

# Machine Learning (CE 40717)

## Fall 2025

Ali Sharifi-Zarchi

CE Department  
Sharif University of Technology

November 23, 2025



## 1 Introduction and Motivation

## 2 Hard-Margin SVM

## 3 Soft-Margin SVM

## 4 Kernel SVM

## 1 Introduction and Motivation

## 2 Hard-Margin SVM

### 3 Soft-Margin SVM

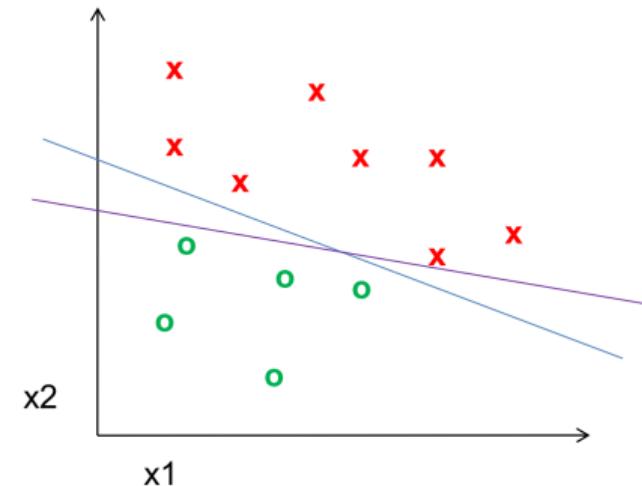
## 4 Kernel SVM

## Support Vector Machines (SVMs)

- Developed in the 1990s by Vapnik and colleagues at Bell Labs, grounded in statistical learning theory.
  - Remained one of the most widely used learning techniques until the resurgence of deep learning.
  - Key aspects that make SVMs particularly interesting:
    - **Strong generalization capabilities**, including margin maximization and structural risk minimization.
    - Extension to non-linear classifiers through the use of **kernel functions**.
    - A **dual formulation** that reveals how linear classifiers can be interpreted as a weighted combination of training examples.
    - Optimization via **stochastic gradient descent (SGD)**, which is also widely used in training neural networks.

## What is the Best Linear Classifier?

- **Logistic Regression**
    - Maximizes the expected likelihood of the label given the data.
    - Every training example contributes to the overall loss.
  - **Support Vector Machine (SVM)**
    - Ensures that all examples are classified with at least a minimal level of confidence.
    - Bases its decision on a minimal subset of examples, known as **support vectors**.



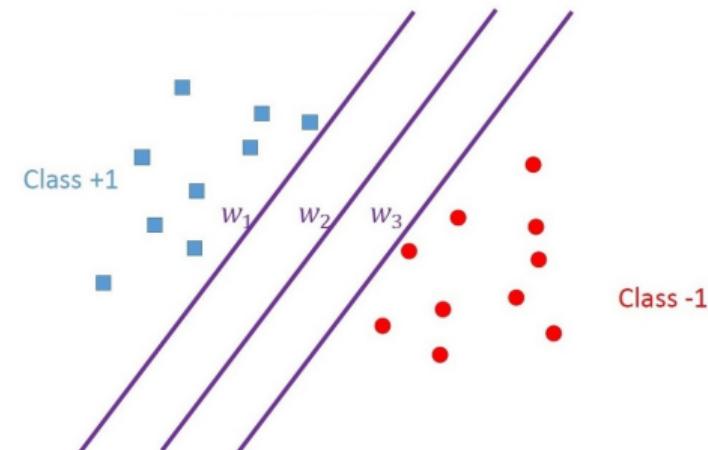
## Multiple Optimal Solutions?

- Given training data  $\{(\mathbf{x}_i, y_i) : 1 \leq i \leq n\}$  drawn i.i.d. from distribution  $D$ .
  - Hypothesis:

$$y = \text{sign}(f_{\mathbf{w}}(\mathbf{x})) = \text{sign}(\mathbf{w}^\top \mathbf{x})$$

- $y = +1$  if  $\mathbf{w}^\top \mathbf{x} > 0$
  - $y = -1$  if  $\mathbf{w}^\top \mathbf{x} < 0$

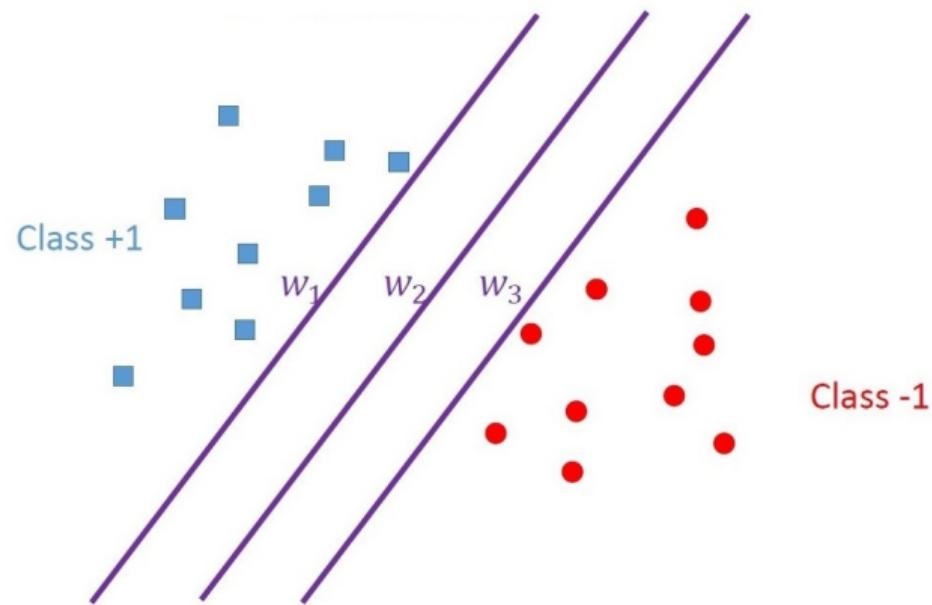
- Let's assume that we can optimize to find  $\mathbf{w}$ .



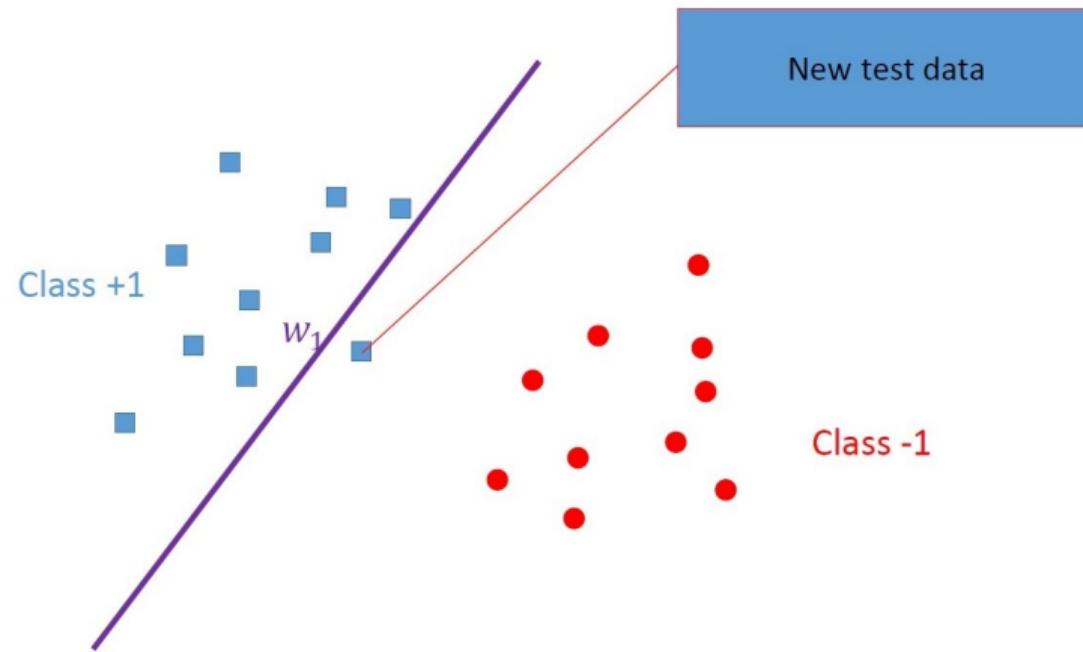
Same empirical loss,  
different test/expected loss.

## What Is a Better Linear Separation?

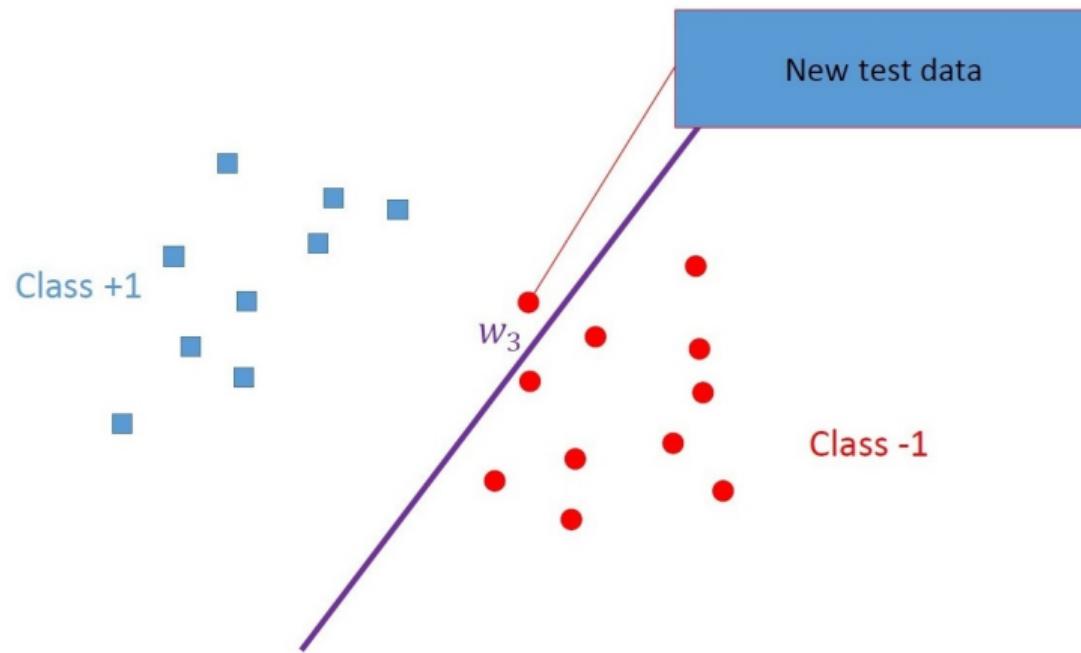
# Which Separator Would You Choose?



