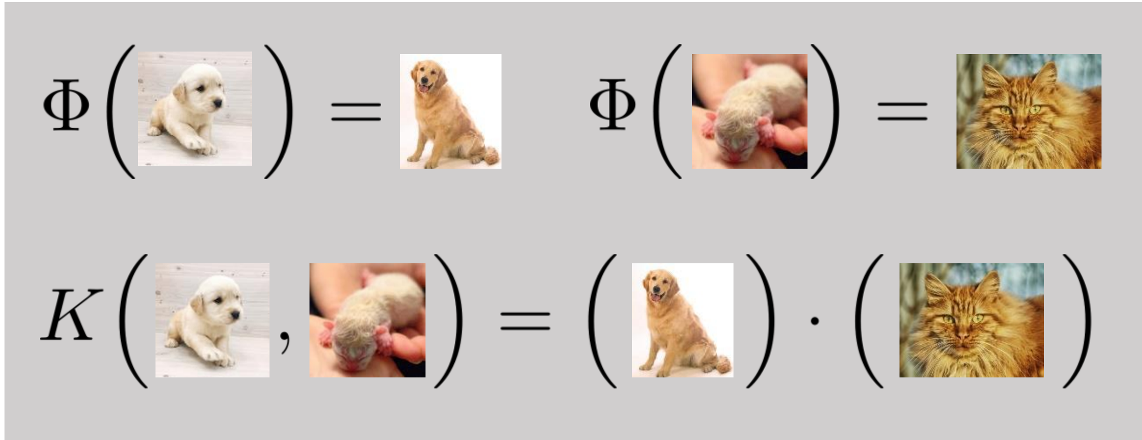



# Kernel-based Learning: Support Vector Machine – Hard Margin

Pilsung Kang

School of Industrial Management Engineering

Korea University

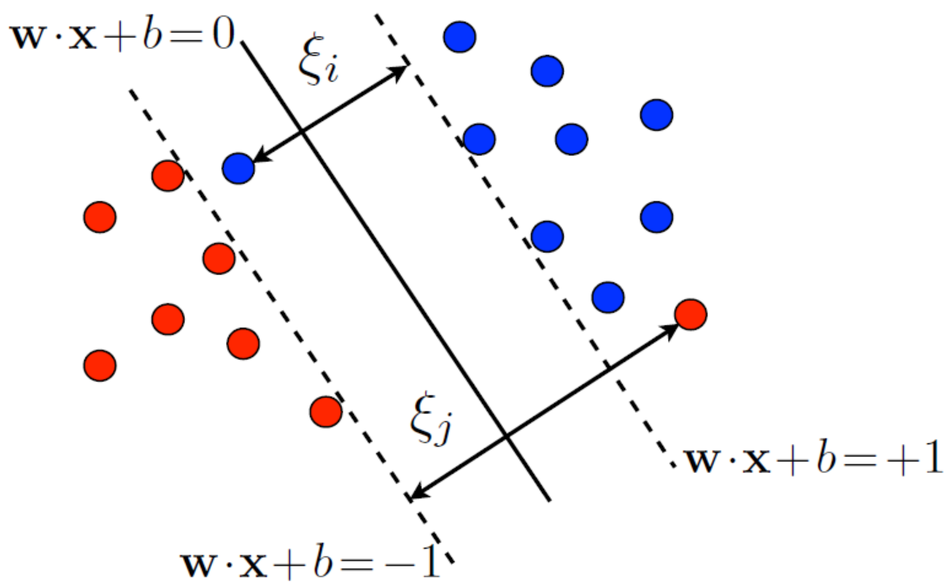
# SVM Case 2: Linear Case & Soft Margin

- Optimization Problem (C-SVM)
  - ✓ Objective function

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

  - ✓ Constraints

$$s.t.\ \ y_i\left(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b\right) \geq 1 - \xi_i, \xi_i \geq 0, \ \ \forall i$$



# SVM Case 2: Linear Case & Soft Margin

- Optimization Problem
  - ✓ Lagrangian Problem

$$\min \quad L_P(\mathbf{w}, b, \alpha_i) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i\left(y_i\left(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b\right) - 1 + \xi_i\right) - \sum_{i=1}^{N}\mu_i\xi_i$$

$$s.t. \quad \alpha_i \geq 0$$

  - ✓ KKT conditions

$$\frac{\partial L_p}{\partial \mathbf{w}} = 0 \quad \Rightarrow \quad \mathbf{w} = \sum_{i=1}^{N}\alpha_i\, y_i\mathbf{x}_i \qquad\qquad \frac{\partial L_p}{\partial b} = 0 \quad \Rightarrow \quad \sum_{i=1}^{N}\alpha_i y_i = 0$$

$$\frac{\partial L_p}{\partial \xi_i} = 0 \quad \Rightarrow \quad C - \alpha_i - \mu_i = 0$$

# SVM Case 2: Linear Case & Soft Margin

- From Primal to Dual

$$\min \quad L_P(\mathbf{w}, b, \alpha_i) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i\big(y_i(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b) - 1 + \xi_i\big) - \sum_{i=1}^{N}\mu_i\xi_i$$

$$s.t. \quad \alpha_i \geq 0$$

$$\max \quad L_D(\alpha_i) = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^{\mathrm{T}}\mathbf{x}_j$$

$$s.t. \quad \sum_{i=1}^{N}\alpha_i y_i = 0 \quad and \quad 0 \leq \alpha_i \leq C$$

- Solution

  ✓ Only requires support vectors

$$f(\mathbf{x}_{new}) = sign\left(\sum_{i=1}^{N}\alpha_i y_i\, \mathbf{x}_i^{\mathrm{T}}\mathbf{x}_{new} + b\right)$$

# SVM Case 2: Linear Case & Soft Margin

- From KKT condition, we know that

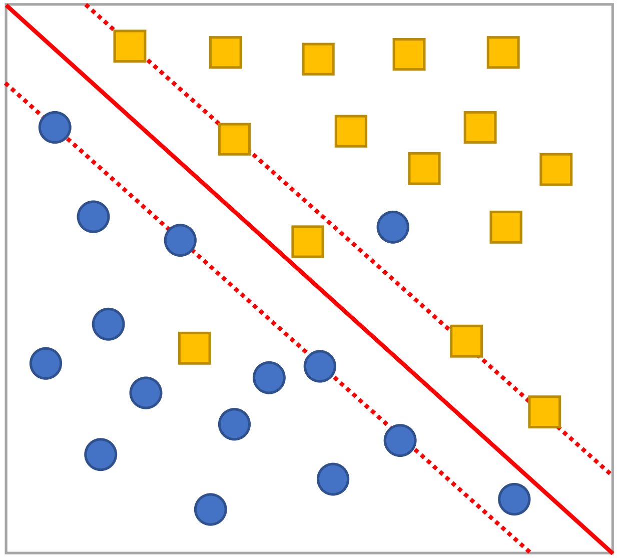
$$\alpha_i \left( y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 + \xi_i \right) = 0$$

