

$$\Phi \left(\begin{array}{c} \text{(Image of a puppy)} \\ \text{(Image of a dog)} \end{array} \right) = \begin{array}{c} \text{(Image of a puppy)} \\ \text{(Image of a dog)} \end{array} \quad \Phi \left(\begin{array}{c} \text{(Image of a puppy)} \\ \text{(Image of a cat)} \end{array} \right) = \begin{array}{c} \text{(Image of a puppy)} \\ \text{(Image of a cat)} \end{array}$$
$$K \left(\begin{array}{c} \text{(Image of a puppy)} \\ , \end{array} \begin{array}{c} \text{(Image of a puppy)} \\ \text{(Image of a cat)} \end{array} \right) = \left(\begin{array}{c} \text{(Image of a dog)} \\ \text{(Image of a dog)} \end{array} \right) \cdot \left(\begin{array}{c} \text{(Image of a cat)} \\ \text{(Image of a cat)} \end{array} \right)$$

Lecture 2: Kernel-based Learning

Pilsung Kang

School of Industrial Management Engineering
Korea University

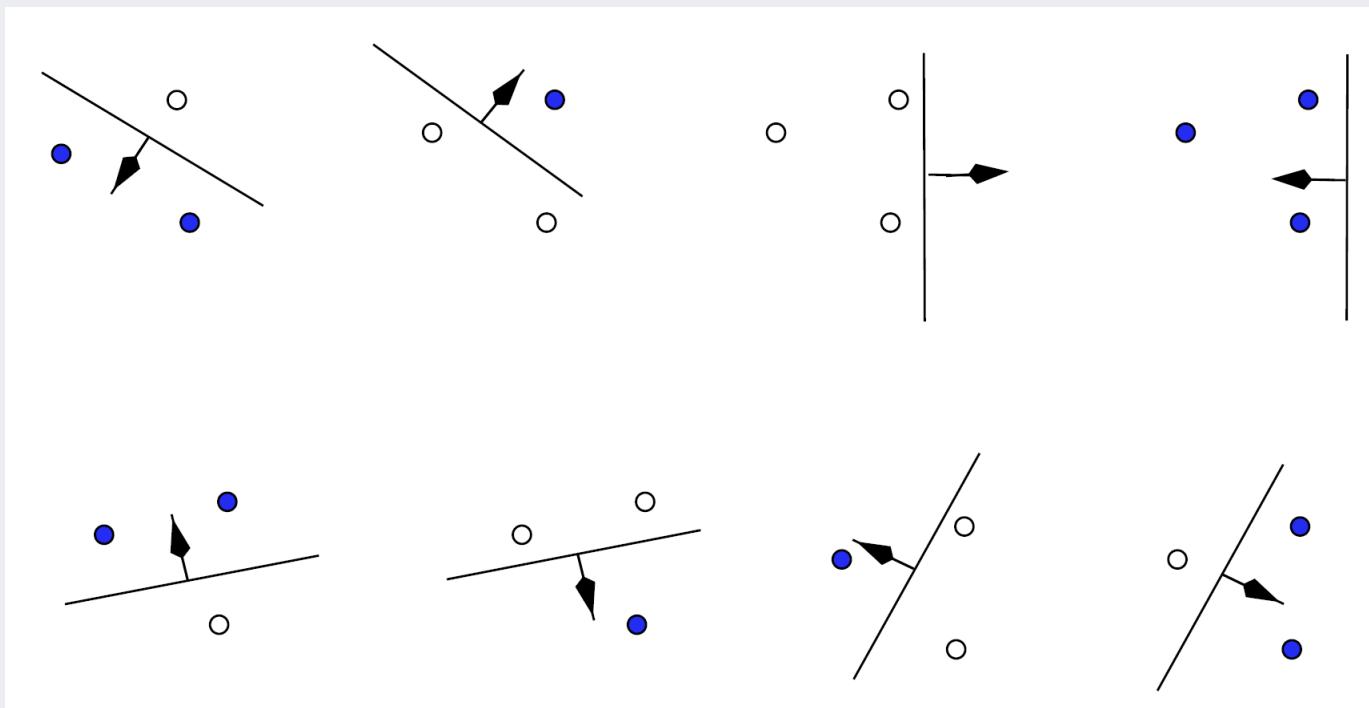
AGENDA

- 01 **Theoretical Foundations**
- 02 Support Vector Machine
- 03 Support Vector Regression
- 04 Kernel Fisher Discriminant Analysis
- 05 Kernel Principal Component Analysis

The Concept of Shatter

- Shatter

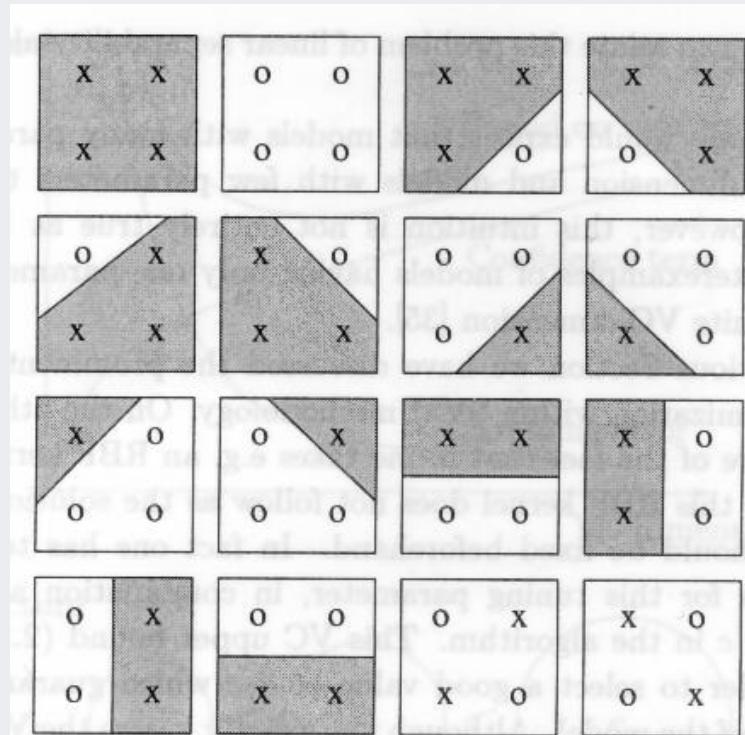
- ✓ A set of points is said to be shattered by a class of functions if, no matter how we assign a binary label to each points, a member of the class can perfectly separate them.
 - A linear classifier can shatter $(n+1)$ instances in n -dimensional space



The Concept of Shatter

- Shatter

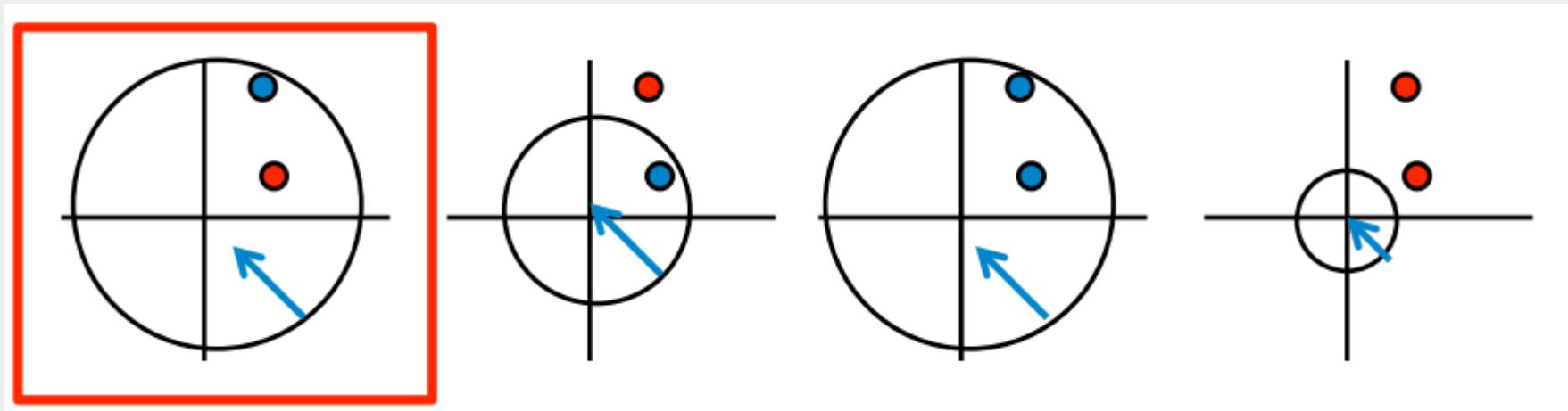
- ✓ A set of points is said to be shattered by a class of functions if, no matter how we assign a binary label to each points, a member of the class can perfectly separate them.
 - Can a linear classifier shatter $(n+2)$ instances in n -dimensional space?



The Concept of Shatter

- **Shatter**

- ✓ A set of points is said to be shattered by a class of functions if, no matter how we assign a binary label to each points, a member of the class can perfectly separate them.
 - A circle cannot shatter 2 instances in 2-dimensional space

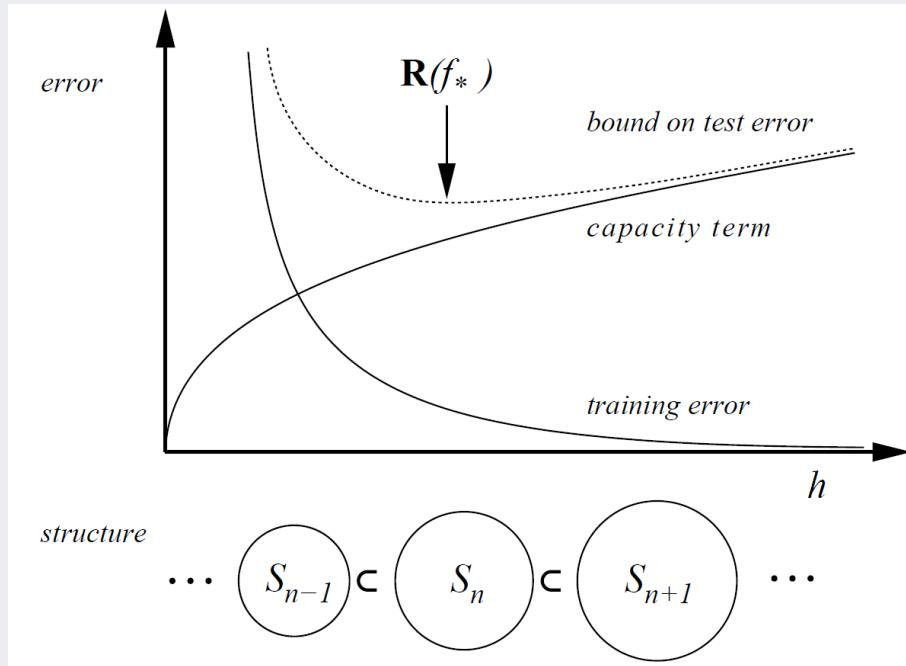


Vapnik-Chervonekis (VC) Dimension

- VC dimension
 - ✓ Measures the capacity of a hypothesis space
 - ✓ Capacity is a measure of complexity and measures the expressive power, richness or flexibility of a set of functions by assessing how wiggly its members can be
 - ✓ The maximum number of points that can be shattered by H is called **VC dimension**

ERM vs. SRM

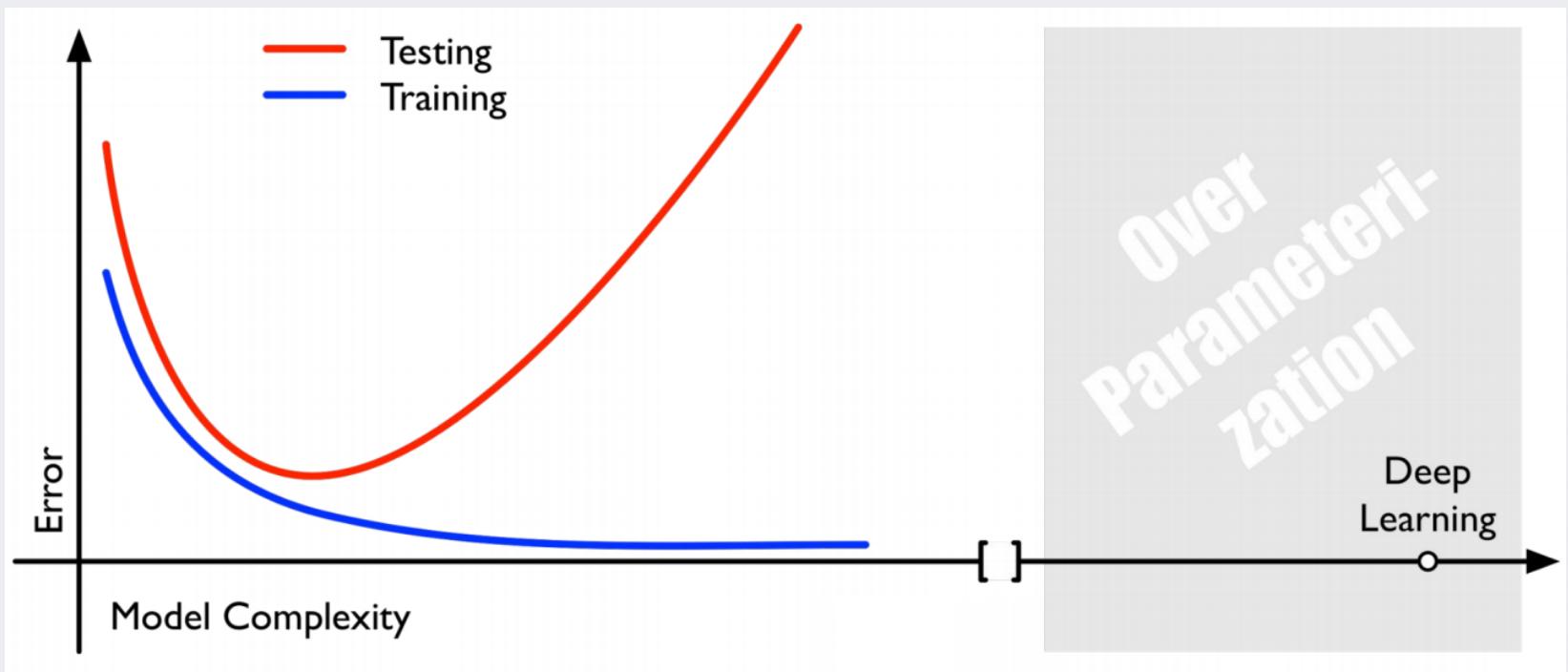
- Structural Risk Minimization (SRM)



- ✓ An inductive principle for model selection used for learning from finite training data
- ✓ Describe a general model of capacity control and provides a trade-off between **hypothesis space complexity (VC dim.)** and the **quality of fitting the training data (empirical error)**

ERM vs. SRM

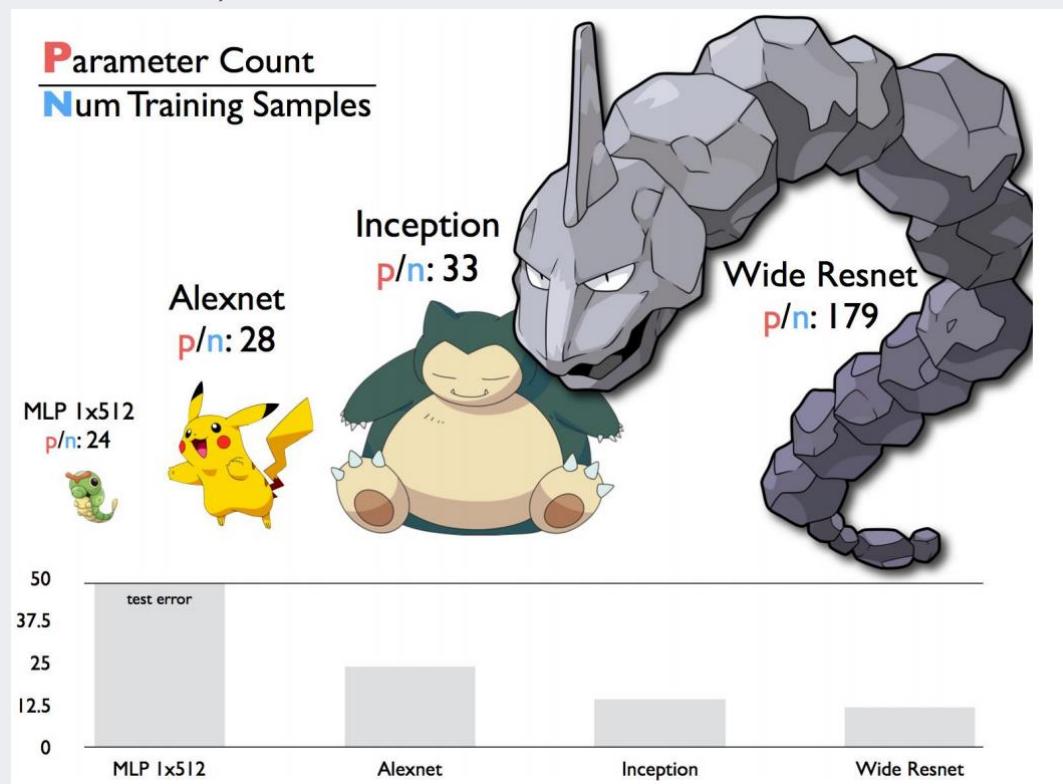
- Structural Risk Minimization (SRM)
 - ✓ Trade-off between **hypothesis space complexity (VC dim.)** and the **quality of fitting the training data (empirical error)**
 - ✓ Understanding Deep Learning Requires Rethinking Generalization (Zhang et al 2017a, 2017b, 2017c, Park 2017)



ERM vs. SRM

- Structural Risk Minimization (SRM)

- ✓ Trade-off between **hypothesis space complexity (VC dim.)** and the **quality of fitting the training data (empirical error)**
- ✓ Understanding Deep Learning Requires Rethinking Generalization (Zhang et al 2017a, 2017b, 2017c, Park 2017)



ERM vs. SRM

- Structural Risk Minimization

Let h denote the VC dimension of the function class F and let R_{emp} be defined as follows using the 0/1 loss.

$$R_{emp}[f] = \frac{1}{n} \sum_{i=1}^n L(f(x_i), y_i)$$

For all $\delta > 0$ and $f \in F$ the inequality bounding the risk below holds with probability of at least $1 - \delta$ for $n > h$.

$$R[f] \leq R_{emp}[f] + \sqrt{\frac{h \left(\ln \frac{2n}{h} + 1 \right) - \ln \left(\frac{\delta}{4} \right)}{n}}$$

Kernel Machine Kids vs. Deep Learning Kids

- Fantastic Four in Deep Learning

[Yann Lecun](#)

Facebook AI Director
Professor @NYU

[Geoffrey Hinton](#)

Fellow @ Google
Professor @U of Toronto

[Yoshua Bengio](#)

Head of MILA

[Andrew Ng](#)

Chief Scientist @Baidu
Co-Founder of Coursera
Professor @Stanford Univ.



Kernel Machine Kids vs. Deep Learning Kids

- The Three Musketeers in Kernel Machine

Yann Lecun

Professor @RHUL

Book: Statistical Learning Theory

Bernhard Scholkopf

Director of Intelligent Systems

@Max Planck Institute

Book: Learning with Kernels

Alexander J. Smola

Professor @CMU

Book: Advances in Kernel Methods



BIG DATA

Facebook's AI team hires Vladimir Vapnik, father of the popular support vector machine algorithm

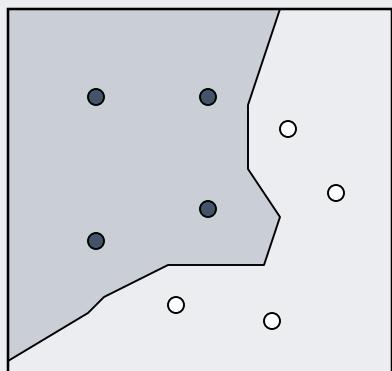
JORDAN NOVET @JORDANNOVET NOVEMBER 25, 2014 1:23 PM

AGENDA

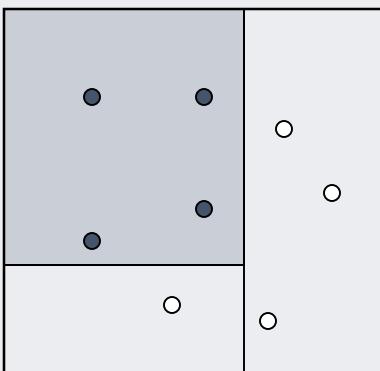
- 01 Theoretical Foundations
- 02 Support Vector Machine
- 03 Support Vector Regression
- 04 Kernel Fisher Discriminant Analysis
- 05 Kernel Principal Component Analysis

Discriminant Function

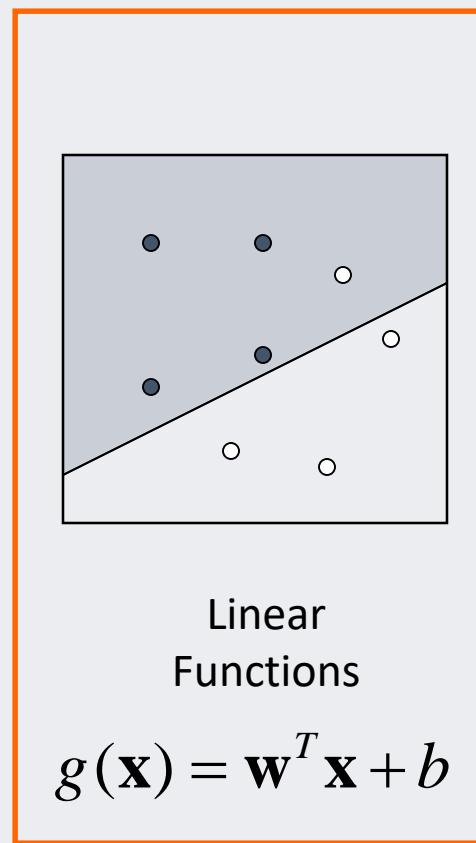
- Discriminant functions in classification