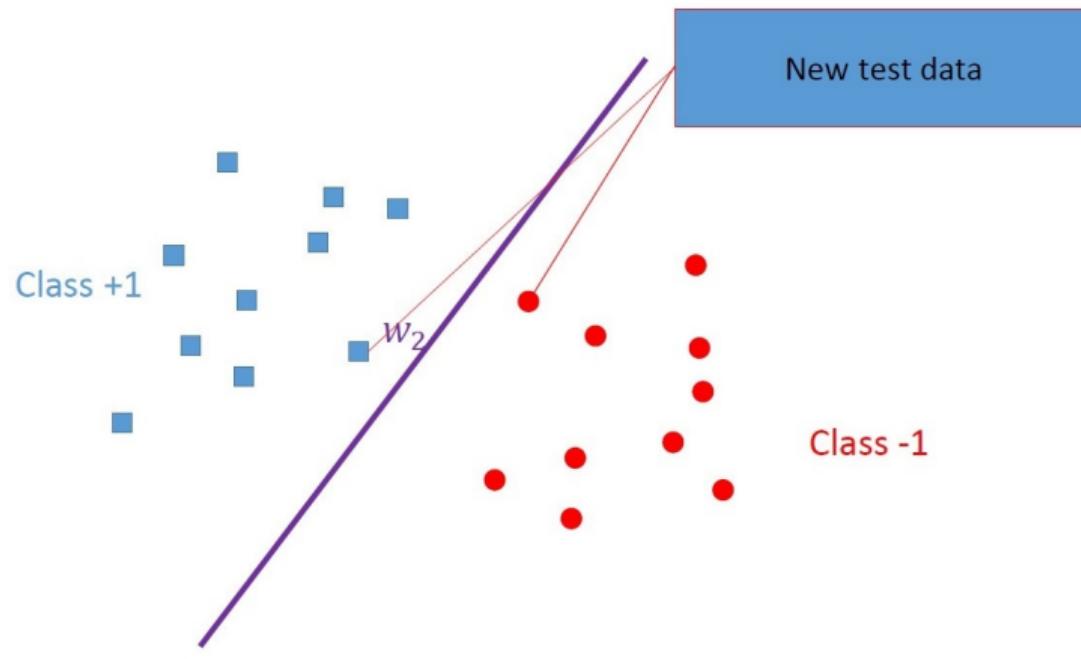
## What Is a Better Linear Separation?



# What Is a Better Linear Separation?

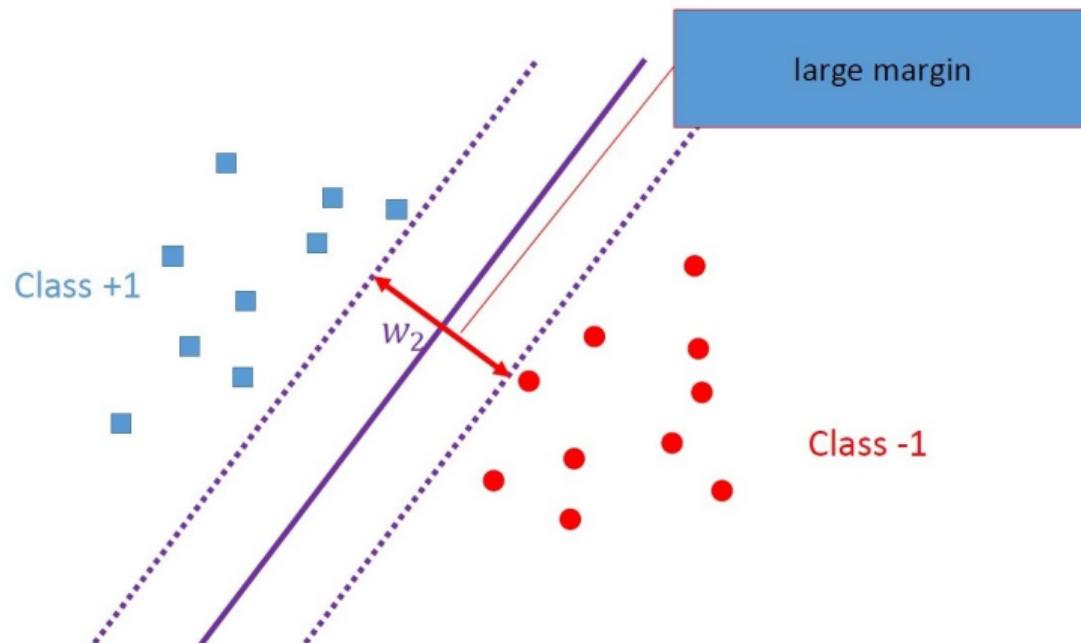


# What Is a Better Linear Separation?

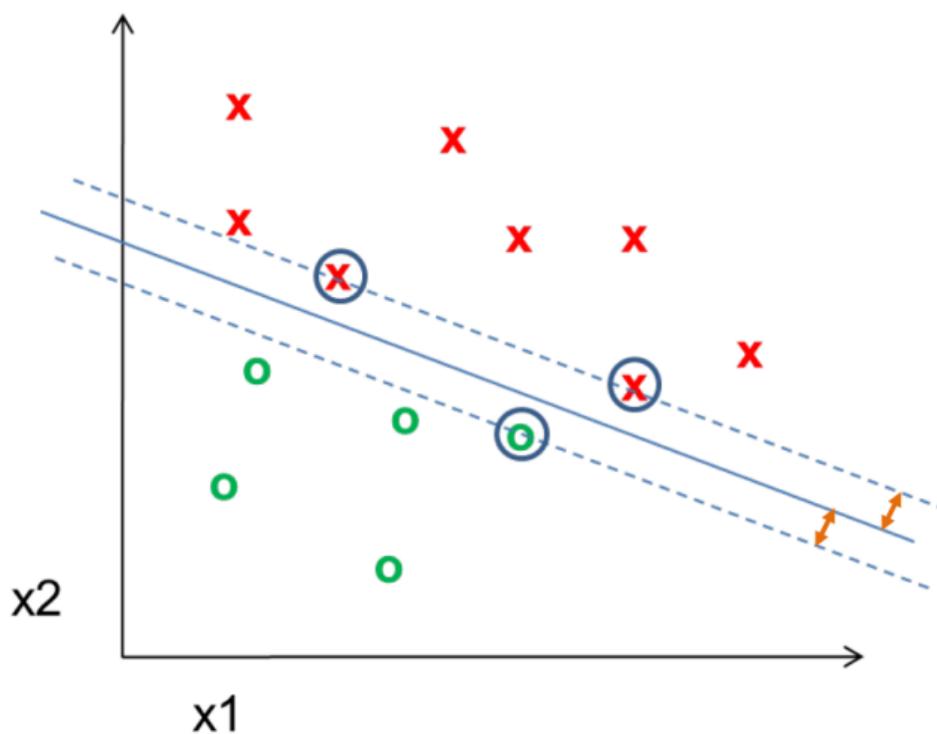


# What Is a Better Linear Separation?

Larger margins lead to better generalization on unseen data.



# SVM Terminology

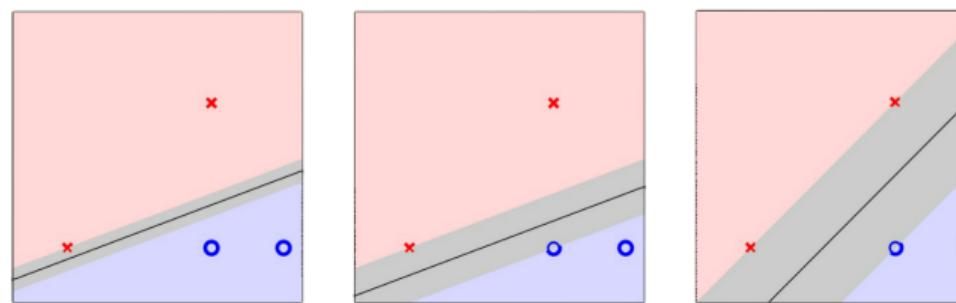
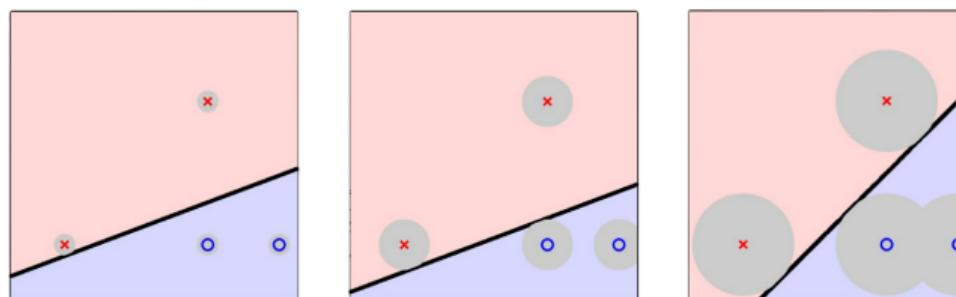


**Margin:** the smallest distance between any training observation and the hyperplane (orange double-headed arrow).

**Support Vector:** a training observation whose distance to the hyperplane is equal to the margin (circled).

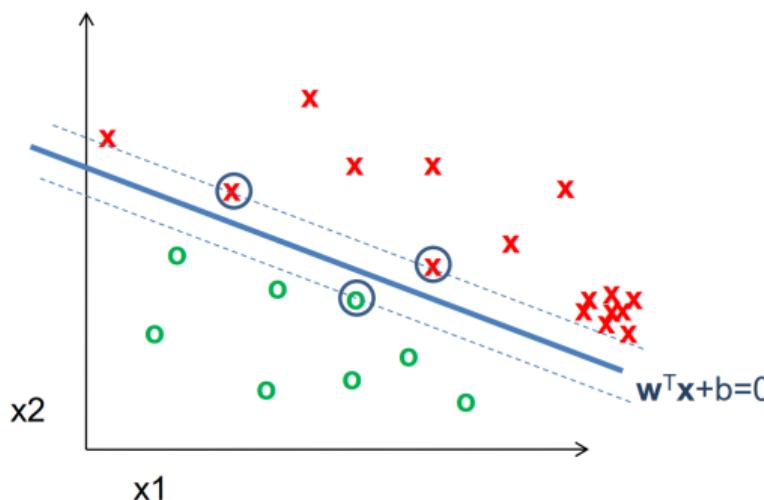
# Fat Hyperplane

We refer to such hyperplanes as **fat hyperplanes**.



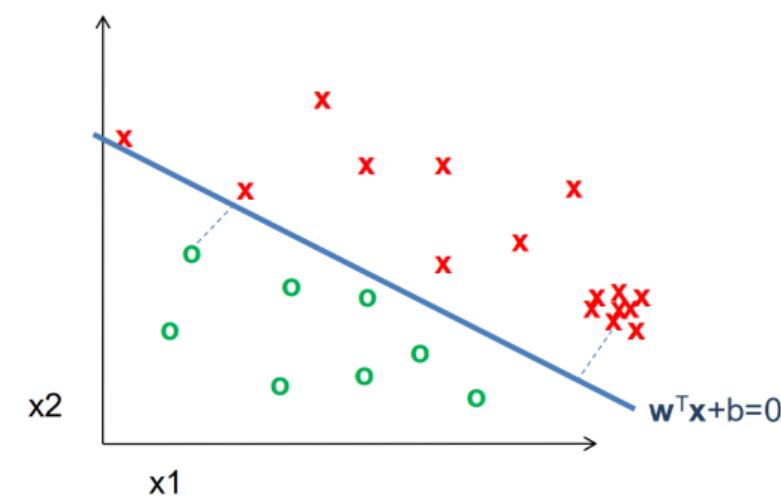
SVMs Minimize  $\mathbf{w}^\top \mathbf{w}$  While Preserving a Margin of 1

## Optimized SVM



The decision boundary depends only on the support vectors (circled).

