

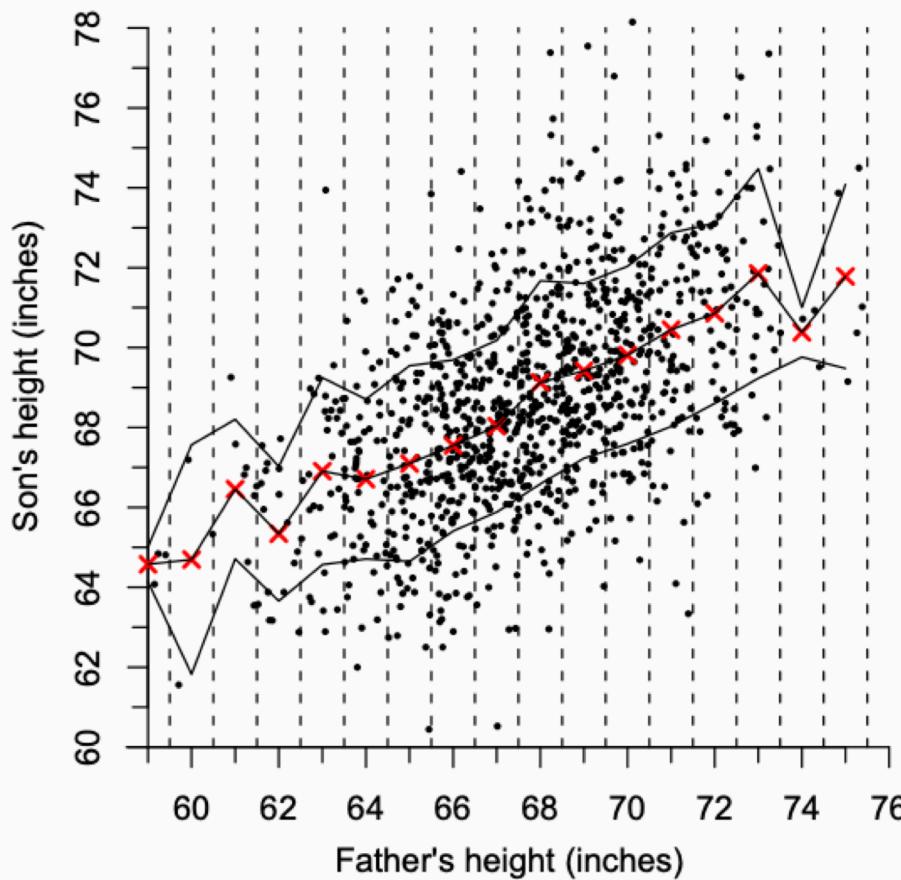
Statistical Learning and Computational Finance Lab.  
Department of Industrial Engineering  
<http://slcf.snu.ac.kr>

## Lecture 8: Regression, Decision Tree, SVM, PCA

# Regression

# Linear Regression

Example : Pearson's Father-and-Son Data



아버지의 키와 아들의 키와의 관계?

$y = \beta_0 + \beta_1 x$  ( $x$  : father's height,  $y$  : son's height)

# Linear Regression

## Regression Model

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} + \varepsilon_i$$

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$$

$$\begin{bmatrix} 1 & x_{11} & \cdots & x_{1p} \\ 1 & x_{21} & \cdots & x_{2p} \\ \vdots & \ddots & \vdots & \vdots \\ 1 & x_{n1} & \cdots & x_{np} \end{bmatrix} \begin{bmatrix} \boldsymbol{\beta}_0 \\ \boldsymbol{\beta}_1 \\ \vdots \\ \boldsymbol{\beta}_p \end{bmatrix} + \begin{bmatrix} \boldsymbol{\varepsilon}_1 \\ \boldsymbol{\varepsilon}_2 \\ \vdots \\ \boldsymbol{\varepsilon}_n \end{bmatrix}$$

Find  $\boldsymbol{\beta} \Rightarrow \text{minimize } \boldsymbol{\varepsilon}^T \boldsymbol{\varepsilon}$

# Linear Regression

Minimize  $\varepsilon^T \varepsilon$

Choose  $\hat{\beta}$  to minimize

$$\begin{aligned} \text{SSE} &= \sum_{i=1}^n e_i^2 = \sum_{i=1}^n (y_i - \beta_0 - \beta_1 x_{i1} - \beta_2 x_{i2} - \dots - \beta_p x_{ip})^2 \\ &= \mathbf{e}^T \mathbf{e} = (\mathbf{y} - \hat{\mathbf{y}})^T (\mathbf{y} - \hat{\mathbf{y}}) = (\mathbf{y} - \mathbf{X}\hat{\beta})^T (\mathbf{y} - \mathbf{X}\hat{\beta}) \end{aligned}$$

To solve the problem

$$\frac{\partial \text{SSE}}{\partial \beta} = 0 \quad \text{implies} \quad 2\mathbf{X}^T(\mathbf{y} - \mathbf{X}\hat{\beta}) = \mathbf{0} \Rightarrow \mathbf{X}^T \mathbf{X} \hat{\beta} = \mathbf{X}^T \mathbf{y} \quad (\text{the normal equations})$$

Solving for  $\hat{\beta}$ , we get (analogous to  $\hat{\beta} = \frac{S_{xy}}{S_{xx}}$ )

$$\hat{\beta} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$

# Ridge Regression, Lasso Regression

$\beta$  값 자체에 대해 Penalty 를 줘서 모델을 단순하게 만들 수 있음

Linear Regression :  $(Y - X\beta)^T(Y - X\beta)$  을 최소화하는  $\beta$ 를 찾자

Ridge Regression :  $(Y - X\beta)^T(Y - X\beta) + \lambda\beta^T\beta$  을 최소화하는  $\beta$  를 찾자

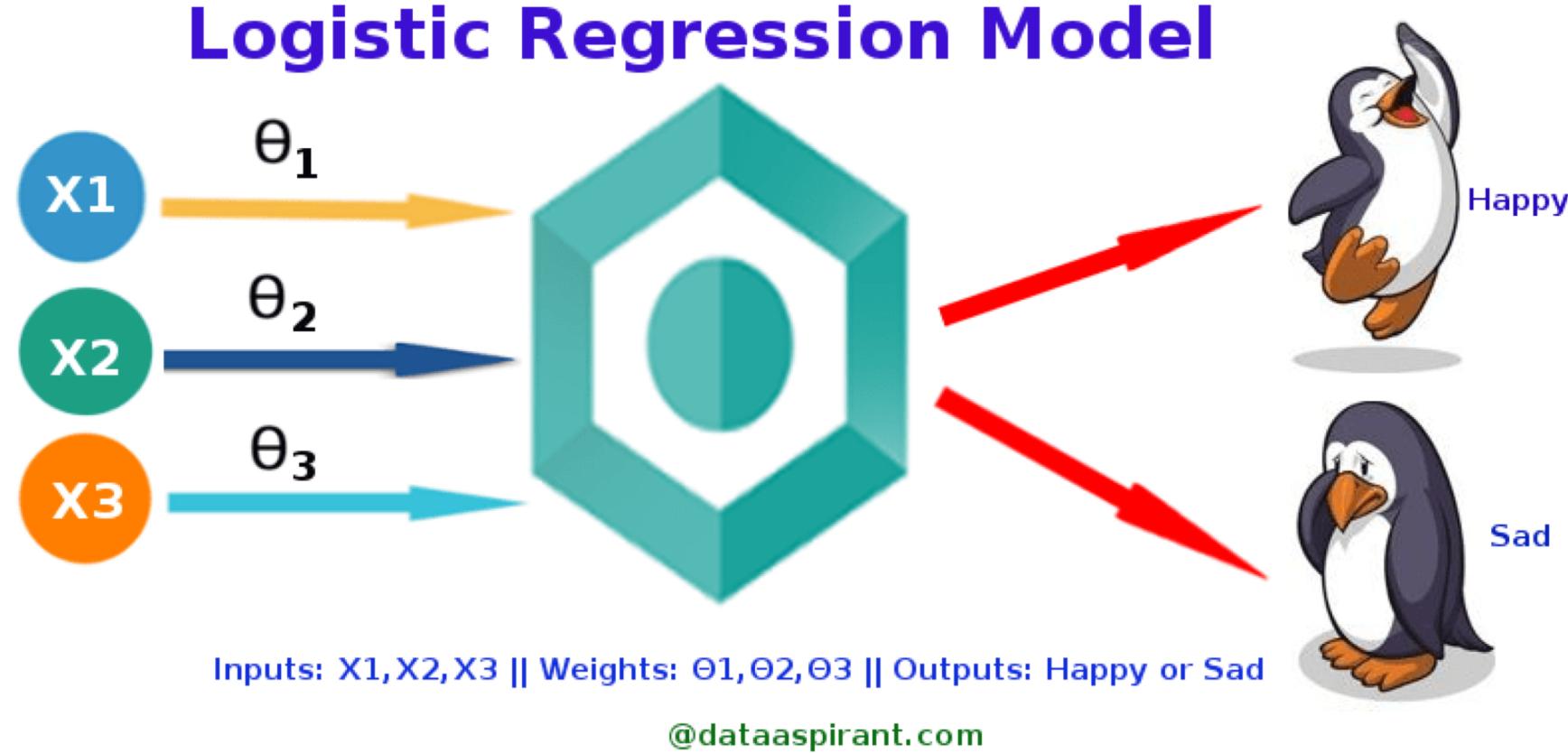
Lasso Regression :  $(Y - X\beta)^T(Y - X\beta) + \lambda|\beta|_1$  을 최소화하는  $\beta$  를 찾자

=>  $\beta$ 가 지나치게 커지는 것을 방지

# Logistic Regression

Linear Regression is not suitable for classification

## Logistic Regression Model



Source: <http://dataaspirant.com/2017/03/02/how-logistic-regression-model-works/>