✓ Thus, the only support vectors have $\alpha_i \neq 0$

✓ In addition,

$$C - \alpha_i - \mu_i = 0 \quad \& \quad \mu_i \xi_i = 0$$

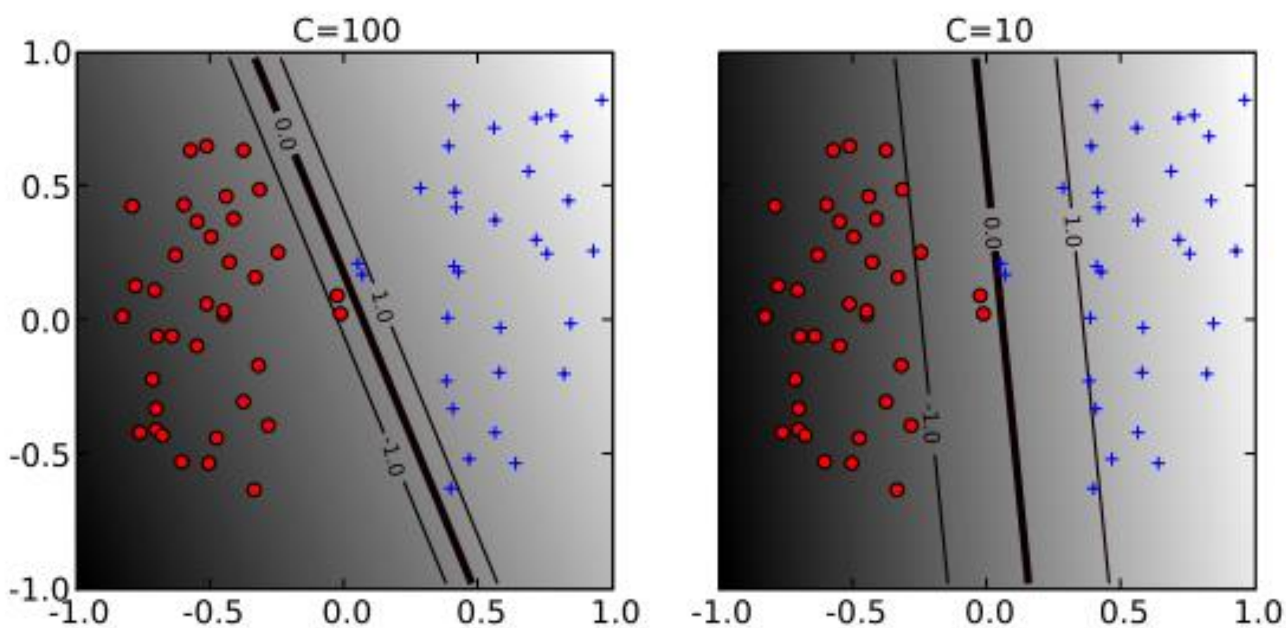
- Case 1: $\alpha_i = 0$ → non-support vectors
- Case 2: $0 < \alpha_i < C$ → support vectors on the margin
- Case 3: $\alpha_i = C$ → support vectors outside the margin



# SVM Case 2: Linear Case & Soft Margin

- Regularization cost C
  - ✓ Large C: narrow margin with a few support vectors
  - ✓ Small C: wide margin with many support vectors

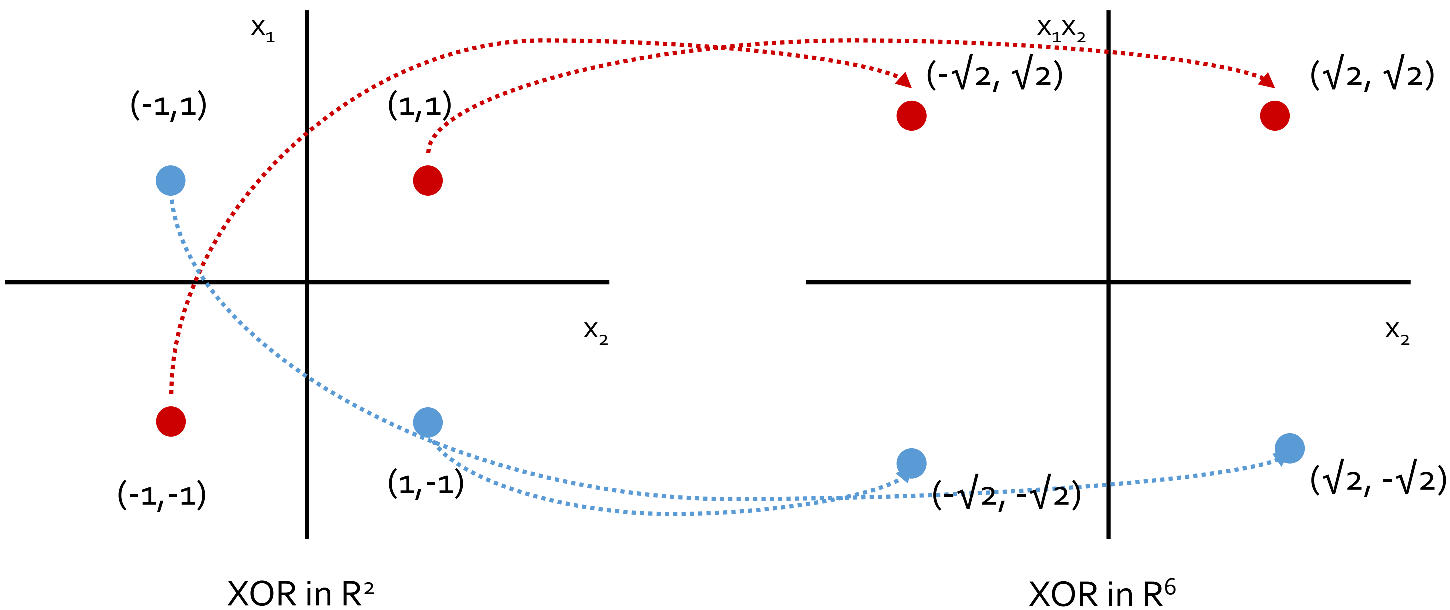
$$\min \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i$$



# SVM Case 3: Non-linear Case & Soft Margin

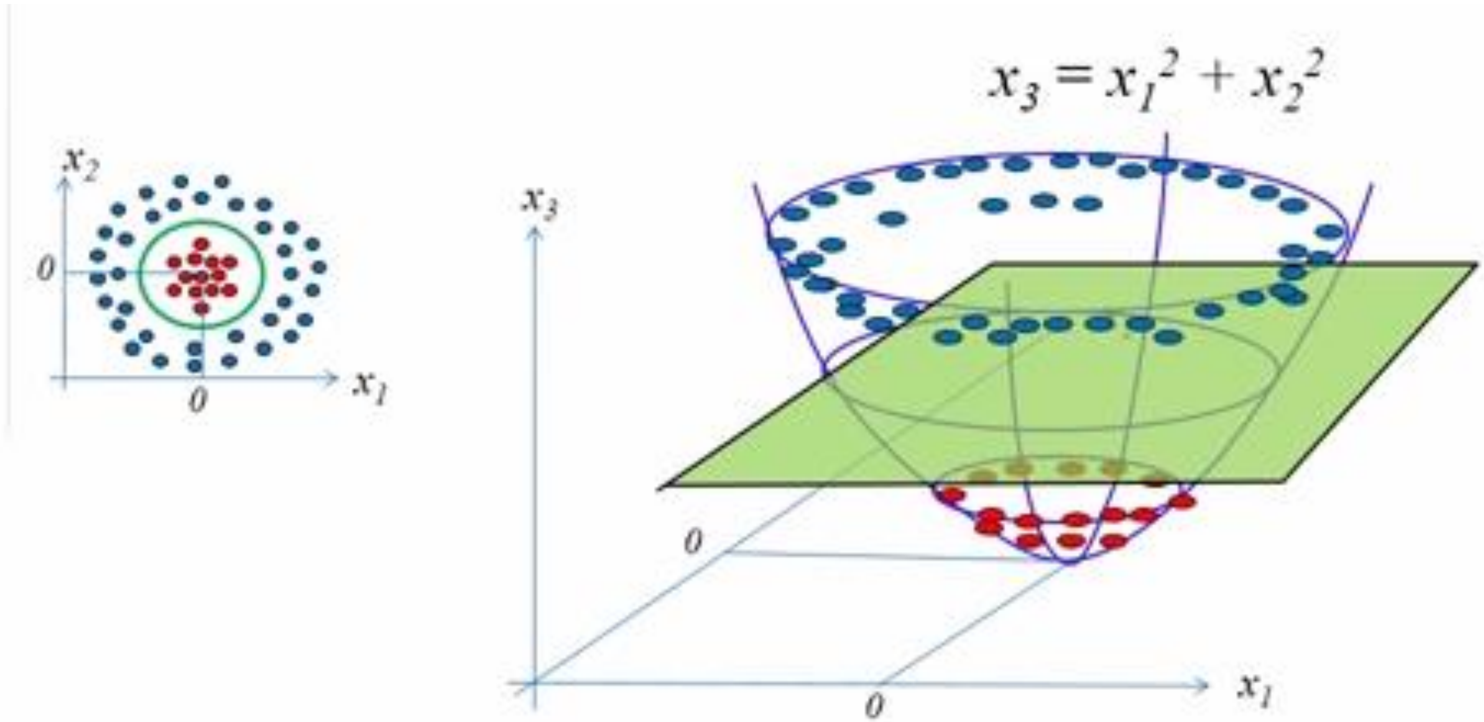
- What if the decision boundary is not linear?
  - ✓ Use a mapping function $\Phi(\mathrm{x})$ to transform an input vector from a lower dimensional space into a higher dimensional space