## Optimized Linear Logistic Regression



It minimizes the sum of the logistic error across all samples, so the boundary tends to lie farther from dense regions.

## 1 Introduction and Motivation

## 2 Hard-Margin SVM

## 3 Soft-Margin SVM

## 4 Kernel SVM

## Why Is It Called a Support Vector?

- **Support:** the maximal-margin hyperplane depends only on these specific training observations.
- **Vector:** each point represents a vector in the  $p$ -dimensional feature space.

If the support vectors are perturbed, the maximal-margin hyperplane will change.

If other training observations are perturbed (provided they are not moved within the margin distance of the hyperplane), the maximal-margin hyperplane remains unaffected.

## Finding $\mathbf{w}$ with a Large Margin

Let  $\mathbf{x}_n$  be the nearest data point to the hyperplane:

$$\mathbf{w}^\top \mathbf{x} = 0.$$

How far is it? Two preliminary technicalities:

- **Normalize  $\mathbf{w}$ :**

$$|\mathbf{w}^\top \mathbf{x}_n| = 1.$$

- **Separate the bias term  $b$ :**

$$\mathbf{w} = (w_1, \dots, w_d)$$

apart from  $b$ .

The hyperplane can now be written as:

$$\mathbf{w}^\top \mathbf{x} + b = 0.$$

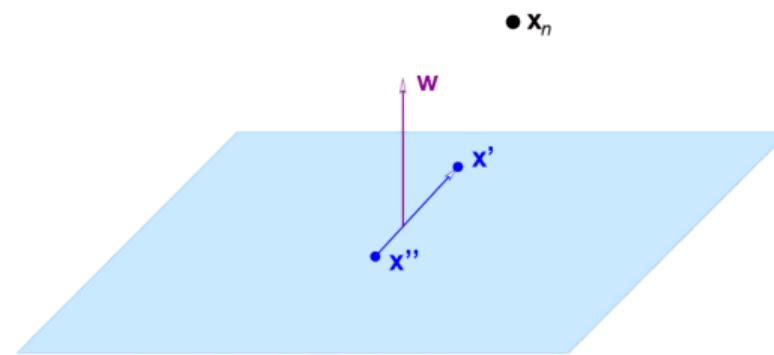
## Computing the Distance

The distance between  $\mathbf{x}_n$  and the hyperplane  $\mathbf{w}^\top \mathbf{x} + b = 0$ , where  $|\mathbf{w}^\top \mathbf{x}_n + b| = 1$ , is given as follows.

The vector  $\mathbf{w}$  is *orthogonal* to the hyperplane in the feature space  $\mathcal{X}$ .

Consider two points  $\mathbf{x}'$  and  $\mathbf{x}''$  lying on the hyperplane:

$$\mathbf{w}^\top \mathbf{x}' + b = 0 \quad \text{and} \quad \mathbf{w}^\top \mathbf{x}'' + b = 0 \implies \mathbf{w}^\top (\mathbf{x}' - \mathbf{x}'') = 0.$$

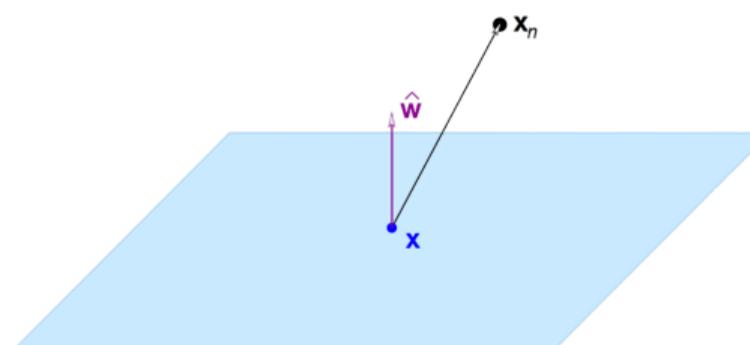


## And the Distance Is ...

The distance between  $\mathbf{x}_n$  and the hyperplane can be derived as follows. Take any point  $\mathbf{x}$  lying on the hyperplane. The distance is the projection of the vector  $(\mathbf{x}_n - \mathbf{x})$  onto  $\mathbf{w}$ :

$$\hat{\mathbf{w}} = \frac{\mathbf{w}}{\|\mathbf{w}\|} \implies \text{distance} = |\hat{\mathbf{w}}^\top (\mathbf{x}_n - \mathbf{x})|.$$

$$\text{distance} = \frac{1}{\|\mathbf{w}\|} |\mathbf{w}^\top \mathbf{x}_n - \mathbf{w}^\top \mathbf{x}| = \frac{1}{\|\mathbf{w}\|} |\mathbf{w}^\top \mathbf{x}_n + b - \mathbf{w}^\top \mathbf{x} - b| = \frac{1}{\|\mathbf{w}\|}.$$



# The Optimization Problem

$$\begin{array}{ll}\text{Maximize} & \frac{1}{\|\mathbf{w}\|} \\ \text{subject to} & \min_{n=1,2,\dots,N} |\mathbf{w}^\top \mathbf{x}_n + b| = 1.\end{array}$$

**Note:**  $|\mathbf{w}^\top \mathbf{x}_n + b| = y_n (\mathbf{w}^\top \mathbf{x}_n + b)$

## Equivalent Formulation

$$\begin{array}{ll}\text{Minimize} & \frac{1}{2} \mathbf{w}^\top \mathbf{w} \\ \text{subject to} & y_n (\mathbf{w}^\top \mathbf{x}_n + b) \geq 1, \quad \text{for } n = 1, 2, \dots, N. \\ & \mathbf{w} \in \mathbb{R}^d, \quad b \in \mathbb{R}.\end{array}$$

**Next step:** Introduce Lagrange multipliers. Since the constraints are inequalities, the **KKT conditions** apply.

## Lagrangian Formulation

We define the Lagrangian function as:

$$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2} \mathbf{w}^\top \mathbf{w} - \sum_{n=1}^N \alpha_n (y_n (\mathbf{w}^\top \mathbf{x}_n + b) - 1),$$

to be minimized with respect to  $\mathbf{w}$  and  $b$ , and maximized with respect to each  $\alpha_n \geq 0$ .

Taking derivatives:

$$\nabla_{\mathbf{w}} \mathcal{L} = \mathbf{w} - \sum_{n=1}^N \alpha_n y_n \mathbf{x}_n = 0,$$

$$\frac{\partial \mathcal{L}}{\partial b} = - \sum_{n=1}^N \alpha_n y_n = 0.$$

## Substituting into the Lagrangian

Using the optimality conditions:

$$\mathbf{w} = \sum_{n=1}^N \alpha_n y_n \mathbf{x}_n, \quad \text{and} \quad \sum_{n=1}^N \alpha_n y_n = 0,$$

we substitute these into the Lagrangian:

$$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2} \mathbf{w}^\top \mathbf{w} - \sum_{n=1}^N \alpha_n (y_n (\mathbf{w}^\top \mathbf{x}_n + b) - 1).$$

This yields the dual objective function:

$$\mathcal{L}(\boldsymbol{\alpha}) = \sum_{n=1}^N \alpha_n - \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N y_n y_m \alpha_n \alpha_m \mathbf{x}_n^\top \mathbf{x}_m.$$

Maximize with respect to  $\boldsymbol{\alpha}$ , subject to  $\alpha_n \geq 0$  for  $n = 1, \dots, N$ , and  $\sum_{n=1}^N \alpha_n y_n = 0$ .



## The Solution: Quadratic Programming

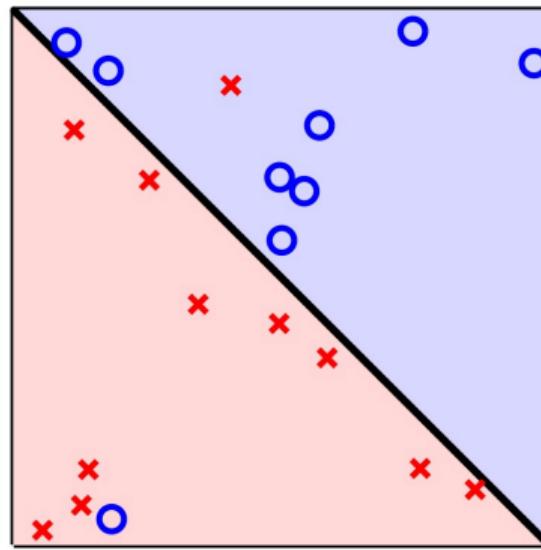
The dual optimization problem can be expressed as a standard quadratic program:

$$\min_{\alpha} \frac{1}{2} \boldsymbol{\alpha}^\top \begin{bmatrix} y_1 y_1 \mathbf{x}_1^\top \mathbf{x}_1 & y_1 y_2 \mathbf{x}_1^\top \mathbf{x}_2 & \dots & y_1 y_N \mathbf{x}_1^\top \mathbf{x}_N \\ y_2 y_1 \mathbf{x}_2^\top \mathbf{x}_1 & y_2 y_2 \mathbf{x}_2^\top \mathbf{x}_2 & \dots & y_2 y_N \mathbf{x}_2^\top \mathbf{x}_N \\ \vdots & \vdots & \ddots & \vdots \\ y_N y_1 \mathbf{x}_N^\top \mathbf{x}_1 & y_N y_2 \mathbf{x}_N^\top \mathbf{x}_2 & \dots & y_N y_N \mathbf{x}_N^\top \mathbf{x}_N \end{bmatrix} \boldsymbol{\alpha} + (-\mathbf{1})^\top \boldsymbol{\alpha},$$

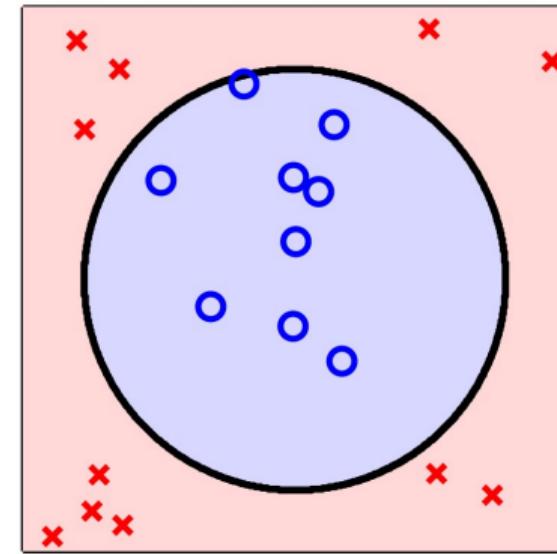
subject to

$$\mathbf{y}^\top \boldsymbol{\alpha} = 0, \quad \text{and} \quad \boldsymbol{\alpha} \geq \mathbf{0}.$$