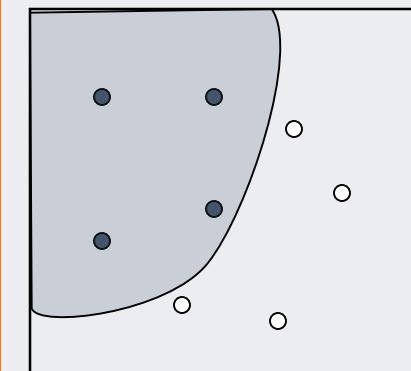
Nearest
Neighbor



Decision
Tree



Linear
Functions



Nonlinear
Functions

Linear Classification

- **Binary Classification Problem**

✓ **Training data:** sample drawn i.i.d. from set $X \in R^d$ according to some distribution D

$$S = \left((x_1, y_1), \dots, (x_n, y_n) \right) \in X \times \{-1, +1\}$$

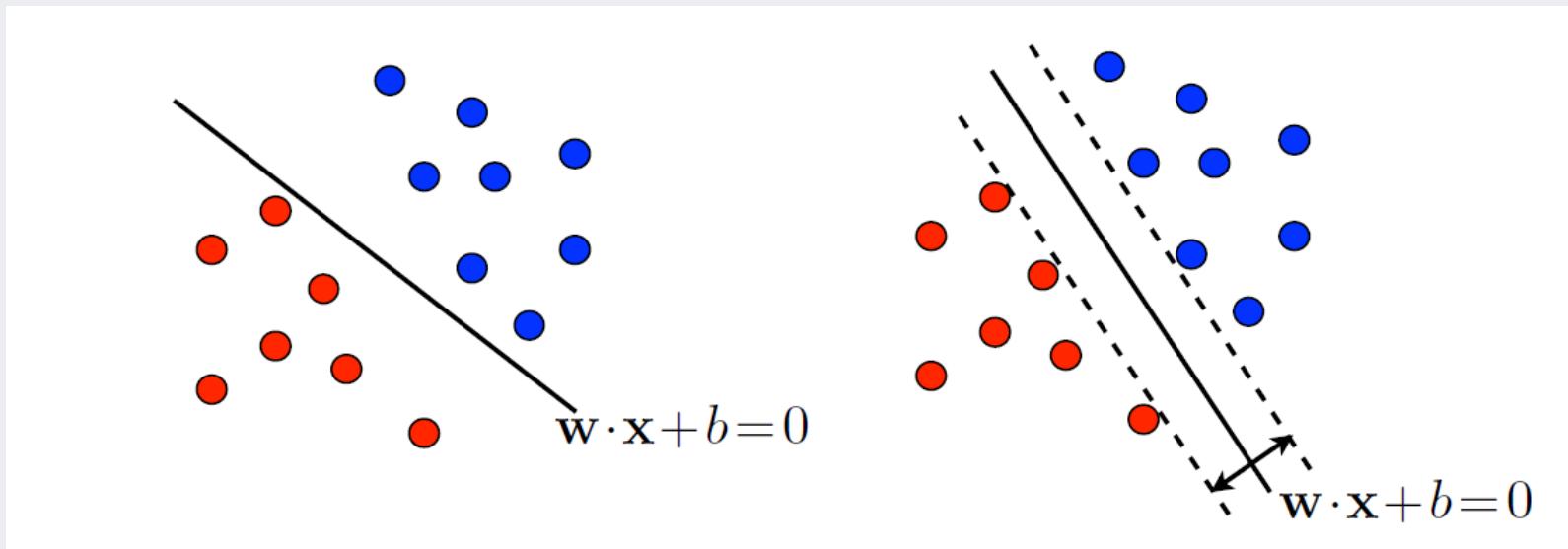
✓ **Problem:** find hypothesis $h : X \rightarrow \{-1, +1\}$ in H (classifier) with small generalization error $R_D(h)$

✓ **Linear classifier**

- Hypothesis based on hyperplanes
- Linear separation in high-dimensional data

Linear Classification

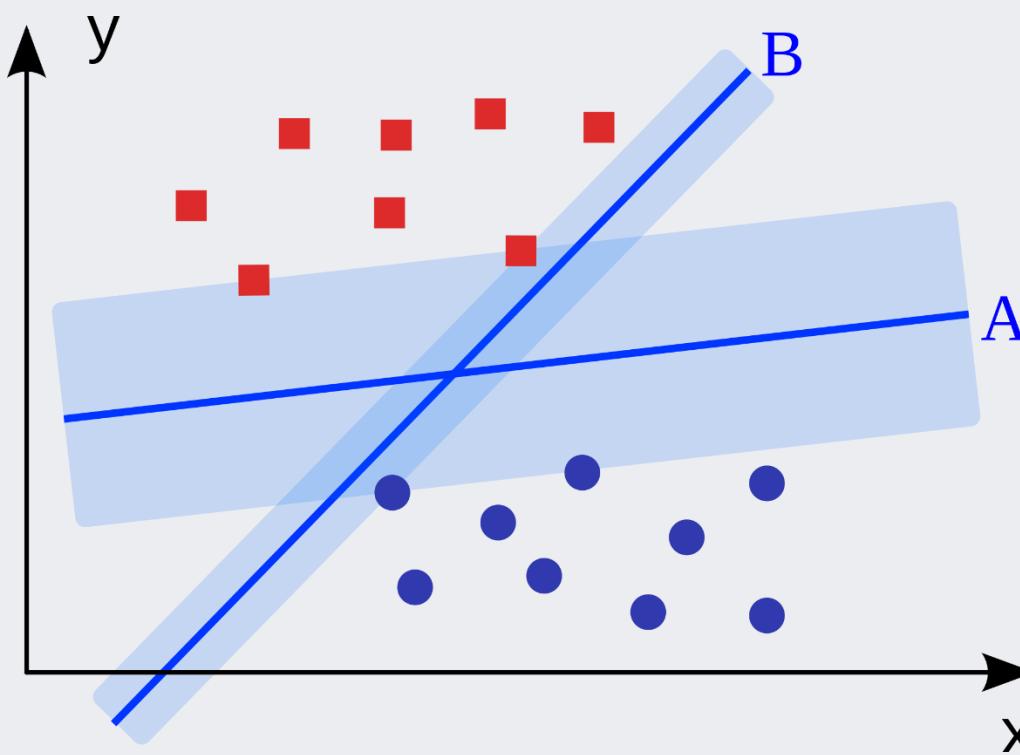
- Binary classification problem



$$H = \{\mathbf{x} \rightarrow \text{sign}(\mathbf{w} \cdot \mathbf{x} + b) : \mathbf{w} \in R^d, b \in R\}$$

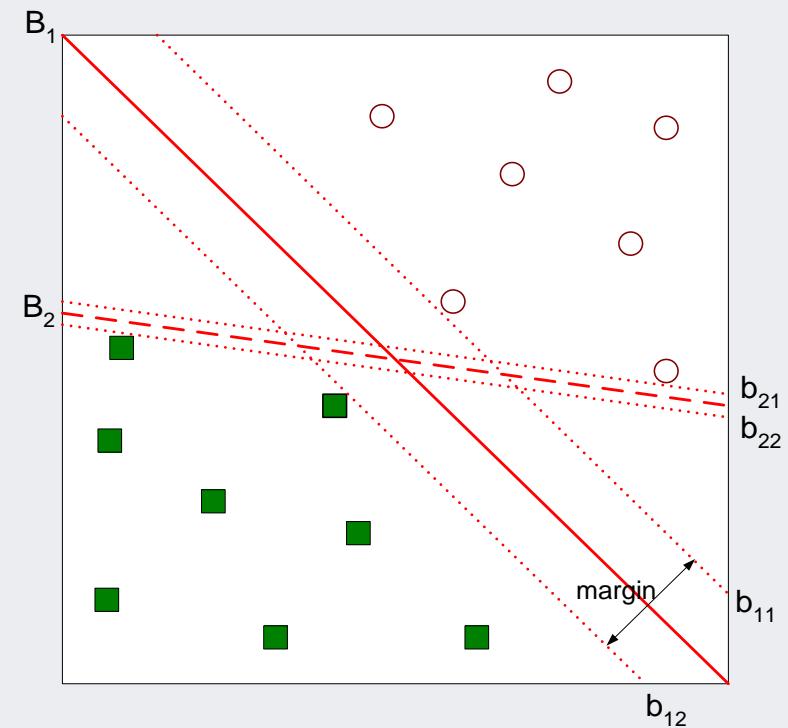
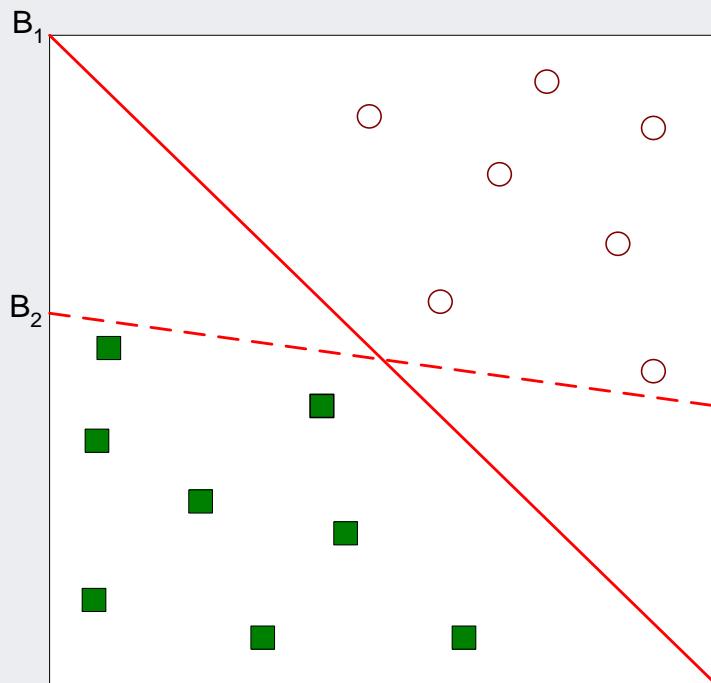
Local Optimum? Global Optimum!

- Which classification boundary is better?
 - ✓ How do you define “better”?



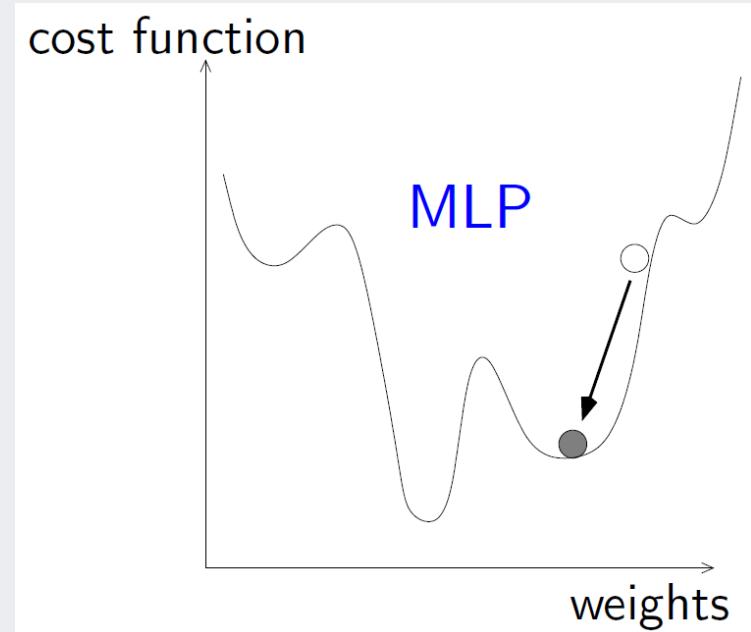
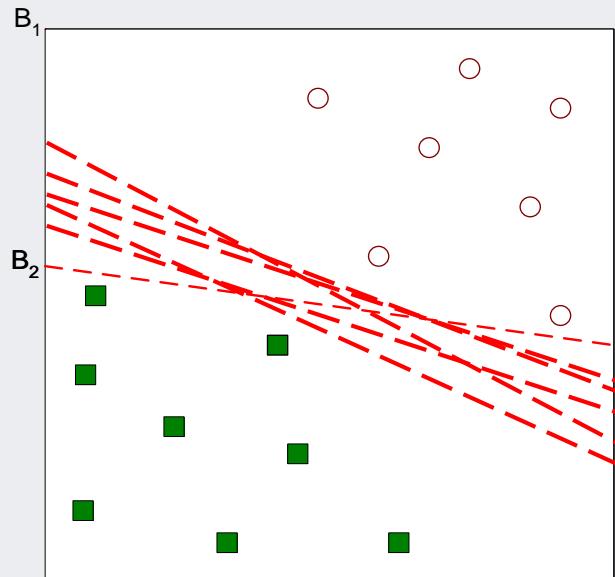
Local Optimum? Global Optimum!

- Which classification boundary is better?
 - ✓ Find the hyperplane that **maximizes the margin!**



Local Optimum? Global Optimum!

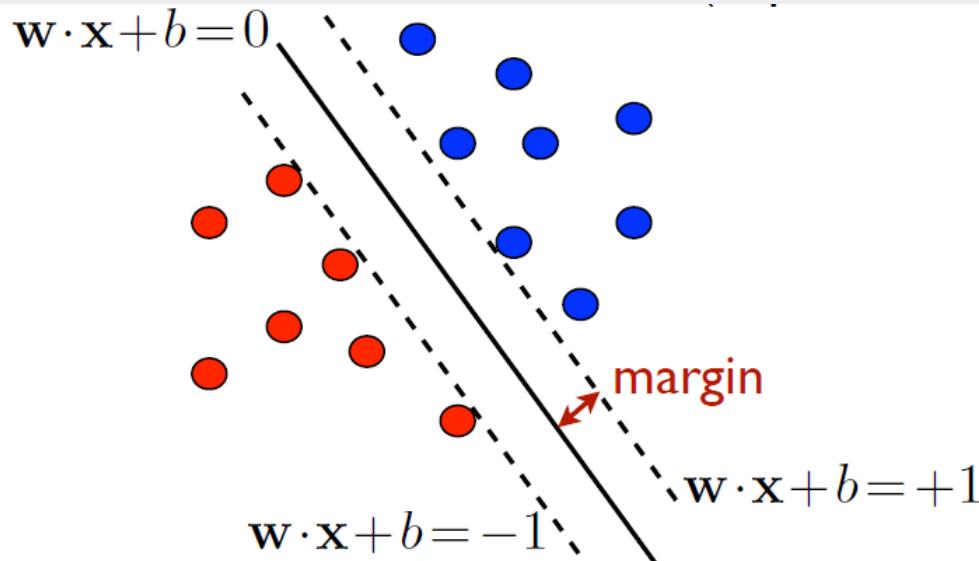
- Artificial neural network (ANN)
 - ✓ Universal approximation of continuous nonlinear functions
 - ✓ Learning from input-output patterns
 - ✓ Parallel network architecture, multiple inputs and outputs
 - ✓ But, existence of many local optimum!



Support Vector Machine: Formulation

Burges (1998)

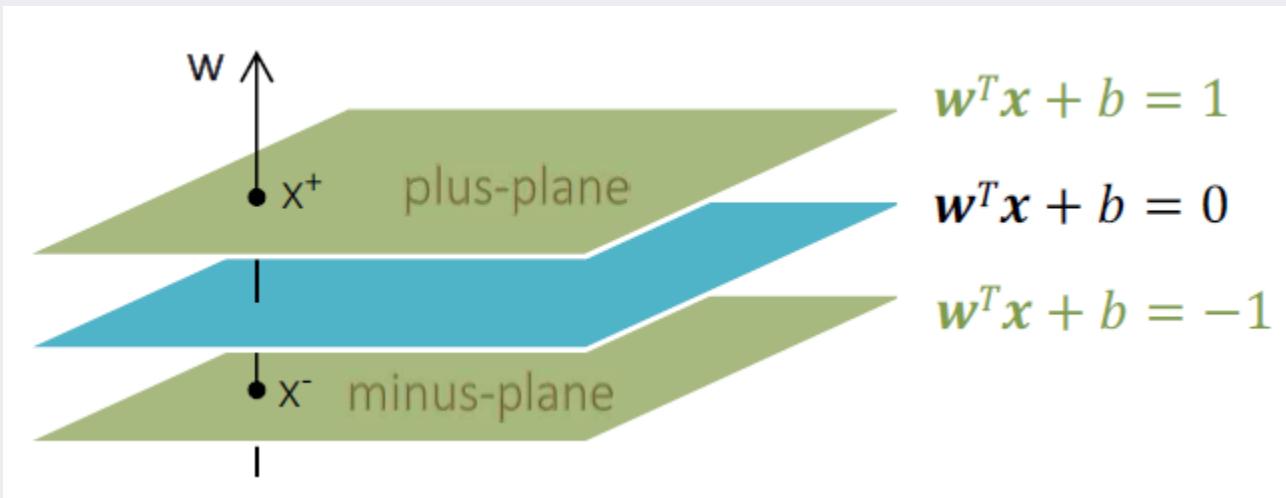
- Optimal hyperplane: Maximize the margin



- **Canonical hyperplane:** w and b chosen such that for closest points $|w \cdot x + b| = 1$.

Support Vector Machine: Formulation

- How to compute the margin?



$$\text{Margin} = \frac{2}{|w|}$$

Margin and VC Dimension

- Relationship between margin and VC dimension

The VC dimension of a separating hyperplane with a margin Δ is bounded as follows

$$h \leq \min\left(\left\lceil \frac{R^2}{\Delta^2} \right\rceil, D\right) + 1$$

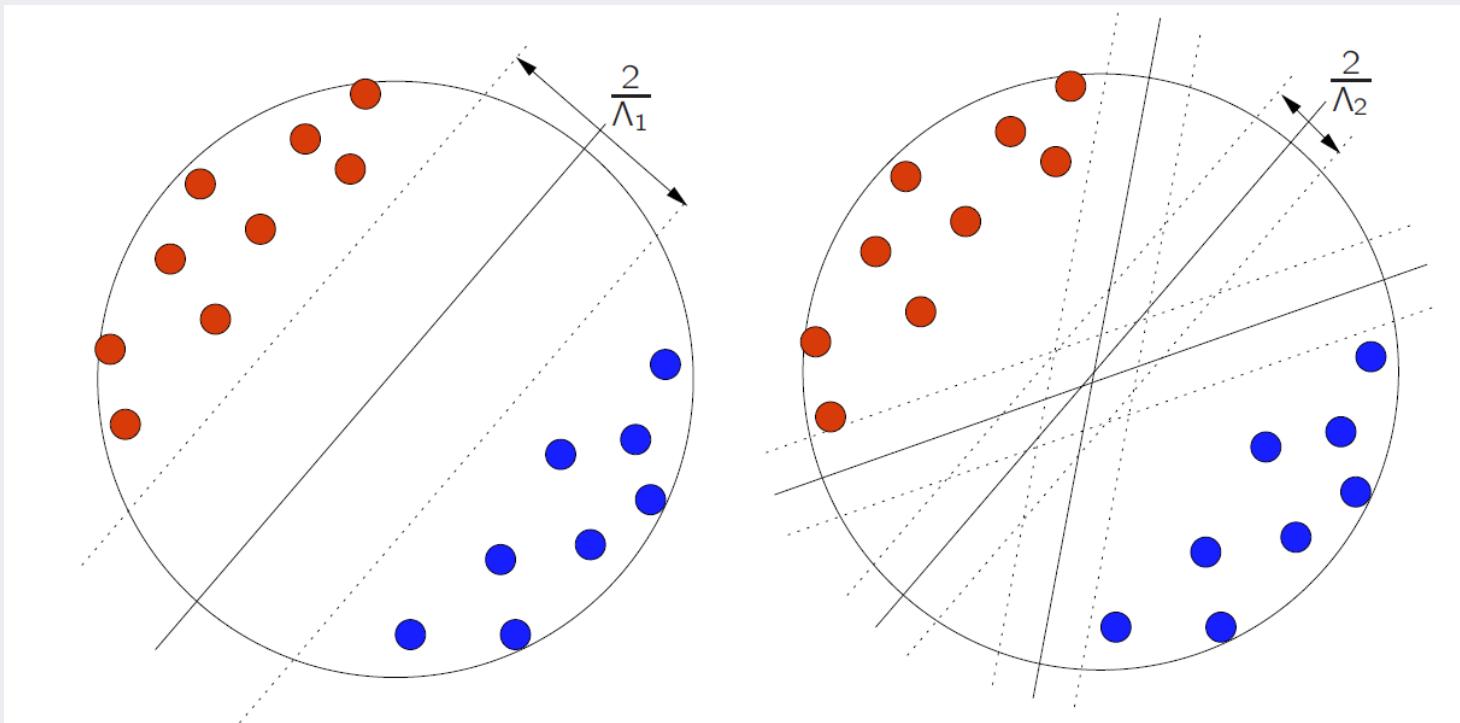
where D is the dimensionality of the input space, and R is the radius of the smallest sphere containing all the input vectors

- Maximizing the margin \rightarrow Minimizing the VC dimension \rightarrow Minimizing the Expected Risk

$$R[f] \leq R_{emp}[f] + \sqrt{\frac{h \left(\ln \frac{2n}{h} + 1 \right) - \ln \left(\frac{\delta}{4} \right)}{n}}$$

Margin and VC Dimension

- An illustrative example



- ✓ If we choose a hyperplane with a large margin (left), there is only a small number of possibilities to separate the data → lower VC dimension

Support Vector Machine: Cases

- Support Vector Machine Formulation

	Hard margin?	Soft margin?
Linearly separable?	Basic form (Case 1)	Introduce penalty terms (Case 2)
Linearly non-separable?	Utilize Kernel Trick	Introduce penalty terms Utilize Kernel Trick (Case 3)

SVM Case I: Linear Case & Hard Margin

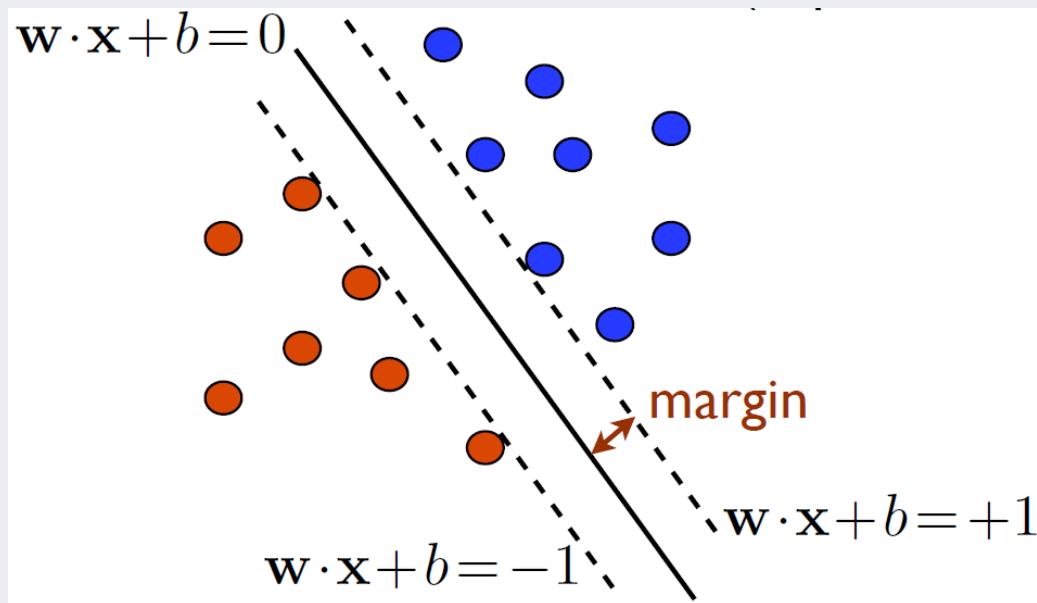
- Optimization Problem

- ✓ Objective function

$$\min \frac{1}{2} \|\mathbf{w}\|^2$$

- ✓ Constraints

$$s.t. \quad y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1$$



SVM Case I: Linear Case & Hard Margin

- Optimization Problem

- ✓ Lagrangian Problem

$$\begin{aligned} \min \quad L_P(\mathbf{w}, b, \alpha_i) &= \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^N \alpha_i (y_i (\mathbf{w}^\top \mathbf{x}_i + b) - 1) \\ \text{s.t.} \quad \alpha_i &\geq 0 \end{aligned}$$

