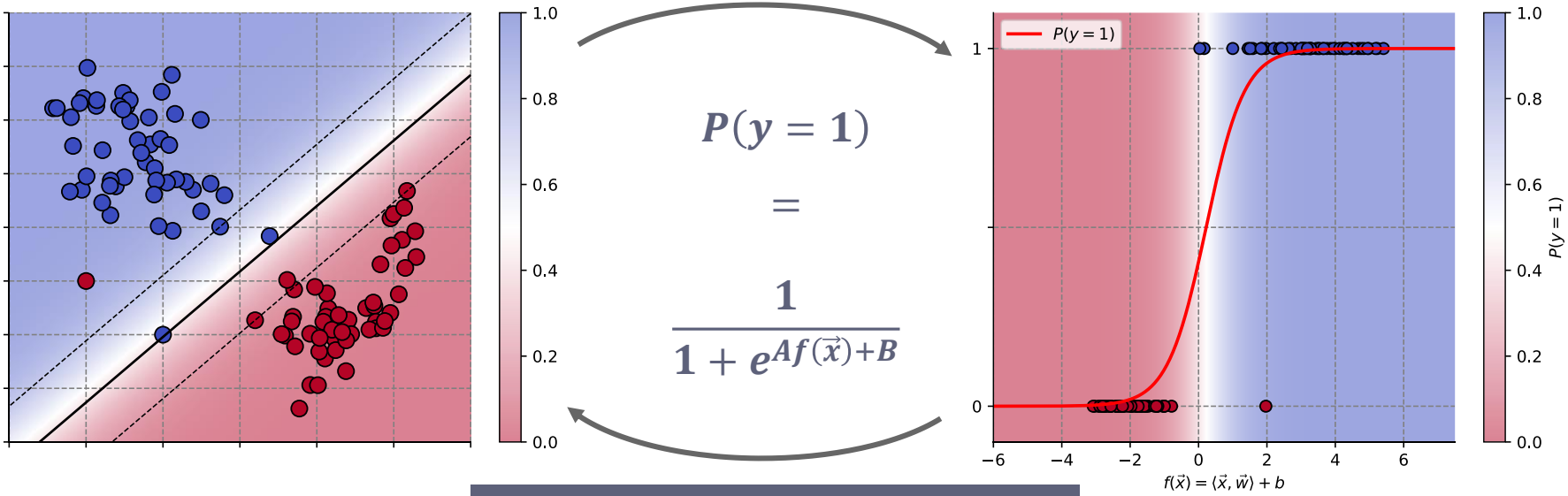


Simple Logistic Regression on SVM outputs:

- $f(\vec{x}) \in \mathbb{R}$ is the independent variable
- $A, B \in \mathbb{R}$ are the model parameters

Predicting probabilities with SVMs

- Having a model which can predict class probabilities is sometimes very useful.
- The SVM's decision function $f(\vec{x}) = \langle \vec{x}, \vec{w} \rangle + b$ provides an unbounded value that is not a probability.