# Non-Separable Data

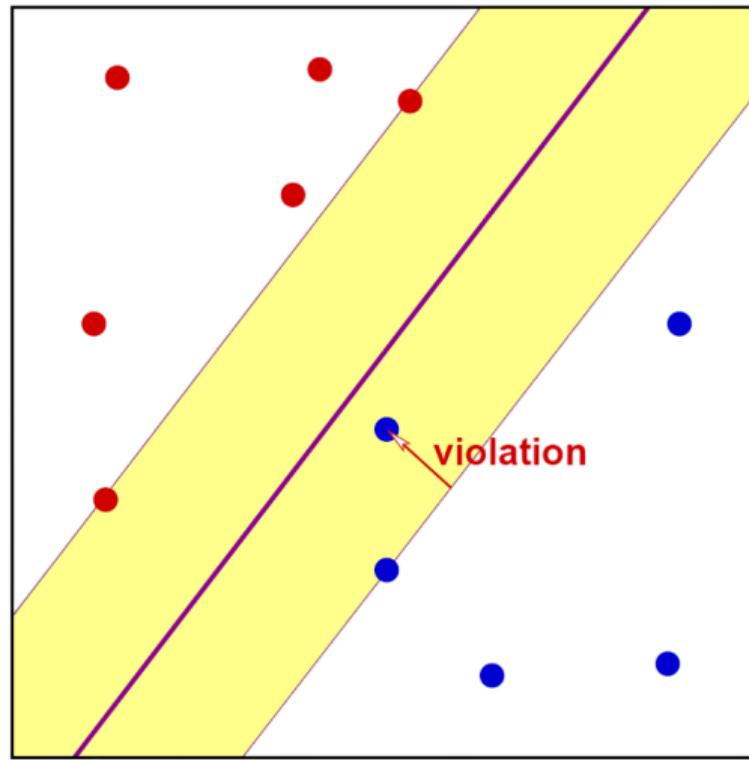


Allow Some Classification Error  
(Tolerate error)



Apply a Nonlinear Transformation

# Margin Violation



# Introducing Slack Variables

Margin violation occurs when:

$$y_n(\mathbf{w}^\top \mathbf{x}_n + b) \geq 1$$

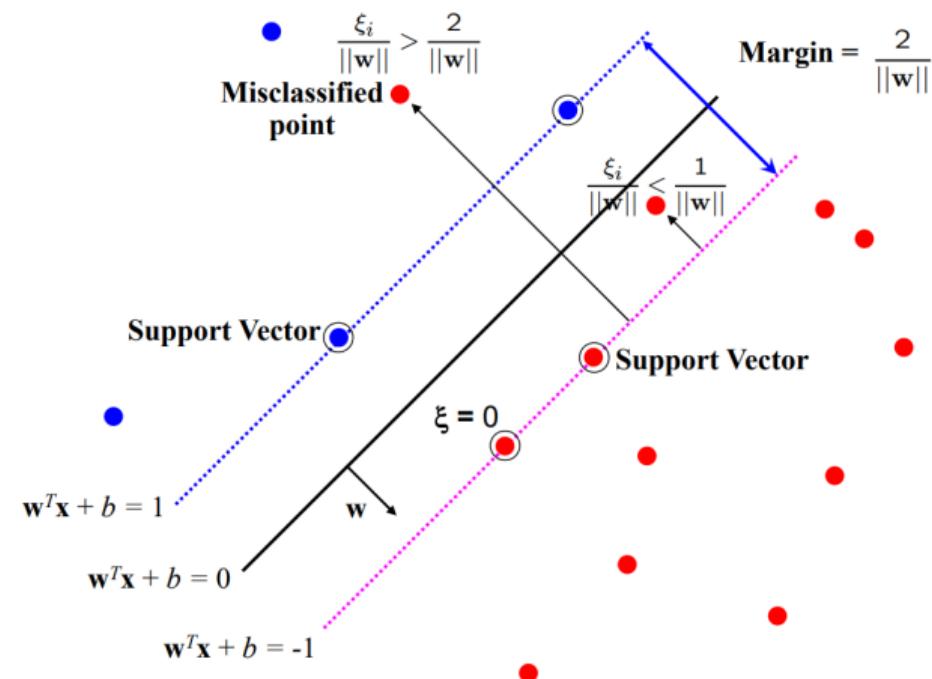
fails to hold.

To quantify this violation:

$$y_n(\mathbf{w}^\top \mathbf{x}_n + b) \geq 1 - \xi_n, \quad \xi_n \geq 0.$$

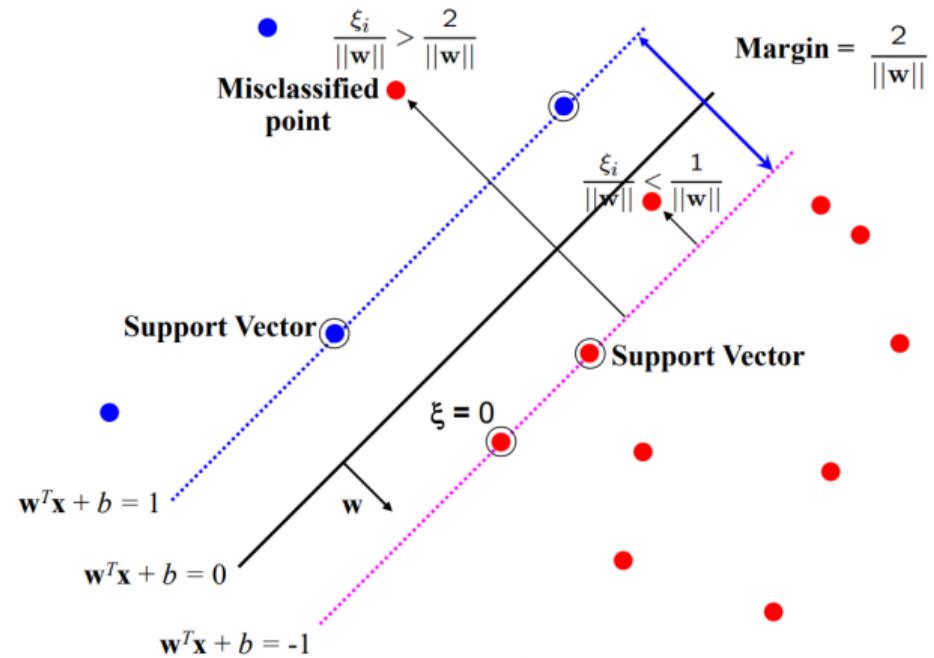
The total violation is:

$$\sum_{n=1}^N \xi_n.$$



# Introducing Slack Variables

- For  $0 < \xi_n \leq 1$ , the point lies between the margin and the correct side of the hyperplane. This is a **margin violation**.
- For  $\xi_n > 1$ , the point is **misclassified**.



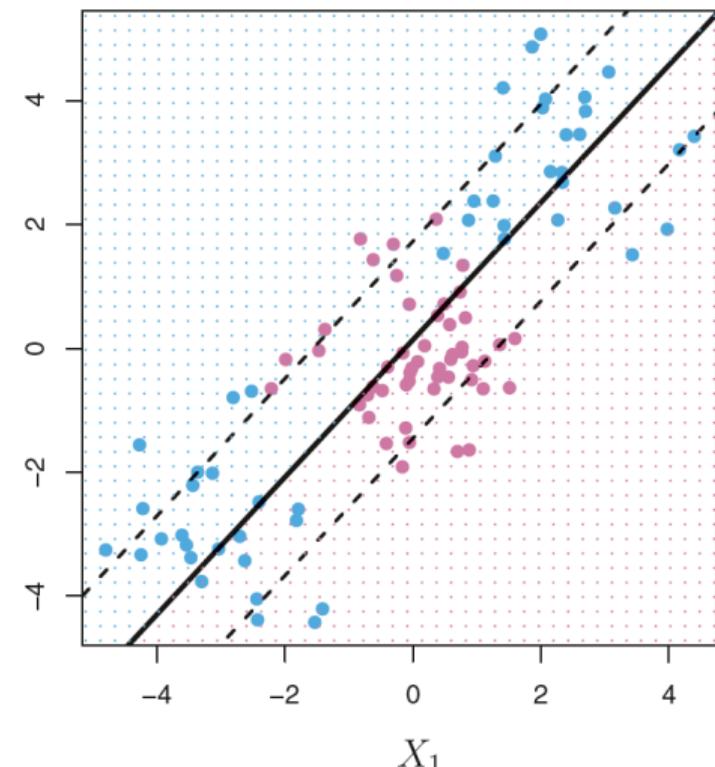
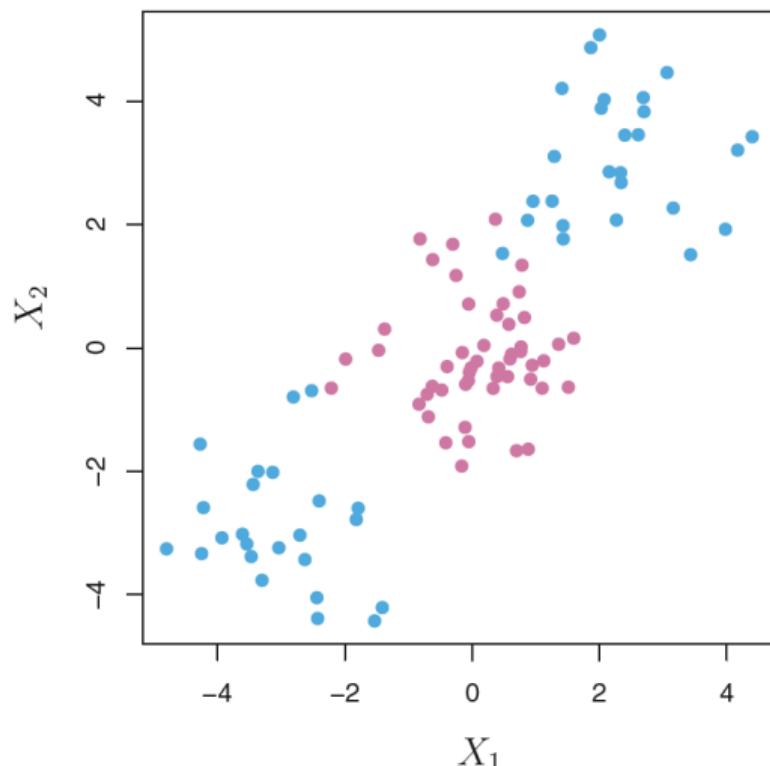
## 1 Introduction and Motivation