- ✓ KKT conditions

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

SVM Case I: Linear Case & Hard Margin

- From Primal to Dual

$$\begin{aligned} \min \quad L_P(\mathbf{w}, b, \alpha_i) &= \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^N \alpha_i (y_i (\mathbf{w}^\top \mathbf{x}_i + b) - 1) \\ \text{s.t.} \quad \alpha_i &\geq 0 \end{aligned}$$

$$\begin{aligned} \max \quad L_D(\alpha_i) &= \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j \\ \text{s.t.} \quad \sum_{i=1}^N \alpha_i y_i &= 0 \quad \text{and} \quad \alpha_i \geq 0 \end{aligned}$$

- Solution

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

SVM Case I: Linear Case & Hard Margin

- From KKT condition, we know that

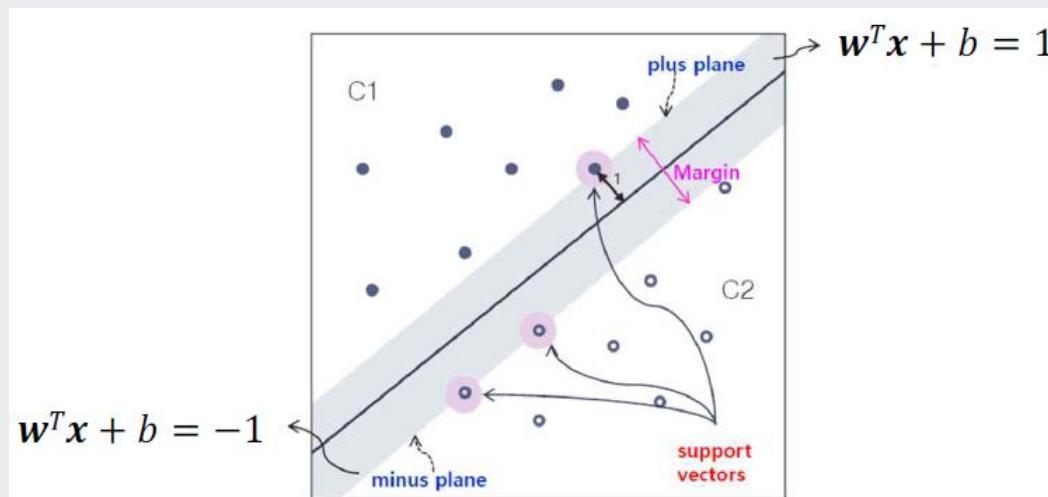
$$\alpha_i(y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1) = 0$$

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

✓ The solution has the form

$$\mathbf{w} = \sum_{i=1}^N \alpha_i y_i \mathbf{x}_i = \sum_{i \in SV} \alpha_i y_i \mathbf{x}_i$$

✓ b can be computed by $y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 = 0$ with a support vector \mathbf{x}_i



SVM Case I: Linear Case & Hard Margin

- Compute the margin

✓ Since the SVs lie on the marginal hyperplanes, for any support vector \mathbf{x}_i , $\mathbf{w}^T \mathbf{x}_i + b = y_i$

$$b = y_i - \sum_{i=1}^N \alpha_i y_i (\mathbf{x}_j, \mathbf{x}_i)$$

✓ Multiplying both sides by $\alpha_i y_i$ and taking the sum leads to

$$\sum_{i=1}^N \alpha_i y_i b = \sum_{i=1}^N \alpha_i y_i^2 - \sum_{i,j=1}^N \alpha_i \alpha_j y_i y_j (\mathbf{x}_i, \mathbf{x}_j)$$

✓ Using the fact that $y_i^2 = 1$

$$0 = \sum_{i=1}^N \alpha_i - \mathbf{w}^T \mathbf{w}$$

$$\rho^2 = \frac{1}{\|\mathbf{w}\|_2^2} = \frac{1}{\sum_{i=1}^N \alpha_i} = \frac{1}{\|\boldsymbol{\alpha}\|_1}$$

SVM Case 2: Linear Case & Soft Margin

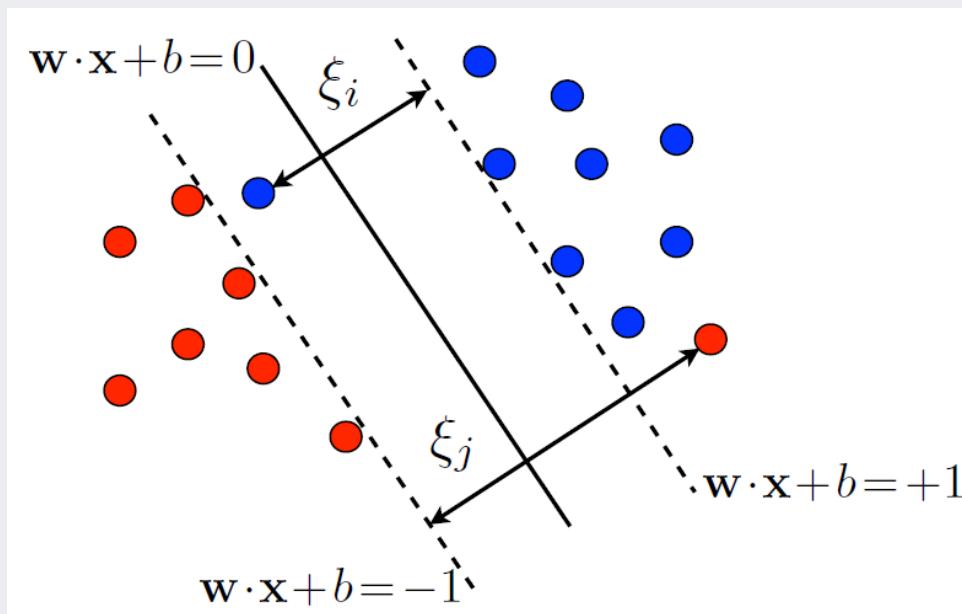
- Optimization Problem (C-SVM)

✓ Objective function

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

✓ Constraints

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



SVM Case 2: Linear Case & Soft Margin

- Optimization Problem

- ✓ Lagrangian Problem

$$\begin{aligned} \min \quad L_P(\mathbf{w}, b, \alpha_i) &= \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^N \xi_i - \sum_{i=1}^N \alpha_i (y_i (\mathbf{w}^\top \mathbf{x}_i + b) - 1 + \xi_i) - \sum_{i=1}^N \mu_i \xi_i \\ \text{s.t.} \quad \alpha_i &\geq 0 \end{aligned}$$

- ✓ KKT conditions

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

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

SVM Case 2: Linear Case & Soft Margin

- From Primal to Dual

$$\begin{aligned} \min \quad L_P(\mathbf{w}, b, \alpha_i) &= \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^N \xi_i - \sum_{i=1}^N \alpha_i (y_i (\mathbf{w}^\top \mathbf{x}_i + b) - 1 + \xi_i) - \sum_{i=1}^N \mu_i \xi_i \\ \text{s.t.} \quad \alpha_i &\geq 0 \end{aligned}$$

$$\begin{aligned} \max \quad L_D(\alpha_i) &= \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j \\ \text{s.t.} \quad \sum_{i=1}^N \alpha_i y_i &= 0 \quad \text{and} \quad 0 \leq \alpha_i \leq C \end{aligned}$$

- Solution

✓ Only requires support vectors

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

SVM Case 2: Linear Case & Soft Margin

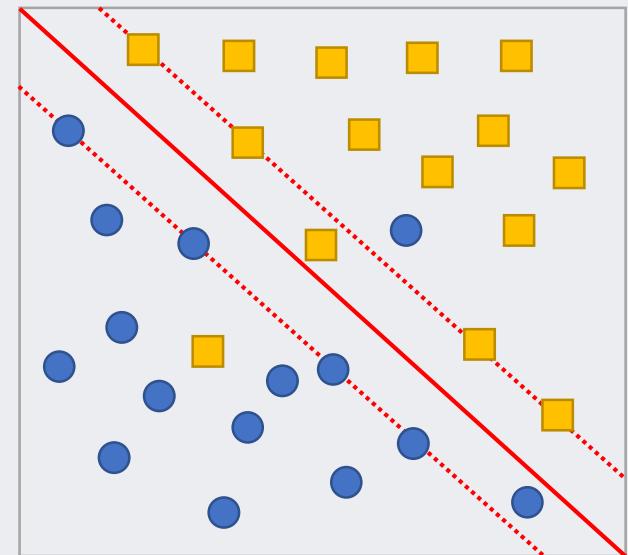
- From KKT condition, we know that

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

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

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

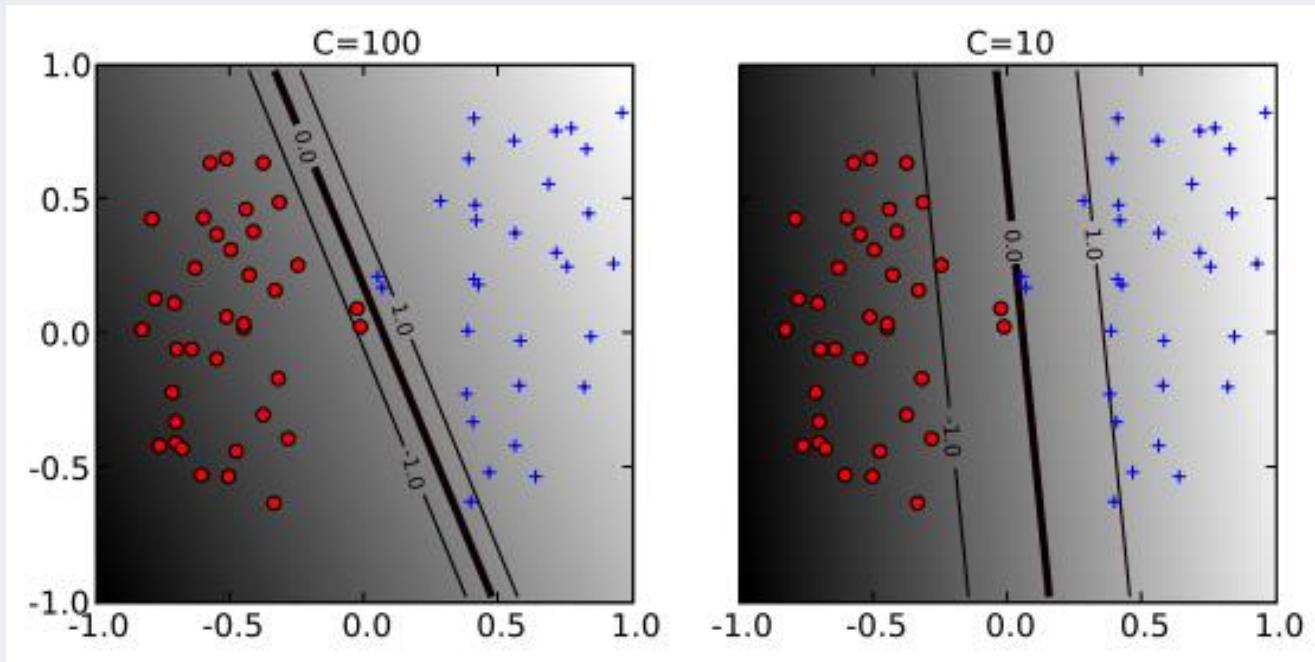
- Case 1: $\alpha_i = 0 \rightarrow$ non-support vectors
- Case 2: $0 < \alpha_i < C \rightarrow$ support vectors on the margin
- Case 3: $\alpha_i = C \rightarrow$ support vectors outside the margin



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

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

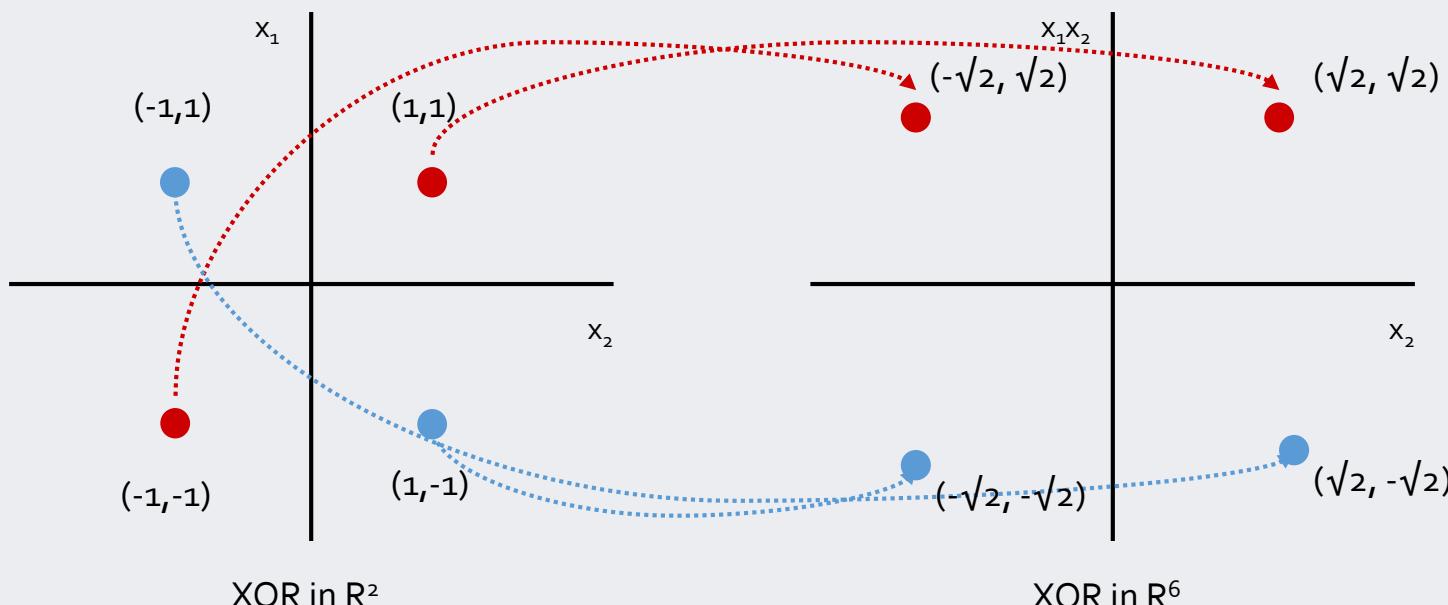


SVM Case 3: Non-linear Case & Soft Margin

- What if the decision boundary is not linear?

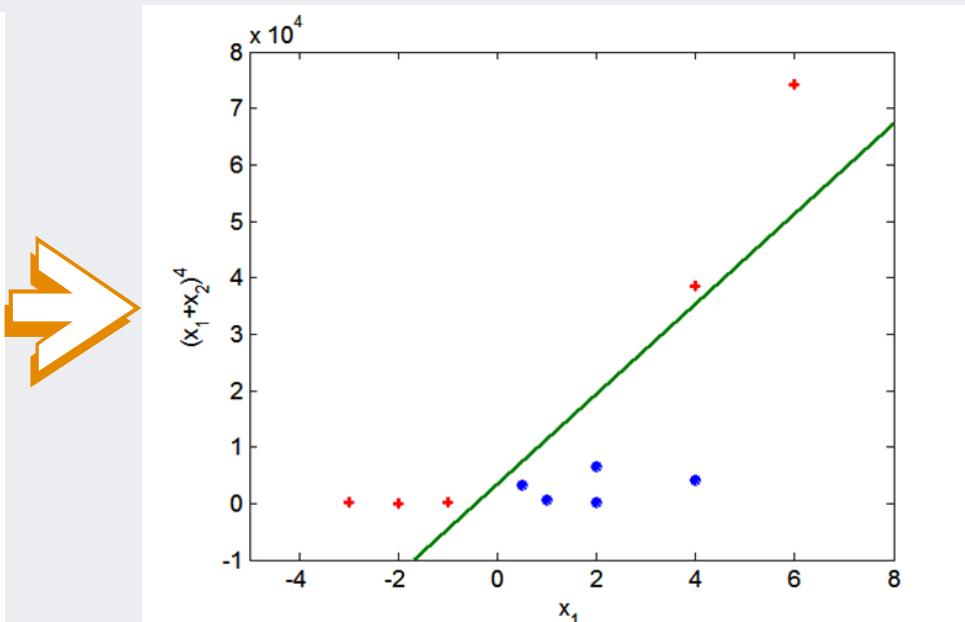
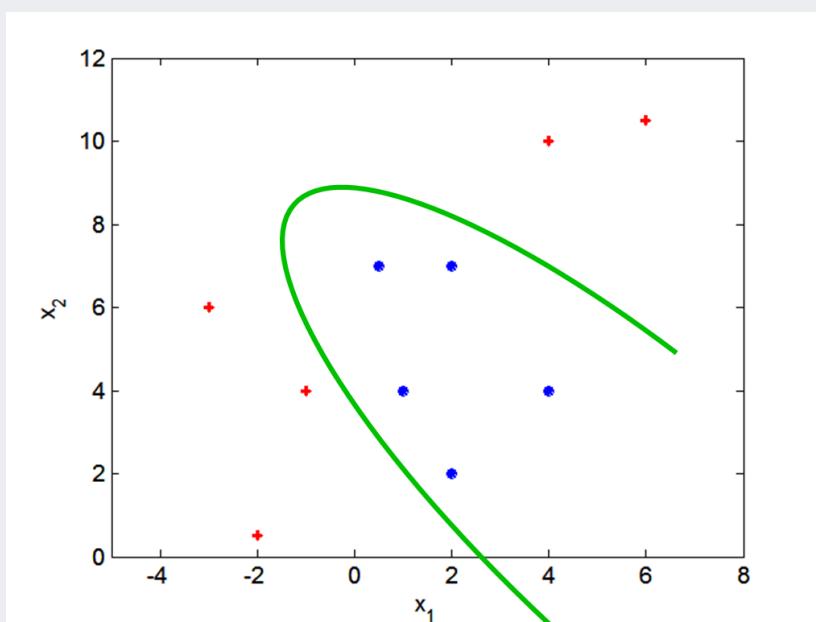
- ✓ Use a **mapping function** $\Phi(x)$ to transform an input vector from a lower dimensional space into a higher dimensional space

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



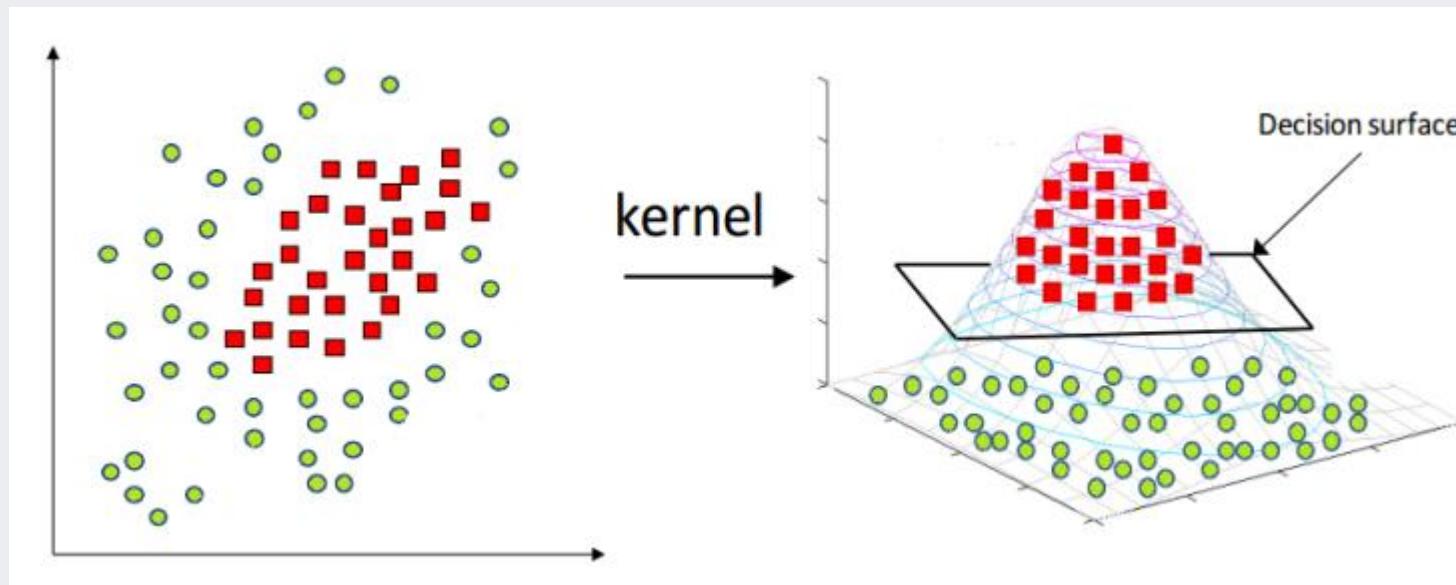
SVM Case 3: Non-linear Case & Soft Margin

- 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

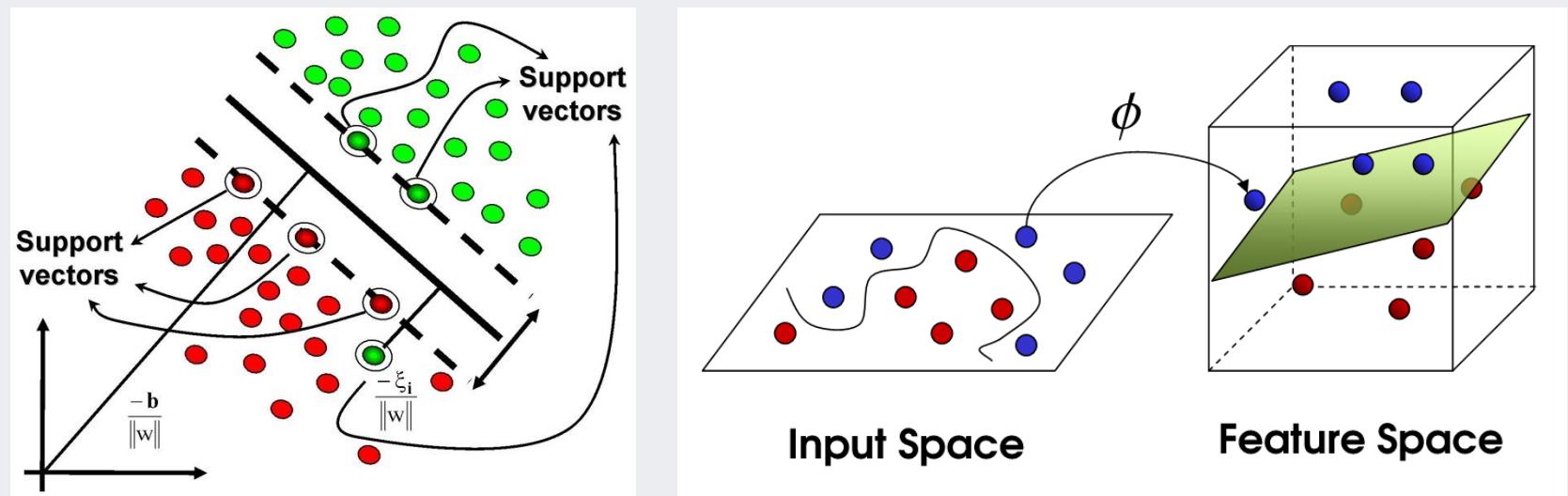
- What if the decision boundary is not linear?
 - ✓ Transform an input vector into a higher feature vector