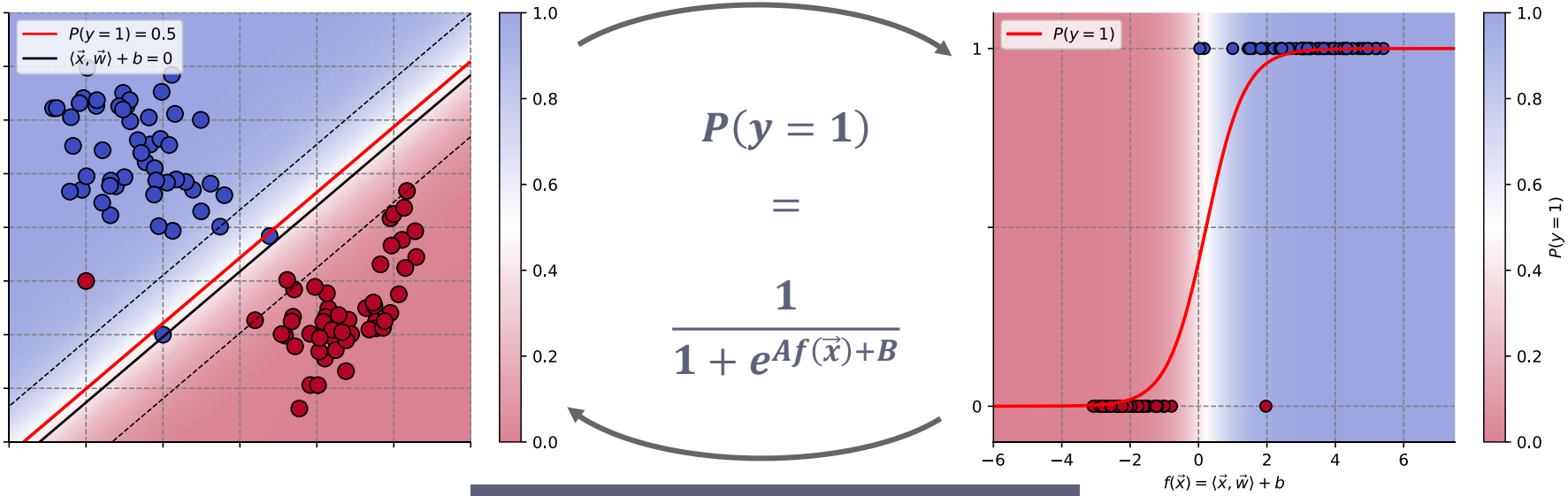


Simple Logistic Regression on SVM outputs:

- $f(\vec{x}) \in \mathbb{R}$ is the independent variable
- $A, B \in \mathbb{R}$ are the model parameters

Predicting probabilities with SVMs

- Having a model which can predict class probabilities is sometimes very useful.
- The SVM's decision function $f(\vec{x}) = \langle \vec{x}, \vec{w} \rangle + b$ provides an unbounded value that is not a probability.

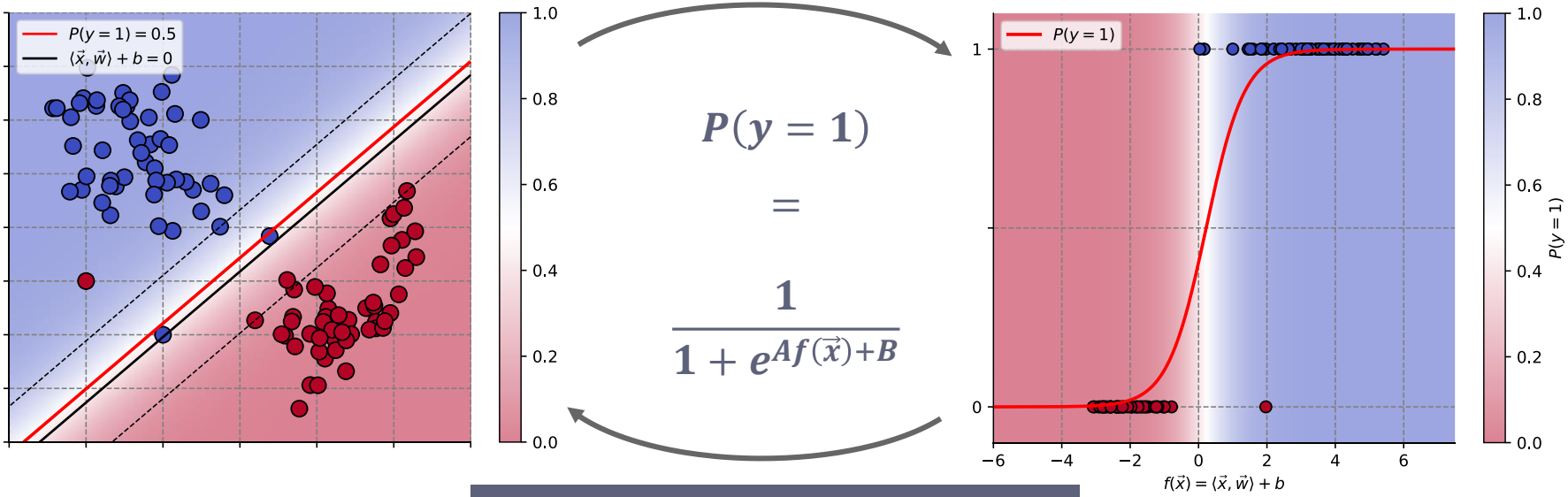


Simple Logistic Regression on SVM outputs:

- $f(\vec{x}) \in \mathbb{R}$ is the independent variable
- $A, B \in \mathbb{R}$ are the model parameters

Predicting probabilities with SVMs

- Having a model which can predict class probabilities is sometimes very useful.
- The SVM's decision function $f(\vec{x}) = \langle \vec{x}, \vec{w} \rangle + b$ provides an unbounded value that is not a probability.