$$\Phi : (\mathbf{x}_1, \mathbf{x}_2) \rightarrow (\mathbf{x}_1^2, \mathbf{x}_2^2, \sqrt{2}\mathbf{x}_1, \sqrt{2}\mathbf{x}_2, \sqrt{2}\mathbf{x}_1\mathbf{x}_2, 1)$$



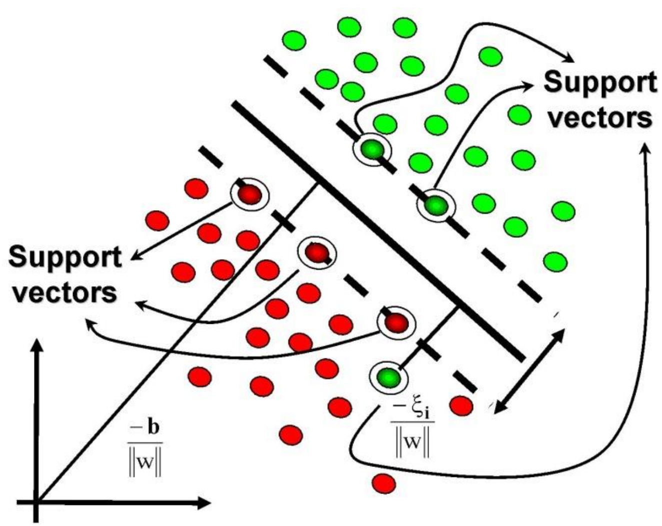
# SVM Case 3: Non-linear Case & Soft Margin

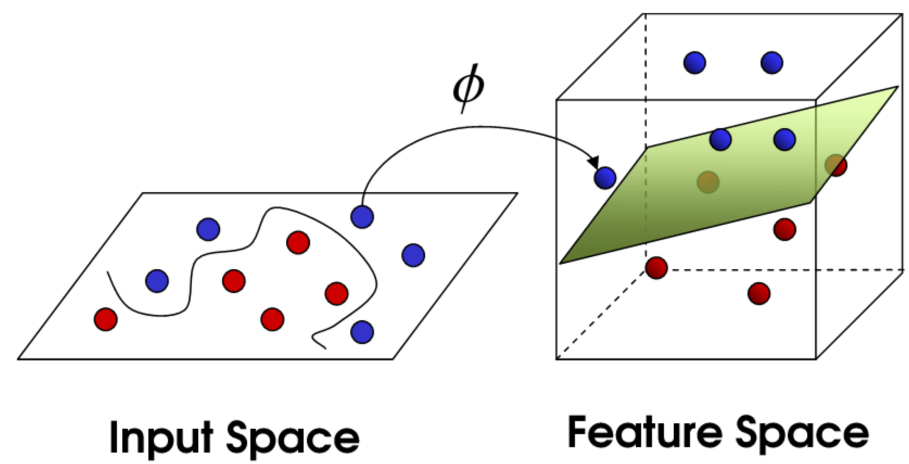
- What if the decision boundary is not linear?
  - ✓ Transform an input vector into a higher feature vector



# SVM Case 3: Non-linear Case & Soft Margin

- Goal: To find the best hyperplane that maximizes the margin and minimize the misclassification error
  - ✓ **Large margin**: preserve the generalization ability
  - ✓ **Flexible**: can generate an arbitrary shape of boundary in the input space

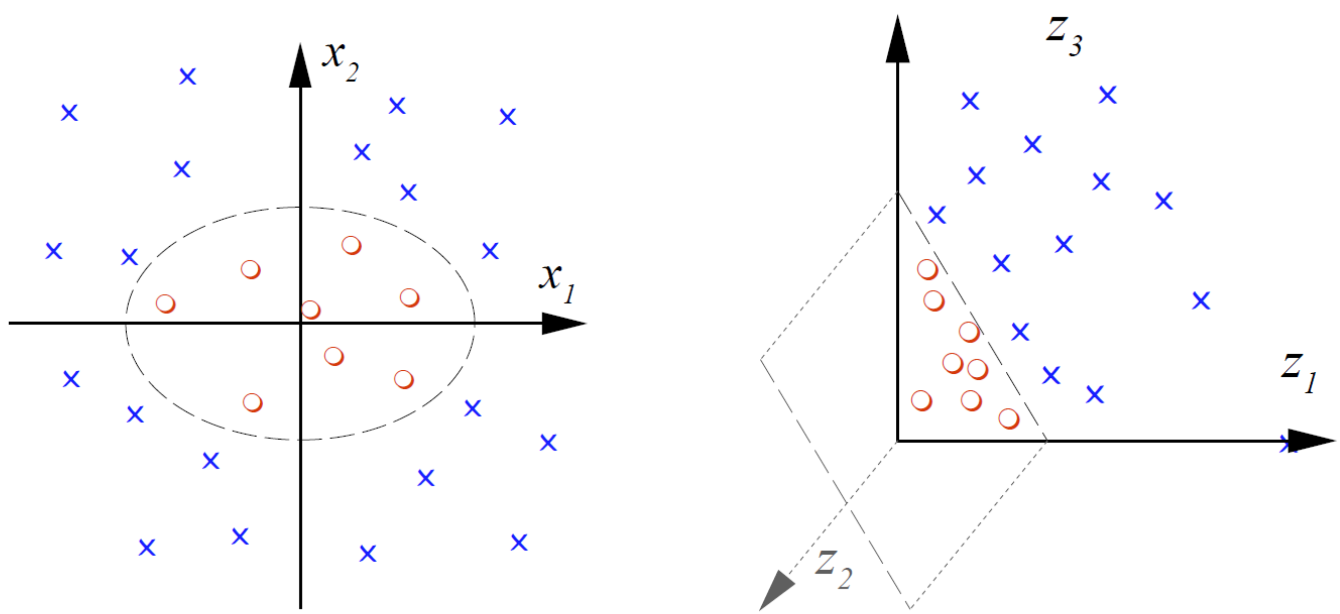


# SVM Case 3: Non-linear Case & Soft Margin

- Goal: To find the best hyperplance that maximizes the margin and minimize the misclassification error
  - ✓ **Large margin**: preserve the generalization ability
  - ✓ **Flexible**: can generate an arbitrary shape of boundary in the input space

$$\Phi : \mathbb{R}^2 \to \mathbb{R}^3$$
$$(x_1, x_2) \mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt{2}\,x_1 x_2, x_2^2)$$