# Logistic Regression

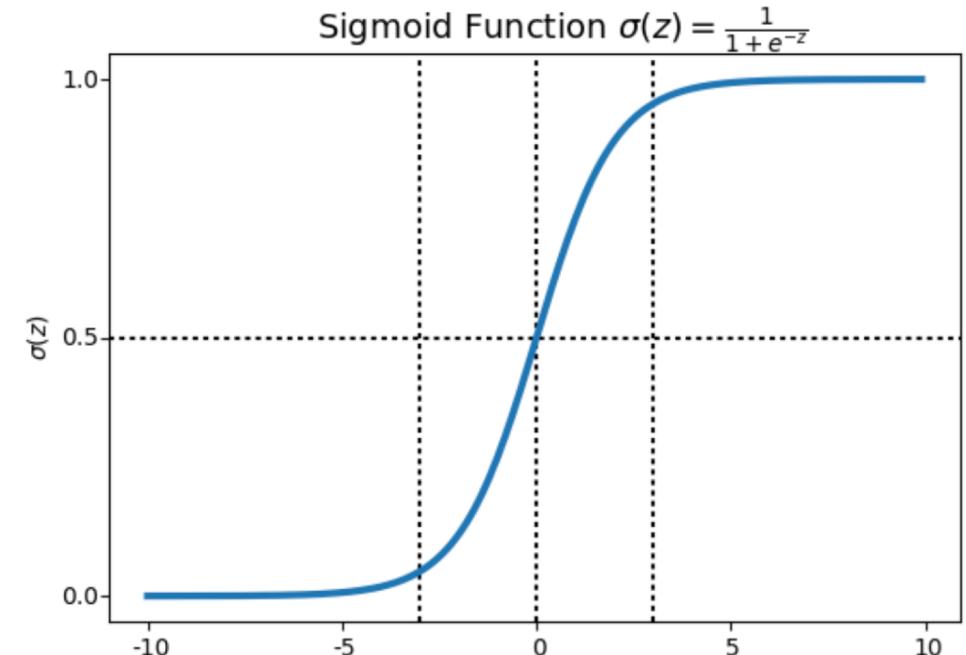
간단한 Logistic Regression 예시 : Binary Classification

Output : 0 or 1

Linear Regression의 경우 output의 범위 제한이 없음  $(-\infty, \infty)$

Regression의 output이 0과 1 사이의 확률이면?

Sigmoid function :  $\frac{1}{1+e^{-z}}$

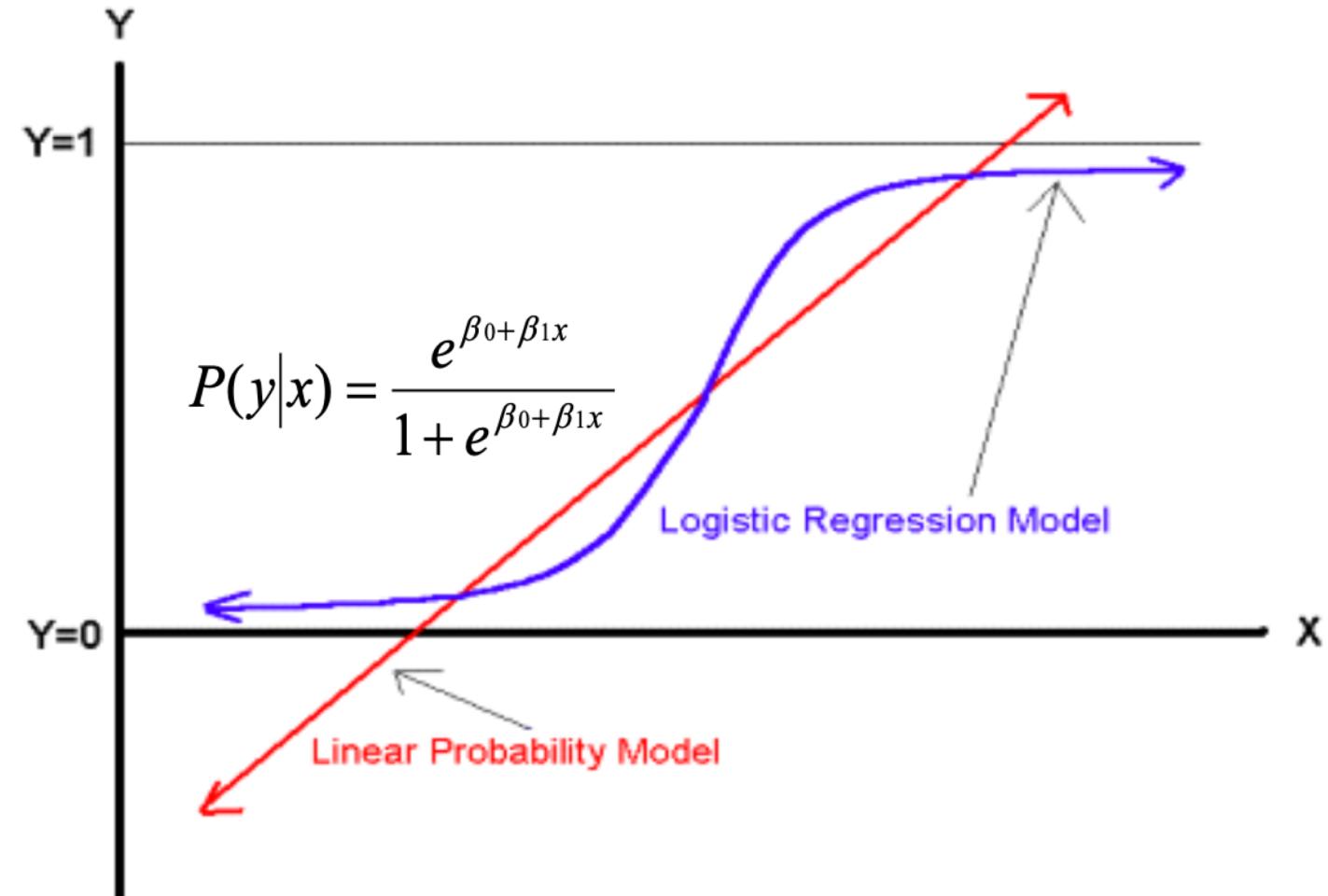


# Logistic Regression

$x$ 가 주어졌을 때  $y=1$  클래스에  
해당할 확률로 생각!

$$P(y = 1|x) = \sigma(\beta^T x) = \frac{1}{1 + e^{-\beta^T x}}$$

Linear Regression과 마찬가지로,  
 $\beta$ 값을 구하는 것이 목표



# Practice

[https://github.com/SLCFLAB/Data-Science-Python/blob/main/Day%208/8\\_1.Regression.ipynb](https://github.com/SLCFLAB/Data-Science-Python/blob/main/Day%208/8_1.Regression.ipynb)

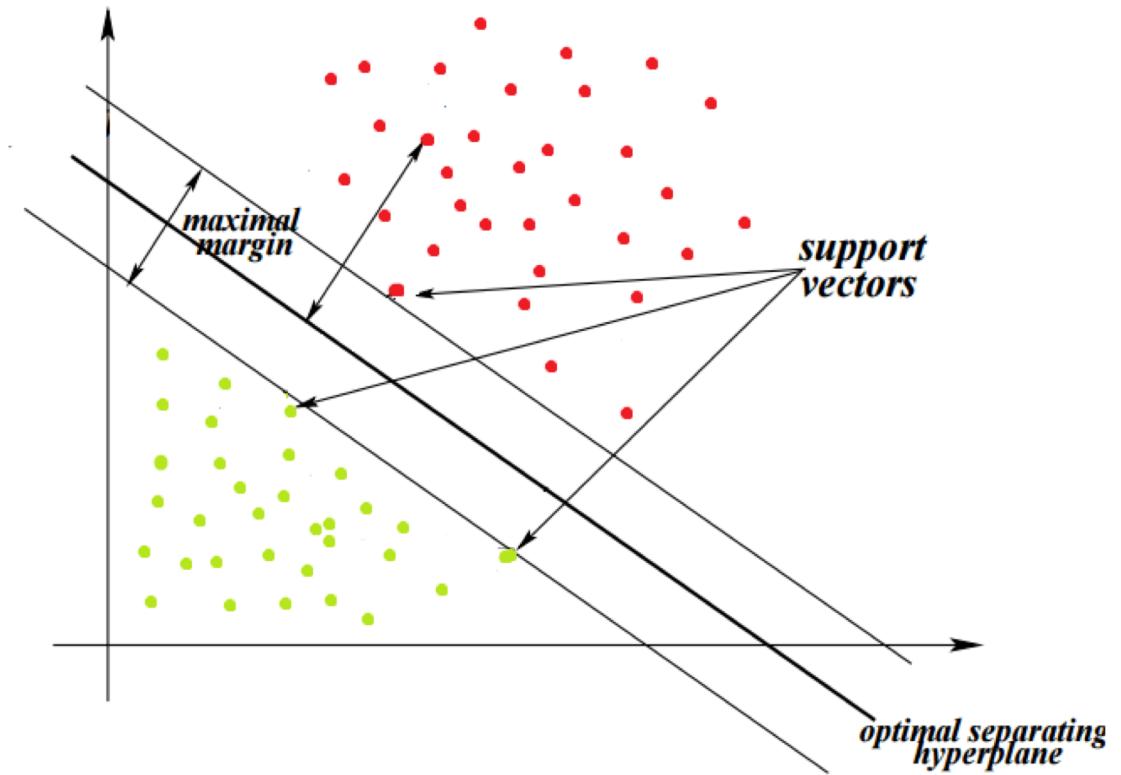
# Support Vector Machine

# Support Vector Machine

What is SVM (Support Vector Machine)?