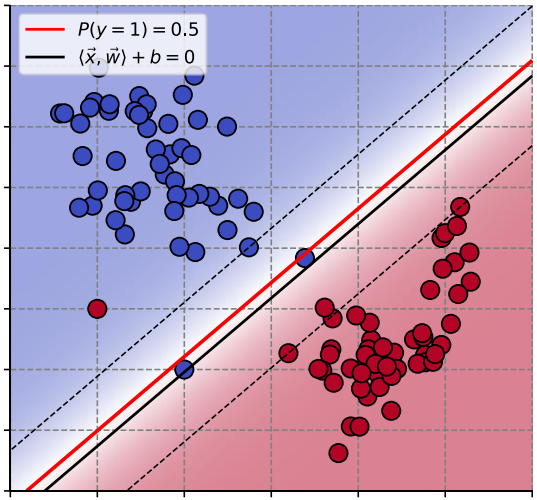
SVM prediction might not always agree with the probability estimate.

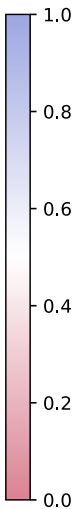
Simple Logistic Regression on SVM outputs:

- $f(\vec{x}) \in \mathbb{R}$ is the independent variable
- $A, B \in \mathbb{R}$ are the model parameters

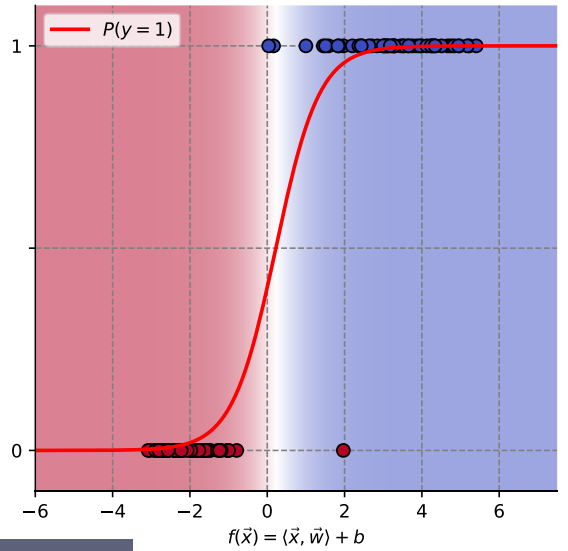
Predicting probabilities with SVMs

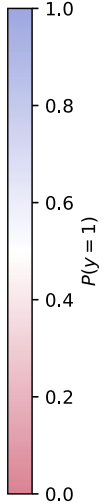
- Having a model which can predict class probabilities is sometimes very useful.
- The SVM's decision function $f(\vec{x}) = \langle \vec{x}, \vec{w} \rangle + b$ provides an unbounded value that is not a probability.





$$P(y=1) = \frac{1}{1 + e^{Af(\vec{x})+B}}$$





SVM prediction might not always agree with the probability estimate.

Simple Logistic Regression on SVM outputs:
- $f(\vec{x}) \in \mathbb{R}$ is the independent variable
- $A, B \in \mathbb{R}$ are the model parameters

Same idea also works for SVM with kernel.

Predicting probabilities with SVMs

- **Platt scaling** is a way of transforming the outputs of a classification model into a probability distribution over classes.
- It works by fitting a logistic model over the uncalibrated classifier scores $f(\vec{x})$.

$$P(y = 1) = \frac{1}{1 + e^{Af(\vec{x})+B}}$$

- To avoid *overfitting*, some additional measures are taken, including fitting the logistic model on a *validation set*, which is not used to train the original non-probabilistic model.

It is mostly used for SVMs, but it can be used for other algorithms as well.

Multi-class SVMs

Multi-class SVM

- SVMs are *binary classifiers* by design. Can we use them for *multi-class* problems?