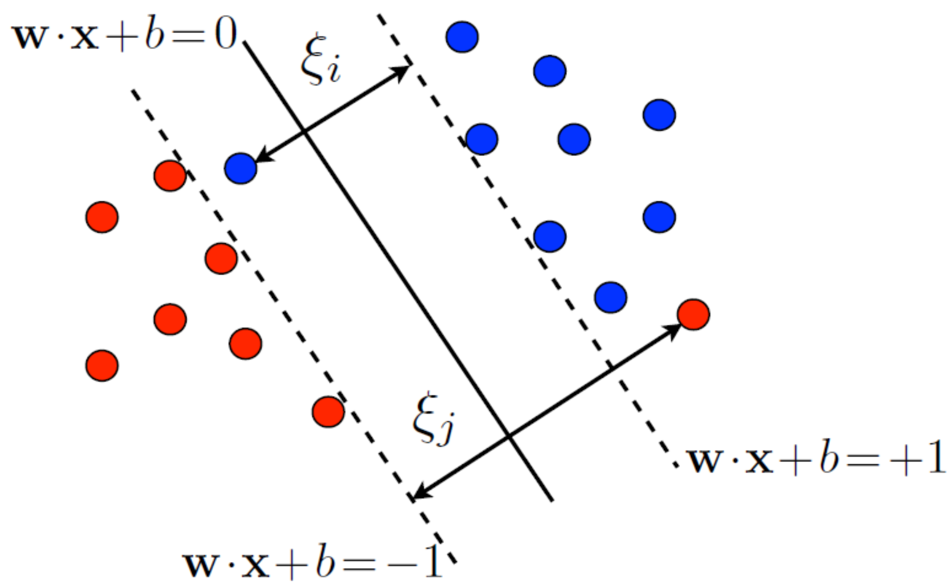
# SVM Case 3: Non-linear Case & Soft Margin

- Optimization Problem (C-SVM)
  - ✓ Objective function

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

  - ✓ Constraints

$$s.t. \quad y_i(\mathbf{w}^T\Phi(\mathbf{x}_i) + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad \forall i$$



# SVM Case 3: Non-linear Case & Soft Margin

- Optimization Problem (C-SVM)
  - ✓ Lagrangian Problem

$$\min \quad L_P(\mathbf{w}, b, \alpha_i) = \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_i$$

$$-\sum_{i=1}^{n}\alpha_i\big(y_i(\mathbf{w}^T\Phi(\mathbf{x}_i) + b) - 1 + \xi_i\big) - \sum_{i=1}^{n}\mu_i\xi_i$$

  - ✓ KKT condition

$$\frac{\partial L_P}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \sum_{i=1}^{n}\alpha_i y_i \Phi(\mathbf{x}_i) \qquad \frac{\partial L_P}{\partial b} = 0 \Rightarrow \sum_{i=1}^{n}\alpha_i y_i = 0$$

$$\frac{\partial L_P}{\partial \xi_i} = 0 \Rightarrow C - \alpha_i - \mu_i = 0$$

# SVM Case 3: Non-linear Case & Soft Margin

✓ Lagrangian Problem (Primal)

$$\min \quad L_P(\mathbf{w}, b, \alpha_i) = \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_i$$

$$- \sum_{i=1}^{n}\alpha_i\big(y_i(\mathbf{w}^T\Phi(\mathbf{x}_i) + b) - 1 + \xi_i\big) - \sum_{i=1}^{n}\mu_i\xi_i$$

✓ Lagrangian Problem (Dual)

$$\max \quad L_D = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \Phi(\mathbf{x}_i)^T\Phi(\mathbf{x}_j)$$

$$s.t. \quad \sum_{i=1}^{n}\alpha_i y_i = 0 \quad and \quad 0 \leq \alpha_i \leq C$$

고려대학교 KOREA UNIVERSITY



DSBA Data Science & Business Analytics

# SVM Case 3: Non-linear Case & Soft Margin

- How to find a mapping function $\Phi$?
  - ✓ Introduce a Kernel Trick

$$\max \quad L_D = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j)$$

$$s.t. \quad \sum_{i=1}^{n} \alpha_i y_i = 0 \quad and \quad 0 \leq \alpha_i \leq C$$

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

$$s.t. \quad \sum_{i=1}^{n} \alpha_i y_i = 0 \quad and \quad 0 \leq \alpha_i \leq C$$

# Kernel Trick

- Generalized inner product

Given two points $\mathbf{x}$ and $\mathbf{x}' \in \mathcal{X}$, we need $\mathbf{z}^\intercal\mathbf{z}'$

Let $\mathbf{z}^\intercal\mathbf{z}' = K(\mathbf{x}, \mathbf{x}')$ (the kernel) "inner product" of $\mathbf{x}$ and $\mathbf{x}'$

Example: $\mathbf{x} = (x_1, x_2) \longrightarrow$ 2nd-order $\Phi$

$$\mathbf{z} = \Phi(\mathbf{x}) = (1, x_1, x_2, x_1^2, x_2^2, x_1 x_2)$$

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

고려대학교 KOREA UNIVERSITY



DSBA Data Science & Business Analytics

# Kernel Trick

- Kernel Trick

Can we compute $K(\mathbf{x}, \mathbf{x}')$ **without** transforming $\mathbf{x}$ and $\mathbf{x}'$ ?

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

$$= 1 + x_1^2 x'^2_1 + x_2^2 x'^2_2 + 2x_1 x'_1 + 2x_2 x'_2 + 2x_1 x'_1 x_2 x'_2$$

This is an inner product!

$$(1, x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}\, x_1 x_2)$$

$$(1, x'^2_1, x'^2_2, \sqrt{2}x'_1, \sqrt{2}x'_2, \sqrt{2}x'_1 x'_2)$$

고려대학교 KOREA UNIVERSITY



DSBA Data Science & Business Analytics

# Kernel Trick

- Polynomial Kernel

$\mathcal{X} = \mathbb{R}^d$ and $\Phi : \mathcal{X} \to \mathcal{Z}$ is polynomial of order $Q$

The "equivalent" kernel $K(\mathbf{x}, \mathbf{x}') = (1 + \mathbf{x}^\intercal \mathbf{x}')^Q$

$$= (1 + x_1 x'_1 + x_2 x'_2 + \cdots + x_d x'_d)^Q$$

Compare for $d = 10$ and $Q = 100$

Can adjust scale: $K(\mathbf{x}, \mathbf{x}') = (a\mathbf{x}^\intercal \mathbf{x}' + b)^Q$

# Kernel Trick

- Gaussian (RBF) Kernel

If $K(\mathbf{x}, \mathbf{x}')$ is an inner product in some space $\mathcal{Z}$, we are good.