## 2 Hard-Margin SVM

## 3 Soft-Margin SVM

## 4 Kernel SVM

## Soft-Margin SVM



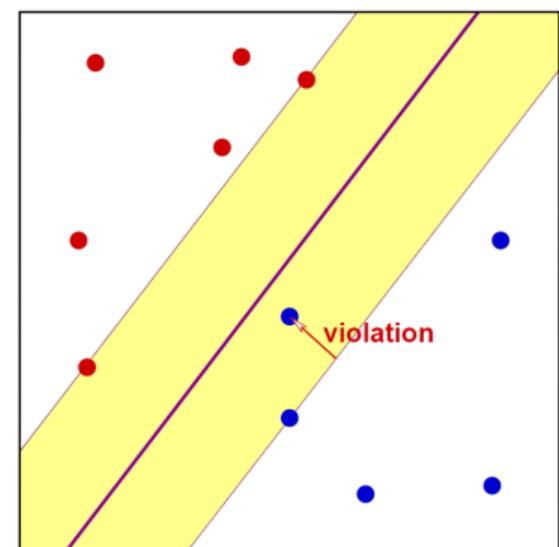
## Soft-Margin SVM

$$\text{minimize}_{\mathbf{w}, b, \xi} \quad \frac{1}{2} \mathbf{w}^\top \mathbf{w} + C \sum_{n=1}^N \xi_n$$

subject to:  $y_n(\mathbf{w}^\top \mathbf{x}_n + b) \geq 1 - \xi_n, \quad \text{for } n = 1, \dots, N,$

$\xi_n \geq 0, \quad \text{for } n = 1, \dots, N,$

$\mathbf{w} \in \mathbb{R}^d, \quad b \in \mathbb{R}, \quad \xi \in \mathbb{R}^N.$



# Soft-Margin SVM

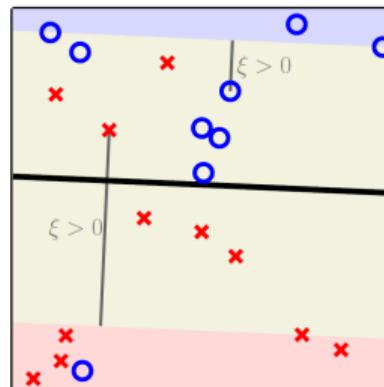
- Balances the **soft in-sample error**

$$\sum_{n=1}^N \xi_n$$

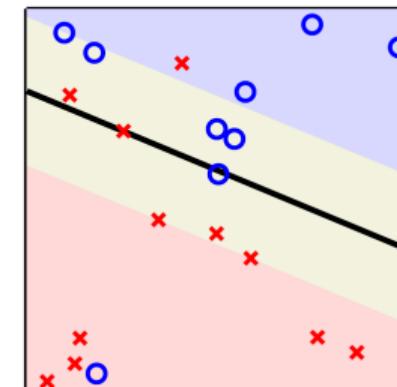
against the **weight norm**

$$\frac{1}{2} \mathbf{w}^\top \mathbf{w}.$$

- Every constraint can be satisfied if  $\xi_n$  is sufficiently large.
- The parameter  $C$  is a **regularization term** that controls the trade-off:
  - Small  $C$  allows the constraints to be relaxed easily  $\Rightarrow$  larger margin.
  - Large  $C$  enforces the constraints more strictly  $\Rightarrow$  narrower margin.
  - $C = \infty$  enforces all constraints, resulting in a **hard-margin SVM**.
- The optimization problem remains **quadratic** and has a unique global minimum.  
Note that there is only one hyperparameter:  $C$ .

Non-Separable Data and the Effect of  $C$ 

$$C = 1$$



$$C = 500$$

minimize <sub>$\mathbf{w}, b, \xi$</sub>

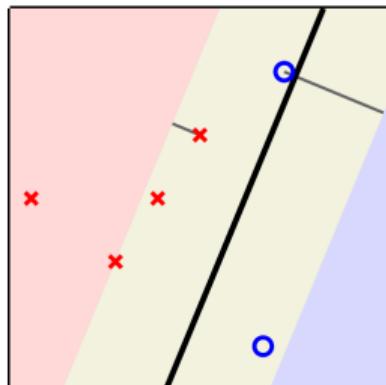
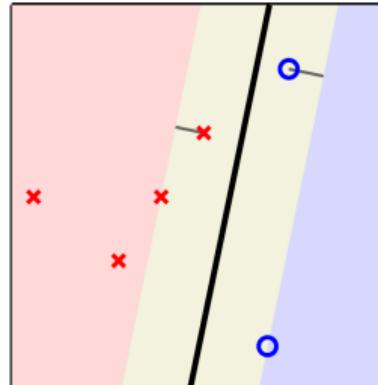
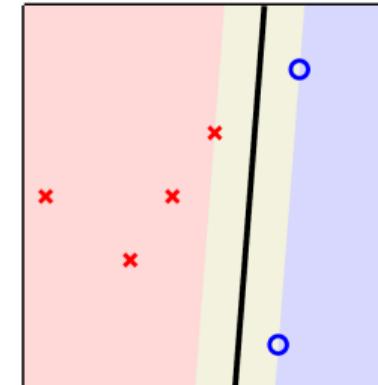
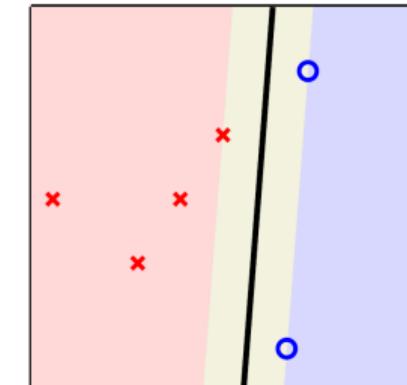
$$\frac{1}{2} \mathbf{w}^\top \mathbf{w} + C \sum_{n=1}^N \xi_n$$

subject to:

$$y_n(\mathbf{w}^\top \mathbf{x}_n + b) \geq 1 - \xi_n, \quad \text{for } n = 1, \dots, N,$$

$$\xi_n \geq 0, \quad \text{for } n = 1, \dots, N.$$

## Soft-Margin SVM with Separable Data

Small  $C$ Medium  $C$ Large  $C$ 

Hard Margin

$$\text{minimize}_{\mathbf{w}, b, \xi} \quad \frac{1}{2} \mathbf{w}^\top \mathbf{w} + C \sum_{n=1}^N \xi_n$$

**Choosing  $C$  Is Important**

subject to:  $y_n(\mathbf{w}^\top \mathbf{x}_n + b) \geq 1 - \xi_n, \quad \text{for } n = 1, \dots, N,$

$$\xi_n \geq 0, \quad \text{for } n = 1, \dots, N.$$

# Lagrange Formulation for the Soft-Margin SVM

$$\mathcal{L}(\mathbf{w}, b, \xi, \alpha, \beta) = \frac{1}{2} \mathbf{w}^\top \mathbf{w} + C \sum_{n=1}^N \xi_n - \sum_{n=1}^N \alpha_n (y_n (\mathbf{w}^\top \mathbf{x}_n + b) - 1 + \xi_n) - \sum_{n=1}^N \beta_n \xi_n$$

Minimize with respect to  $\mathbf{w}$ ,  $b$ , and  $\xi$ , and maximize with respect to each  $\alpha_n \geq 0$  and  $\beta_n \geq 0$ .

$$\nabla_{\mathbf{w}} \mathcal{L} = \mathbf{w} - \sum_{n=1}^N \alpha_n y_n \mathbf{x}_n = 0$$

$$\frac{\partial \mathcal{L}}{\partial b} = - \sum_{n=1}^N \alpha_n y_n = 0$$

$$\frac{\partial \mathcal{L}}{\partial \xi_n} = C - \alpha_n - \beta_n = 0$$

## Soft-Margin SVM: Dual Problem

$$\max_{\alpha} \left( \sum_{n=1}^N \alpha_n - \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N \alpha_n \alpha_m y_n y_m \mathbf{x}_n^\top \mathbf{x}_m \right)$$

Subject to:

$$\sum_{n=1}^N \alpha_n y_n = 0, \quad 0 \leq \alpha_n \leq C, \quad n = 1, \dots, N.$$

After solving this quadratic optimization problem, the optimal weight vector is given by:

$$\mathbf{w} = \sum_{n=1}^N \alpha_n y_n \mathbf{x}_n.$$

# Non-Linearily Separable Data

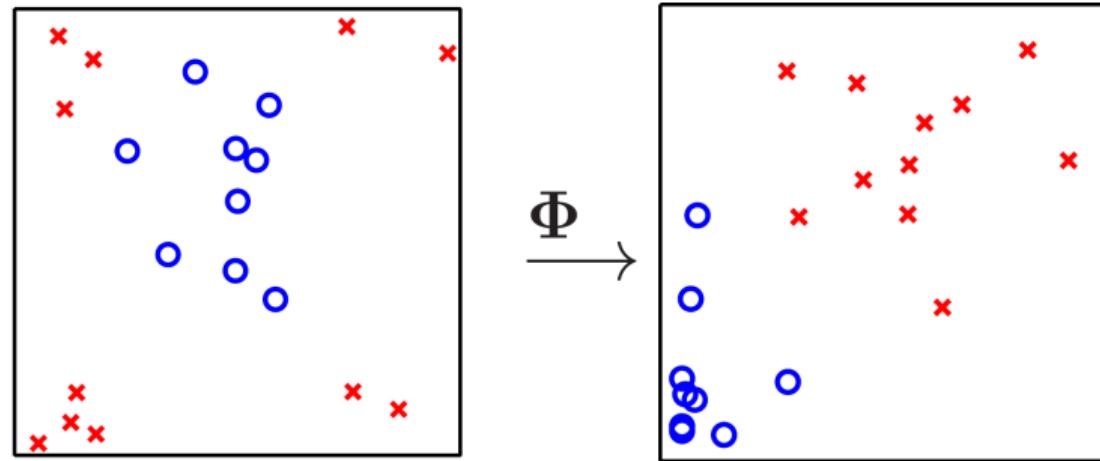
## Noisy Data or Overlapping Classes

(as discussed earlier: soft margin)

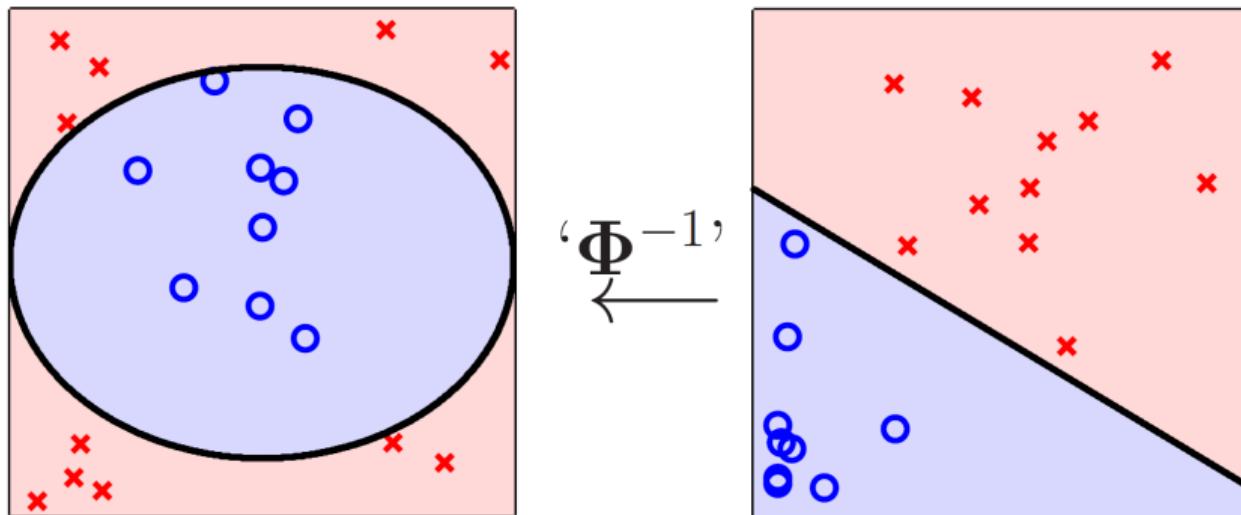
- Nearly linearly separable

## Non-Linear Decision Surface

- Transform data into a new feature space



## Mechanics of the Nonlinear Transform



## Soft-Margin SVM in a Transformed Space: Primal Problem

- Primal problem:

$$\min_{\mathbf{w}, b, \xi} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{n=1}^N \xi_n$$

Subject to:

$$y_n (\mathbf{w}^\top \phi(\mathbf{x}_n) + b) \geq 1 - \xi_n, \quad n = 1, \dots, N,$$

$$\xi_n \geq 0, \quad n = 1, \dots, N.$$

- $\mathbf{w} \in \mathbb{R}^m$ : weight vector to be learned.
- If  $m \gg d$  (i.e., the feature space is very high-dimensional), the model contains many more parameters to estimate.