Multi-class SVM

- SVMs are *binary classifiers* by design. Can we use them for *multi-class* problems?
- **One-versus-rest** (OVR), sometimes called (one-versus-all, OVA)
  - Train n classifiers, one for each class, where the negative examples are all the other classes
  - At inference time, run all classifiers and pick the class with the highest margin (most confident)
- **One-versus-one** (OVO)
  - Train $n(n-1)/2$ classifiers, one for each pair of classes
  - At inference time, run all classifiers and pick the class which was selected by most of them

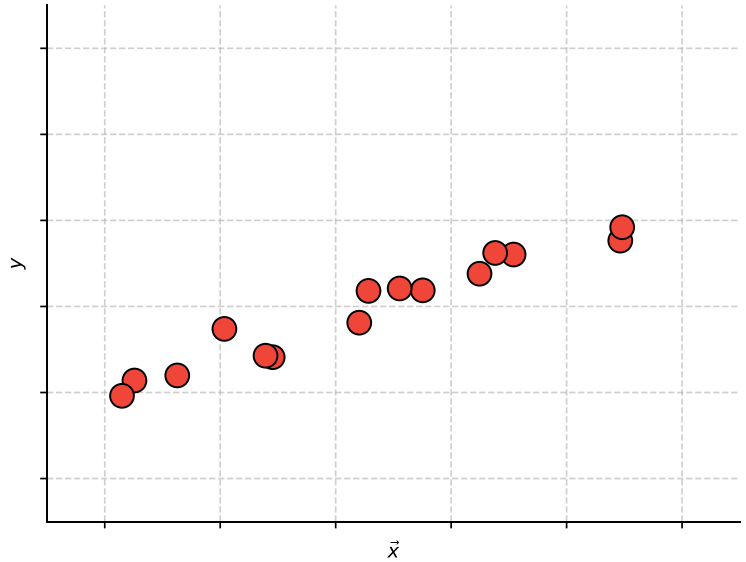
Multi-class SVM

- SVMs are *binary classifiers* by design. Can we use them for *multi-class* problems?
- **One-versus-rest** (OVR), sometimes called (one-versus-all, OVA)
  - Train n classifiers, one for each class, where the negative examples are all the other classes
  - At inference time, run all classifiers and pick the class with the highest margin (most confident)
- **One-versus-one** (OVO)
  - Train $n(n-1)/2$ classifiers, one for each pair of classes
  - At inference time, run all classifiers and pick the class which was selected by most of them
- There are also (less common) methods which optimize a joint loss function over all classes.
  - e.g. **"On the Algorithmic Implementation of Multiclass Kernel-based Vector Machines"**
    - The *"Crammer-Singer Loss"* Koby Crammer, Yoram Singer, 2001

SVM Regression

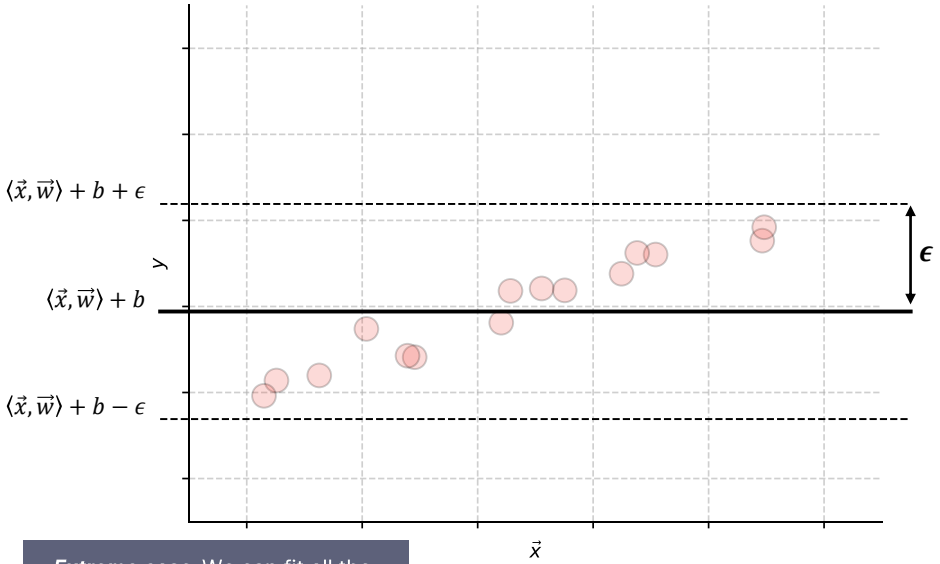
SVM Regression

- Inference – Use the SVM's decision function as prediction $\hat{y} = \langle \vec{x}, \vec{w} \rangle + b$
- Training – Find the *flattest* function which can fit the training data inside a tube of radius ϵ