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

Infinite-dimensional $\mathcal{Z}$ : take simple case

$$K(x, x') = \exp\left(-(x - x')^2\right)$$

$$= \exp\left(-x^2\right) \exp\left(-x'^2\right) \underbrace{\sum_{k=0}^{\infty} \frac{2^k (x)^k (x')^k}{k!}}_{\exp(2xx')}$$

Taylor expansion of exponential function

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots \text{ for all } x$$

고려대학교 KOREA UNIVERSITY

DSBA Data Science & Business Analytics

# Kernel Trick

- Idea

  ✓ Define K: $X \times X \to R$, called Kernel, such that

$$\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j) = K(\mathbf{x}_i, \mathbf{x}_j)$$

  ✓ K is often interpreted as a similarity measure

- Benefits

  ✓ Efficiency

  - K is often more efficient to compute than Φ and the dot product

  ✓ Flexibility

  - K can be chosen arbitrarily so long as the existence of Φ is guaranteed (Mercer's condition)

# Kernel Trick

- Conditions on Kernel functions

For any symmetric function K: $\mathrm{X} \times \mathrm{X} \rightarrow R$ which is square integrable ($\mathrm{L}_2$-space) in $\mathrm{X} \times \mathrm{X}$ and which satisfies

$$\int_{\mathrm{X}\times\mathrm{X}} f(\mathbf{x})K(\mathbf{x},\mathbf{x}')f(\mathbf{x}')dxdx' \geq 0 \text{ for all } f \in L_2(\mathrm{X})$$

there exist functions $\phi_i : \mathrm{X} \rightarrow R$ and numbers $\lambda_i \geq 0$ such that

$$K(\mathbf{x},\mathbf{x}') = \sum_i \lambda_i \phi_i(\mathbf{x})\phi_i(\mathbf{x}') \text{ for all } \mathbf{x},\mathbf{x}' \in \mathrm{X}$$

- Interpretation

  ✓ Double integral is the continuous version of a vector-matrix-vector multiplication

$$\sum_i \sum_j K(\mathbf{x}_i, \mathbf{x}_j)\alpha_i \alpha_j \geq 0$$

고려대학교 KOREA UNIVERSITY



DSBA Data Science & Business Analytics

# Kernel Trick

- Conditions on Kernel functions

$K(\mathbf{x}, \mathbf{x}')$ is a valid kernel iff

1. It is symmetric and 2. The matrix:

$$\begin{bmatrix} K(\mathbf{x}_1, \mathbf{x}_1) & K(\mathbf{x}_1, \mathbf{x}_2) & \dots & K(\mathbf{x}_1, \mathbf{x}_N) \\ K(\mathbf{x}_2, \mathbf{x}_1) & K(\mathbf{x}_2, \mathbf{x}_2) & \dots & K(\mathbf{x}_2, \mathbf{x}_N) \\ \dots & \dots & \dots & \dots \\ K(\mathbf{x}_N, \mathbf{x}_1) & K(\mathbf{x}_N, \mathbf{x}_2) & \dots & K(\mathbf{x}_N, \mathbf{x}_N) \end{bmatrix}$$

is **positive semi-definite**

for any $\mathbf{x}_1, \cdots, \mathbf{x}_N$ (Mercer's condition)

# Kernel Trick

- Type of Kernel Functions
  - ✓ Polynomial

$$K(x,y) = (x \cdot y + c)^d, \quad c > 0$$

  - ✓ Gaussian (RBF)

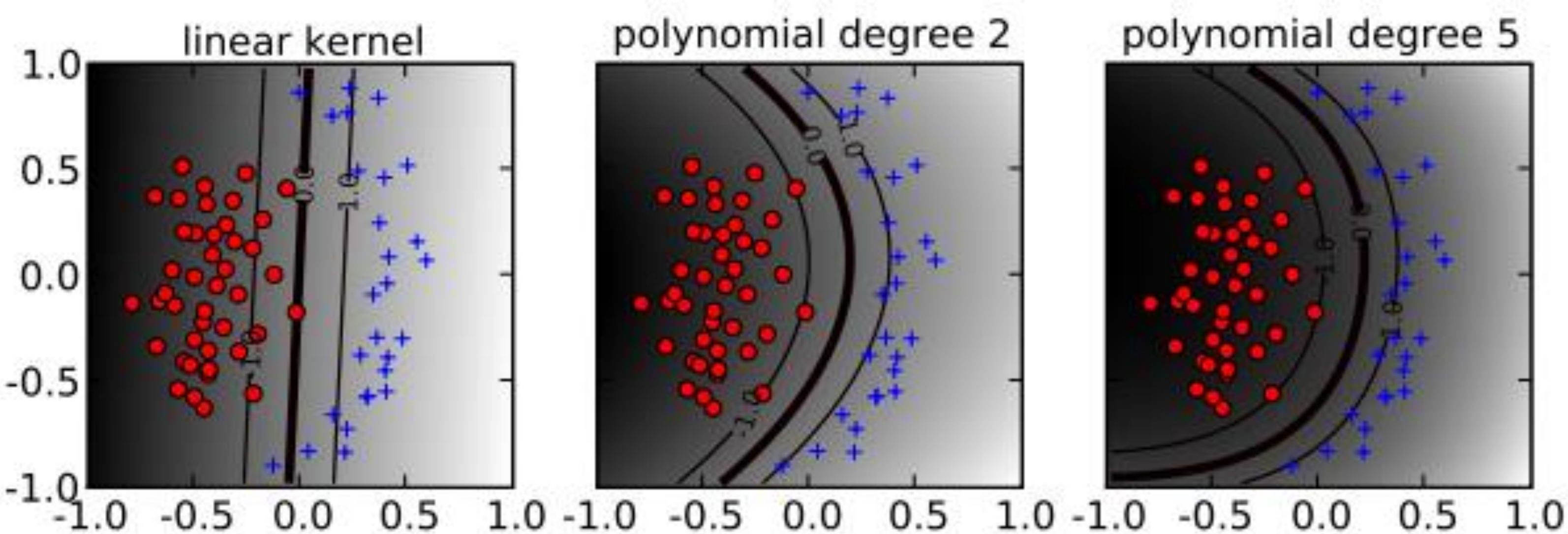
$$K(x,y) = \exp\left(-\frac{||x-y||^2}{2\sigma^2}\right), \quad \sigma \neq 0$$