<http://blog.hackerearth.com/simple-tutorial-svm-parameter-tuning-python-r>

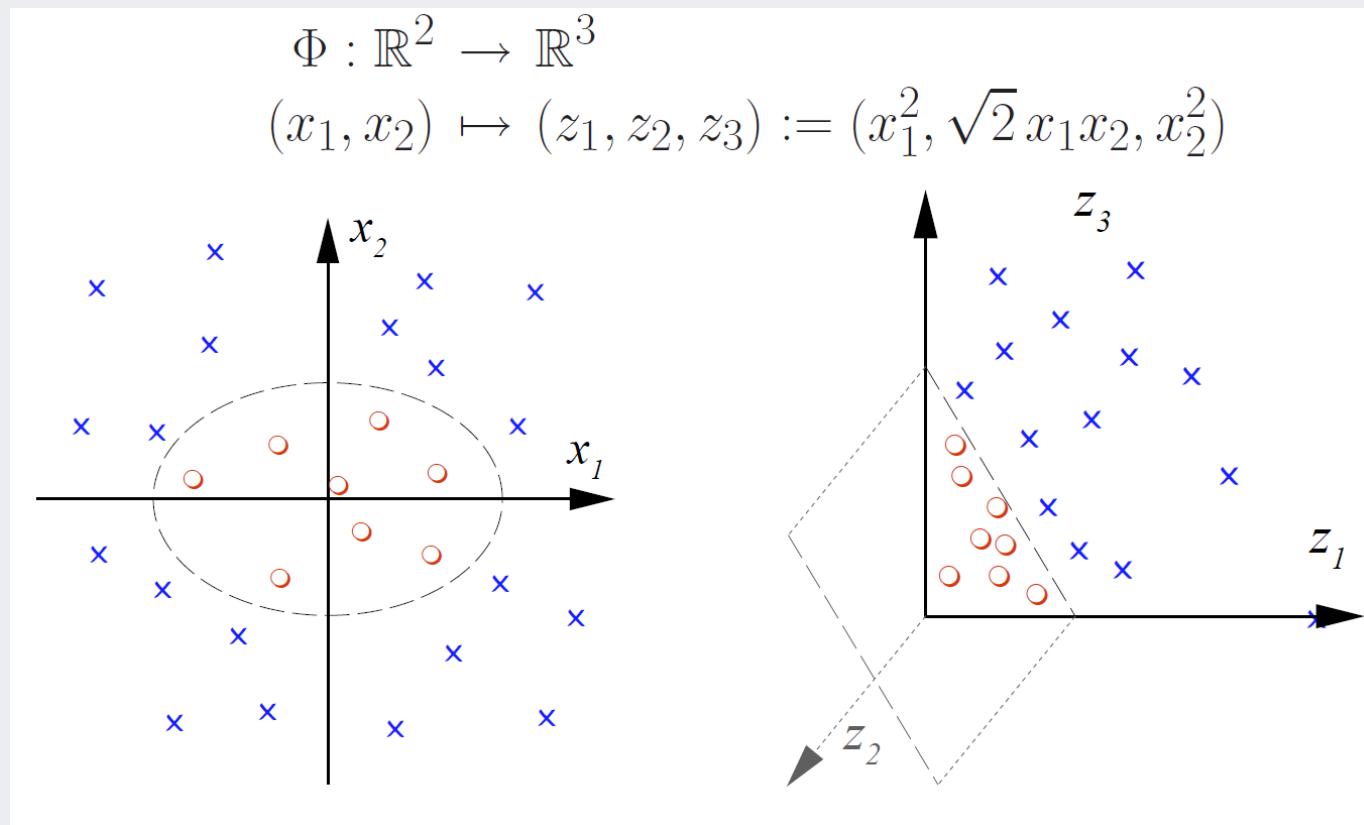
SVM Case 3: Non-linear Case & Soft Margin

- 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 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

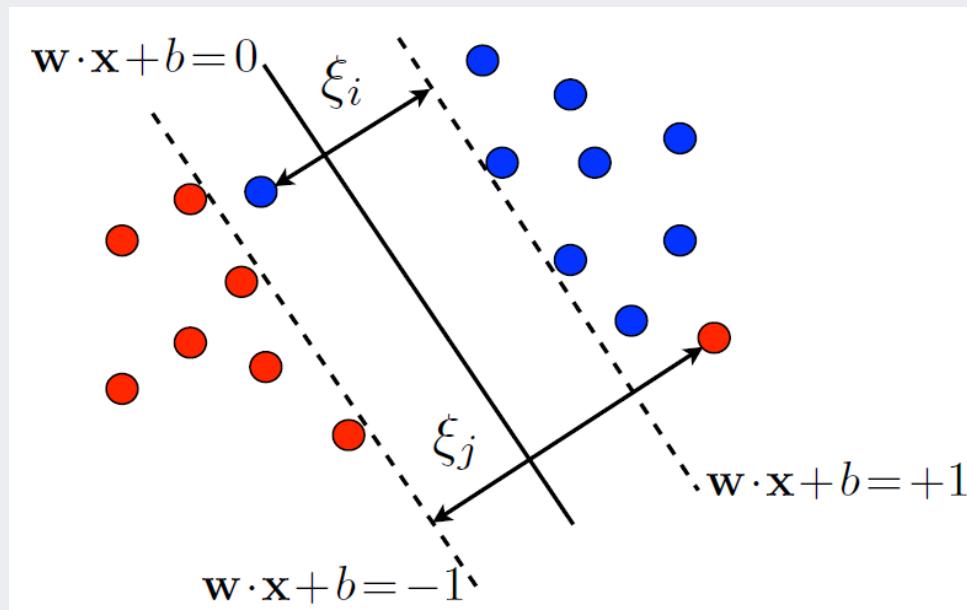
- Optimization Problem (C-SVM)

✓ Objective function

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

✓ 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

$$\begin{aligned} \min \quad L_P(\mathbf{w}, b, \alpha_i) = & \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \xi_i \\ & - \sum_{i=1}^n \alpha_i (y_i (\mathbf{w}^T \Phi(\mathbf{x}_i) + b) - 1 + \xi_i) - \sum_{i=1}^n \mu_i \xi_i \end{aligned}$$

✓ KKT condition

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

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

SVM Case 3: Non-linear Case & Soft Margin

✓ Lagrangian Problem (Primal)

$$\begin{aligned} \min \quad L_P(\mathbf{w}, b, \alpha_i) = & \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \xi_i \\ & - \sum_{i=1}^n \alpha_i (y_i (\mathbf{w}^T \Phi(\mathbf{x}_i) + b) - 1 + \xi_i) - \sum_{i=1}^n \mu_i \xi_i \end{aligned}$$

✓ Lagrangian Problem (Dual)

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

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

SVM Case 3: Non-linear Case & Soft Margin

- How to find a mapping function Φ ?

✓ Introduce a **Kernel Trick**

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

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

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

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

Kernel Trick

- Generalized inner product

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

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

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

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

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

Kernel Trick

- Kernel Trick

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

Example: Consider $K(\mathbf{x}, \mathbf{x}') = (1 + \mathbf{x}^\top \mathbf{x}')^2 = (1 + x_1x'_1 + x_2x'_2)^2$

$$= 1 + x_1^2 x'^2_1 + x_2^2 x'^2_2 + 2x_1x'_1 + 2x_2x'_2 + 2x_1x'_1 x_2x'_2$$

This is an inner product!

$$(1, x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2)$$

$$(1, x'^2_1, x'^2_2, \sqrt{2}x'_1, \sqrt{2}x'_2, \sqrt{2}x'_1x'_2)$$

Kernel Trick

- **Polynomial Kernel**

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

The “equivalent” kernel $K(\mathbf{x}, \mathbf{x}') = (1 + \mathbf{x}^\top \mathbf{x}')^Q$

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

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

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

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

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

Taylor expansion of exponential function

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

Kernel Trick

- Idea

- ✓ Define $K: X \times X \rightarrow R$, 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 $K: X \times X \rightarrow R$ which is square integrable (L_2 -space) in $X \times X$ and which satisfies

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

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

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

- **Interpretation**

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

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

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 $\mathbf{x}_1, \dots, \mathbf{x}_N$ (Mercer's condition)

Kernel Trick

- Type of Kernel Functions

- ✓ Polynomial

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

- ✓ Gaussian (RBF)

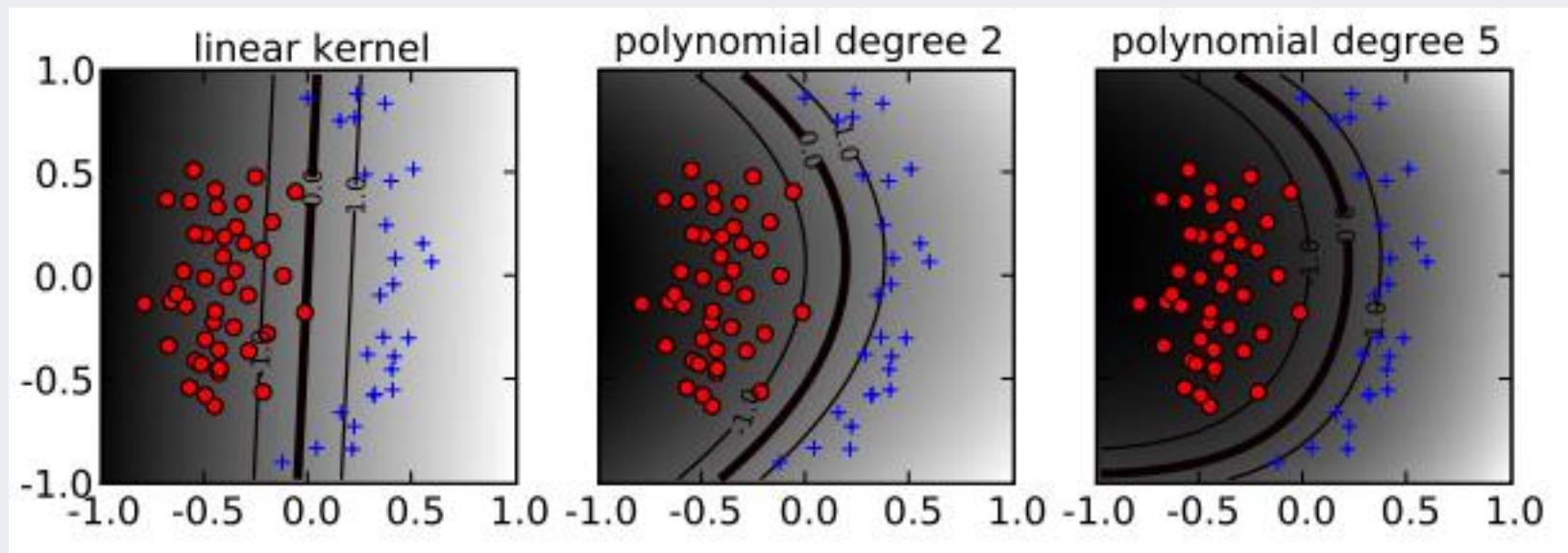
$$K(x, y) = \exp\left(-\frac{\|x - y\|^2}{2\sigma^2}\right), \quad \sigma \neq 0$$

- ✓ Sigmoid

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

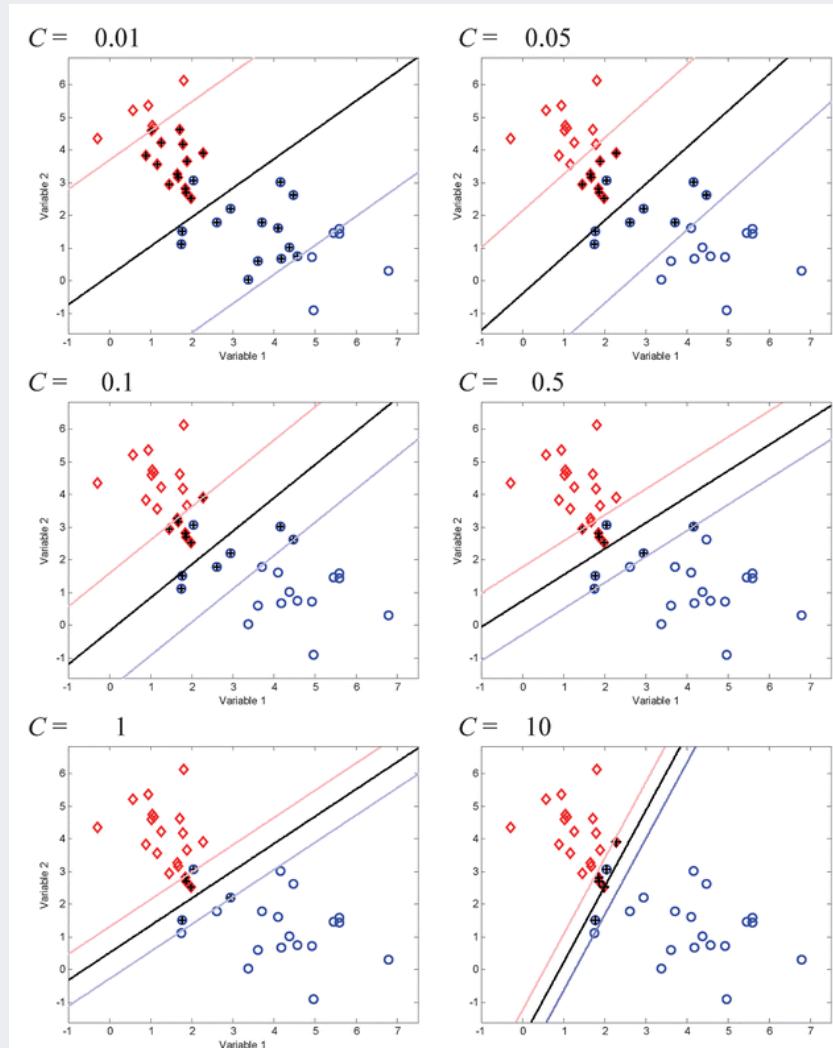
Effects of Different Kernels

- 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



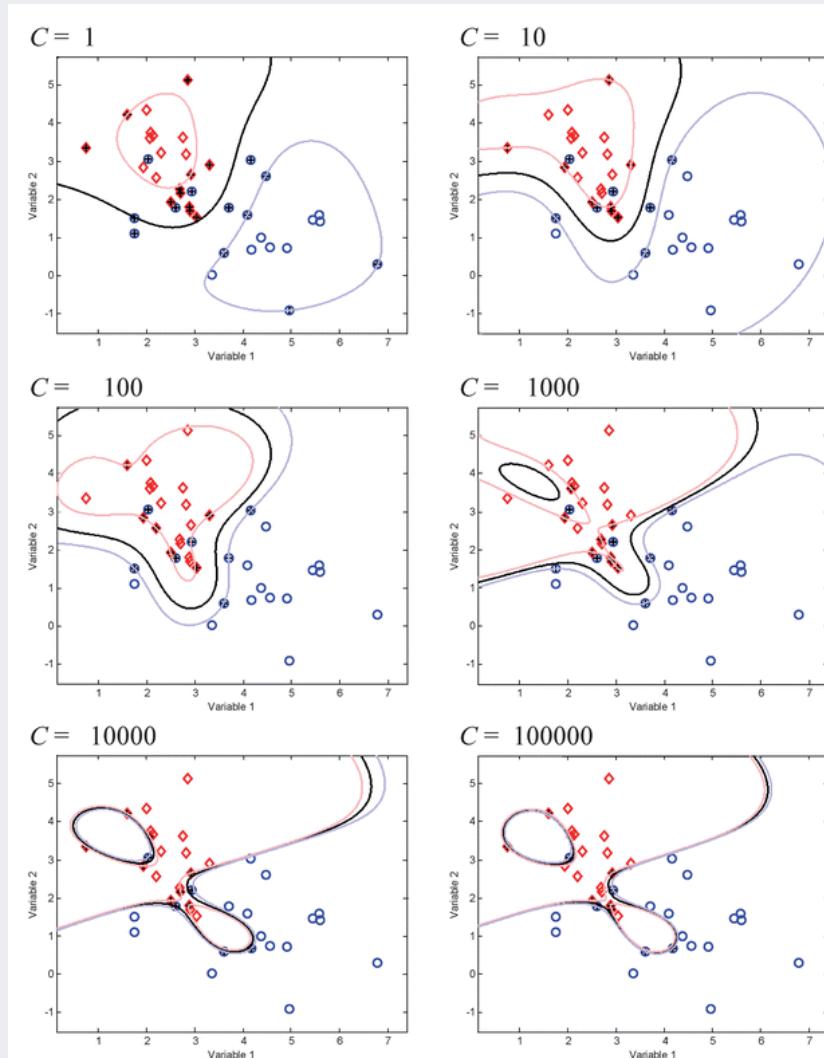
Effects of Different Costs

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



Effects of Different Costs

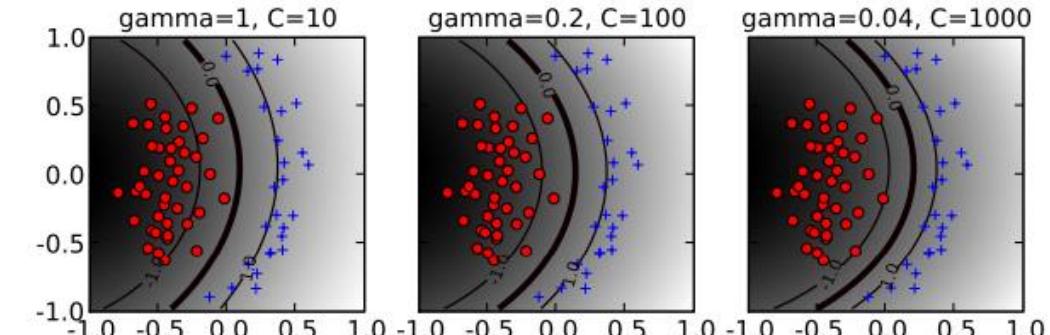
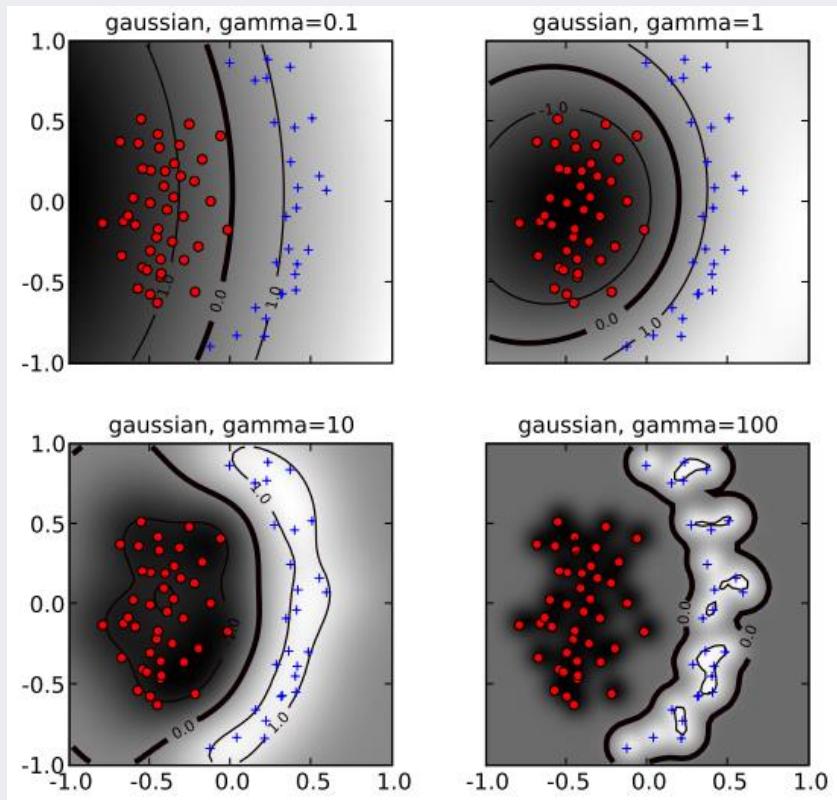
- Margins and SVs with different cost values (RBF kernel)



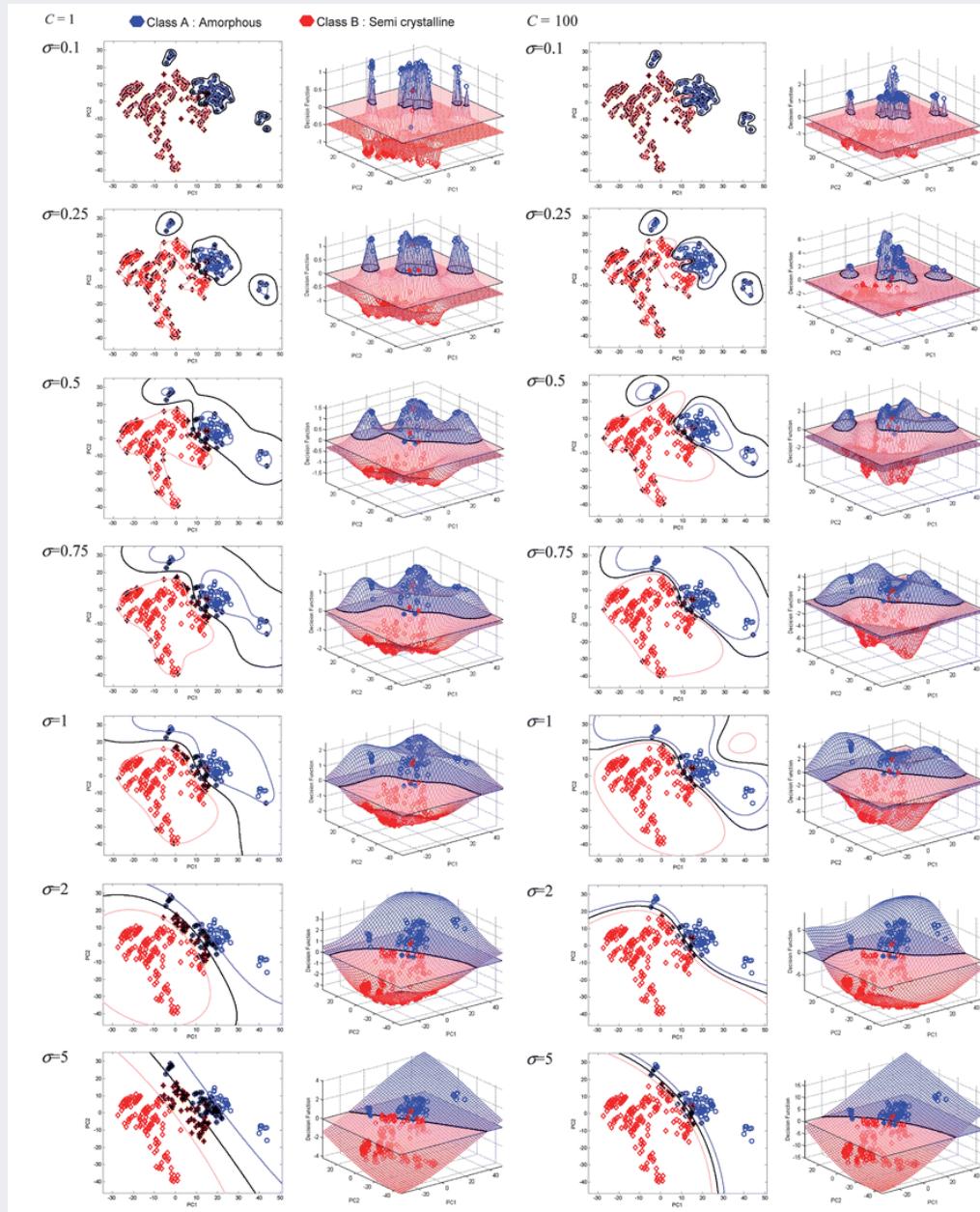
Effects of Different Kernels

- Kernel width (sigma) for RBF Kernel

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



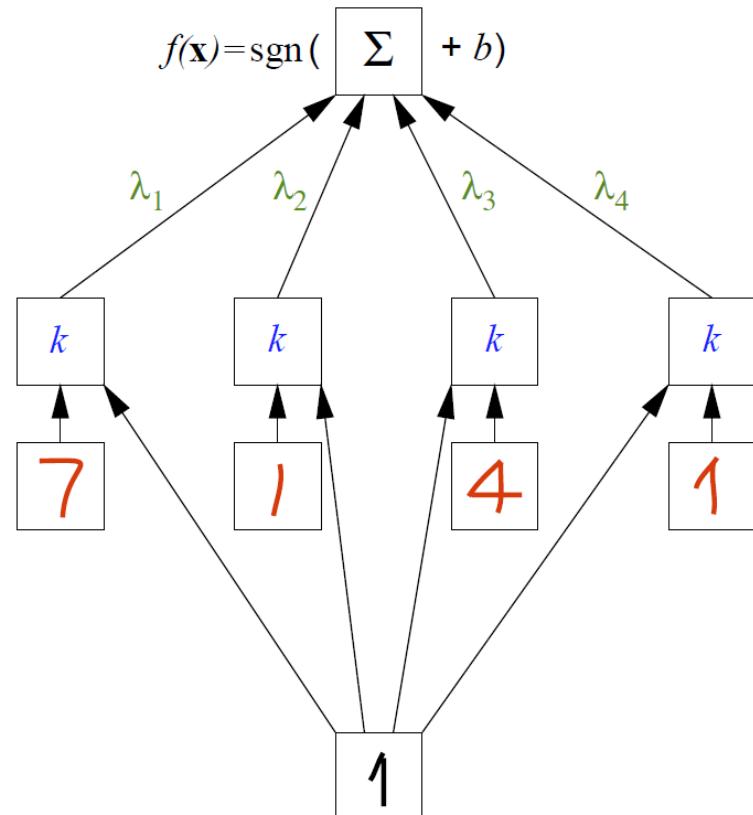
Effects of Kernels and Misclassification Cost