SVM Regression

- Inference – Use the SVM's decision function as prediction $\hat{y} = \langle \vec{x}, \vec{w} \rangle + b$
- Training – Find the *flattest* function which can fit the training data inside a tube of radius ϵ



Extreme case: We can fit all the data points with a function that does not change at all ($\|\vec{w}\| = 0$)

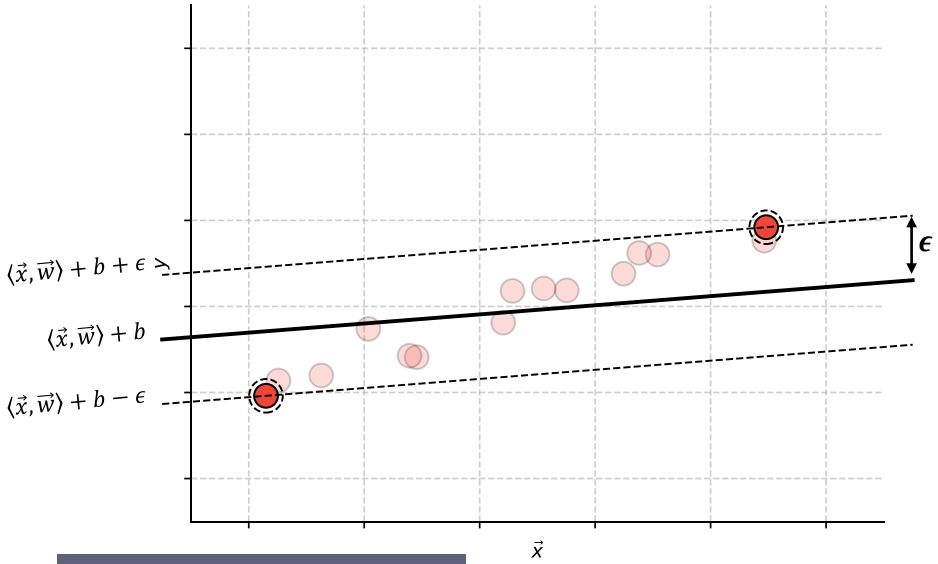
- *"Flat" function* means it does not change much with parameters (i.e. small $\|\vec{w}\|$):
- In other words, we find the *smallest* $\|\vec{w}\|$ for which the prediction error for all training points is *at most* $\pm\epsilon$:

$$\textbf{minimize} \quad \frac{\|\vec{w}\|^2}{2}$$

$$\textbf{subject to} \quad \left| y^{(i)} - \langle \vec{x}^{(i)}, \vec{w} \rangle - b \right| \leq \epsilon$$

SVM Regression

- Inference – Use the SVM's decision function as prediction $\hat{y} = \langle \vec{x}, \vec{w} \rangle + b$
- Training – Find the *flattest* function which can fit the training data inside a tube of radius ϵ



When ϵ is smaller, $\vec{w}$ and b need to be adjusted for all the points to fit.