## Soft-Margin SVM in a Transformed Space: Dual Problem

- Optimization problem:

$$\max_{\alpha} \left\{ \sum_{n=1}^N \alpha_n - \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N \alpha_n \alpha_m y_n y_m \phi(\mathbf{x}_n)^\top \phi(\mathbf{x}_m) \right\}$$

Subject to:

$$\sum_{n=1}^N \alpha_n y_n = 0, \quad 0 \leq \alpha_n \leq C, \quad n = 1, \dots, N.$$

- Since only inner products  $\phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j)$  appear, only the coefficients  $\alpha = [\alpha_1, \dots, \alpha_N]$  need to be learned.
- It is not necessary to learn  $m$  parameters explicitly, unlike the primal problem.

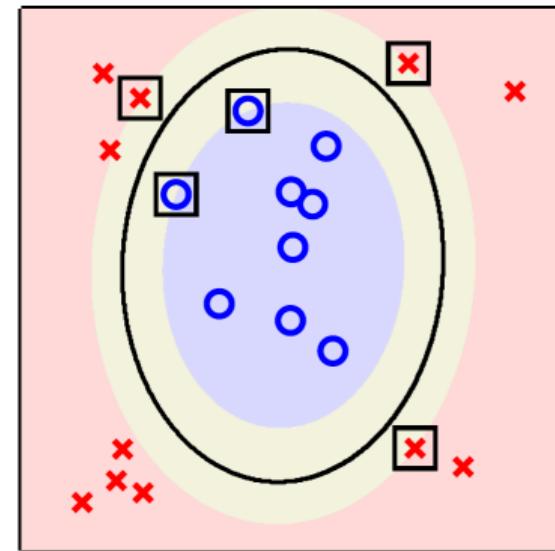
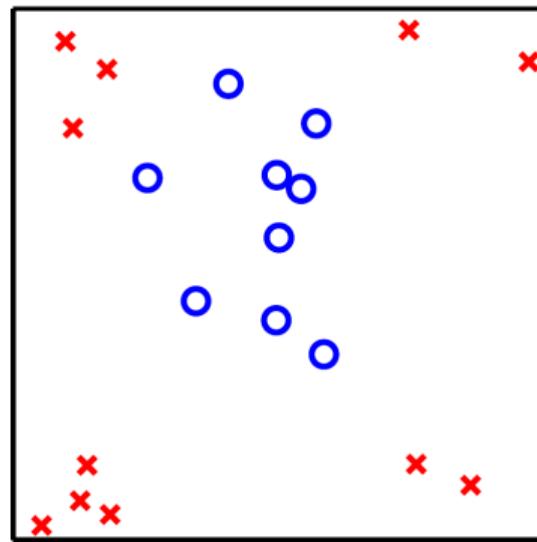
# Classifying a New Data Point

$$\hat{y} = \text{sign}(b + \mathbf{w}^\top \phi(\mathbf{x}))$$

where  $\mathbf{w} = \sum_{\alpha_n > 0} \alpha_n y_n \phi(\mathbf{x}_n)$

and  $b = y_s - \mathbf{w}^\top \phi(\mathbf{x}_s)$  for any support vector  $s$ .

## Non-Linearly Separable Data



## 1 Introduction and Motivation

## 2 Hard-Margin SVM

## 3 Soft-Margin SVM

## 4 Kernel SVM

## Kernel SVM

- Learns a linear decision boundary in a high-dimensional feature space without explicitly computing the mapped data.
- Let

$$\phi(\mathbf{x})^\top \phi(\mathbf{x}') = K(\mathbf{x}, \mathbf{x}') \quad (\text{kernel function})$$

- **Example:** for  $\mathbf{x} = [x_1, x_2]$  and a second-order mapping  $\phi(\cdot)$ :

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

Then,

$$K(\mathbf{x}, \mathbf{x}') = 1 + x_1 x'_1 + x_2 x'_2 + x_1^2 x'^2_1 + x_2^2 x'^2_2 + x_1 x_2 x'_1 x'_2$$

## Kernel Trick

- Compute  $K(\mathbf{x}, \mathbf{x}')$  directly, without explicitly transforming  $\mathbf{x}$  and  $\mathbf{x}'$ .
- **Example:** consider

$$\begin{aligned} K(\mathbf{x}, \mathbf{x}') &= (1 + \mathbf{x}^\top \mathbf{x}')^2 \\ &= (1 + x_1 x'_1 + x_2 x'_2)^2 \\ &= 1 + 2x_1 x'_1 + 2x_2 x'_2 + x_1^2 x'^2_1 + x_2^2 x'^2_2 + 2x_1 x_2 x'_1 x'_2 \end{aligned}$$

This corresponds to an inner product in the feature space:

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

$$\boldsymbol{\phi}(\mathbf{x}') = [1, \sqrt{2}x'_1, \sqrt{2}x'_2, x'^2_1, x'^2_2, \sqrt{2}x'_1 x'_2]$$

## Polynomial Kernel: Degree Two

- Instead of computing the explicit feature mapping, we use:

$$K(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^\top \mathbf{x}' + 1)^2$$

which corresponds to an implicit feature mapping.

- For a  $d$ -dimensional input vector:

$$\mathbf{x} = [x_1, \dots, x_d]^\top$$

- The corresponding feature mapping is:

$$\phi(\mathbf{x}) = [1, \sqrt{2}x_1, \dots, \sqrt{2}x_d, x_1^2, \dots, x_d^2, \sqrt{2}x_1x_2, \dots, \sqrt{2}x_1x_d, \sqrt{2}x_2x_3, \dots, \sqrt{2}x_{d-1}x_d]^\top$$

## Polynomial Kernel

- This kernel can be generalized to a  $d$ -dimensional input vector with polynomial feature mappings of order  $M$ :

$$\begin{aligned} K(\mathbf{x}, \mathbf{x}') &= (1 + \mathbf{x}^\top \mathbf{x}')^M \\ &= (1 + x_1 x'_1 + x_2 x'_2 + \cdots + x_d x'_d)^M \end{aligned}$$

- Example:** SVM decision boundary using a polynomial kernel:

$$\begin{aligned} b + \mathbf{w}^\top \phi(\mathbf{x}) &= 0 \\ \Rightarrow b + \sum_{\alpha_i > 0} \alpha_i y_i \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}) &= 0 \\ \Rightarrow b + \sum_{\alpha_i > 0} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) &= 0 \\ \Rightarrow b + \sum_{\alpha_i > 0} \alpha_i y_i (1 + \mathbf{x}_i^\top \mathbf{x})^M &= 0 \quad \Rightarrow \quad \text{Decision boundary is a polynomial of order } M. \end{aligned}$$

## Why Kernel?

- Kernel functions  $K$  can be computed efficiently, with a cost proportional to the input dimensionality  $d$  instead of the (potentially much larger) feature space dimension  $m$ .
- Example:** consider a second-order polynomial transformation:

$$\phi(\mathbf{x}) = [1, x_1, \dots, x_d, x_1^2, x_1x_2, \dots, x_dx_d]^\top, \quad m = 1 + d + d^2$$

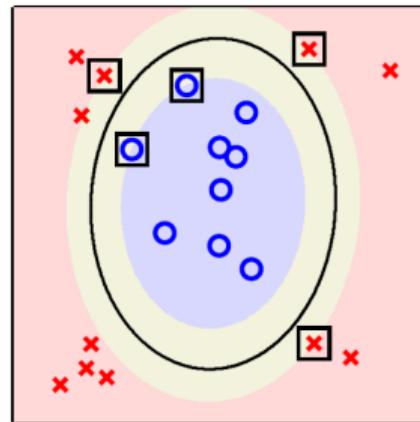
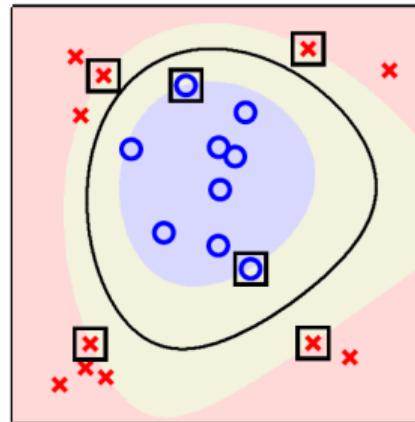
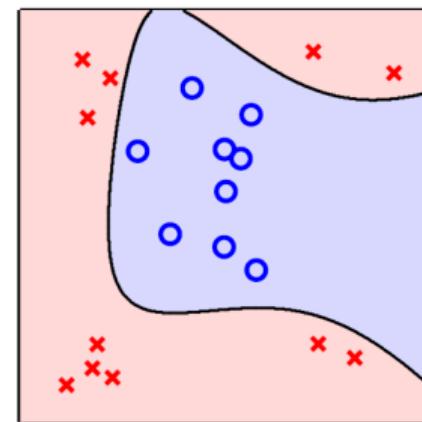
- Computing the inner product explicitly:

$$\phi(\mathbf{x})^\top \phi(\mathbf{x}') = 1 + \sum_{i=1}^d x_i x'_i + \sum_{i=1}^d \sum_{j=1}^d x_i x_j x'_i x'_j \quad O(m)$$

- Using the kernel trick instead:

$$\phi(\mathbf{x})^\top \phi(\mathbf{x}') = 1 + (\mathbf{x}^\top \mathbf{x}') + (\mathbf{x}^\top \mathbf{x}')^2 \quad O(d)$$

## Nonlinear Transform and SVM

 $\Phi_2 + \text{SVM}$  $\Phi_3 + \text{SVM}$  $\Phi_3 + \text{pseudoinverse algorithm}$ 

- The mapping  $\Phi_3$  has almost twice as many parameters as  $\Phi_2$ .
- The  $\Phi_3$ -SVM does not exhibit significant overfitting compared to  $\Phi_3$ -regression.
- The number of support vectors did not double.
- Higher-dimensional mappings are feasible as long as the number of support vectors remains small or the margin stays large.