supervised learning models for classification  
and regression problems

Find a hyperplane that maximizes the margin of  
the classifier



Source: [tinyurl.com/43544syy](http://tinyurl.com/43544syy)

# Support Vector Machine

$$w \cdot x + b = -1$$

$$w \cdot x + b = +1$$

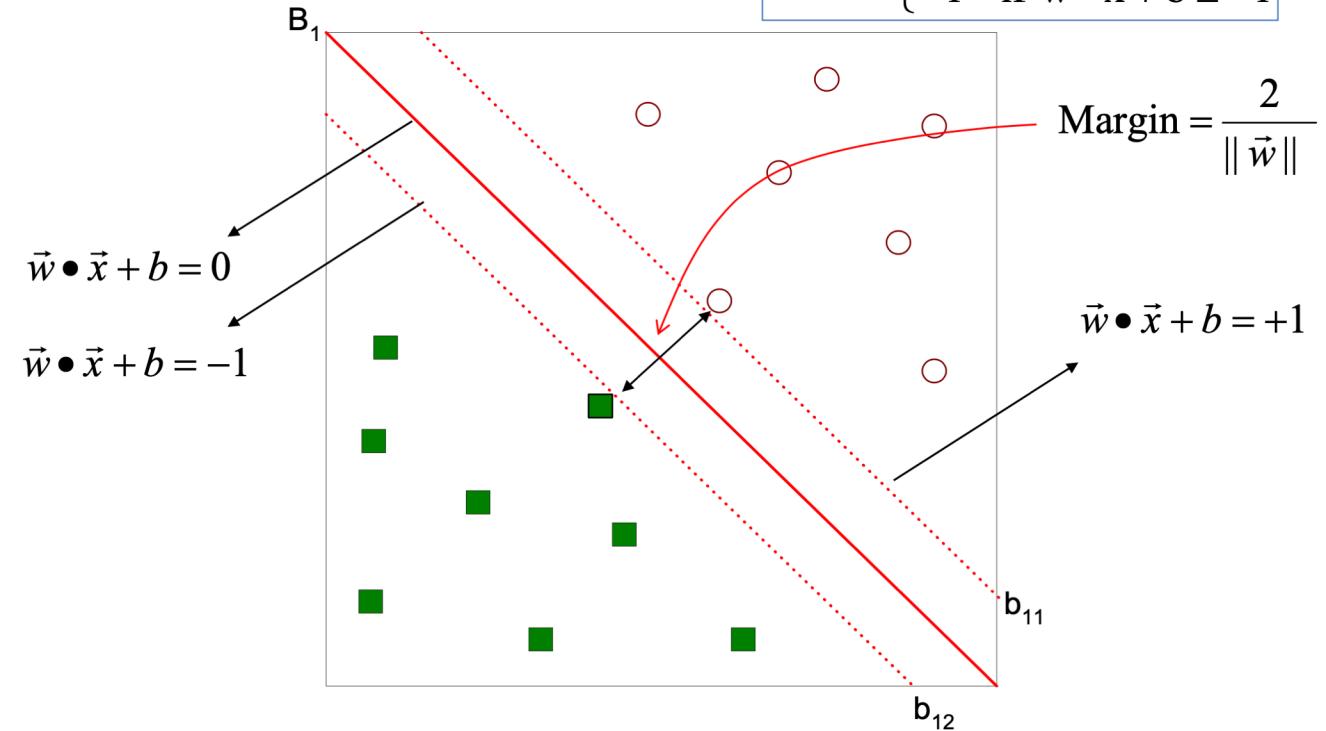
$$\text{Margin} = \frac{2}{\|w\|}$$

maximize margin  $\Leftrightarrow$  minimize  $\|w\|$

$$\min_{w,b} \Phi(w) = \frac{1}{2} w^T w$$

subject to  $y_i(w^T x_i + b) \geq 1, \quad i = 1, \dots, N$

$$f(\vec{x}) = \begin{cases} 1 & \text{if } \vec{w} \bullet \vec{x} + b \geq 1 \\ -1 & \text{if } \vec{w} \bullet \vec{x} + b \leq -1 \end{cases}$$



# Support Vector Machine

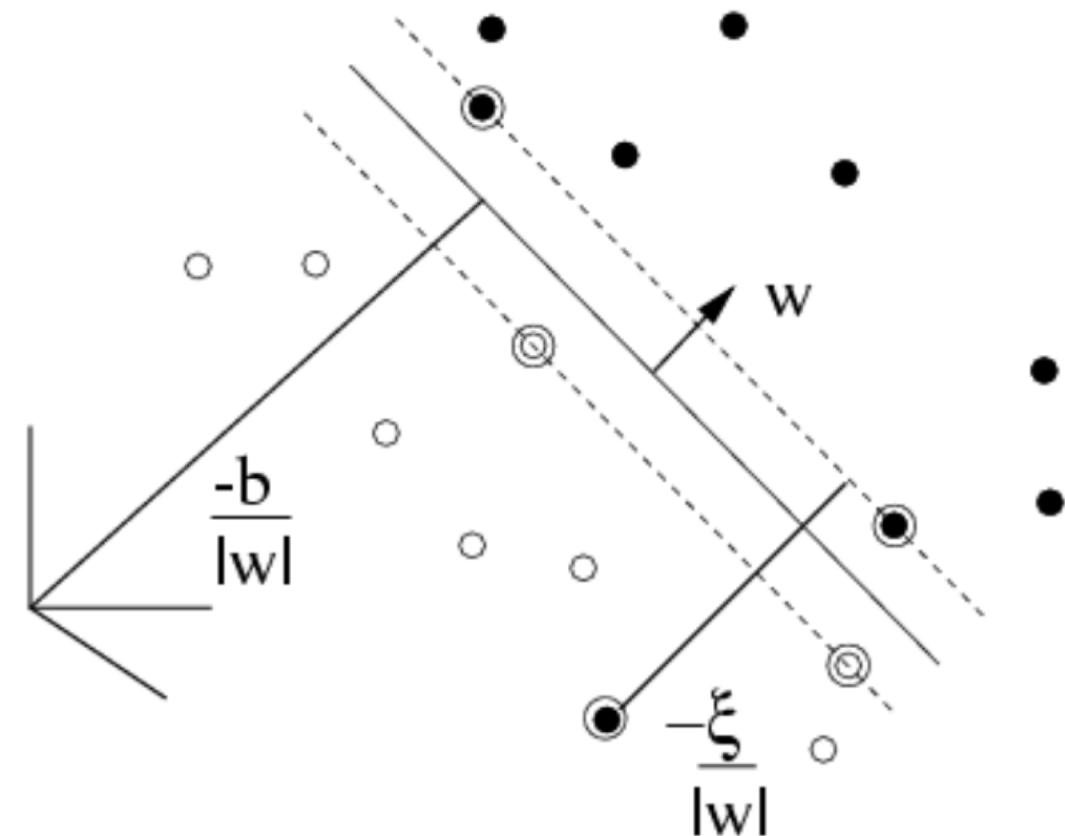
Not linearly separable ?

Consider slack variables : $\xi$

Allow for outliers

Objective function

$$\frac{\|w\|^2}{2} \longrightarrow \frac{\|w\|^2}{2} + C \left( \sum_i \xi_i \right)^k$$

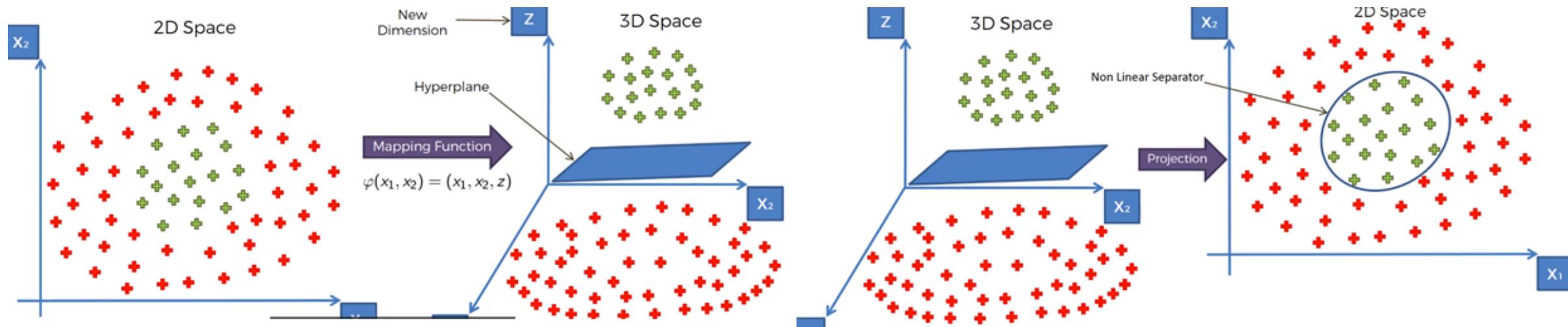


# Support Vector Machine

## Kernel SVM

고차원 공간으로 바꾸어서 linearly separable하게끔 만듬

kernel 종류에 따라 다양 : polynomial kernel, rbf kernel, exponential kernel, ...