SVM Architecture

- Class decision for a new instance \mathbf{x}

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



classification

weights

comparison: $k(\mathbf{x}, \mathbf{x}_i)$, e.g. $k(\mathbf{x}, \mathbf{x}_i) = (\mathbf{x} \cdot \mathbf{x}_i)^d$

support vectors
 $\mathbf{x}_1 \dots \mathbf{x}_4$

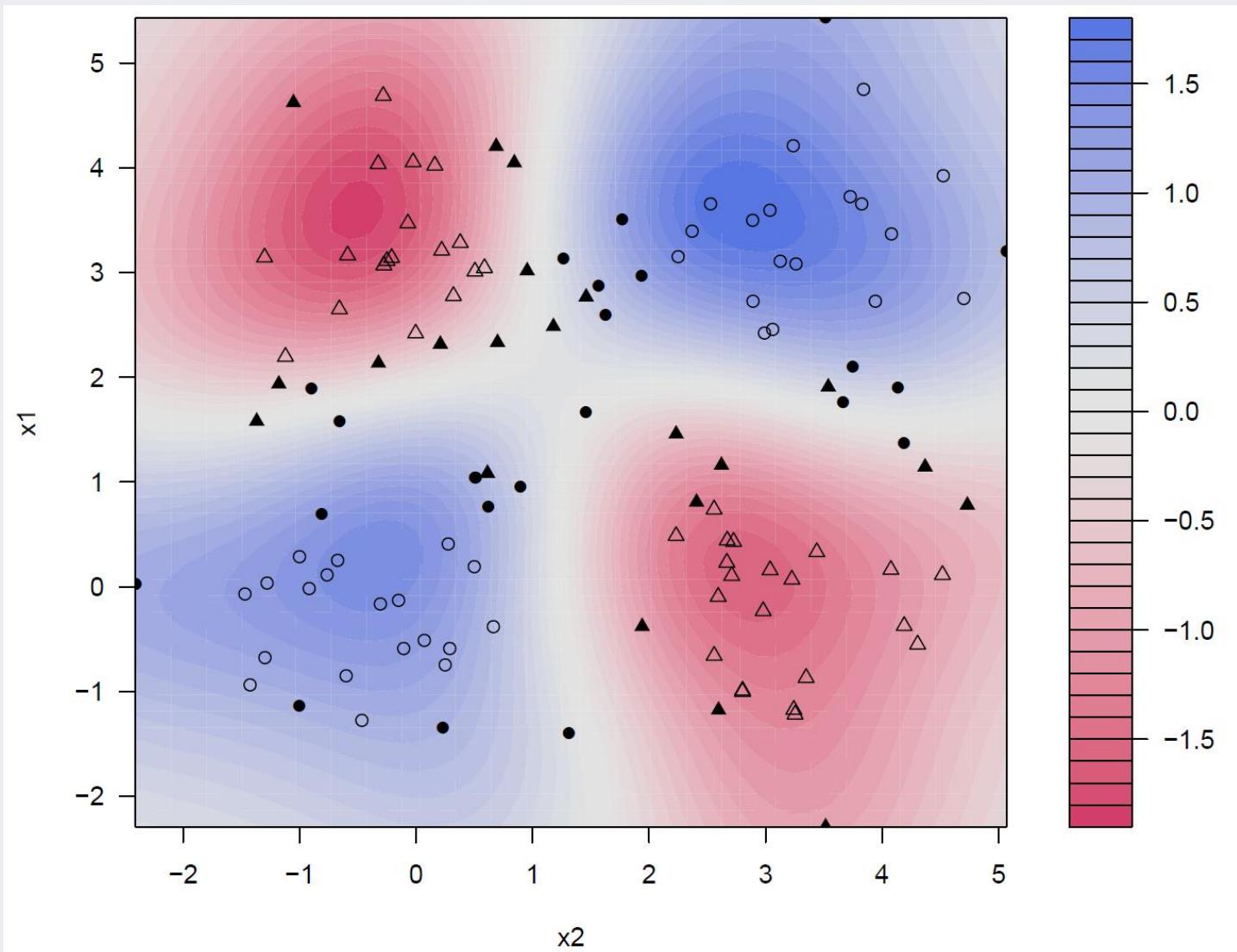
$f(\mathbf{x}) = \text{sgn}(\sum \lambda_i k(\mathbf{x}, \mathbf{x}_i) + b)$

$k(\mathbf{x}, \mathbf{x}_i) = \exp(-\|\mathbf{x} - \mathbf{x}_i\|^2 / c)$

$k(\mathbf{x}, \mathbf{x}_i) = \tanh(\kappa(\mathbf{x} \cdot \mathbf{x}_i) + \theta)$

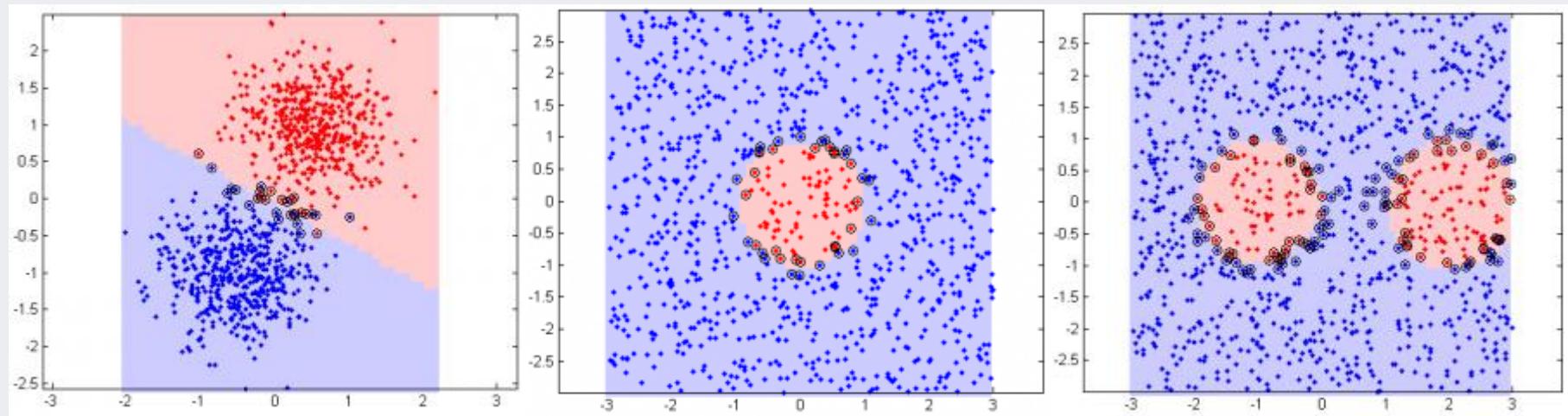
SVM Decision Boundary & Support Vectors

- XOR Problem



SVM Decision Boundary & Support Vectors

- Other examples



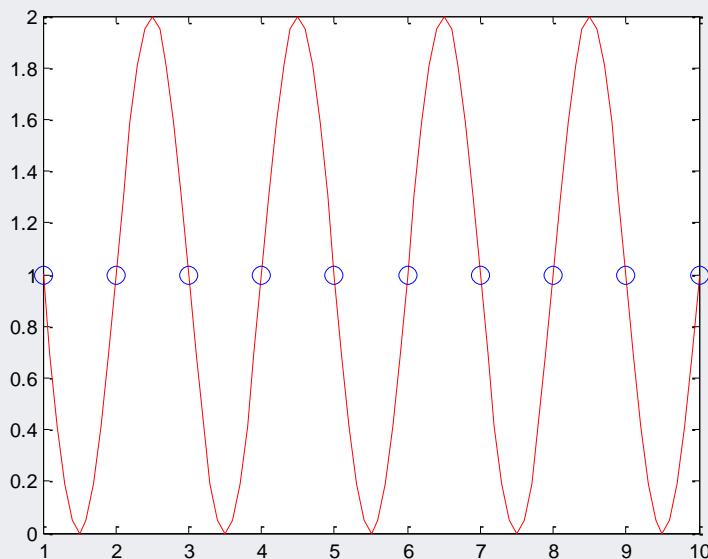
<http://www.alivelearn.net/?p=912>

AGENDA

- 01 Theoretical Foundations
- 02 Support Vector Machine
- 03 Support Vector Regression
- 04 Kernel Fisher Discriminant Analysis
- 05 Kernel Principal Component Analysis

Fitting Function

- Two objectives of function fitting
 - ✓ To fit a function, we minimize an error measure, called also **loss function**
 - ✓ We also like the function to be simple
 - Fewest basis functions
 - Simplest basis functions
 - Flatness is desirable

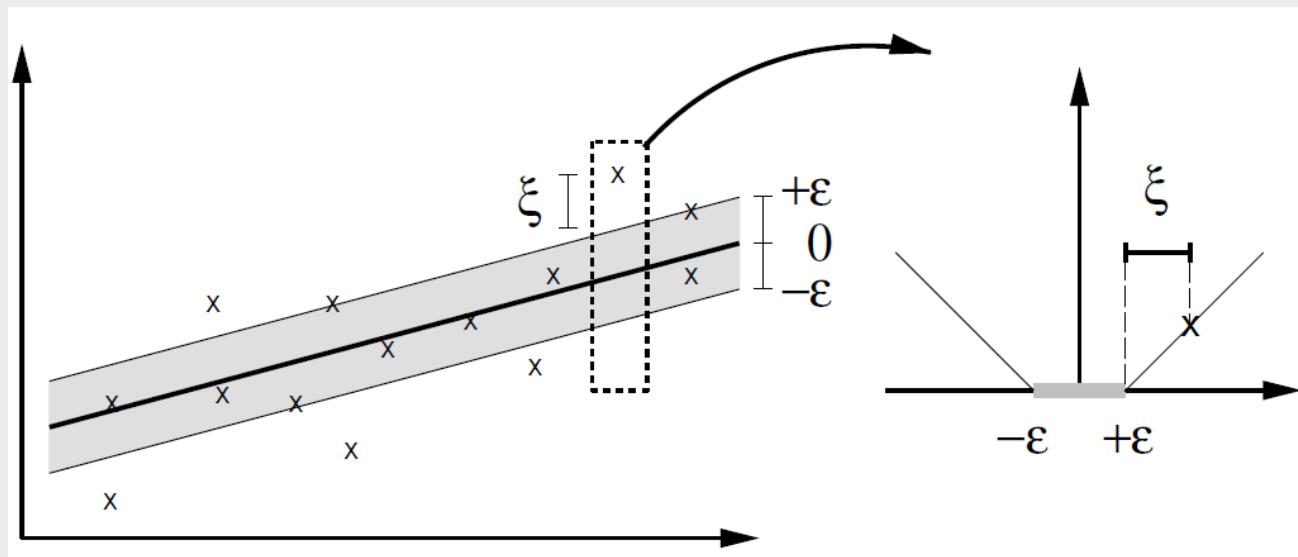


Why not draw a flat line rather than a sine function?

Support Vector Regression (SVR)

Smola and Scholkopf (2004)

- Combine loss function and flatness as a single objective
 - ✓ SVM was developed in the 1960s
 - ✓ Its extension to regression, i.e. SVR, is developed in 1997
- ϵ -SVR
 - ✓ Loss function



Support Vector Regression (SVR)

- SVR Formulation

- ✓ Estimating a linear regression

$$f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b$$

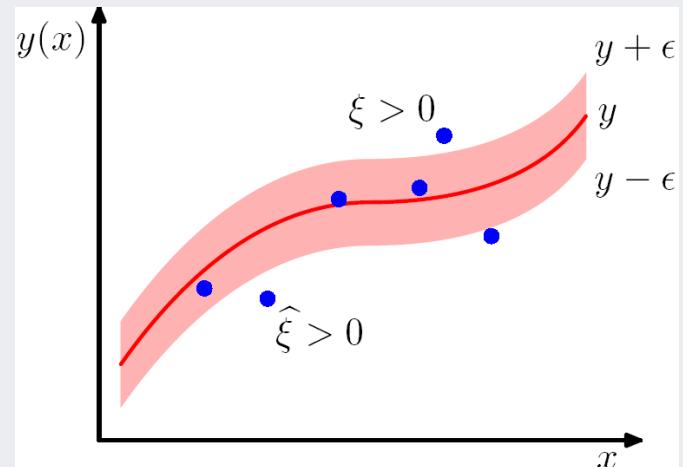
- ✓ with precision ϵ by minimizing

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

$$s.t. \quad (\mathbf{w}^T \mathbf{x}_i + b) - y_i \leq \epsilon + \xi_i$$

$$y_i - (\mathbf{w}^T \mathbf{x}_i + b) \leq \epsilon + \xi_i^*$$

$$\xi_i, \xi_i^* \geq 0$$



Support Vector Regression (SVR)

- Primal Lagrangian

$$L_P = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n (\xi_i + \xi_i^*) - \sum_{i=1}^n (\eta_i \xi_i + \eta_i^* \xi_i^*)$$

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

$$\alpha_i^{(*)}, \eta_i^{(*)} \geq 0$$

- Take the derivative w.r.t. b , \mathbf{w} , ξ

$$\frac{\partial L}{\partial b} = \sum_{i=1}^n (\alpha_i - \alpha_i^*) = 0 \quad \frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^n (\alpha_i^* - \alpha_i) \mathbf{x}_i = 0$$

$$\frac{\partial L}{\partial \xi_i^{(*)}} = C - \alpha_i^{(*)} - \eta_i^{(*)}$$

Support Vector Regression (SVR)

- Dual Lagrangian Problem

$$L_D = -\frac{1}{2} \sum_{i,j=1}^n (\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j) \mathbf{x}_i^T \mathbf{x}_j - \epsilon \sum_{i,j=1}^n (\alpha_i + \alpha_i^*) + \sum_{i,j=1}^n y_i(\alpha_i^* - \alpha_i)$$

$$s.t. \quad \sum_{i=1}^n (\alpha_i - \alpha_i^*) = 0, \quad \alpha_i, \alpha_i^* \in [0, C]$$

- Decision Function

$$\mathbf{w} = \sum_{i=1}^n (\alpha_i^* - \alpha_i) \mathbf{x}_i \quad \Rightarrow \quad f(\mathbf{x}) = \sum_{i=1}^n (\alpha_i^* - \alpha_i) \mathbf{x}_i^T \mathbf{x} + b$$

Support Vector Regression (SVR)

- Dual Lagrangian Problem with Kernel Trick

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

$$s.t. \quad \sum_{i=1}^n (\alpha_i - \alpha_i^*) = 0, \quad \alpha_i, \alpha_i^* \in [0, C]$$

- Decision Function

$$\mathbf{w} = \sum_{i=1}^n (\alpha_i^* - \alpha_i) \Phi(\mathbf{x}_i) \quad \Rightarrow \quad f(\mathbf{x}) = \sum_{i=1}^n (\alpha_i^* - \alpha_i) K(\mathbf{x}_i, \mathbf{x}) + b$$

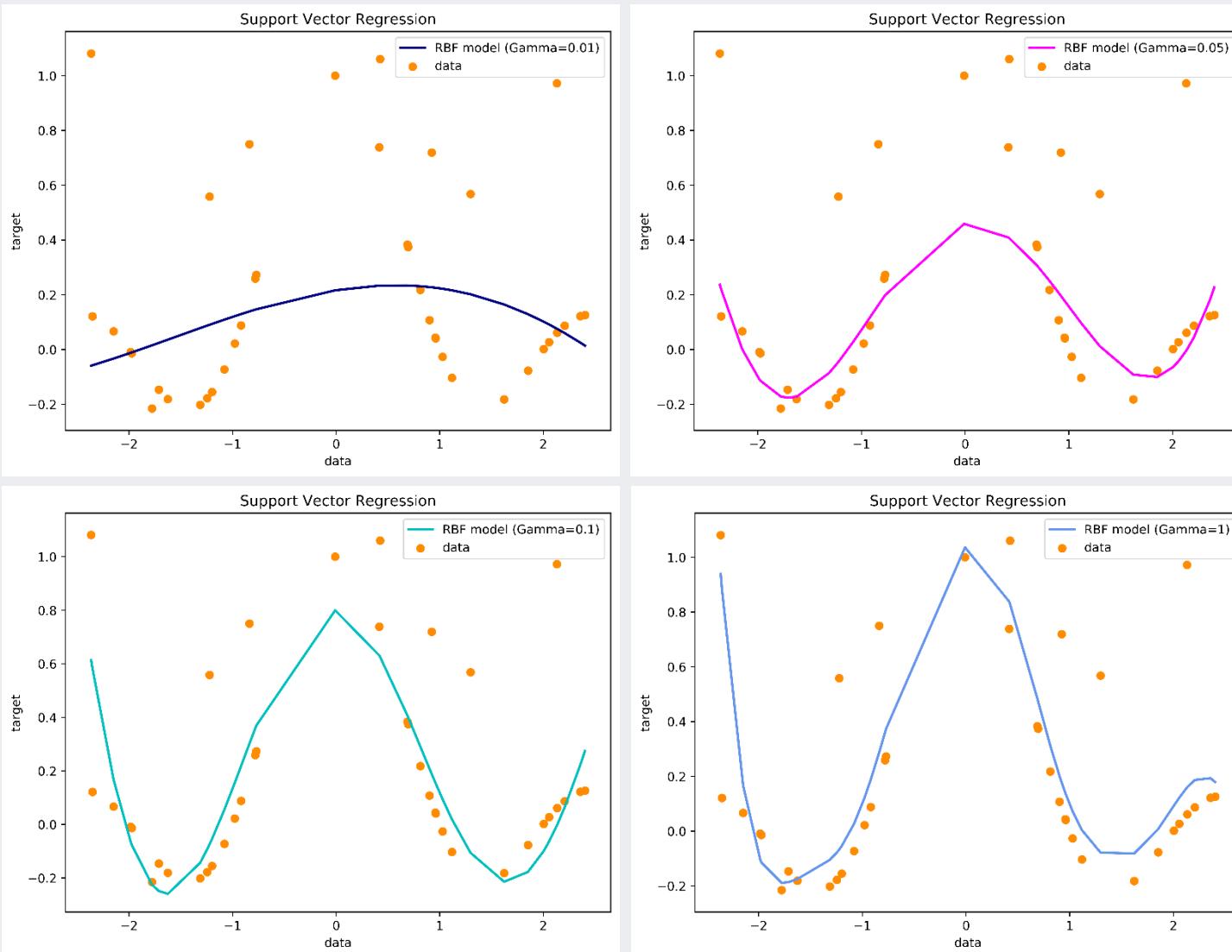
Support Vector Regression (SVR)

- Various Loss Functions for SVR

	loss function	density model
ε -insensitive	$c(\xi) = \xi _\varepsilon$	$p(\xi) = \frac{1}{2(1+\varepsilon)} \exp(- \xi _\varepsilon)$
Laplacian	$c(\xi) = \xi $	$p(\xi) = \frac{1}{2} \exp(- \xi)$
Gaussian	$c(\xi) = \frac{1}{2}\xi^2$	$p(\xi) = \frac{1}{\sqrt{2\pi}} \exp(-\frac{\xi^2}{2})$
Huber's robust loss	$c(\xi) = \begin{cases} \frac{1}{2\sigma}(\xi)^2 & \text{if } \xi \leq \sigma \\ \xi - \frac{\sigma}{2} & \text{otherwise} \end{cases}$	$p(\xi) \propto \begin{cases} \exp(-\frac{\xi^2}{2\sigma}) & \text{if } \xi \leq \sigma \\ \exp(\frac{\sigma}{2} - \xi) & \text{otherwise} \end{cases}$
Polynomial	$c(\xi) = \frac{1}{p} \xi ^p$	$p(\xi) = \frac{p}{2\Gamma(1/p)} \exp(- \xi ^p)$
Piecewise polynomial	$c(\xi) = \begin{cases} \frac{1}{p\sigma^{p-1}}(\xi)^p & \text{if } \xi \leq \sigma \\ \xi - \sigma \frac{p-1}{p} & \text{otherwise} \end{cases}$	$p(\xi) \propto \begin{cases} \exp(-\frac{\xi^p}{p\sigma^{p-1}}) & \text{if } \xi \leq \sigma \\ \exp(\sigma \frac{p-1}{p} - \xi) & \text{otherwise} \end{cases}$