## Gaussian (RBF) Kernel

- If  $K(\mathbf{x}, \mathbf{x}')$  represents an inner product in some transformed space of  $\mathbf{x}$ , it is a valid kernel.
- Gaussian or Radial Basis Function (RBF) kernel:

$$K(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2)$$

- For the one-dimensional case with  $\gamma = 1$ :

$$\begin{aligned} K(x, x') &= \exp(-(x - x')^2) \\ &= \exp(-x^2) \exp(-x'^2) \exp(2xx') \\ &= \color{red}{\exp(-x^2)} \color{magenta}{\exp(-x'^2)} \sum_{k=0}^{\infty} \frac{(2xx')^k}{k!} \end{aligned}$$

## Some Common Kernel Functions

- Linear kernel:

$$K(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top \mathbf{x}'$$

- Polynomial kernel:

$$K(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^\top \mathbf{x}' + 1)^M$$

- Gaussian (RBF) kernel:

$$K(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2)$$

- Sigmoid (tanh) kernel:

$$K(\mathbf{x}, \mathbf{x}') = \tanh(a \mathbf{x}^\top \mathbf{x}' + b)$$

# Kernel Formulation of SVM

Optimization problem:

$$\max_{\alpha} \left[ \sum_{n=1}^N \alpha_n - \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N \alpha_n \alpha_m y_n y_m K(\mathbf{x}_n, \mathbf{x}_m) \right]$$

subject to:  $\sum_{n=1}^N \alpha_n y_n = 0, \quad 0 \leq \alpha_n \leq C, \quad n = 1, \dots, N$

Equivalent quadratic form:

$$\mathbf{Q} = \begin{bmatrix} y_1 y_1 K(\mathbf{x}_1, \mathbf{x}_1) & \cdots & y_1 y_N K(\mathbf{x}_1, \mathbf{x}_N) \\ \vdots & \ddots & \vdots \\ y_N y_1 K(\mathbf{x}_N, \mathbf{x}_1) & \cdots & y_N y_N K(\mathbf{x}_N, \mathbf{x}_N) \end{bmatrix}$$

## Gaussian (RBF) Kernel

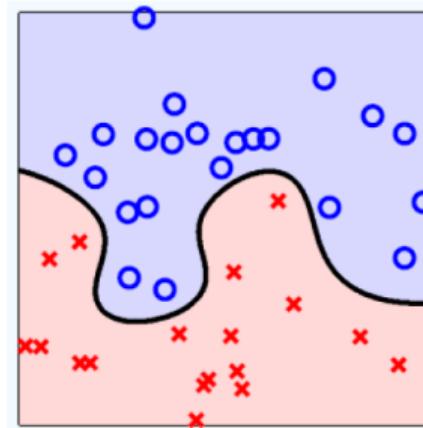
**Example:** SVM decision boundary with a Gaussian kernel

- Considers a Gaussian function centered around each training point:

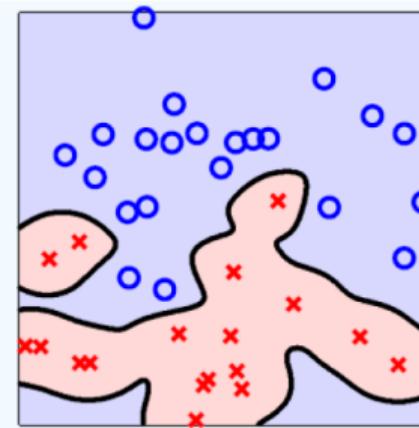
$$b + \sum_{\alpha_i > 0} \alpha_i y^{(i)} \exp(-\gamma \| \mathbf{x} - \mathbf{x}_i \|^2) = 0$$

- An SVM with a Gaussian kernel can, in principle, classify *any* training set.
  - The training error becomes zero as  $\sigma \rightarrow 0$ .
    - In this case, all samples become support vectors — indicating potential **overfitting**.

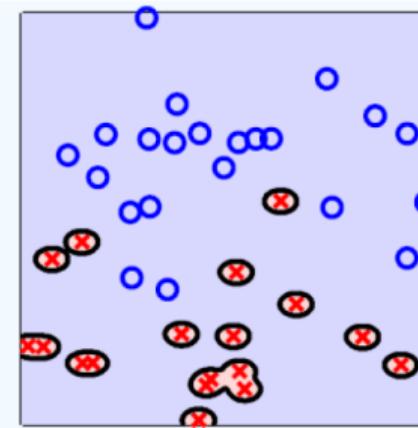
## Hard-Margin Example



$$\exp(-1\|\mathbf{x} - \mathbf{x}'\|^2)$$



$$\exp(-10\|\mathbf{x} - \mathbf{x}'\|^2)$$

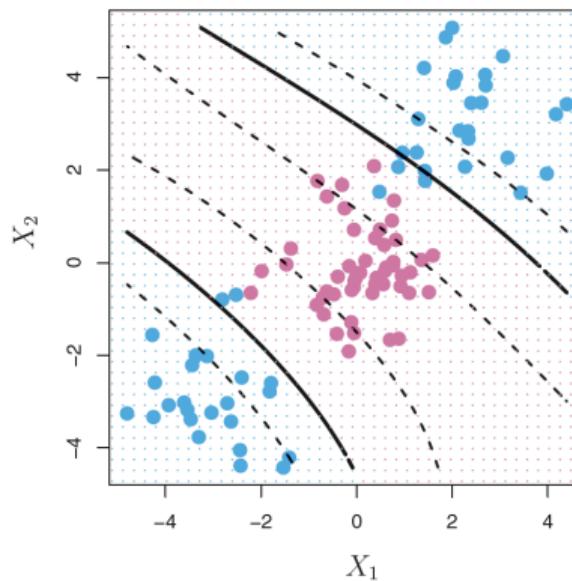


$$\exp(-100\|\mathbf{x} - \mathbf{x}'\|^2)$$

For a **narrow Gaussian** (i.e., small  $\sigma$ ), even maintaining a large margin cannot prevent overfitting.

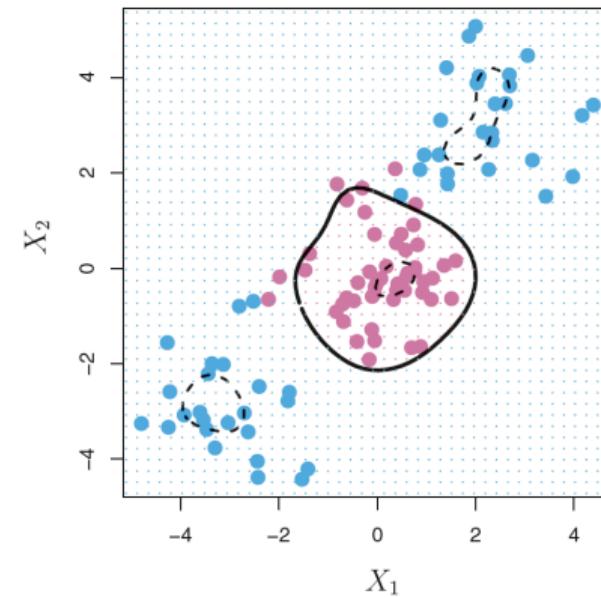
# SVM with Polynomial and Gaussian Kernels

SVM with a Polynomial Kernel (degree 3)



$$K(\mathbf{x}_i, \mathbf{x}'_i) = \left(1 + \sum_{j=1}^p x_{ij}x'_{ij}\right)^d$$

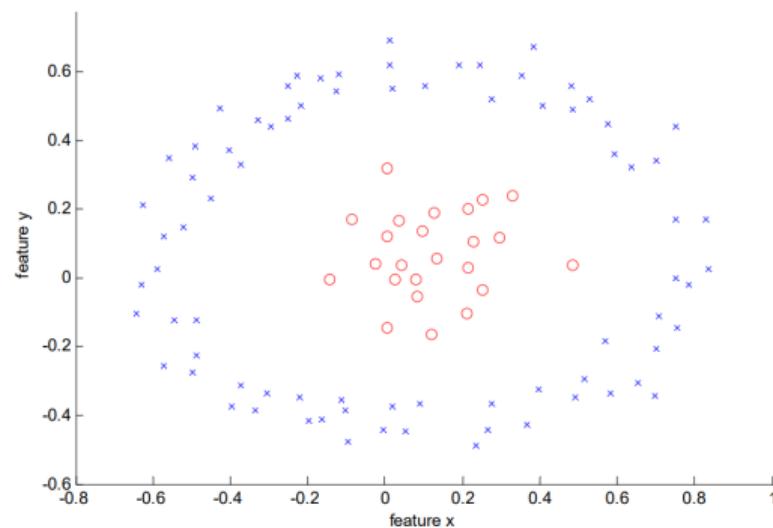
SVM with a Radial Kernel



$$K(\mathbf{x}_i, \mathbf{x}'_i) = \exp\left(-\gamma \sum_{j=1}^p (x_{ij} - x'_{ij})^2\right)$$

# SVM with a Gaussian (RBF) Kernel: Example

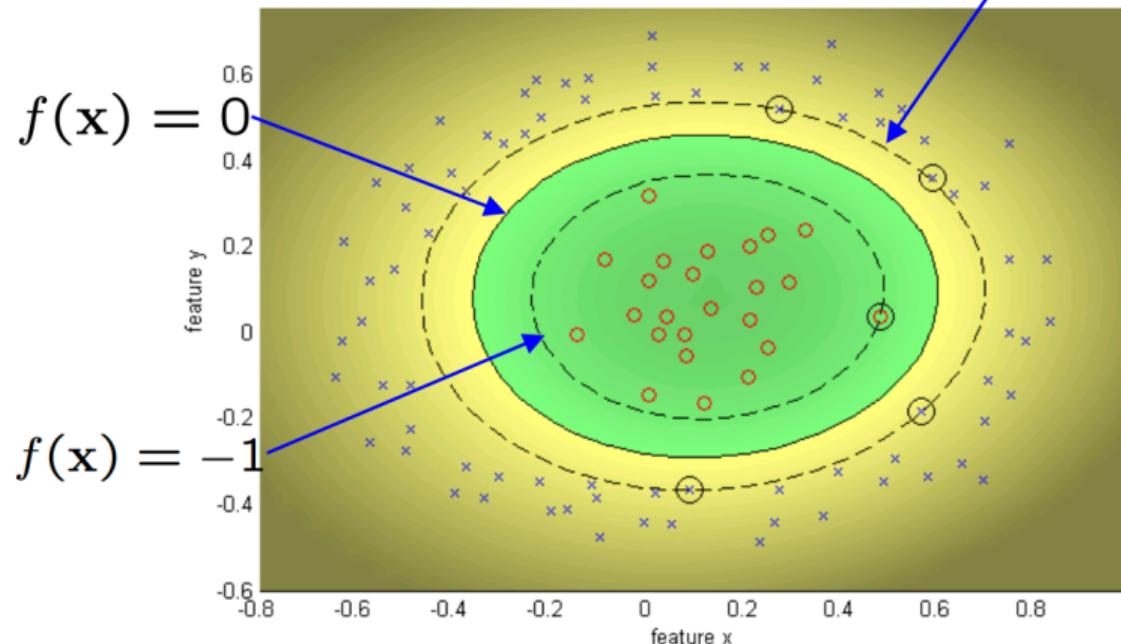
$$f(\mathbf{x}) = w_0 + \sum_{a_i > 0} \alpha_i y^{(i)} \exp\left(-\gamma \|\mathbf{x} - \mathbf{x}^{(i)}\|^2\right)$$