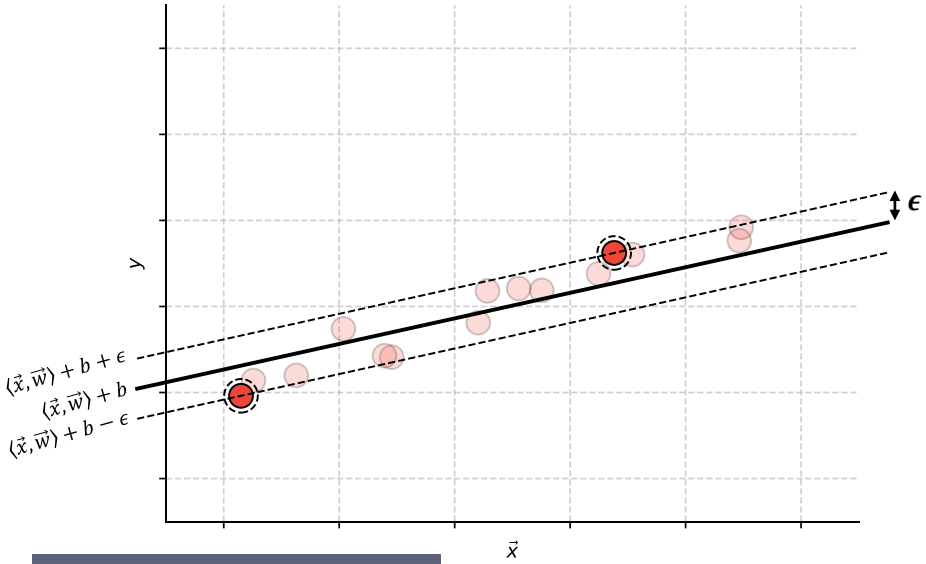
- *"Flat" function* means it does not change much with parameters (i.e. small $\|\vec{w}\|$):
- In other words, we find the *smallest* $\|\vec{w}\|$ for which the prediction error for all training points is *at most* $\pm\epsilon$:

$$\textbf{minimize} \quad \frac{\|\vec{w}\|^2}{2}$$

$$\textbf{subject to} \quad \left| y^{(i)} - \langle \vec{x}^{(i)}, \vec{w} \rangle - b \right| \leq \epsilon$$

SVM Regression

- Inference – Use the SVM's decision function as prediction $\hat{y} = \langle \vec{x}, \vec{w} \rangle + b$
- Training – Find the *flattest* function which can fit the training data inside a tube of radius ϵ



When ϵ is smaller, $\vec{w}$ and b need to be adjusted for all the points to fit.

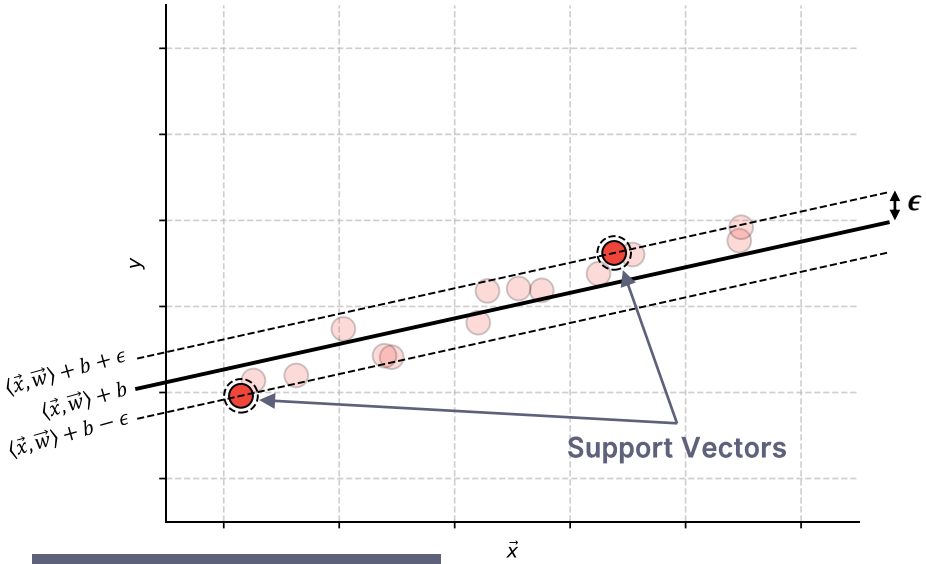
- *"Flat" function* means it does not change much with parameters (i.e. small $\|\vec{w}\|$):
- In other words, we find the *smallest* $\|\vec{w}\|$ for which the prediction error for all training points is *at most* $\pm\epsilon$:

$$\textbf{minimize} \quad \frac{\|\vec{w}\|^2}{2}$$

$$\textbf{subject to} \quad \left| y^{(i)} - \langle \vec{x}^{(i)}, \vec{w} \rangle - b \right| \leq \epsilon$$

SVM Regression

- Inference – Use the SVM's decision function as prediction $\hat{y} = \langle \vec{x}, \vec{w} \rangle + b$
- Training – Find the *flattest* function which can fit the training data inside a tube of radius ϵ



When ϵ is smaller, $\vec{w}$ and b need to be adjusted for all the points to fit.