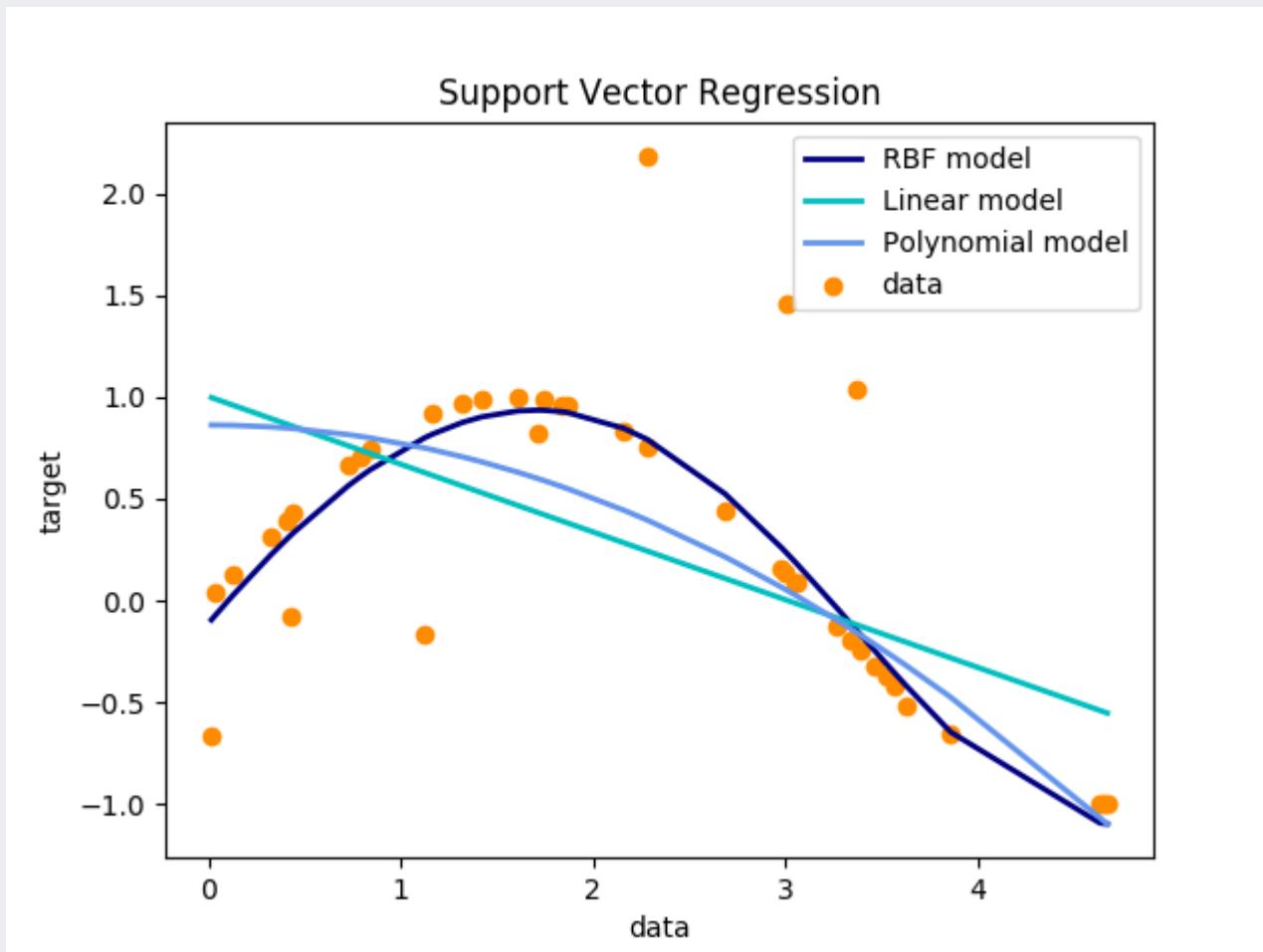
Support Vector Regression (SVR)

- Fitted functions with different ϵ



Support Vector Regression (SVR)

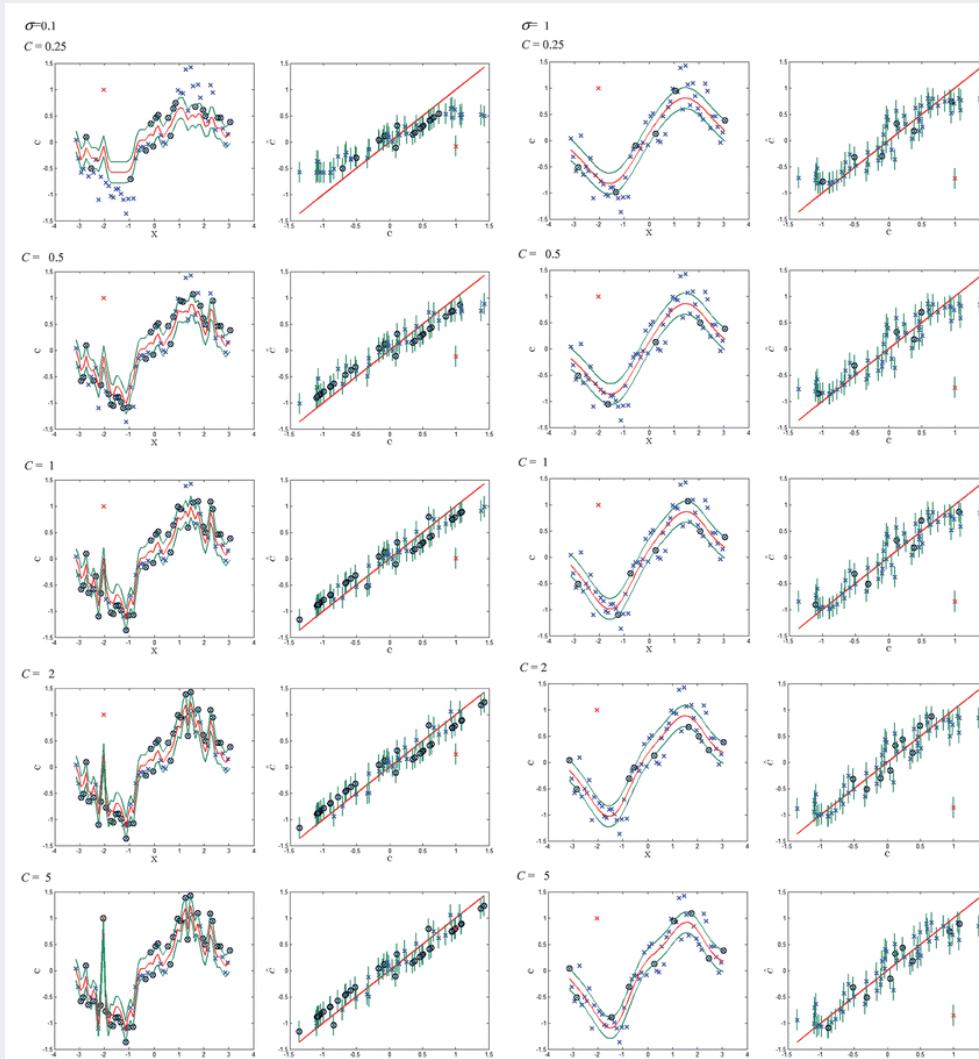
- Fitted function with different Kernel functions



http://scikit-learn.org/stable/auto_examples/svm/plot_svm_regression.html#sphx-glr-auto-examples-svm-plot-svm-regression-py

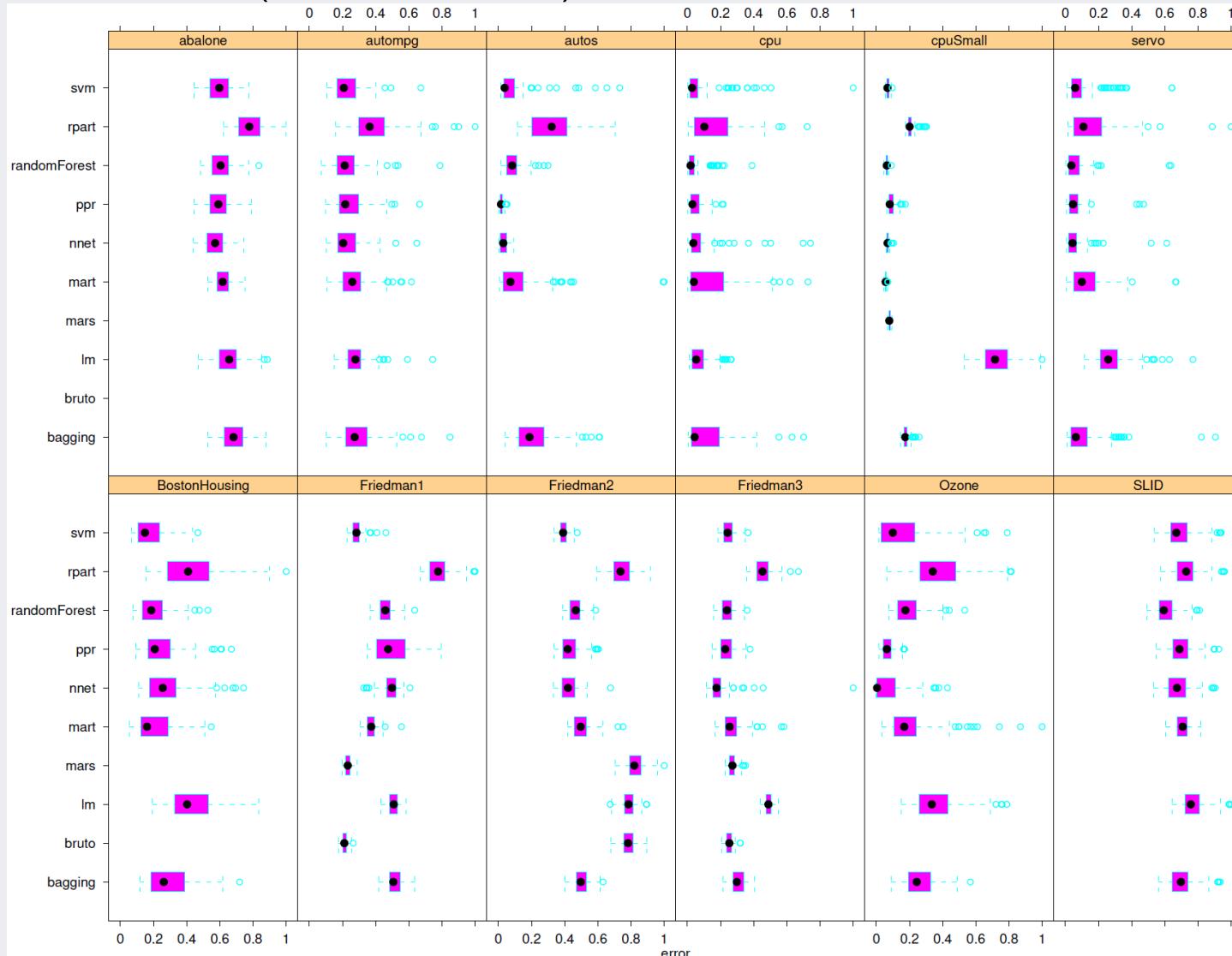
Support Vector Regression (SVR)

- SVR with different sigma and cost combinations



Support Vector Regression (SVR)

- SVR Performance (in terms of MSE)



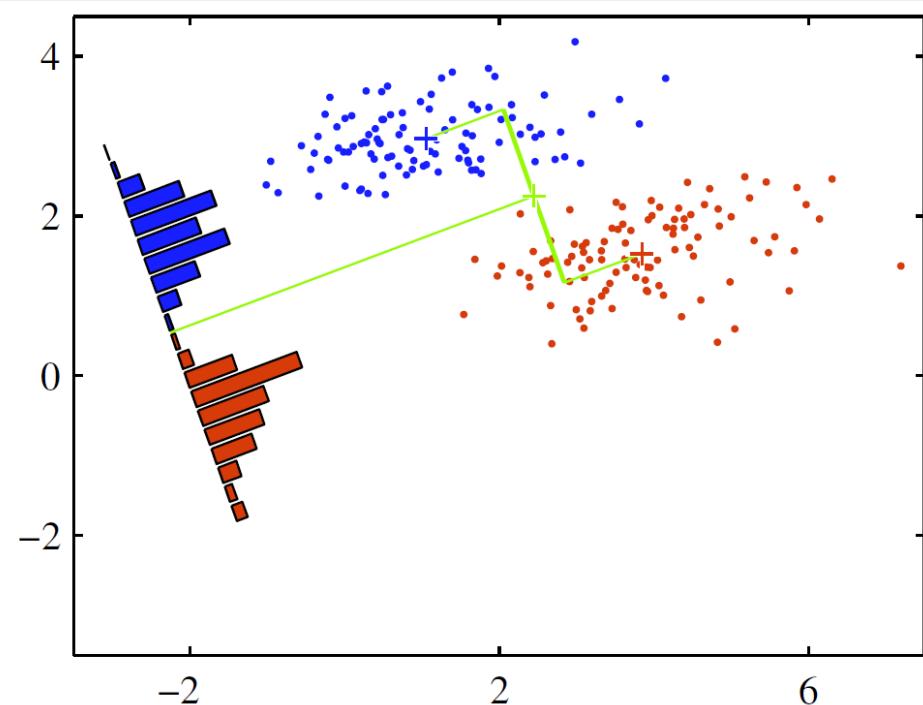
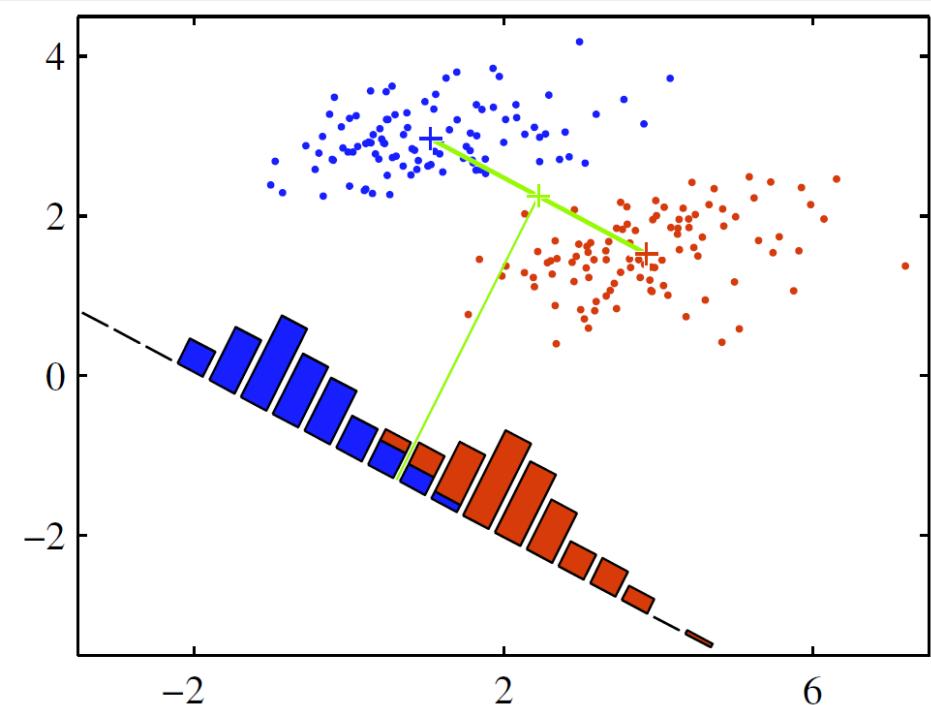
AGENDA

- 01 Theoretical Foundations
- 02 Support Vector Machine
- 03 Support Vector Regression
- 04 Kernel Fisher Discriminant Analysis
- 05 Kernel Principal Component Analysis

Linear Discriminant Analysis

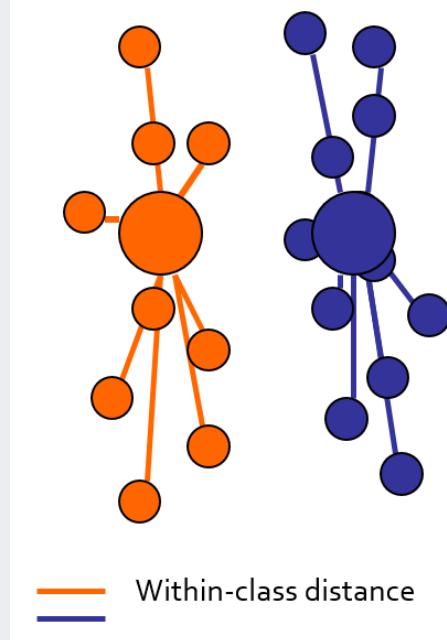
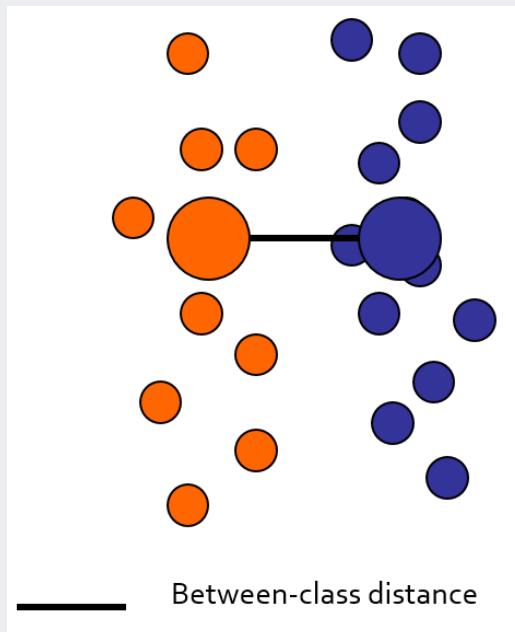
- LDA

- ✓ Find a line to which two classes are well separated after projection



Linear Discriminant Analysis

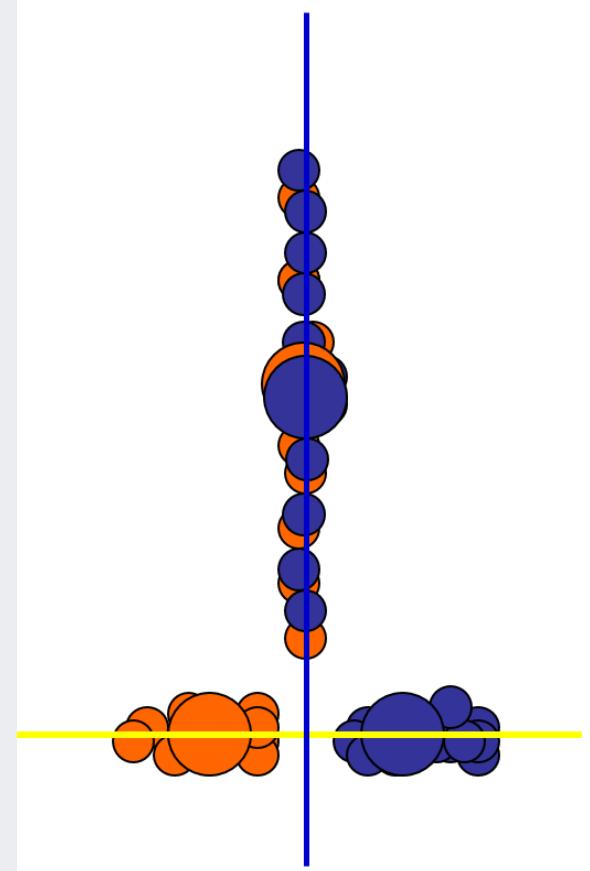
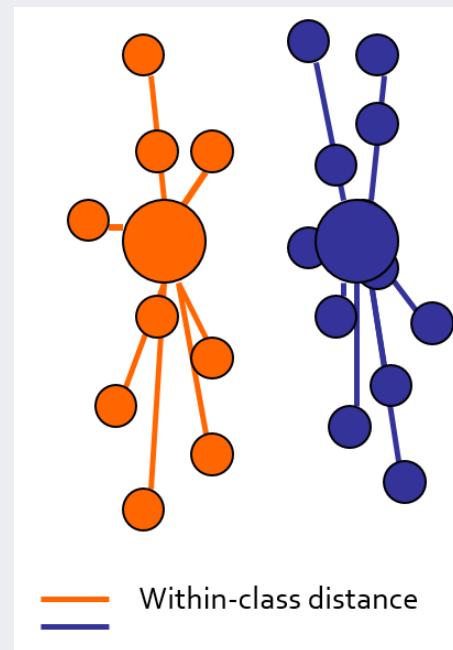
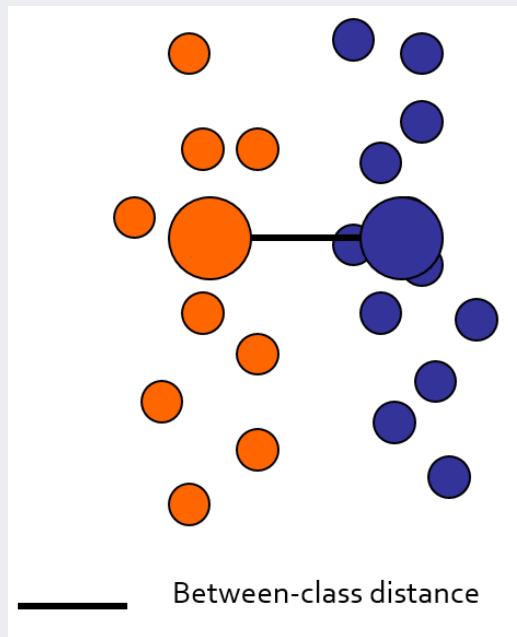
- Two type of class distances
 - ✓ Between-class distance
 - Distance between the centroids of different classes
 - ✓ Within-class distance
 - Accumulated distance of an instance to the centroid of its class



Linear Discriminant Analysis

- (Fisher's) Linear Discriminant Analysis

- ✓ Find most discriminant projection by **maximizing between-class distance (variance)** and **minimizing within-class distance (variance)**



Linear Discriminant Analysis

- Fisher's LDA (cont')

- ✓ Take the D-dimensional input vector \mathbf{x} and project it down to one dim.

$$y = \mathbf{w}^T \mathbf{x}$$

- ✓ Consider a two-class problem in which there are N_1 & N_2 observations in C_1 and C_2 , respectively.

$$\mathbf{m}_1 = \frac{1}{N_1} \sum_{n \in C_1} \mathbf{x}_n, \quad \mathbf{m}_2 = \frac{1}{N_2} \sum_{n \in C_2} \mathbf{x}_n$$

- ✓ Objective I: Choose w to maximize the separation of the projected class means (between class variance)

$$m_2 - m_1 = \mathbf{w}^T (\mathbf{m}_2 - \mathbf{m}_1), \quad m_k = \mathbf{w}^T \mathbf{m}_k$$

Linear Discriminant Analysis

- Fisher's LDA (cont')

- ✓ Objective 2: Choose w to minimize the variance in each class after projection (within class variance)

$$s_k^2 = \sum_{n \in C_k} (y_n - m_k)^2$$

- ✓ Fisher's criterion

- The ratio of the between-class variance to the within-class variance

$$J(\mathbf{w}) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2} = \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}}$$

$$\mathbf{S}_B = (\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^T$$

$$\mathbf{S}_W = \sum_{n \in C_1} (\mathbf{x}_n - \mathbf{m}_1)(\mathbf{x}_n - \mathbf{m}_1)^T + \sum_{n \in C_2} (\mathbf{x}_n - \mathbf{m}_2)(\mathbf{x}_n - \mathbf{m}_2)^T$$

Linear Discriminant Analysis

- Fisher's LDA (cont')