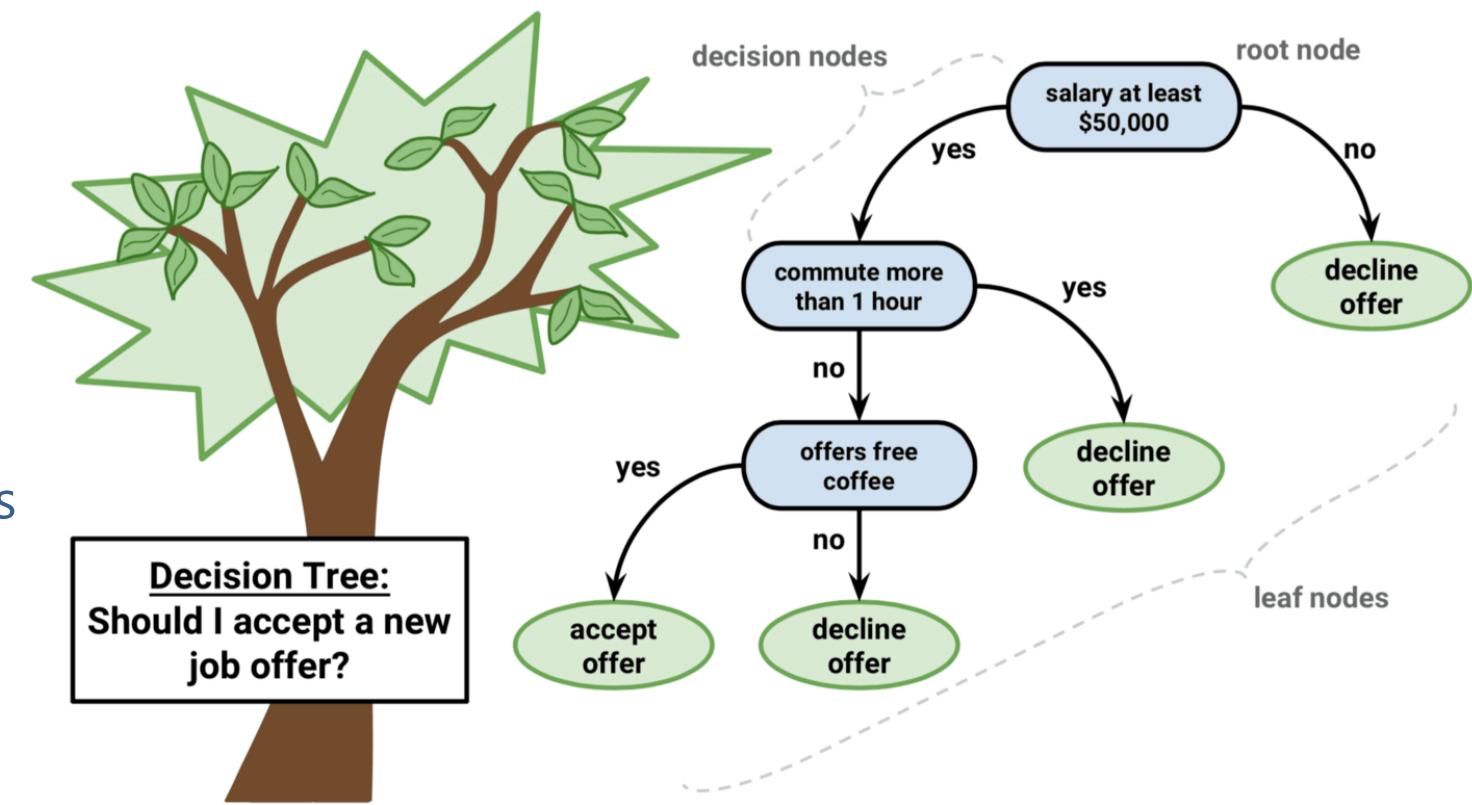
# Practice

[https://github.com/SLCFLAB/Data-Science-Python/blob/main/Day%208/8\\_2.SVM.ipynb](https://github.com/SLCFLAB/Data-Science-Python/blob/main/Day%208/8_2.SVM.ipynb)

# Decision Tree

# Decision Tree

Non-parametric supervised learning method for classification  
predicts the value of a target variable by learning simple decision rules inferred from the data features



# Decision Tree

## Advantages

Simple to understand and to visualize

Cost of using the tree is logarithmic in the number of data points

Able to handle both numerical and categorical data

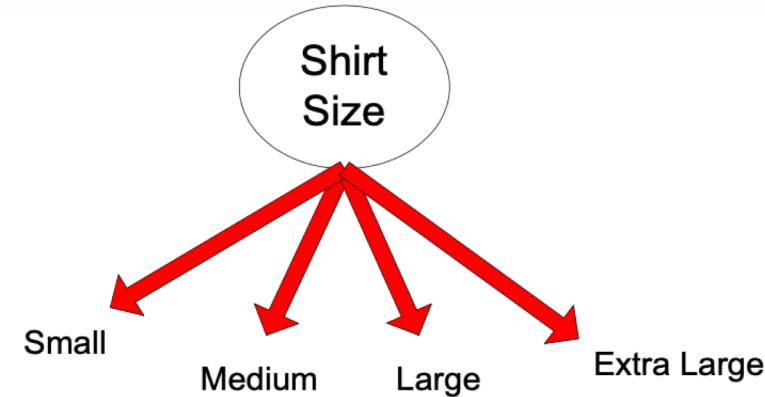
## Disadvantages

Overfitting problem

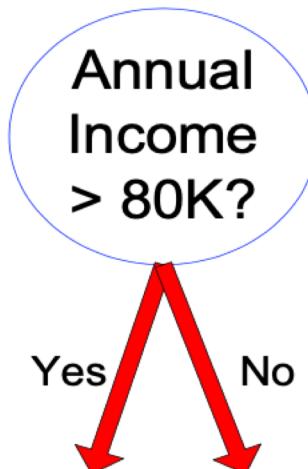
Does not take into account interactions between attributes

# Decision Tree

Multi way split



Binary split



# Decision Tree

How to determine the best split?

Calculate information gain

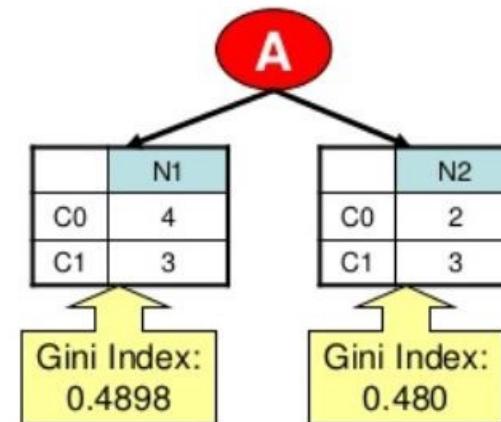
Gini index

$$GINI(t) = 1 - \sum_j [p(j | t)]^2$$

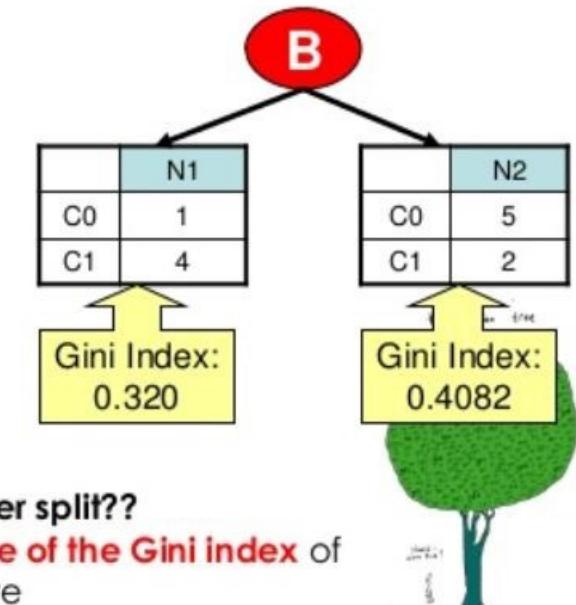
Entropy

Misclassification error

Suppose there are two ways (A and B) to split the data into smaller subset.



Which one is a better split??  
Compute the **weighted average of the Gini index** of both attribute