  - ✓ Sigmoid

$$K(x,y) = \tanh(a(x \cdot y) + b), \quad a, b \geq 0$$

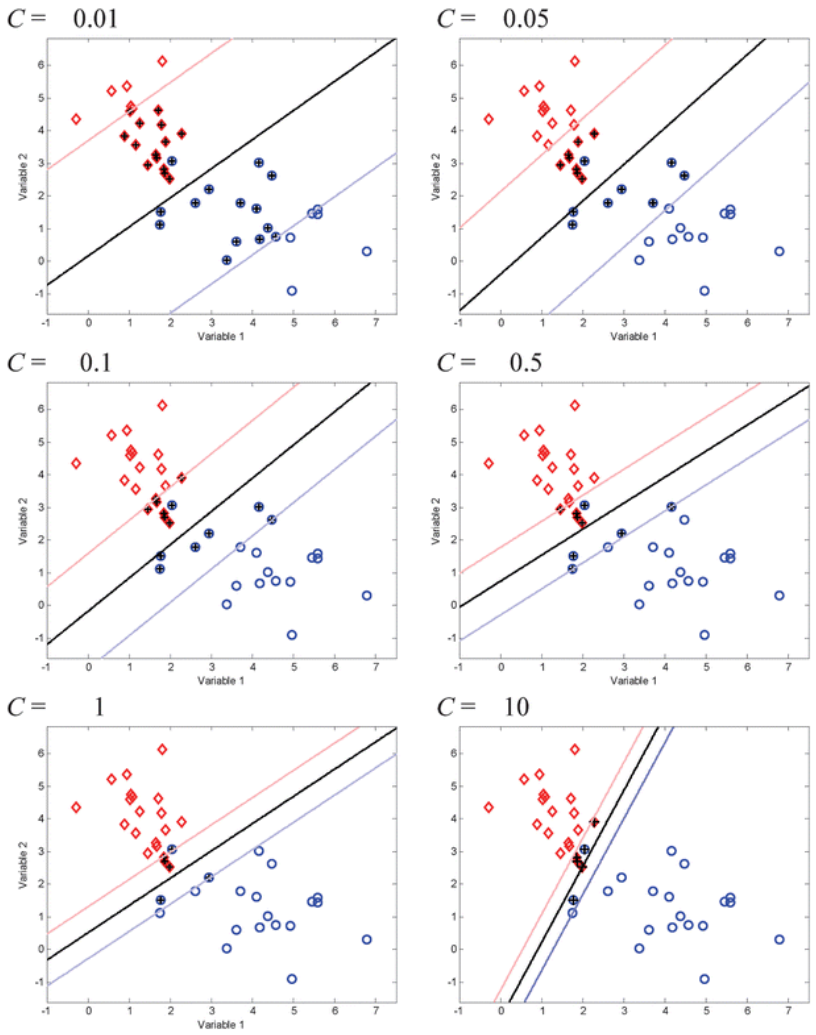
# Effects of Different Kernels

- Decision boundary for different kernels
  - ✓ Linear kernel: only generate linear decision boundary
  - ✓ Non-linear kernel: can generate complex shape of decision boundary
  - ✓ Gaussian(RBF) kernel is commonly used



# Effects of Different Costs

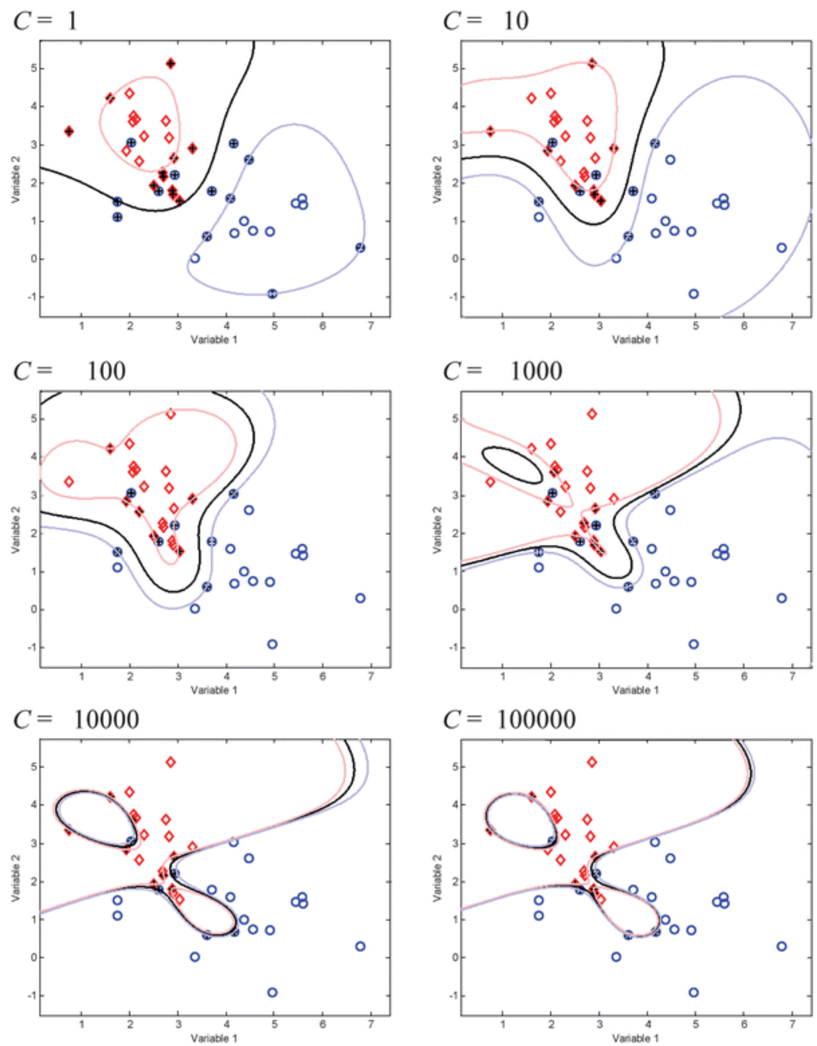
- Margins and SVs with different cost values (Linear kernel)




http://pubs.rsc.org/-/content/articlehtml/2010/an/b918972f

# Effects of Different Costs

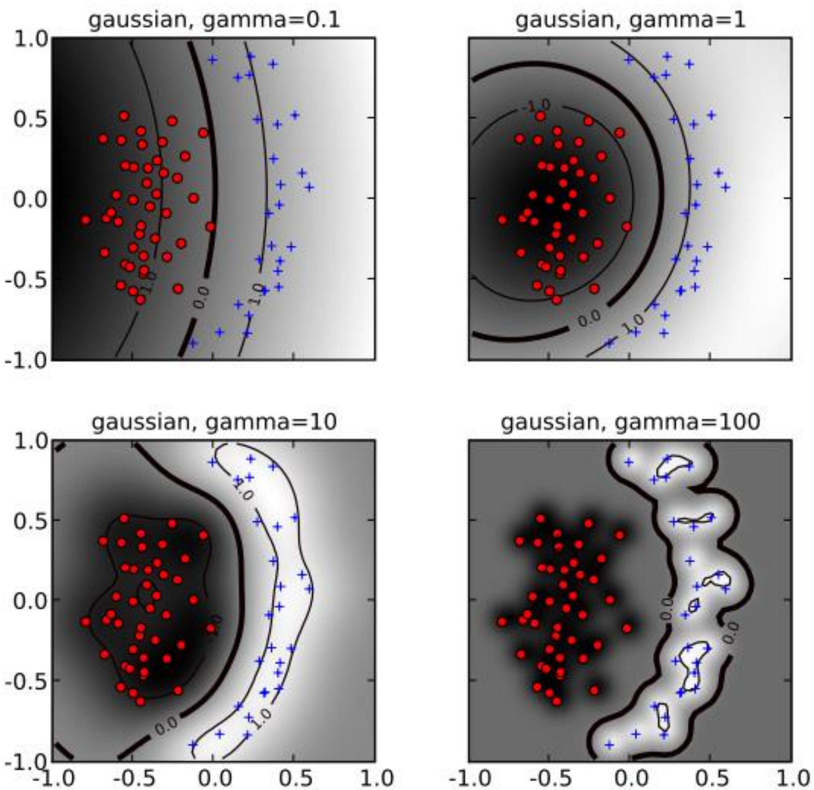
- Margins and SVs with different cost values (RBF kernel)

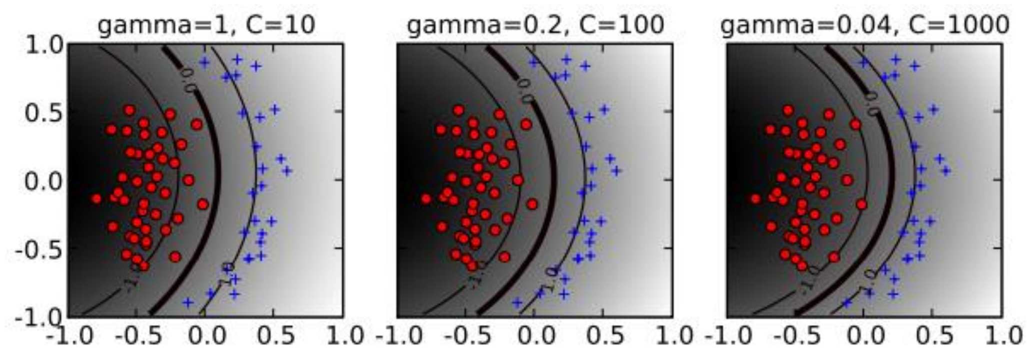



http://pubs.rsc.org/-/content/articlehtml/2010/an/b918972f

# Effects of Different Kernels

- Kernel width (sigma) for RBF Kernel
  - ✓ The smaller the sigma, the more complicated decision boundary is generated
  - ✓ In many libraries, gamma ($=1/\text{sigma}^2$) is commonly used