- *"Flat" function* means it does not change much with parameters (i.e. small $\|\vec{w}\|$):
- In other words, we find the *smallest* $\|\vec{w}\|$ for which the prediction error for all training points is *at most* $\pm\epsilon$:

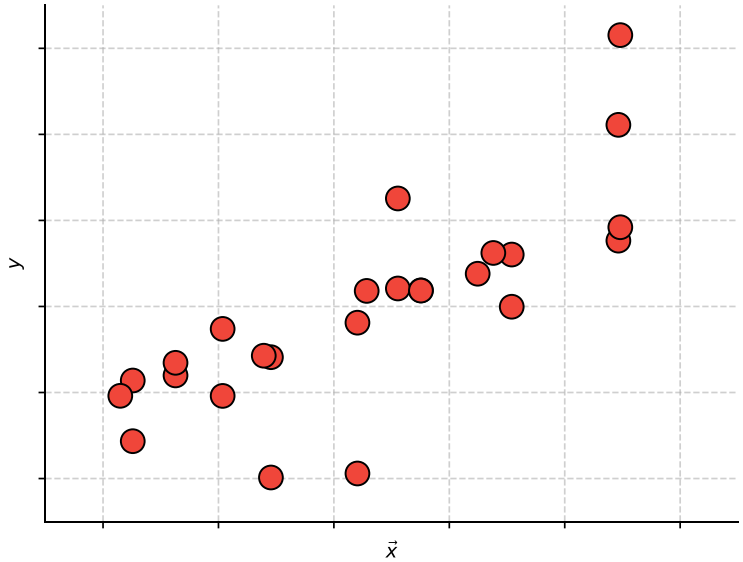
$$\textbf{minimize} \quad \frac{\|\vec{w}\|^2}{2}$$

$$\textbf{subject to} \left| y^{(i)} - \langle \vec{x}^{(i)}, \vec{w} \rangle - b \right| \leq \epsilon$$

- The model can be expressed only in terms of *dot product* with some training points (*"support vectors"*) $\Longrightarrow$ we can use *kernel functions*.

SVM Regression

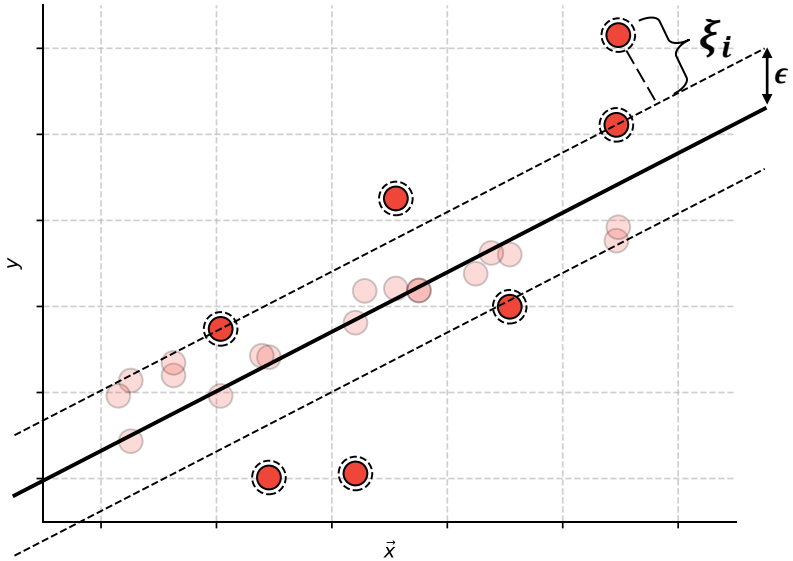
- Inference – Use the SVM's decision function as prediction $\hat{y} = \langle \vec{x}, \vec{w} \rangle + b$
- Training – Find the *flattest* function which can fit the training data inside a tube of radius ϵ



- Sometimes it is impossible to find a function to find all the training inside a $\pm\epsilon$ range.