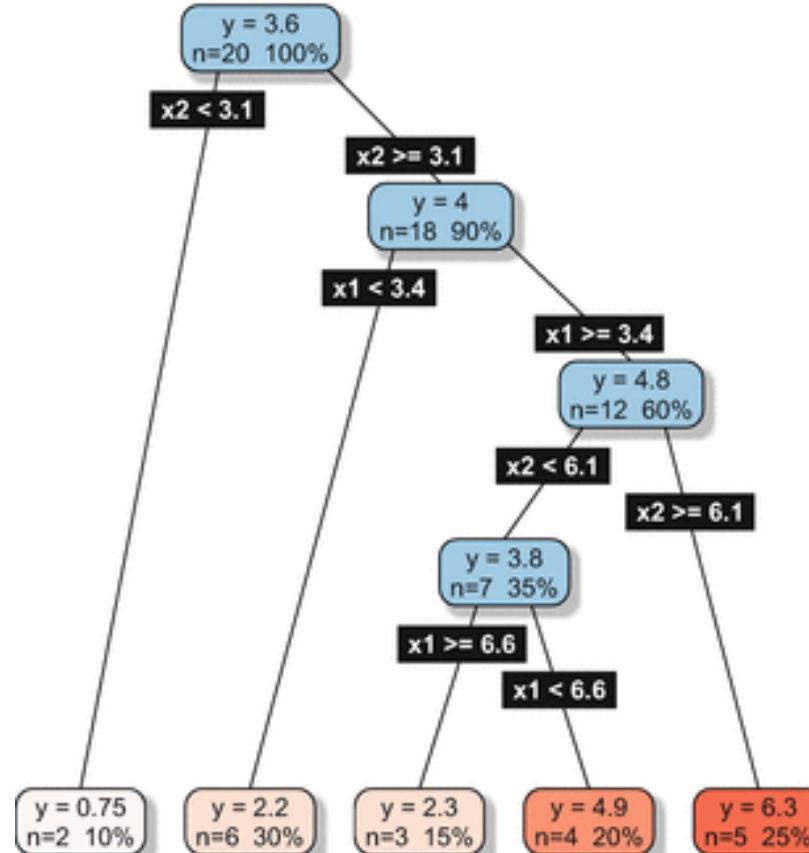
# Regression Tree

Regression case

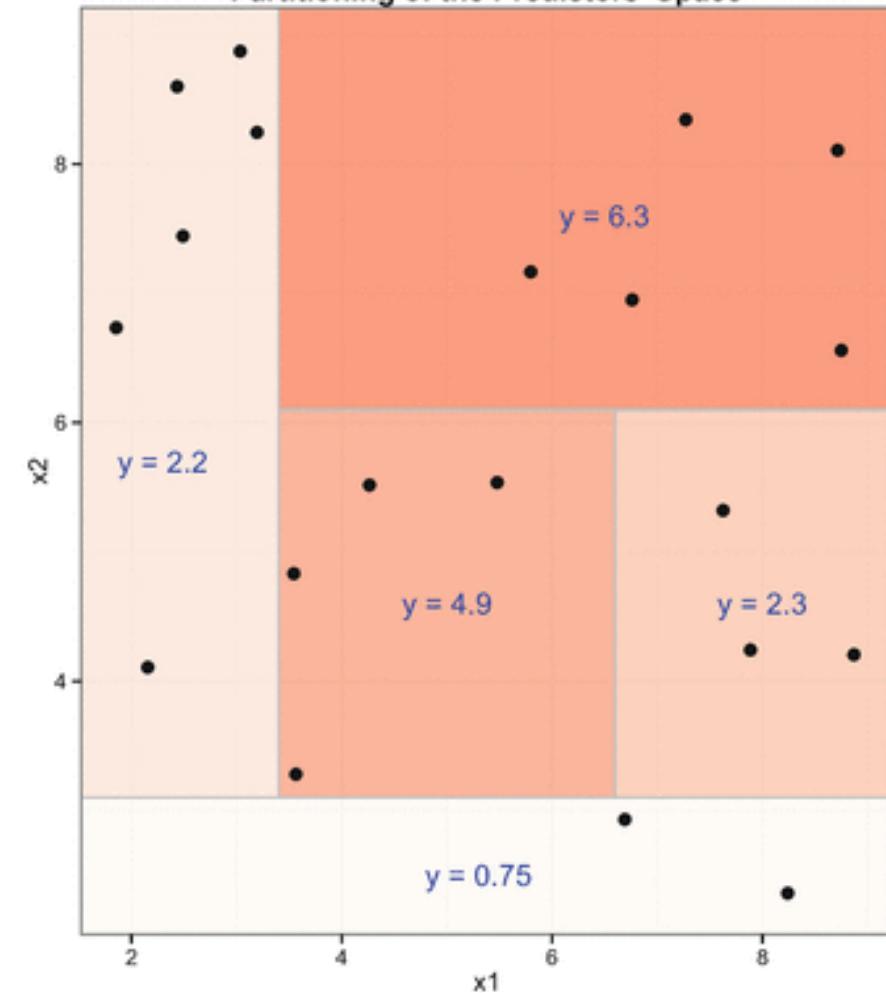
$Average(y|x_i \in R_m)$

속한 그룹에 있는 다른  
데이터들의  $y$ 값의  
평균으로 근사시킴

Example of a Regression Tree

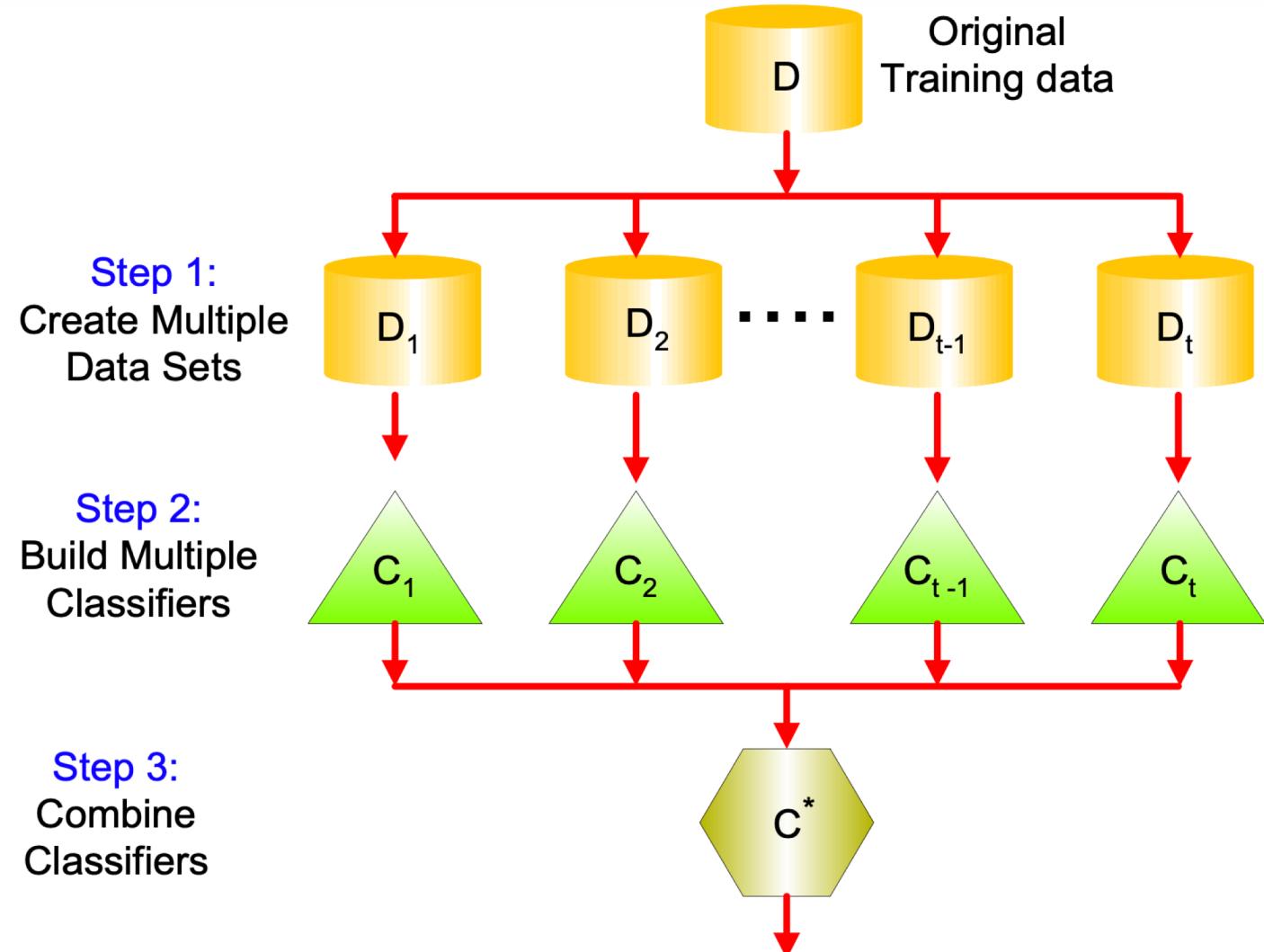


Partitioning of the Predictors' Space



# Ensemble

트레이닝 데이터를 여러 개로  
나누어 각 데이터셋에 대해 모델  
여러 개 학습  
학습된 분류 모델들이 내는 결과를  
종합 (평균, 다수결 등) 하여 최종  
결과 선정



# Bagging

## Bootstrap Aggregation

입력 데이터를 모델 수만큼 나눈  
뒤 각각 학습시킴

출력되어 나온 예측 값들로  
voting하여 majority vote를 받은  
값이 최종 예측값이 됨

Original Data:

x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
y	1	1	1	-1	-1	-1	-1	1	1	1

x	0.1	0.2	0.2	0.3	0.4	0.4	0.5	0.6	0.9	0.9
y	1	1	1	1	-1	-1	-1	-1	1	1

x	0.1	0.2	0.3	0.4	0.5	0.5	0.9	1	1	1
y	1	1	1	-1	-1	-1	1	1	1	1

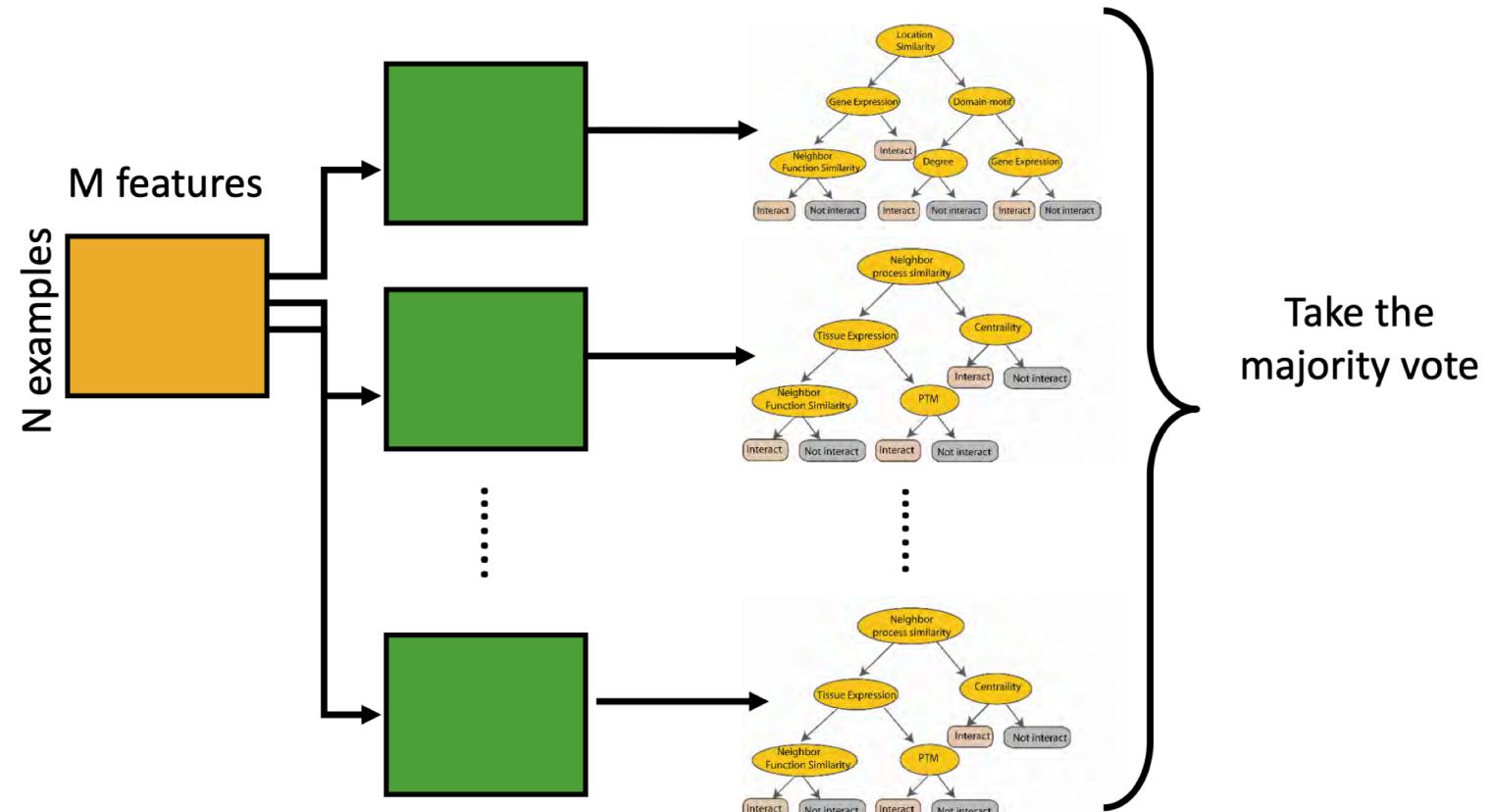
x	0.1	0.2	0.3	0.4	0.4	0.5	0.7	0.7	0.8	0.9
y	1	1	1	-1	-1	-1	-1	-1	1	1

x	0.1	0.2	0.5	0.5	0.5	0.7	0.7	0.8	0.9	1
y	1	1	-1	-1	-1	-1	-1	1	1	1

# Random Forest

bagging의 extension

각 데이터셋에 대해 모든  
feature를 다 쓰는 것이 아니라,  
feature들 중 일부만을 랜덤하게  
골라서 사용



# Boosting

## AdaBoost

Records that are wrongly classified will have their weights increased

Records that are classified correctly will have their weights decreased

e.g. data 4 is hard to classify during Round 1 and Round 2 => weights increased

Original Data	1	2	3	4	5	6	7	8	9	10
Boosting (Round 1)	7	3	2	8	7	9	4	10	6	3
Boosting (Round 2)	5	4	9	4	2	5	1	7	4	2
Boosting (Round 3)	4	4	8	10	4	5	4	6	3	4

# Boosting

AdaBoost

Boosting Round 1:

x	0.1	0.4	0.5	0.6	0.6	0.7	0.7	0.7	0.8	1
y	1	-1	-1	-1	-1	-1	-1	-1	1	1

Boosting Round 2:

x	0.1	0.1	0.2	0.2	0.2	0.2	0.3	0.3	0.3	0.3
y	1	1	1	1	1	1	1	1	1	1

Boosting Round 3:

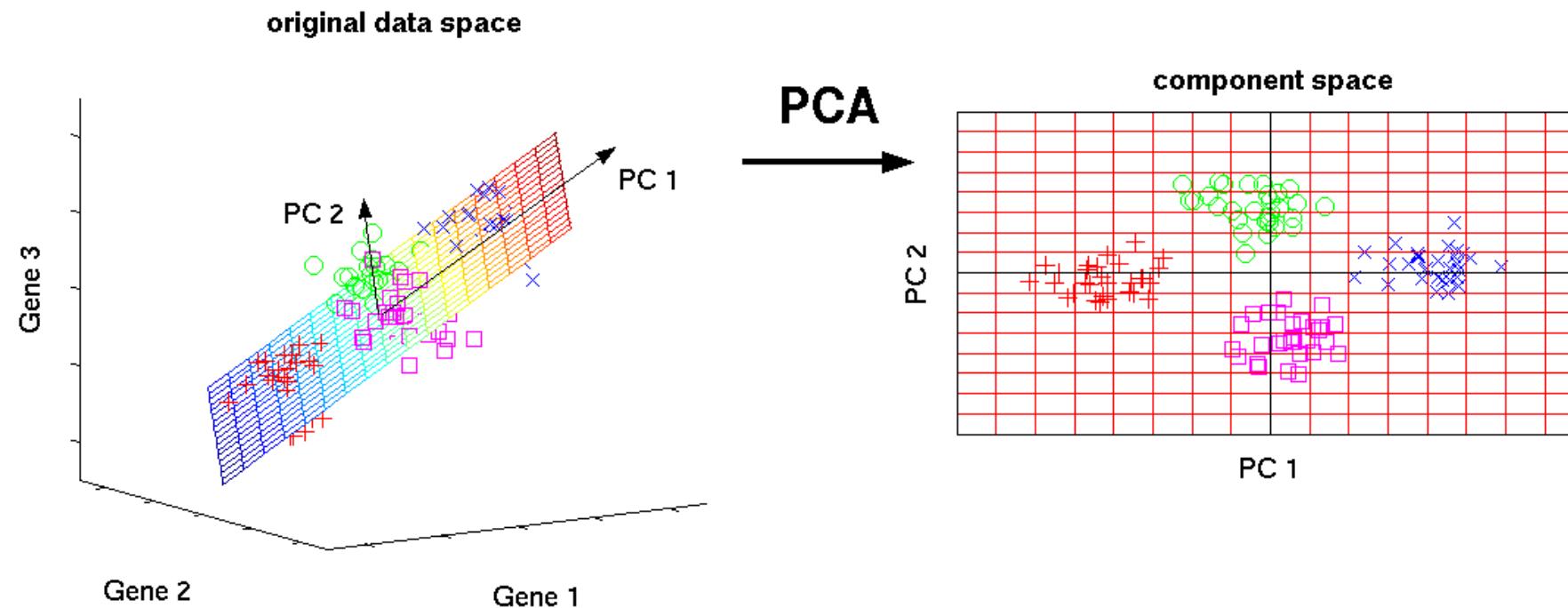
x	0.2	0.2	0.4	0.4	0.4	0.4	0.5	0.6	0.6	0.7
y	1	1	-1	-1	-1	-1	-1	-1	-1	-1

# Principal Components Analysis

# PCA

Dimensionality-reduction technique

"Curse of Dimensionality" – how to handle it?



# PCA

Variance를 최대화하는 축을 찾자 (SVD 분해와 관련)

$$\mathbf{X}\mathbf{V} = U\Sigma \rightarrow \mathbf{Z}_c = \mathbf{X}\mathbf{V}_{:,1:c} = U\Sigma_{:,1:c}$$

$$\mathbf{Z}_c = \begin{bmatrix} \mathbf{z}_1^T \\ \mathbf{z}_2^T \\ \vdots \\ \mathbf{z}_N^T \end{bmatrix} = \mathbf{X}\mathbf{V}_{:,1:c} = \begin{bmatrix} \mathbf{x}_1^T \mathbf{v}_1 & \mathbf{x}_1^T \mathbf{v}_2 & \cdots & \mathbf{x}_1^T \mathbf{v}_c \\ \mathbf{x}_2^T \mathbf{v}_1 & \mathbf{x}_2^T \mathbf{v}_2 & \cdots & \mathbf{x}_N^T \mathbf{v}_c \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{x}_N^T \mathbf{v}_1 & \mathbf{x}_N^T \mathbf{v}_2 & \cdots & \mathbf{x}_N^T \mathbf{v}_c \end{bmatrix} = U\Sigma_{:,1:c} = \begin{bmatrix} \vdots & \vdots & \cdots & \vdots \\ \sigma_1 \mathbf{u}_1 & \sigma_2 \mathbf{u}_2 & \cdots & \sigma_c \mathbf{u}_c \\ \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \cdots & \vdots \end{bmatrix}$$