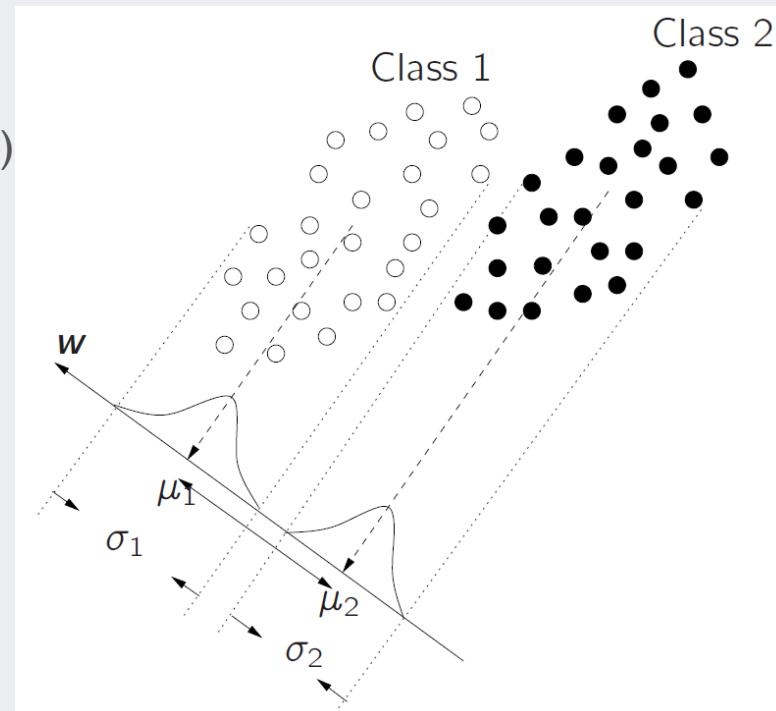
- ✓ Find w

- Differentiating the Fisher's criterion w.r.t. w , then $J(w)$ is maximized when

$$(w^T S_B w) S_W w = (w^T S_W w) S_B w$$

- $S_B w$ is always in the direction of $(m_2 - m_1)$
 - Can drop the scalar factor $(w^T S_B w)$ and $(w^T S_W w)$
 - Then, obtain *Fisher's linear discriminant*

$$w \propto S_W^{-1} (m_2 - m_1)$$



Kernel Fisher Discriminant (KFD)

Mika (2002)

- Extend the LDA formulation by introducing kernels

✓ KFD formulation

- The full covariance of a dataset \mathbf{Z} in the feature space by

$$\mathbf{C}^\Phi = \frac{1}{N} \sum_{n=1}^N (\Phi(\mathbf{x}_n) - \mathbf{m}^\Phi)(\Phi(\mathbf{x}_n) - \mathbf{m}^\Phi)^T, \quad \mathbf{m}^\Phi = \frac{1}{N} \sum_{n=1}^N \Phi(\mathbf{x}_n)$$

- The within-class variance and the between-class variance in the feature space are given by

$$\mathbf{S}_W^\Phi = \sum_{i=1,2} \sum_{n=1}^{N_i} (\Phi(\mathbf{x}_n^i) - \mathbf{m}_i^\Phi)(\Phi(\mathbf{x}_n^i) - \mathbf{m}_i^\Phi)^T$$

$$\mathbf{S}_B = (\mathbf{m}_2^\Phi - \mathbf{m}_1^\Phi)(\mathbf{m}_2^\Phi - \mathbf{m}_1^\Phi)^T \qquad \mathbf{m}_i^\Phi = \frac{1}{N_i} \sum_{j=1}^{N_i} \Phi(\mathbf{x}_j^i)$$

Kernel Fisher Discriminant (KFD)

Mika (2002)

- Extend the LDA formulation by introducing kernels

- ✓ Objective functions

$$J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{S}_B^\Phi \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W^\Phi \mathbf{w}}$$

- ✓ Projected vector

$$\mathbf{w} = \sum_{n=1}^N \alpha_n \Phi(\mathbf{x}_n), \quad \alpha_n \in R$$

- ✓ Projected mean

$$\mathbf{w}^T \mathbf{m}_i^\Phi = \frac{1}{N_i} \sum_{n=1}^N \sum_{k=1}^{N_i} \alpha_n (\Phi(\mathbf{x}_n) \cdot \Phi(\mathbf{x}_k^i)) = \frac{1}{N_i} \sum_{n=1}^N \sum_{k=1}^{N_i} \alpha_n \mathbf{K}(\mathbf{x}_n, \mathbf{x}_k^i) = \boldsymbol{\alpha}^T \boldsymbol{\mu}_i$$

$$(\boldsymbol{\mu}_i)_n = \frac{1}{N_i} \sum_{k=1}^{N_i} \mathbf{K}(\mathbf{x}_n, \mathbf{x}_k^i)$$

Kernel Fisher Discriminant (KFD)

Mika (2002)

- Extend the LDA formulation by introducing kernels

- ✓ Objective function (Numerator)

$$\mathbf{w}^T \mathbf{S}_B^\Phi \mathbf{w} = \mathbf{w}^T (\mathbf{m}_2^\Phi - \mathbf{m}_1^\Phi) (\mathbf{m}_2^\Phi - \mathbf{m}_1^\Phi)^T \mathbf{w}$$

$$= \boldsymbol{\alpha}^T (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1) (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)^T \boldsymbol{\alpha}$$

$$= \boldsymbol{\alpha}^T \mathbf{M} \boldsymbol{\alpha}, \text{ where } \mathbf{M} = (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1) (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)^T$$

Kernel Fisher Discriminant (KFD)

Mika (2002)

- Extend the LDA formulation by introducing kernels
 - ✓ Objective function (Denominator)

$$\begin{aligned}\mathbf{w}^T \mathbf{S}_W^\Phi \mathbf{w} &= \left(\sum_{i=1}^N \alpha_i \Phi(\mathbf{x}_i) \right) \left(\sum_{j=1,2} \sum_{n=i}^{N_i} (\Phi(\mathbf{x}_n^j) - \mathbf{m}_j^\Phi) (\Phi(\mathbf{x}_n^j) - \mathbf{m}_j^\Phi)^T \right) \sum_{k=1}^N \alpha_k \Phi(\mathbf{x}_k) \\ &= \sum_{j=1,2} \sum_{i=1}^N \sum_{n=1}^{N_i} \sum_{k=1}^N \left(\alpha_i \Phi(\mathbf{x}_i) (\Phi(\mathbf{x}_n^j) - \mathbf{m}_j^\Phi) (\Phi(\mathbf{x}_n^j) - \mathbf{m}_j^\Phi)^T \alpha_k \Phi(\mathbf{x}_k) \right) \\ &= \sum_{j=1,2} \sum_{i=1}^N \sum_{n=1}^{N_i} \sum_{k=1}^N \left(\alpha_i \mathbf{K}(\mathbf{x}_i, \mathbf{x}_n^j) - \frac{1}{N_j} \sum_{p=1}^{N_j} \alpha_i \mathbf{K}(\mathbf{x}_i, \mathbf{x}_p^j) \right) \times \\ &\quad \left(\alpha_k \mathbf{K}(\mathbf{x}_k, \mathbf{x}_n^j) - \frac{1}{N_j} \sum_{q=1}^{N_j} \alpha_k \mathbf{K}(\mathbf{x}_k, \mathbf{x}_q^j) \right)\end{aligned}$$

Kernel Fisher Discriminant (KFD)

Mika (2002)

- Extend the LDA formulation by introducing kernels

- ✓ Objective function (Denominator)

$$\sum_{j=1,2} \sum_{i=1}^N \sum_{n=1}^{N_i} \sum_{k=1}^N \left(\alpha_i \mathbf{K}(\mathbf{x}_i, \mathbf{x}_n^j) - \frac{1}{N_j} \sum_{p=1}^{N_j} \alpha_i \mathbf{K}(\mathbf{x}_i, \mathbf{x}_p^j) \right) \left(\alpha_k \mathbf{K}(\mathbf{x}_k, \mathbf{x}_n^j) - \frac{1}{N_j} \sum_{q=1}^{N_j} \alpha_k \mathbf{K}(\mathbf{x}_k, \mathbf{x}_q^j) \right)$$

$$= \sum_{j=1,2} \left(\sum_{i=1}^N \sum_{n=1}^{N_i} \sum_{k=1}^N \left(\alpha_i \alpha_k \mathbf{K}(\mathbf{x}_i, \mathbf{x}_n^j) \mathbf{K}(\mathbf{x}_k, \mathbf{x}_n^j) - \frac{2\alpha_i \alpha_k}{N_j} \sum_{p=1}^{N_j} \mathbf{K}(\mathbf{x}_i, \mathbf{x}_n^j) \mathbf{K}(\mathbf{x}_k, \mathbf{x}_p^j) \right. \right. \\ \left. \left. + \frac{\alpha_i \alpha_k}{N_j^2} \sum_{p=1}^{N_j} \sum_{q=1}^{N_j} \mathbf{K}(\mathbf{x}_i, \mathbf{x}_p^j) \mathbf{K}(\mathbf{x}_k, \mathbf{x}_q^j) \right) \right)$$

$$= \sum_{j=1,2} \left(\sum_{i=1}^N \sum_{n=1}^{N_i} \sum_{k=1}^N \left(\alpha_i \alpha_k \mathbf{K}(\mathbf{x}_i, \mathbf{x}_n^j) \mathbf{K}(\mathbf{x}_k, \mathbf{x}_n^j) - \frac{\alpha_i \alpha_k}{N_j} \sum_{p=1}^{N_j} \mathbf{K}(\mathbf{x}_i, \mathbf{x}_n^j) \mathbf{K}(\mathbf{x}_k, \mathbf{x}_p^j) \right) \right)$$

Kernel Fisher Discriminant (KFD)

Mika (2002)

- Extend the LDA formulation by introducing kernels

- ✓ Objective function (Denominator)

$$\sum_{j=1,2} \left(\sum_{i=1}^N \sum_{n=1}^{N_i} \sum_{k=1}^N \left(\alpha_i \alpha_k \mathbf{K}(\mathbf{x}_i, \mathbf{x}_n^j) \mathbf{K}(\mathbf{x}_k, \mathbf{x}_n^j) - \frac{\alpha_i \alpha_k}{N_j} \sum_{p=1}^{N_j} \mathbf{K}(\mathbf{x}_i, \mathbf{x}_n^j) \mathbf{K}(\mathbf{x}_k, \mathbf{x}_p^j) \right) \right)$$

$$= \sum_{j=1,2} \boldsymbol{\alpha}^T \mathbf{K}_j \mathbf{K}_j^T \boldsymbol{\alpha} - \boldsymbol{\alpha}^T \mathbf{K}_j \mathbf{1}_{l_j} \mathbf{K}_j^T \boldsymbol{\alpha}$$

$$= \boldsymbol{\alpha}^T \mathbf{N} \boldsymbol{\alpha}, \text{ where } \mathbf{N} = \sum_{j=1,2} \mathbf{K}_j (\mathbf{I} - \mathbf{1}_{l_j}) \mathbf{K}_j^T$$

Kernel Fisher Discriminant (KFD)

Mika (2002)

- Extend the LDA formulation by introducing kernels

- ✓ Objective function

$$J(\boldsymbol{\alpha}) = \frac{\boldsymbol{\alpha}^T \mathbf{M} \boldsymbol{\alpha}}{\boldsymbol{\alpha}^T \mathbf{N} \boldsymbol{\alpha}}$$

- ✓ Take the first derivative and set it equal to 0

$$(\boldsymbol{\alpha}^T \mathbf{M} \boldsymbol{\alpha}) \mathbf{N} \boldsymbol{\alpha} = (\boldsymbol{\alpha}^T \mathbf{N} \boldsymbol{\alpha}) \mathbf{M} \boldsymbol{\alpha}$$

- ✓ Since $\mathbf{M} \boldsymbol{\alpha} = (\mathbf{M}_2 - \mathbf{M}_1)(\mathbf{M}_2 - \mathbf{M}_1)^T \boldsymbol{\alpha} = \lambda(\mathbf{M}_2 - \mathbf{M}_1)$,

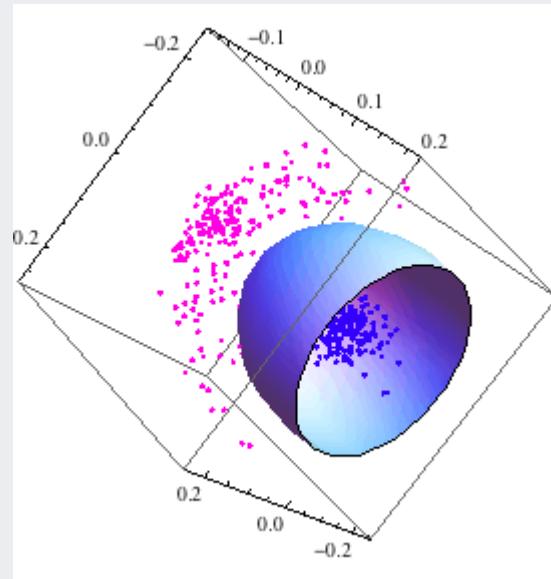
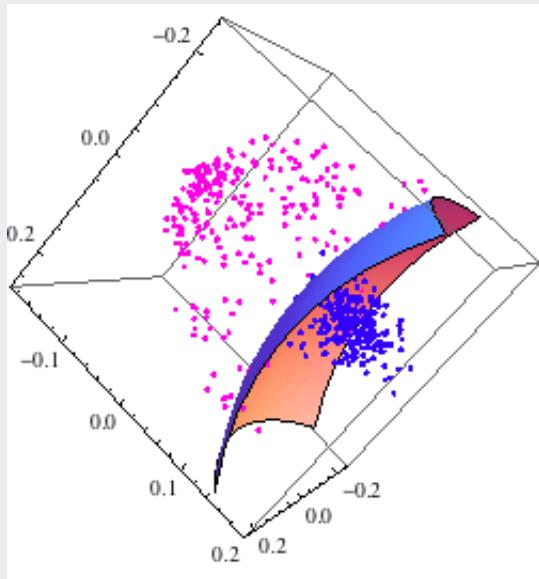
$$\boldsymbol{\alpha} = \mathbf{N}^{-1}(\mathbf{M}_2 - \mathbf{M}_1)$$

- ✓ Given the solution for $\boldsymbol{\alpha}$, the projection of a new data point is given by

$$y(\mathbf{x}) = (\mathbf{w} \cdot \Phi(\mathbf{x})) = \sum_{n=1}^N \alpha_n \mathbf{K}(\mathbf{x}_n, \mathbf{x})$$

KFD Example

- KFD with a polynomial kernel
- KFD with a polynomial kernel



<http://www.mathematica-journal.com/2011/07/fisher-discrimination-with-kernels/>

KFD Performance

- Classification Performance

- ✓ Benchmark data sets

	dimensionality	Size of training set		test set
Banana	2	400	4900	
B.Cancer	9	200	77	
Diabetes	8	468	300	
German	20	700	300	
Heart	13	170	100	
Ringnorm	20	400	7000	
F.Sonar	9	666	400	
Thyroid	5	140	75	
Titanic	3	150	2051	
Waveform	21	400	4600	

KFD Performance

- Classification Performance

- ✓ Classification accuracy

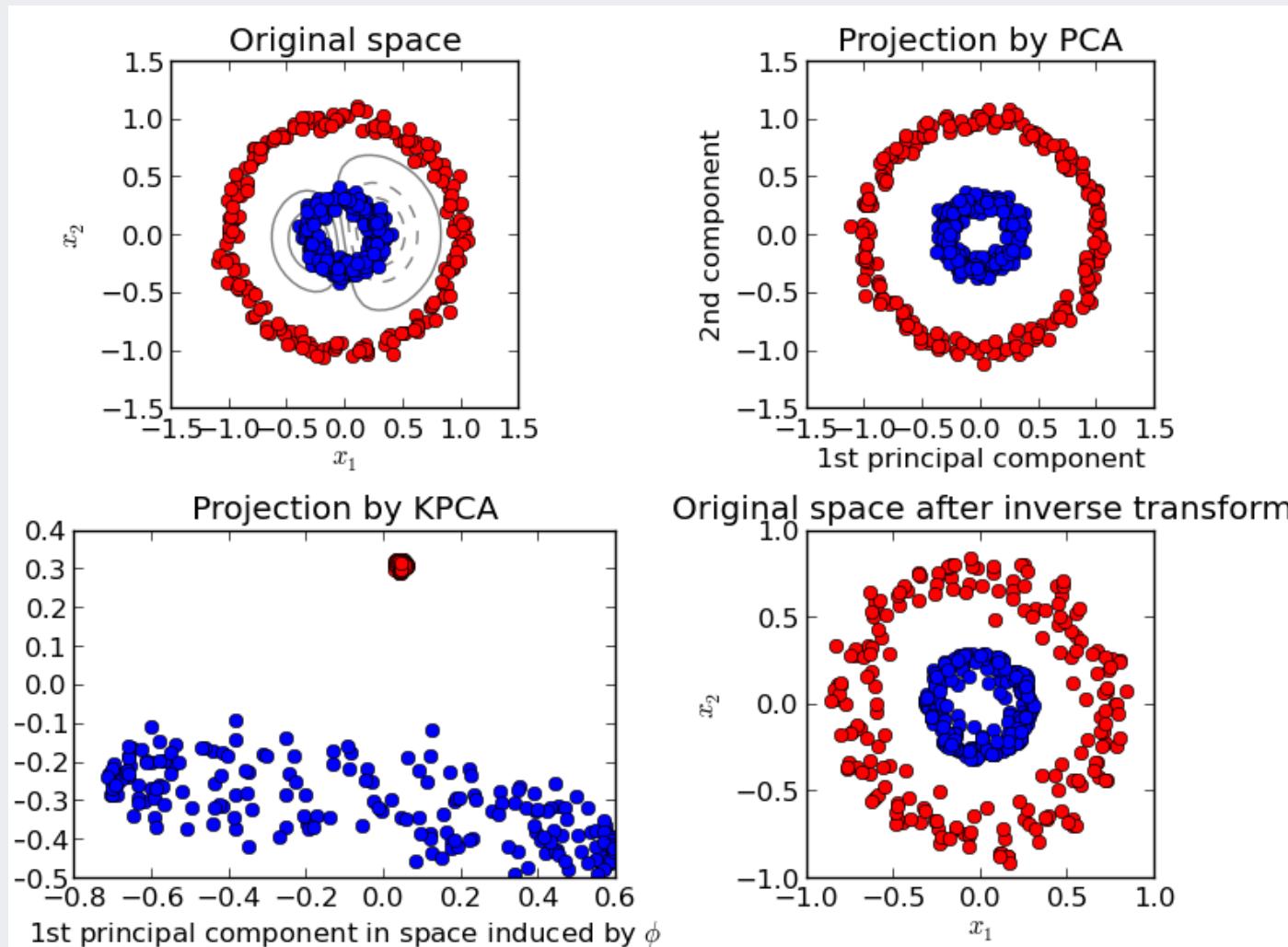
	RBF	AB	AB_R	SVM	KFD
Banana	10.8±0.06	12.3±0.07	10.9±0.04	11.5±0.07	10.8±0.05
B.Cancer	27.6±0.47	30.4±0.47	26.5±0.45	26.0±0.47	25.8±0.46
Diabetes	24.3±0.19	26.5±0.23	23.8±0.18	23.5±0.17	23.2±0.16
German	24.7±0.24	27.5±0.25	24.3±0.21	23.6±0.21	23.7±0.22
Heart	17.6±0.33	20.3±0.34	16.5±0.35	16.0±0.33	16.1±0.34
Ringnorm	1.7±0.02	1.9±0.03	1.6±0.01	1.7±0.01	1.5±0.01
F.Sonar	34.4±0.20	35.7±0.18	34.2±0.22	32.4±0.18	33.2±0.17
Thyroid	4.5±0.21	4.4±0.22	4.6±0.22	4.8±0.22	4.2±0.21
Titanic	23.3±0.13	22.6±0.12	22.6±0.12	22.4±0.10	23.2±0.20
Waveform	10.7±0.11	10.8±0.06	9.8±0.08	9.9±0.04	9.9±0.04
Average	18.0%	20.2%	17.5%	17.2%	17.2%

AGENDA

- 01 Theoretical Foundations
- 02 Support Vector Machine
- 03 Support Vector Regression
- 04 Kernel Fisher Discriminant Analysis
- 05 Kernel Principal Component Analysis

Kernel PCA

- What if the embedding is not linear?



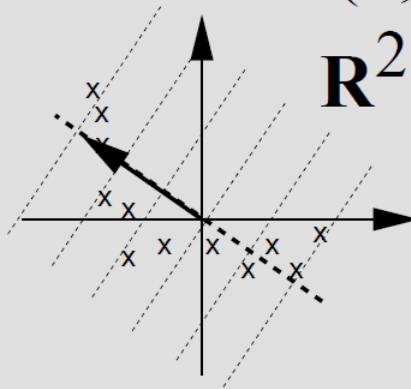
Kernel PCA

Schölkopf et al. (1998)

- Motivation

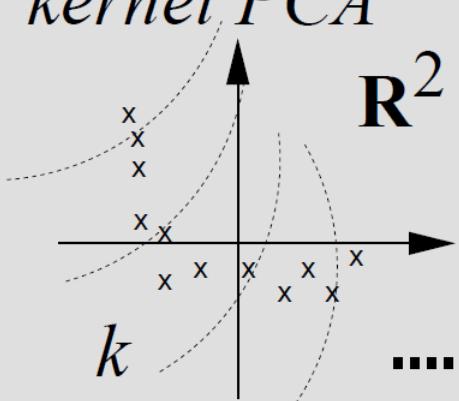
linear PCA

$$k(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y})$$

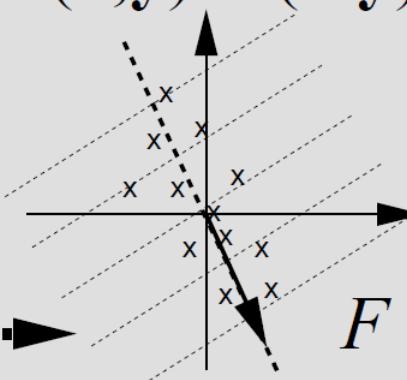


kernel PCA

$$\text{e.g. } k(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y})^d$$



Φ



Kernel PCA

- Kernel PCA Procedure

- ✓ Assumption: the projected new features have zero mean

$$\mathbf{m}^\Phi = \frac{1}{N} \sum_{i=1}^N \Phi(\mathbf{x}_i) = \mathbf{0}$$

- ✓ The covariance matrix of the projected features is M by M, calculated by

$$\mathbf{C}^\Phi = \frac{1}{N} \sum_{i=1}^N (\Phi(\mathbf{x}_i) - \mathbf{m}^\Phi)(\Phi(\mathbf{x}_i) - \mathbf{m}^\Phi)^T = \frac{1}{N} \sum_{i=1}^N \Phi(\mathbf{x}_i)\Phi(\mathbf{x}_i)^T$$

- ✓ Its eigenvalues and eigenvectors are given by

$$\mathbf{C}^\Phi \mathbf{v}_k = \lambda_k \mathbf{v}_k$$

- where k = 1, 2, ..., M.