## SVM Gaussian kernel: Example

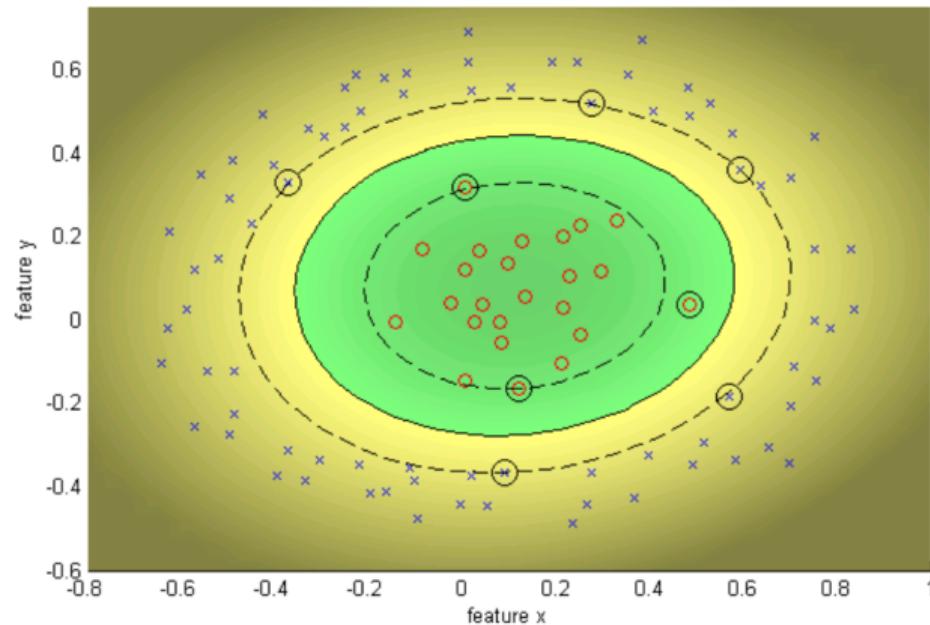
$$\sigma = 1.0 \quad C = \infty$$

$$f(\mathbf{x}) = 1$$



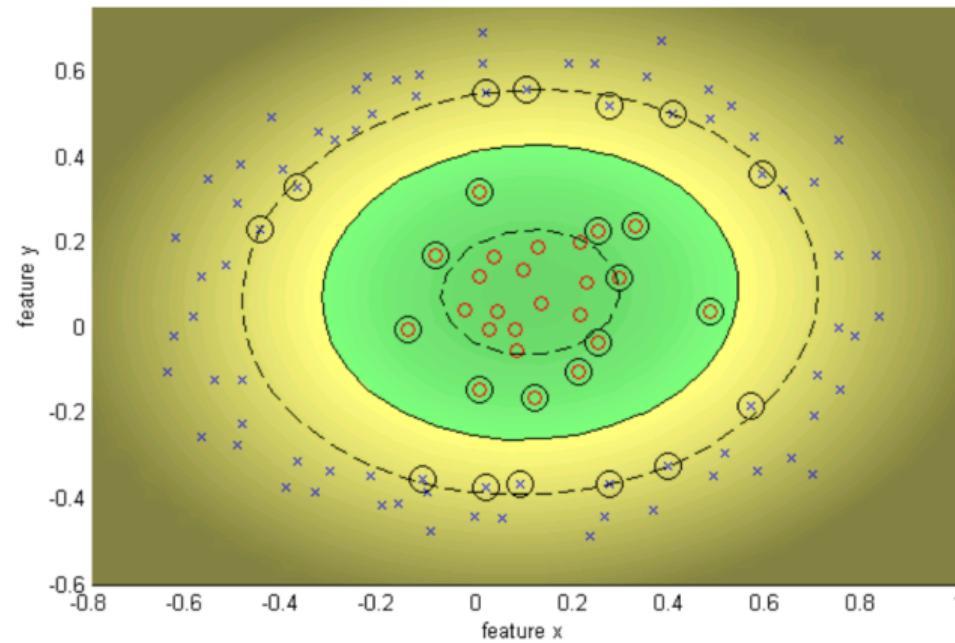
## SVM Gaussian Kernel: Example

$$\sigma = 1.0 \quad C = 100$$



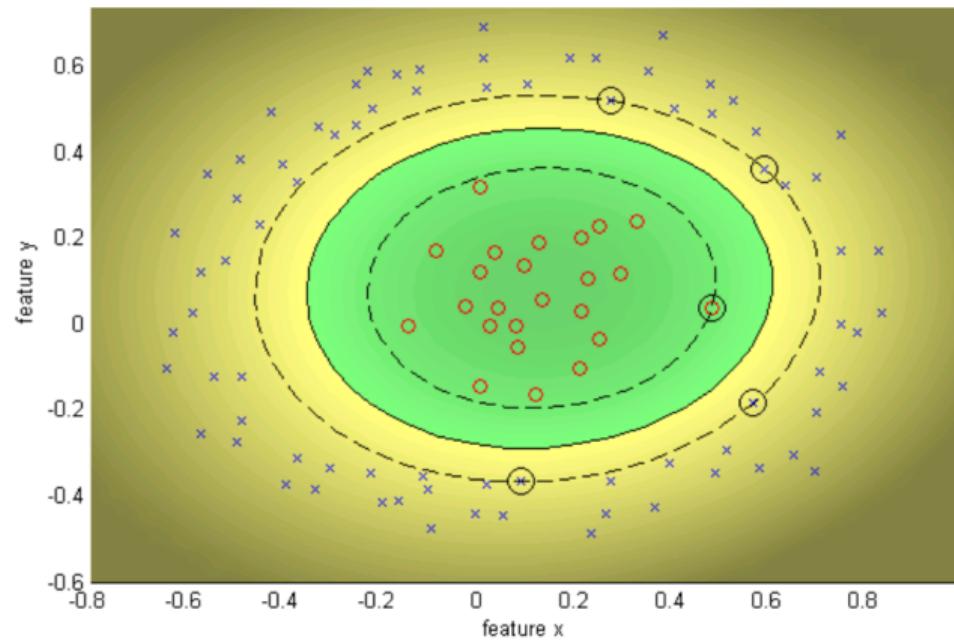
## SVM Gaussian Kernel: Example

$$\sigma = 1.0 \quad C = 10$$



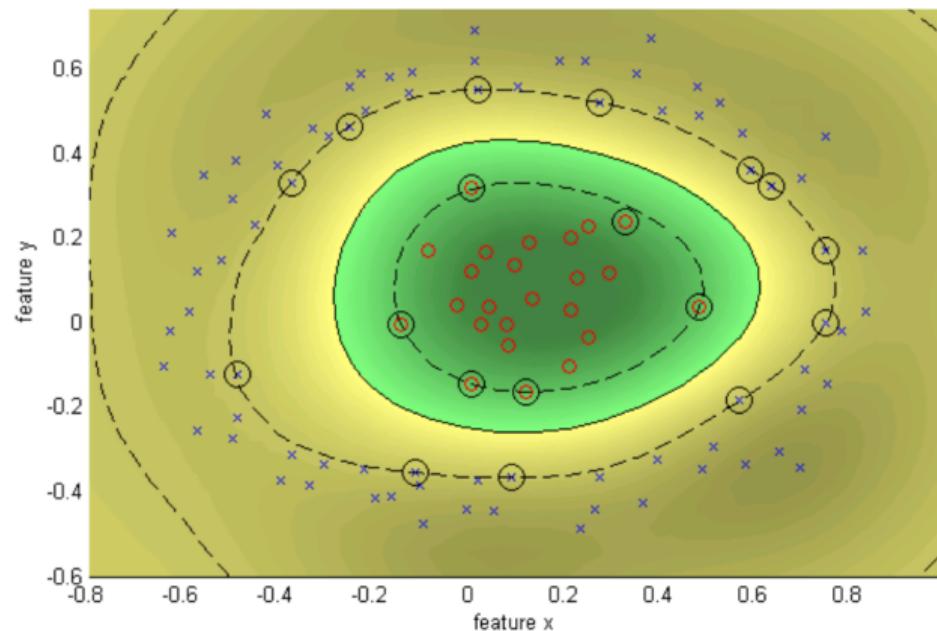
## SVM Gaussian Kernel: Example

$$\sigma = 1.0 \quad C = \infty$$



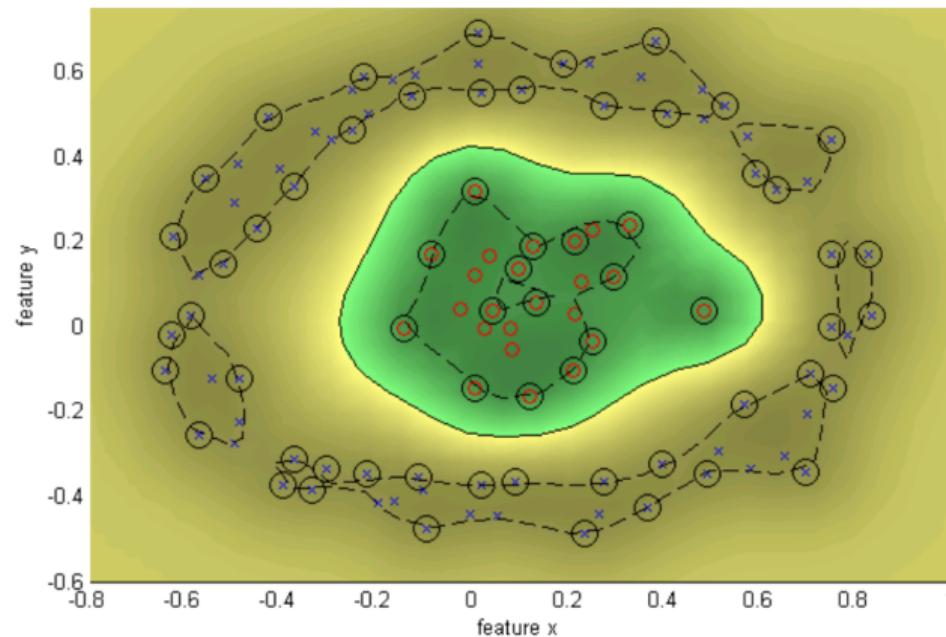
## SVM Gaussian Kernel: Example

$$\sigma = 0.25 \quad C = \infty$$



## SVM Gaussian Kernel: Example

$$\sigma = 0.1 \quad C = \infty$$



# Contributions

- This slide has been prepared thanks to:
  - Mahdi Aghaei

Introduction and Motivation  
oooooooooooo

Hard-Margin SVM  
oooooooooooo

Soft-Margin SVM  
oooooooooooo

Kernel SVM  
oooooooooooooooooooo●