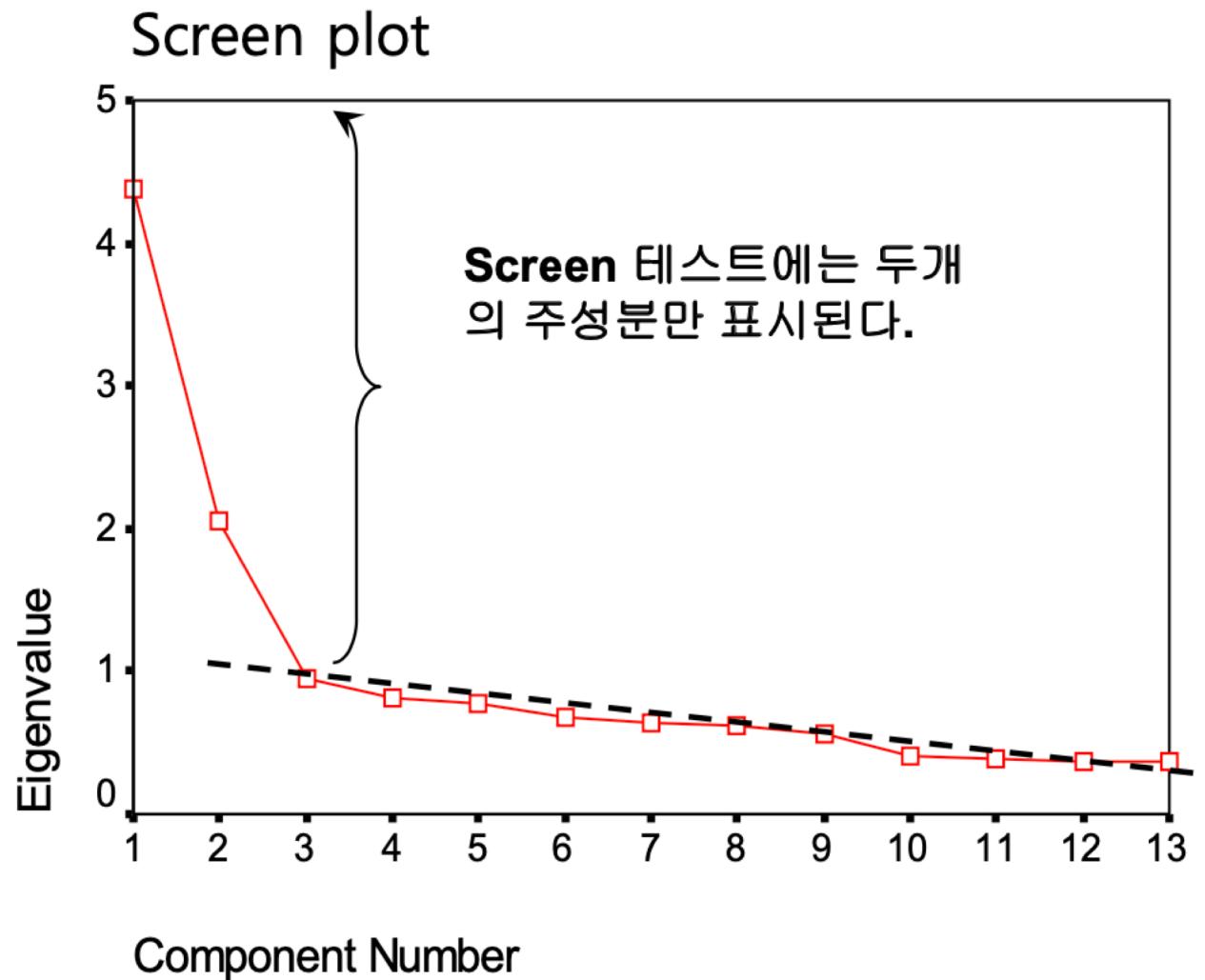
Variance를 최대화하는 축 순서대로 : SVD 분해 후 나오는  $\sigma_1 u_1$  (첫번째 축),  $\sigma_2 u_2$  (두번째 축) ...

축들의 Variance의 합은 곧 “설명력”

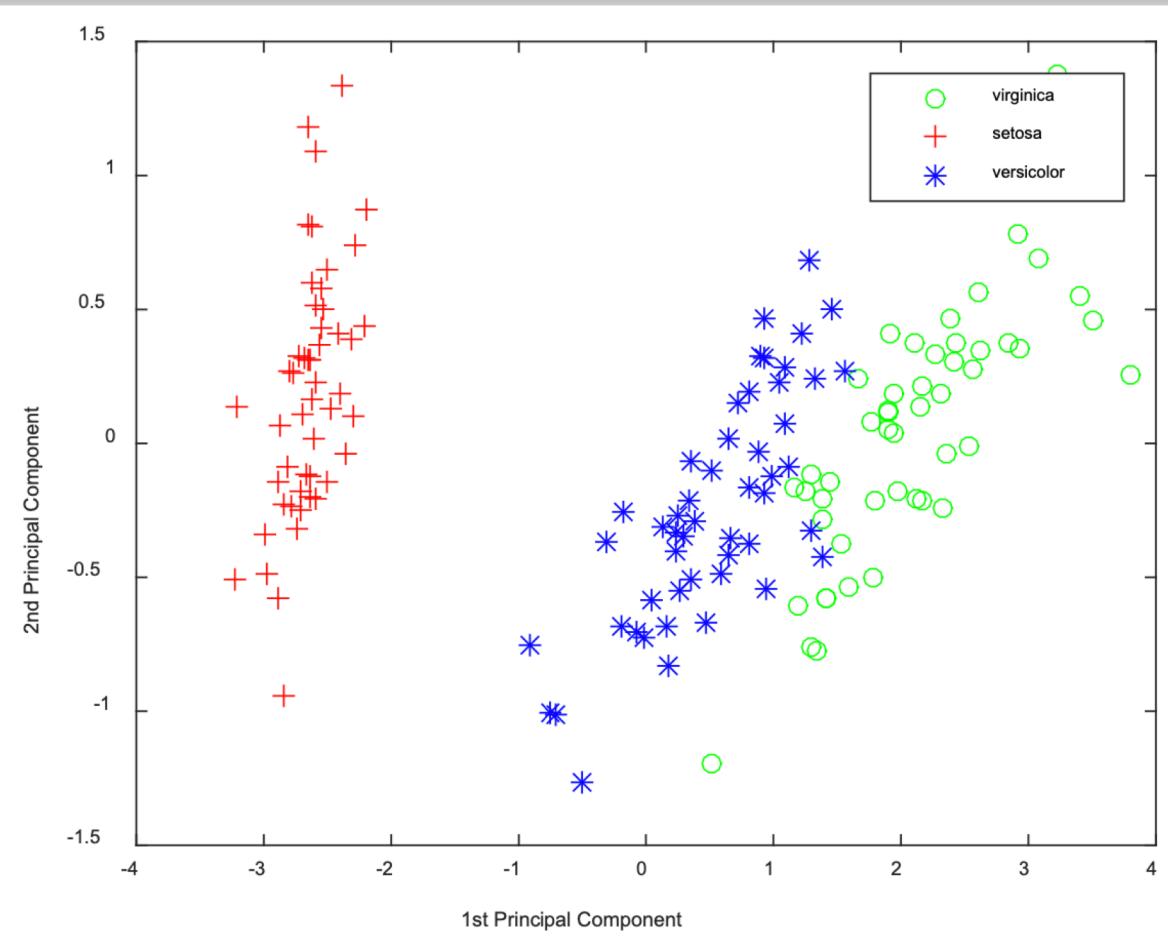
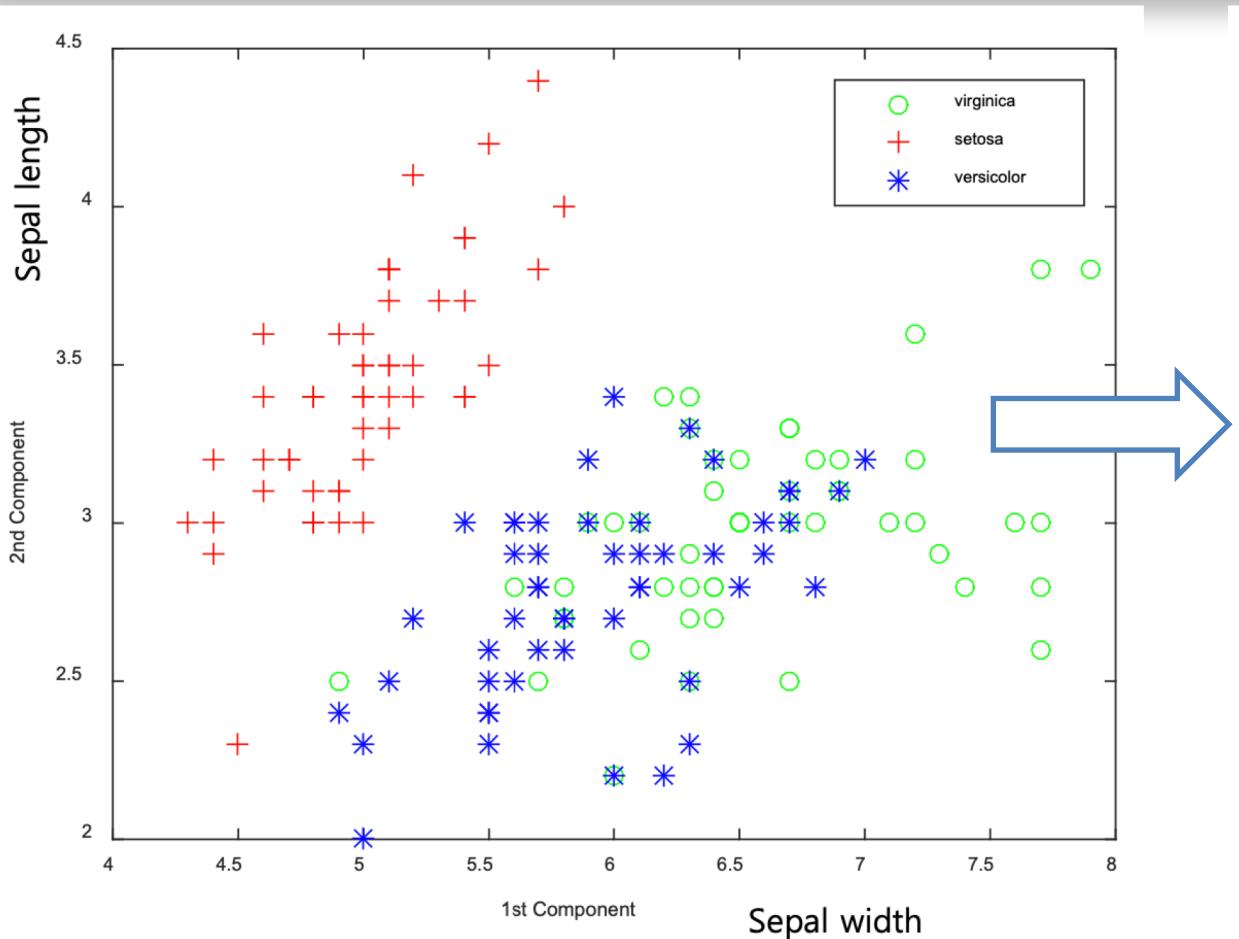
$var(z_1 + z_2 + \cdots) =$  Eigenvalue의 합과 같음