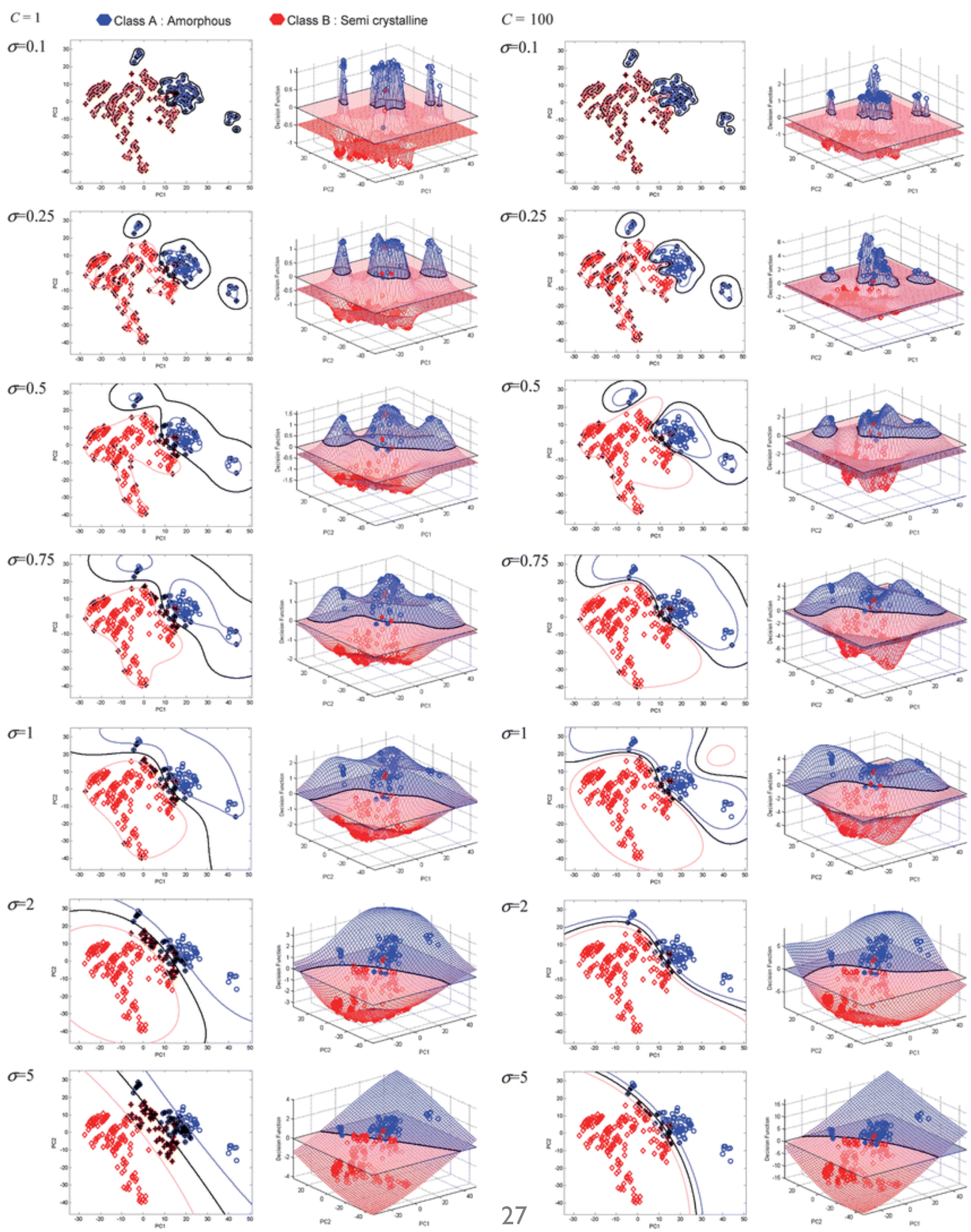




KOREA UNIVERSITY



DSBA Data Science & Business Analytics

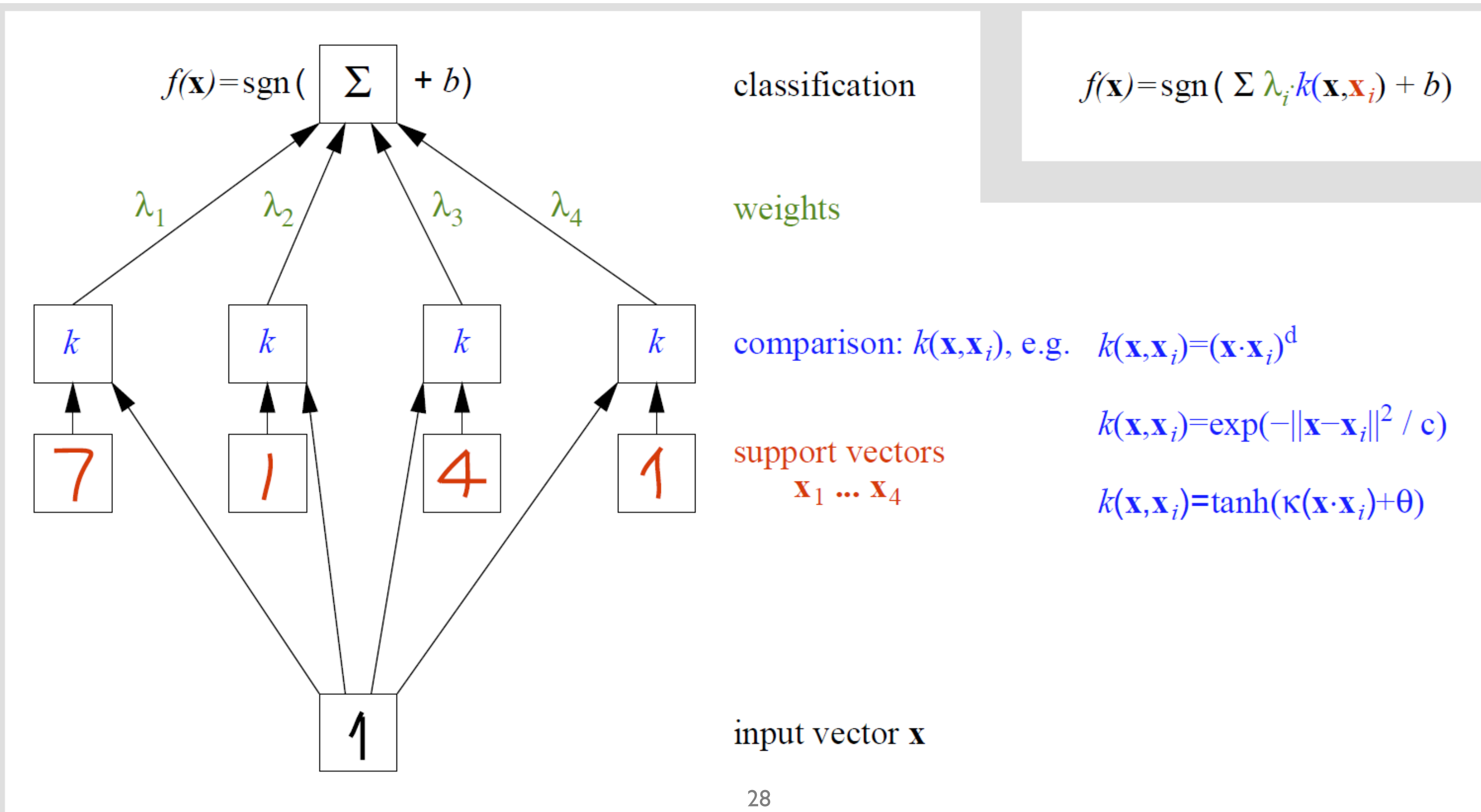
# Effects of Kernels and Misclassification Cost