Kernel PCA

- Kernel PCA Procedure

- ✓ From the previous two equations, we have

$$\frac{1}{N} \sum_{i=1}^N \Phi(\mathbf{x}_i) (\Phi(\mathbf{x}_i)^T \mathbf{v}_k) = \lambda_k \mathbf{v}_k$$

- ✓ which can be rewritten as

$$\mathbf{v}_k = \frac{1}{N} \sum_{i=1}^N \alpha_{ki} \Phi(\mathbf{x}_i)$$

- ✓ By substituting \mathbf{v}_k above, we have

$$\frac{1}{N} \sum_{i=1}^N \Phi(\mathbf{x}_i) \Phi(\mathbf{x}_i)^T \sum_{j=1}^N \alpha_{kj} \Phi(\mathbf{x}_j) = \lambda_k \sum_{i=1}^N \alpha_{kj} \Phi(\mathbf{x}_i)$$

Kernel PCA

- ✓ If we define the kernel function

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

- ✓ And multiply both side of the last equation in the previous page by $\Phi(\mathbf{x}_l)$

$$\frac{1}{N} \sum_{i=1}^N \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_l) \sum_{j=1}^N \alpha_{kj} \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j) = \lambda_k \sum_{i=1}^N \alpha_{kj} \Phi(\mathbf{x}_l)^T \Phi(\mathbf{x}_i)$$

$$\frac{1}{N} \sum_{i=1}^N \mathbf{K}(\mathbf{x}_l, \mathbf{x}_i) \sum_{j=1}^N \alpha_{kj} \mathbf{K}(\mathbf{x}_i, \mathbf{x}_j) = \lambda_k \sum_{i=1}^N \alpha_{kj} \mathbf{K}(\mathbf{x}_l, \mathbf{x}_i)$$

- ✓ Then, we can use the matrix notation

$$\mathbf{K}^2 \boldsymbol{\alpha}_k = \lambda_k N \mathbf{K} \boldsymbol{\alpha}_k$$

- Where $\boldsymbol{\alpha}_k$ is the N-dimensional column vector $\boldsymbol{\alpha}_k = (\alpha_{k1}, \alpha_{k2}, \dots, \alpha_{kN})^T$

Kernel PCA

- ✓ The eigenvector problem becomes

$$\mathbf{K}\boldsymbol{\alpha}_k = \lambda_k N \boldsymbol{\alpha}_k$$

- ✓ And the resulting kernel PCA can be calculated using

$$y_k(\mathbf{x}) = \Phi(\mathbf{x})^T \mathbf{v}_k = \sum_{i=1}^N \alpha_{ki} \mathbf{K}(\mathbf{x}, \mathbf{x}_i)$$

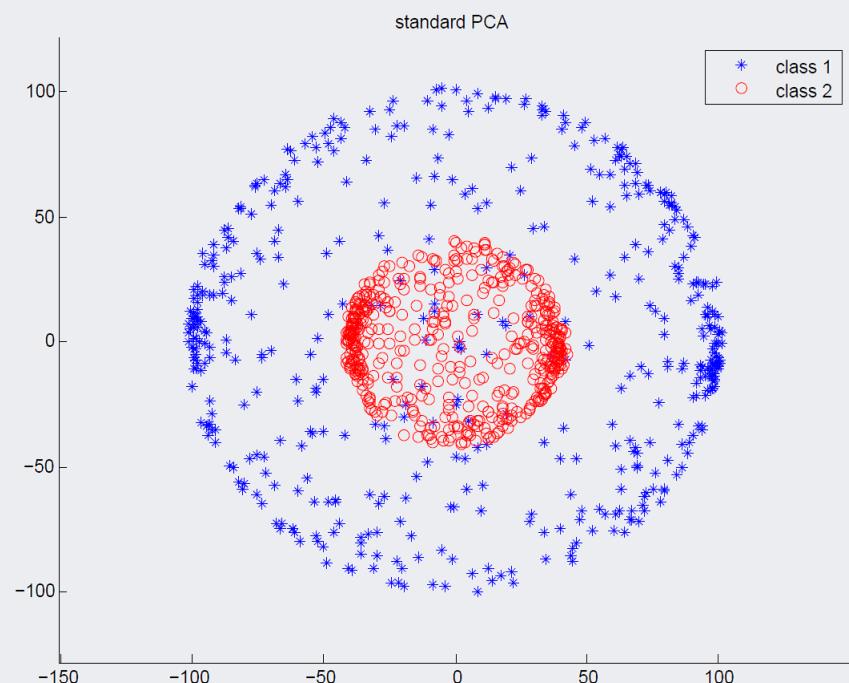
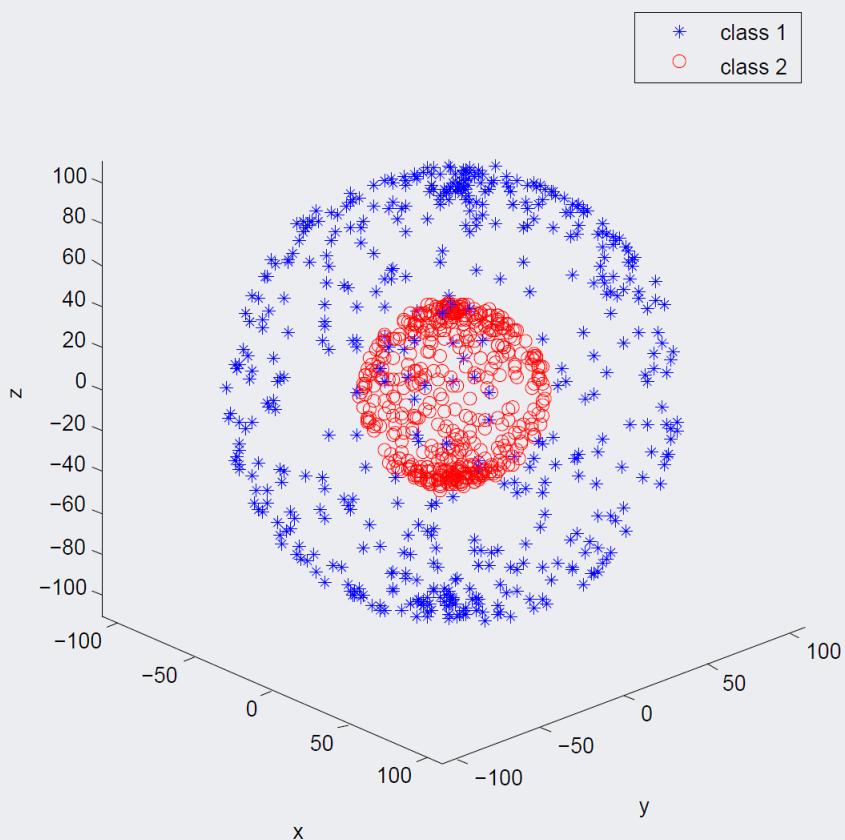
- ✓ If the projected dataset does not have zero mean, use the Gram matrix

$$\begin{aligned}\tilde{\mathbf{K}} &= (\mathbf{I} - \mathbf{1}_N \mathbf{1}_N^T) \mathbf{K} (\mathbf{I} - \mathbf{1}_N \mathbf{1}_N^T) \\ &= \mathbf{K} - \mathbf{1}_N \mathbf{K} - \mathbf{K} \mathbf{1}_N + \mathbf{1}_N \mathbf{K} \mathbf{1}_N\end{aligned}$$

- Where $\mathbf{1}_N$ is the N by N matrix with all elements equal to $1/N$

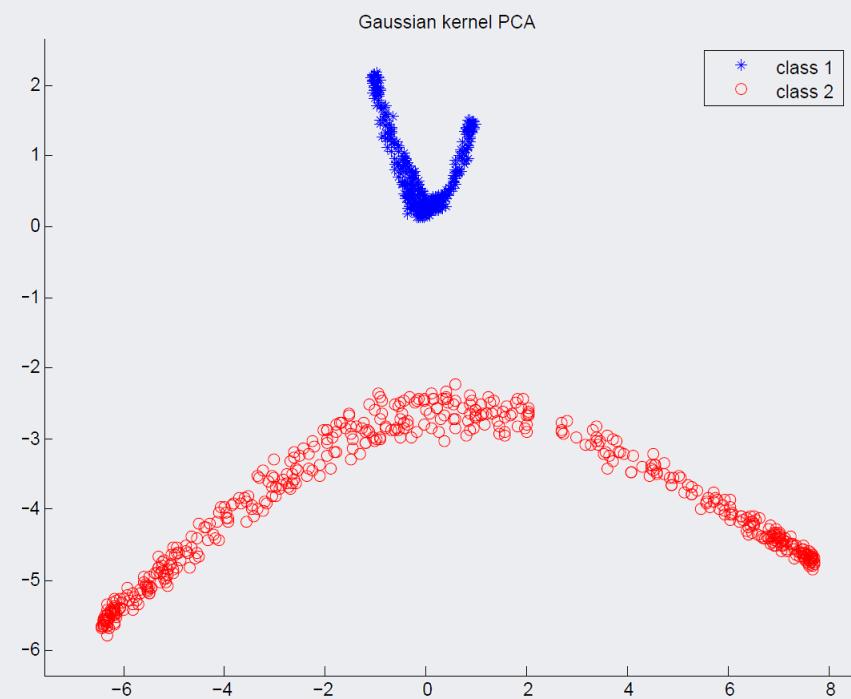
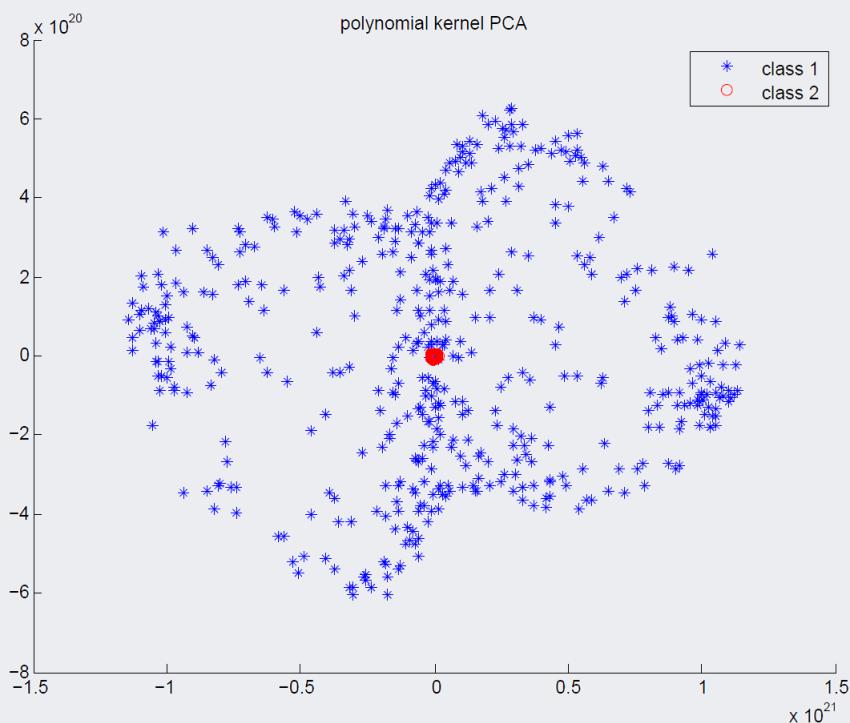
Kernel PCA

- PCA for two sphere sets



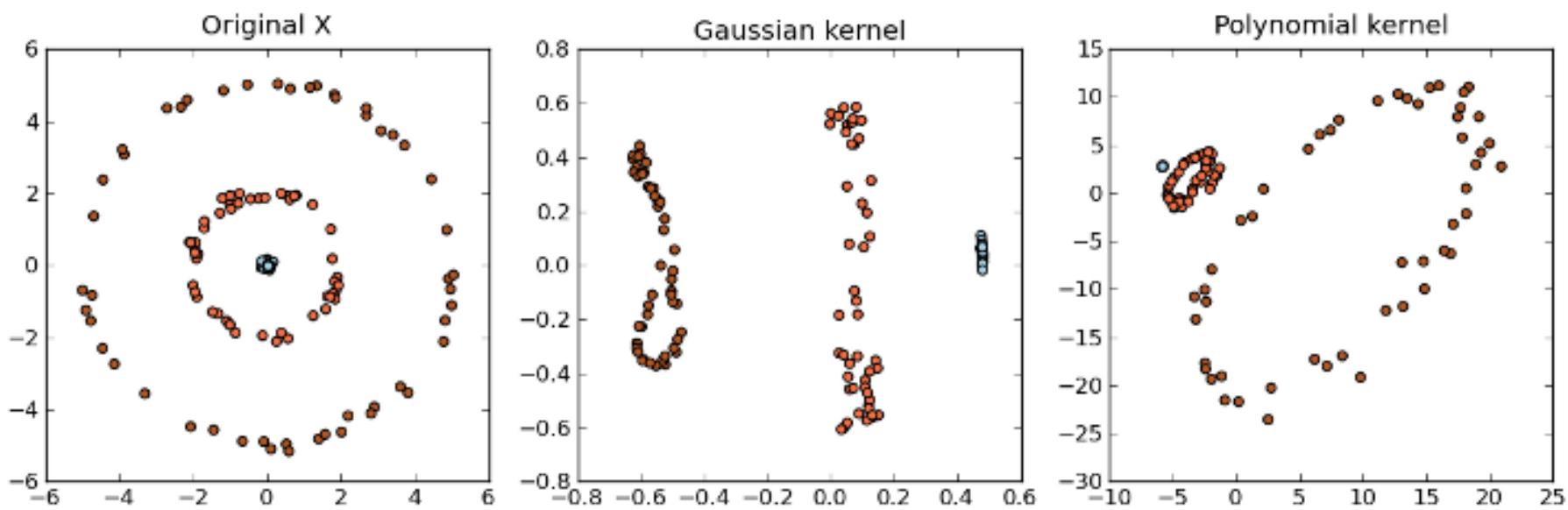
Kernel PCA

- KPCA for two sphere sets



Kernel PCA

- Projection result according to different kernel types



Kernel PCA: Application

- De-noising images

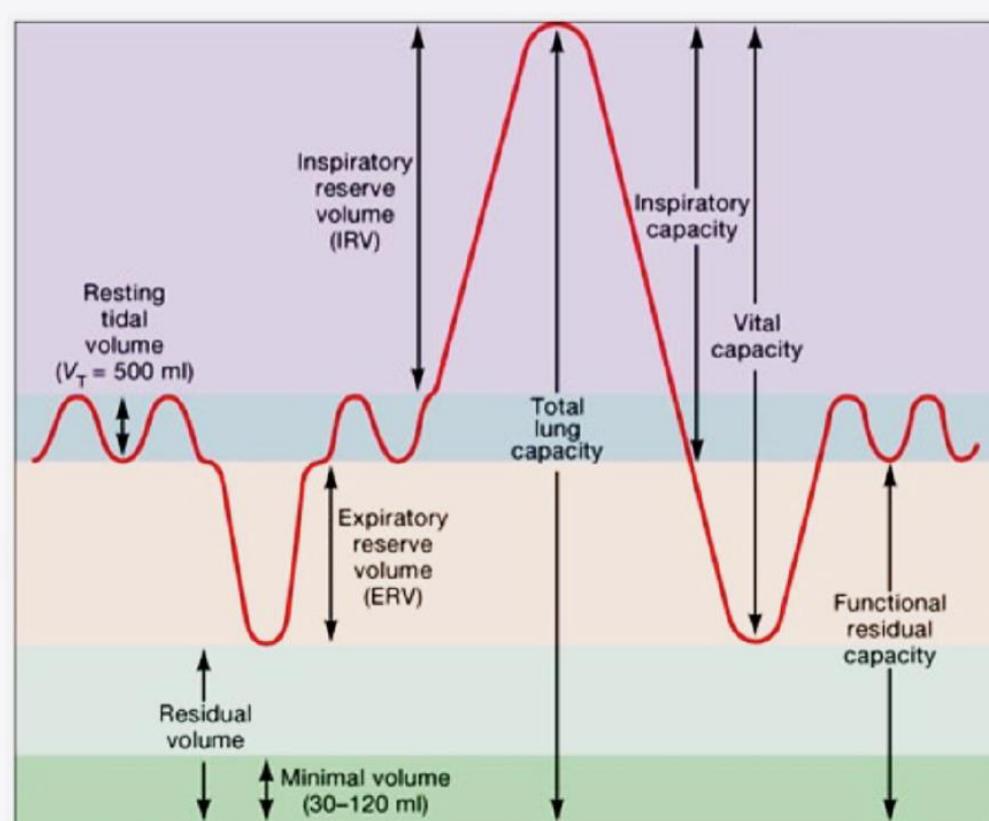
	Gaussian noise	'speckle' noise
PCA	orig.	
	noisy	
	$n = 1$	
	4	
	16	
	64	
	256	
	$n = 1$	
	4	
	16	
KPCA	64	
	256	

KPCA Application

Widjaja et al. (2012)

- Question:

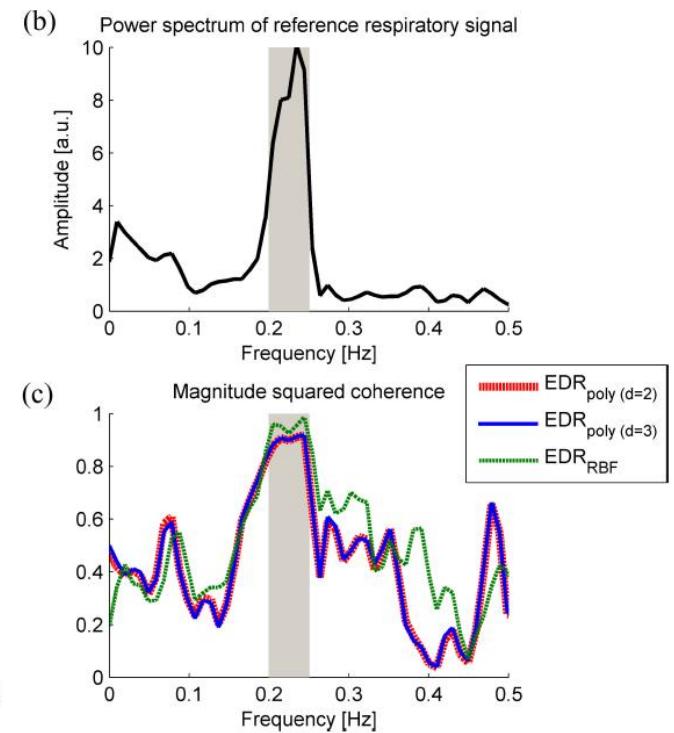
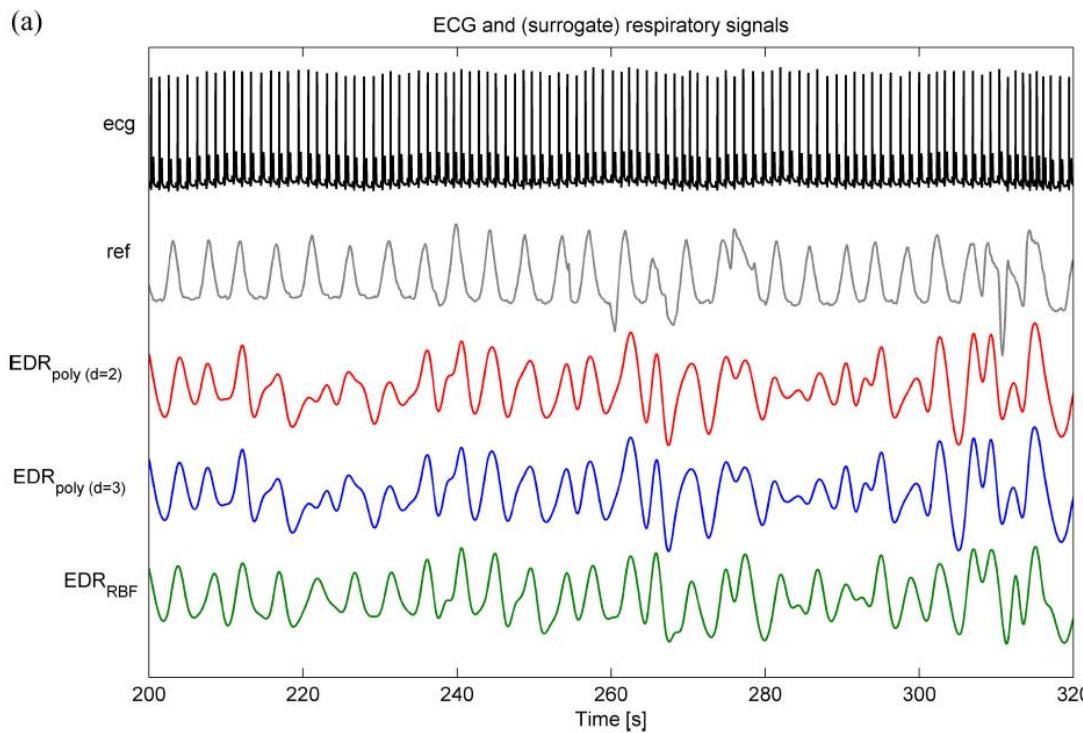
- ✓ How can spontaneous respiration be derived from the **Electrocardiograms**?
- ✓ An example of EDR signal



KPCA Application

Widjaja et al. (2012)

- KPCA for ECG signal





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.
- Mika, S. Kernel Fisher Discriminants. Ph.D Thesis. https://opus4.kobv.de/opus4-tuberlin/files/482/mika_sebastian.pdf
- Mika, S., Schölkopf, B., Smola, A. J., Müller, K. R., Scholz, M., and Rätsch, G. (1998). Kernel PCA and de-noising in feature spaces. In *Proceedings of Neural Information Processing Systems (NIPS'98)*.
- 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.
- Schölkopf, B., Smola, A., and Müller, K. R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation* 10(5): 1299-1319.
- Smola A.J. and Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and Computing* 14(3): 199-222.
- Widjaja, D., Varon, C., Dorado, A., Suykens, J.A., & Van Huffel, S. (2012). Application of kernel principal component analysis for single-lead-ECG-derived respiration. *IEEE Transactions on Biomedical Engineering*, 59(4), 1169-1176.
- Zhang, C., Bengio, S., Hardt, M., Recht, B., & Vinyals, O. (2016). Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*.

References

Other materials

- <http://www.cs.nyu.edu/~mohri/icml2011-tutorial/>
- Suykens, J. (2003). [Least Squares Support Vector Machines](#). IJCNN 2003 Tutorial.
- Abu-Mostafa, Y. (2012). [Lecture 14: Support Vector Machines](#). Learning From Data. Caltech.
- <http://www.ci.tuwien.ac.at/~meyer/svm/final.pdf>
- <http://pubs.rsc.org/en/content/articlehtml/2010/an/b918972f>
- Zhang, C., Bengio, S., Hardt, M., Recht, B., & Vinyals, O. (2017b). Understanding deep learning requires rethinking generalization. International Conference on Learning Representations. [Slides] <http://pluskid.org/slides/ICLR2017.key>
- Zhang, C., Bengio, S., Hardt, M., Recht, B., & Vinyals, O. (2017c). Understanding deep learning requires rethinking generalization. International Conference on Learning Representations. [Poster] <http://pluskid.org/slides/ICLR2017-Poster.pdf>
- Park, J.S. (2017). Deep learning and data. DSBA Lab Seminar. <http://dsba.korea.ac.kr/wp/wp-content/seminar/Paper%20Review/Deep%20Learning%20and%20Data%20-%20%EB%B0%95%EC%9E%AC%EC%84%A0.pdf>