구한 각 축은  $x_k$  들의 linear combination

# PCA

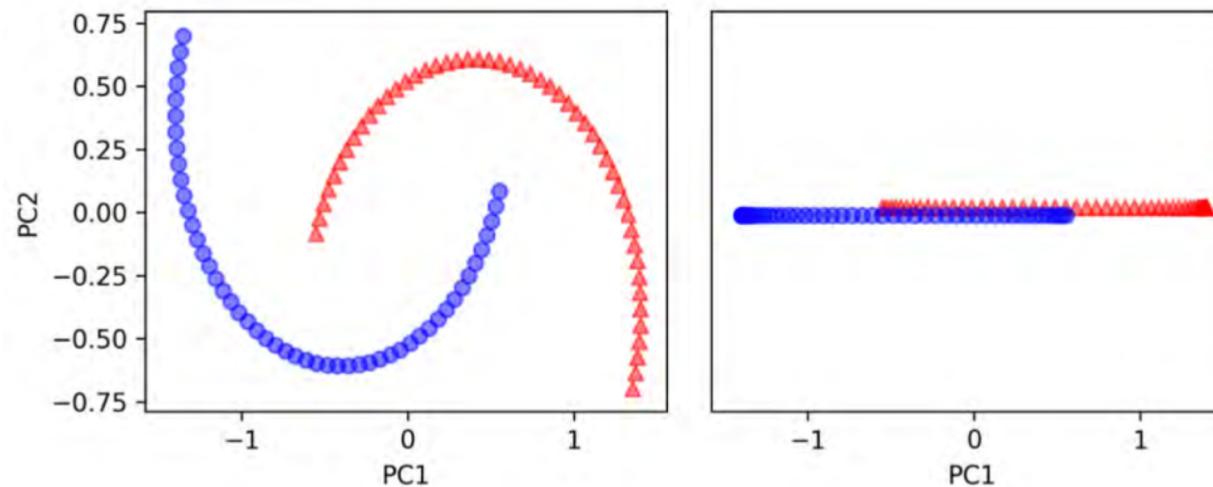


# PCA

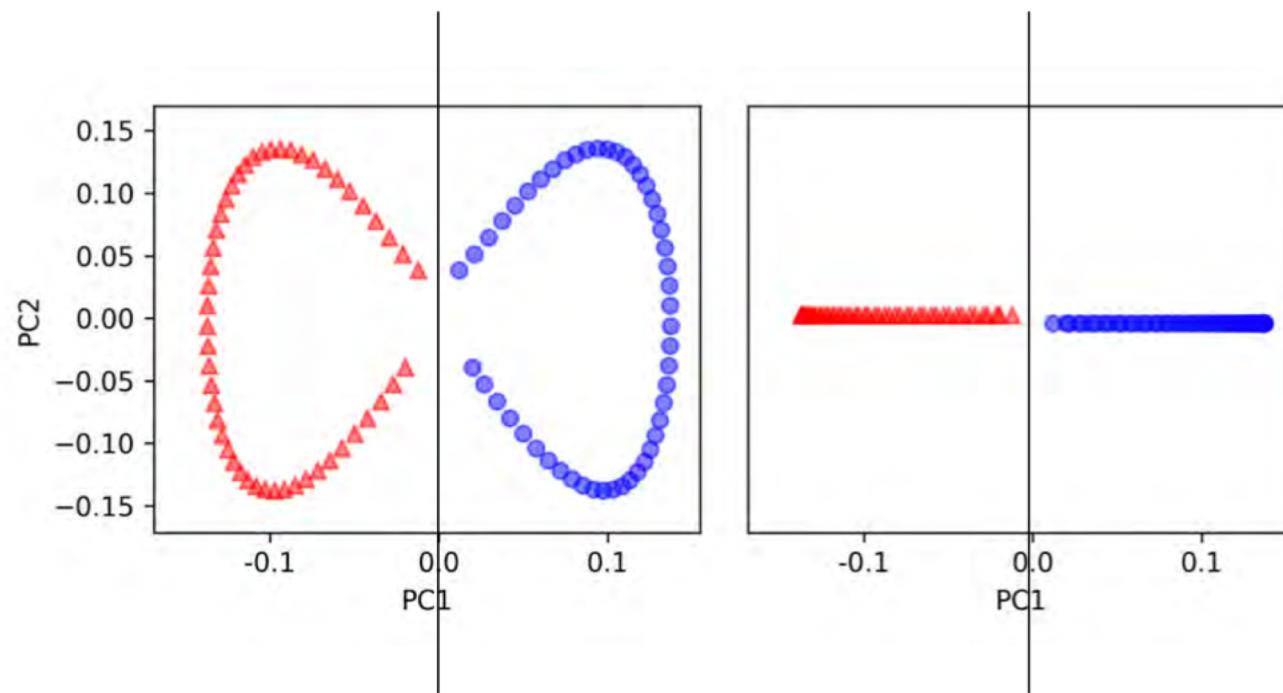


# Kernel PCA

Standard PCA로 분리하기 어려운 경우



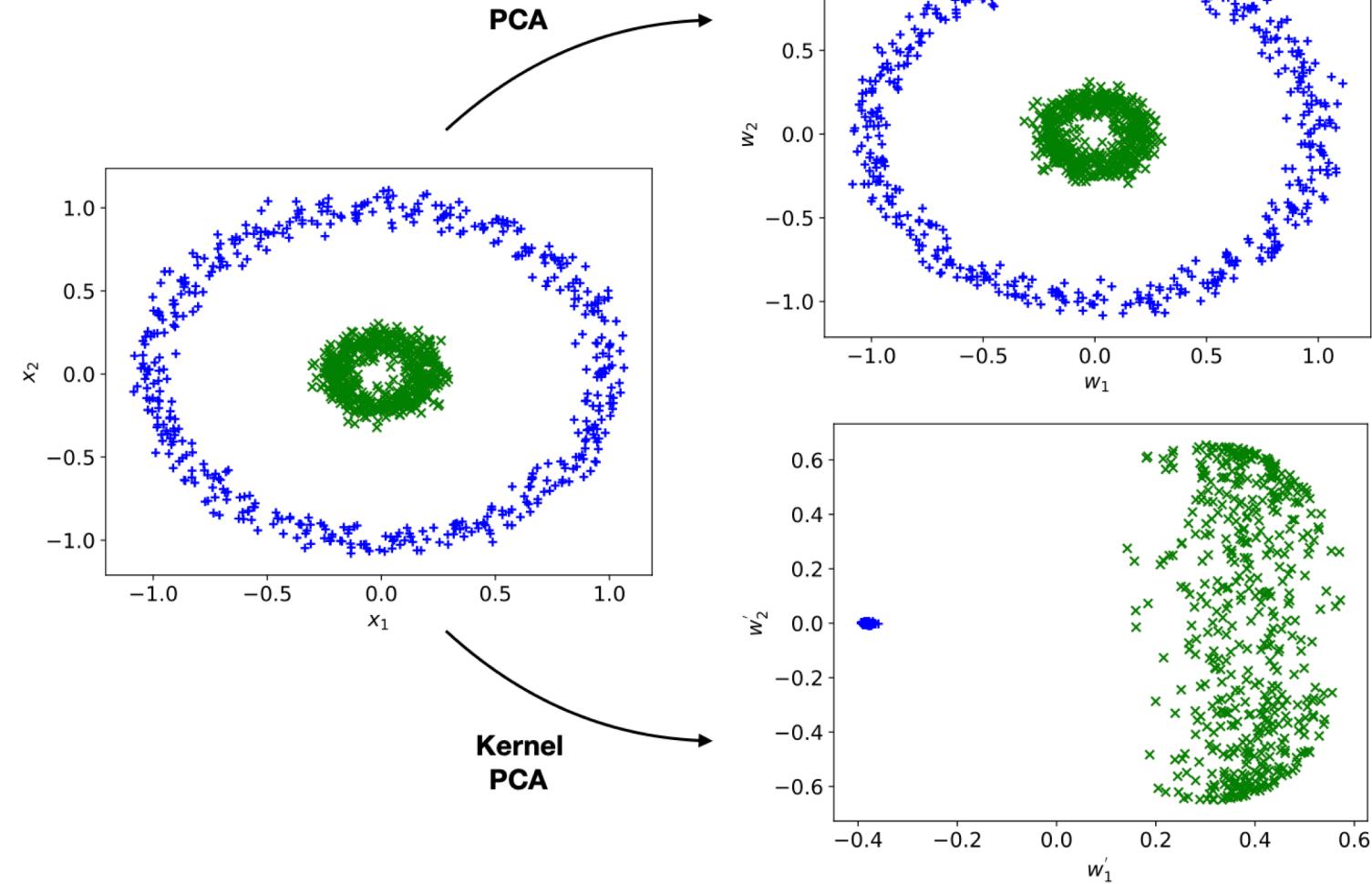
# Kernel PCA



Idea : Kernel SVM과 유사하게, kernel 사용하여 고차원으로 mapping 시킨 후 고차원에서 PCA 진행

# Kernel PCA

Other examples



# References

<https://galton.uchicago.edu/~yibi/teaching/stat220/17aut/Lectures/L24.pdf>

<https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc>

Hands-on machine learning

<https://medium.com/pursuitnotes/day-12-kernel-svm-non-linear-svm-5fdefe77836c>