SVM Regression

- Inference – Use the SVM's decision function as prediction $\hat{y} = \langle \vec{x}, \vec{w} \rangle + b$
- Training – Find the *flattest* function which can fit the training data inside a tube of radius ϵ



- Sometimes it is impossible to find a function to find all the training inside a $\pm\epsilon$ range.
- We introduce "slack variables":

$$\textbf{minimize} \quad \frac{\|\vec{w}\|^2}{2} + C\sum_i \xi_i$$
$$\textbf{subject to } \left|y^{(i)} - \langle \vec{x}^{(i)}, \vec{w} \rangle - b\right| \leq \epsilon + \xi_i$$

or

$$\mathcal{L}(\vec{x}) \stackrel{\text{def}}{=} \max(0, |y - \langle \vec{x}, \vec{w} \rangle - b| - \epsilon)$$

ϵ-insensitive Loss

$$\Rightarrow \textbf{minimize} \left(C\sum_i \mathcal{L}(\vec{x}^{(i)}) + \frac{\|\vec{w}\|^2}{2} \right)$$

SVMs in Python

```
from sklearn.svm import SVC, SVR

clf = SVC(C = 100, kernel = 'linear', probability = True) # soft-margin SVM classifier
clf.fit(X, y)
clf.predict([x]) # predicted class for x
clf.decision_function([x]) # actual value of the decision function at point x
clf.predict_proba([x]) # predicted probability for x (only works if probability was True at init)

clf.n_support_ # number of support vectors per class
clf.support_ # indices of support vectors
clf.support_vectors_ # support vectors (selected rows from X: X[clf.support_])
clf.dual_coef_ # lagrange multipliers multipled by labels (α_i · y_i) Obs: y_i = ±1, not your labels
clf.intercept_ # b
clf.coef_ # w⃗ = clf.dual_coef.dot(clf.support_vectors_), only available for linear kernel

reg = SVR(C = 10, epsilon = 0.2, kernel = 'rbf') # svm regression
```

SVMs in Python

```
from sklearn.svm import LinearSVC, LinearSVR
# LinearSVC implements the svm classifier with linear kernel, but it is more flexible than SVC.
    # It can optimize either the primal or dual
    # It can use L_2 or L_1 for ||w|| minimization
    # It can minimize the hinge or squared-hinge losses (squared-hinge is the default)
    # For multi-class problems, it can also use the crammer-singer loss

clf = LinearSVC(C = 100, dual = False, penalty = 'l2', loss = 'hinge')
clf.fit(X, y)
clf.predict([x]) # predicted class for x
clf.decision_function([x]) # actual value of the decision function at point x
```

History of SVMs

- **“Pattern recognition using generalized portrait method”**
  V. Vapnik and A. Lerner, 1963
  - The idea of “*large margin*” classifiers

- **“A training algorithm for optimal margin classifiers”**
  B. Boser, I. Guyon and V. Vapnik, 1992
  - Large margin *”non-linear”* classifiers with the use of the *“kernel trick”*

- **“Support vector networks”**
  C. Cortes and V. Vapnik, 1995
  - Classifying *“non-separable”* data

Vladimir Vapnik

Conclusions

- **SVMs** separate classes using a **“maximum-margin” hyperplane**.
  - In other words, it tries to make the *decision boundary* as far away as possible from all training points.
  - The model only depends on a few selected training points, called **“support vectors”.**
- A **kernel function** maps data points into a *higher-dimensional feature space*, where they are easier to separate.
  - The mapping is implicit, since the kernel function computes the dot product directly in the high-dimensional space.
  - Both the optimization and the decision can be expressed only in terms of *dot product* with the training points
- The **soft-margin SVM** permits some training mistakes as part of the optimization problem in order to obtain better generalization.
  - It can be reformulated in terms of minimizing the **Hinge-loss**
- We can use Platt scaling (*logistic regression* on SVM outputs) to obtain **class probability** estimates.
- By minimizing the **epsilon-insensitive loss**, SVMs can be used for *regression* problems.
- To use SVMs for *multi-class problems*, the most common method is to combine multiple binary SVMs.

Keywords

Support Vector Machine | SVM

Linear Separability | Hyperplane | Maximum-margin

Linear Classifier | (Non-)Probabilistic Classification

Primal Form | Dual Form | Lagrange Multipliers | Kernel Method

Hard-margin | Soft-margin | Hinge Loss

Platt Scaling | Epsilon-insensitive Loss