# SVM Architecture

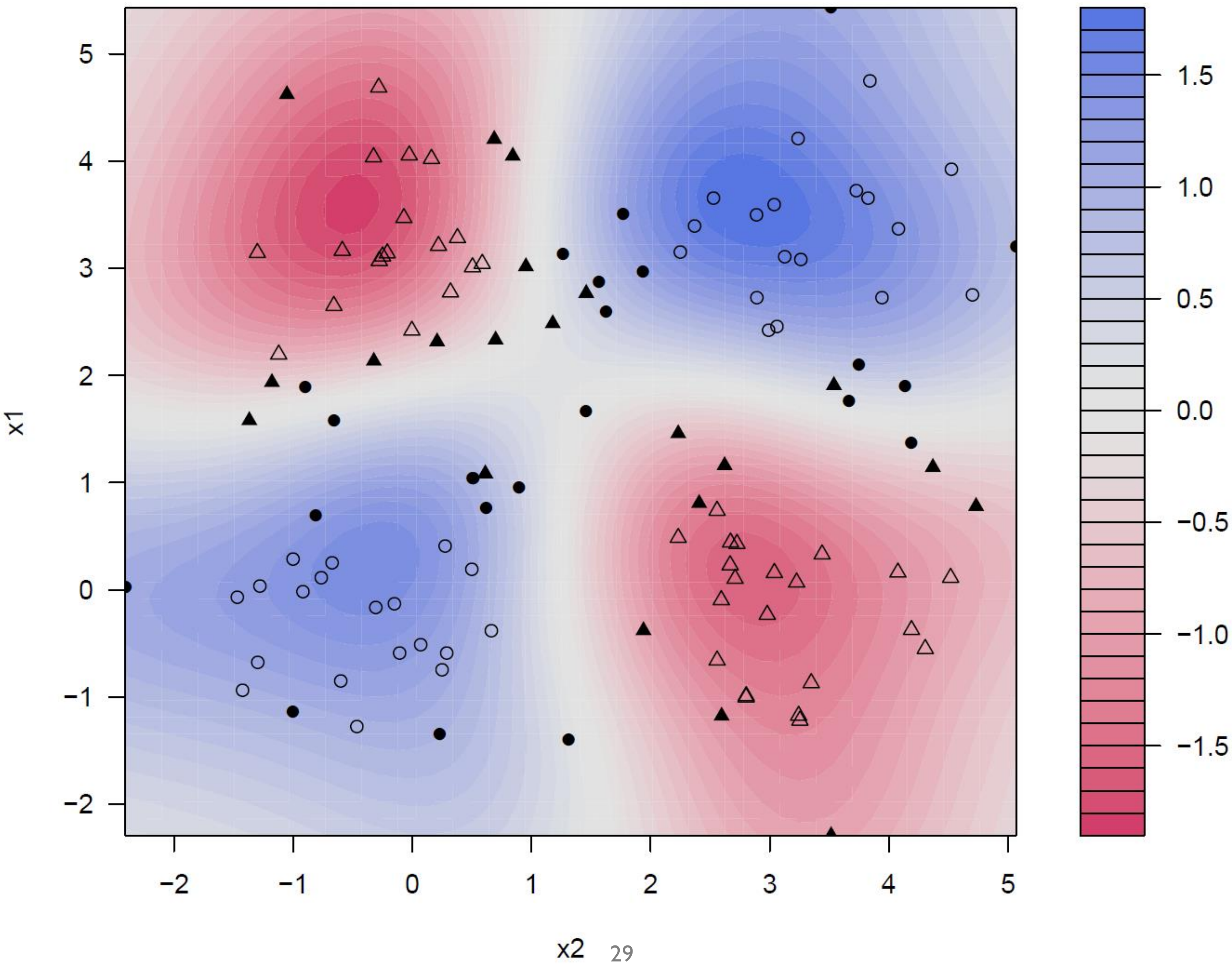
- Class decision for a new instance **x**

$$f(\mathbf{x}) = sign\left(\sum_{i=1}^{N} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b\right)$$



# SVM Decision Boundary & Support Vectors

- XOR Problem

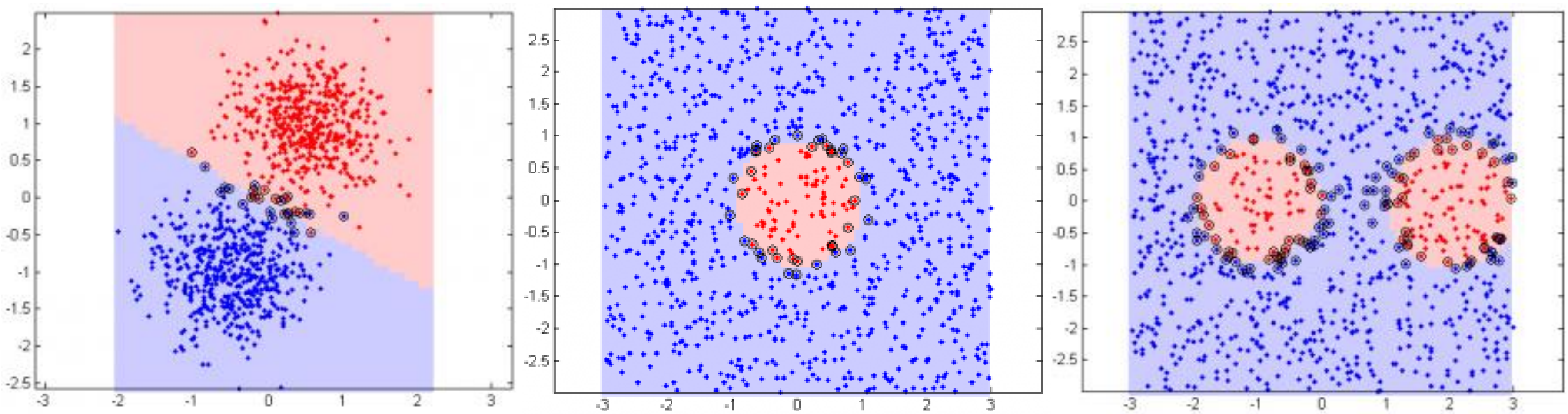



고려대학교 KOREA UNIVERSITY

DSBA Data Science & Business Analytics

# SVM Decision Boundary & Support Vectors

- Other examples




http://www.alivelearn.net/?p=912


Q & A

# References

Research Papers


- Burges, C.J.C. (1998). A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery 2: 121-167.
- Müller, K., Mika, S., Rätsch, G., Tsuda, K., and Schölkopf, B. (2001). An introduction to kernel-based learning algorithms. IEEE Transactions on Neural Networks 12(2): 181-201.