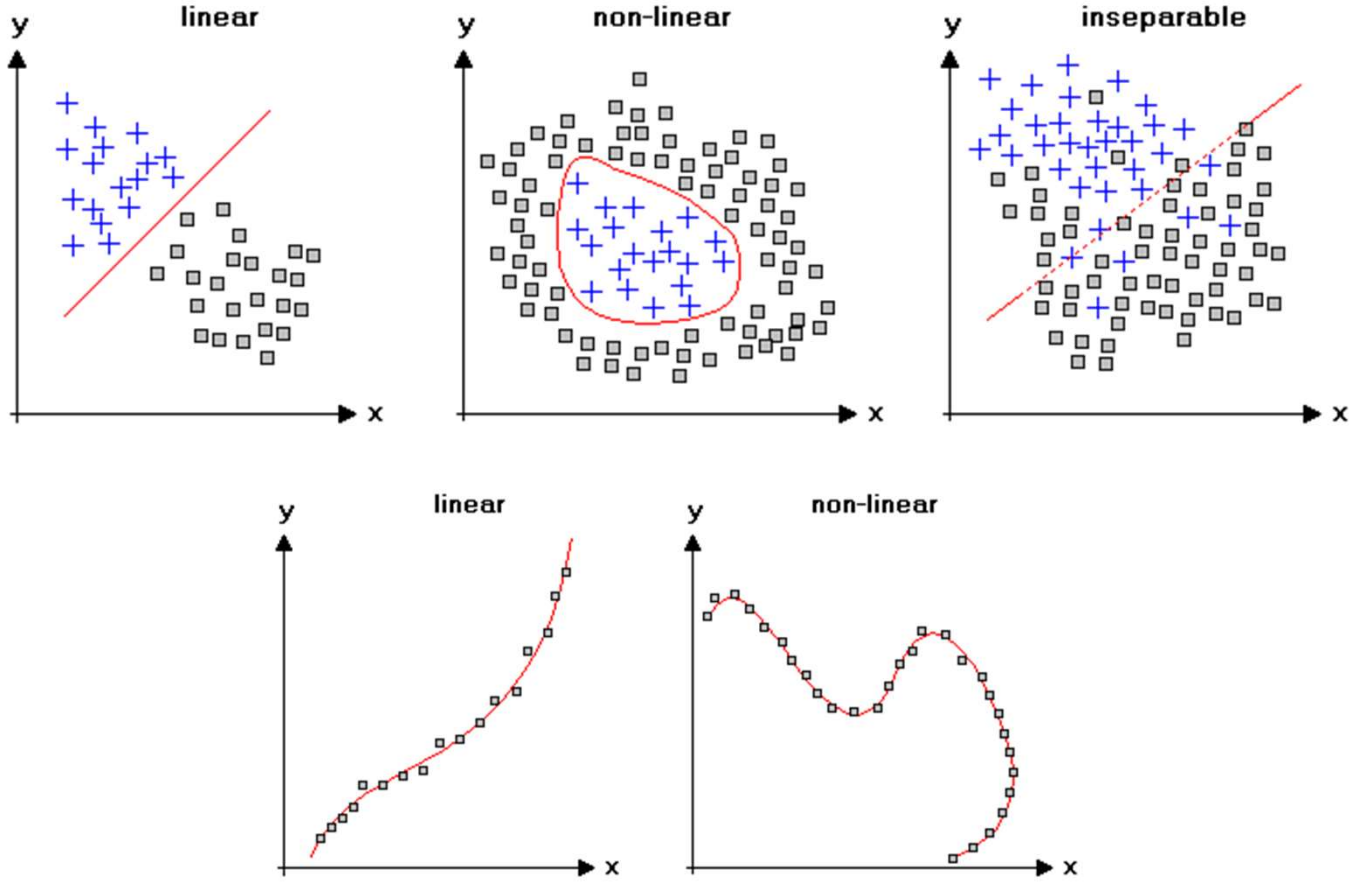
# 선형회귀모형을 넘어 (KNN)

시스템경영공학부
이지환 교수

# Complex patterns of real-world dataset

- 선형 모형들의 중요 가정
  - 변수의 선형조합으로 y값을 예측
  - 하지만 실제 세계의 데이터는 선형적인 패턴을 가지고 있지 않은 경우가 많음



# 머신러닝 모형의 분류

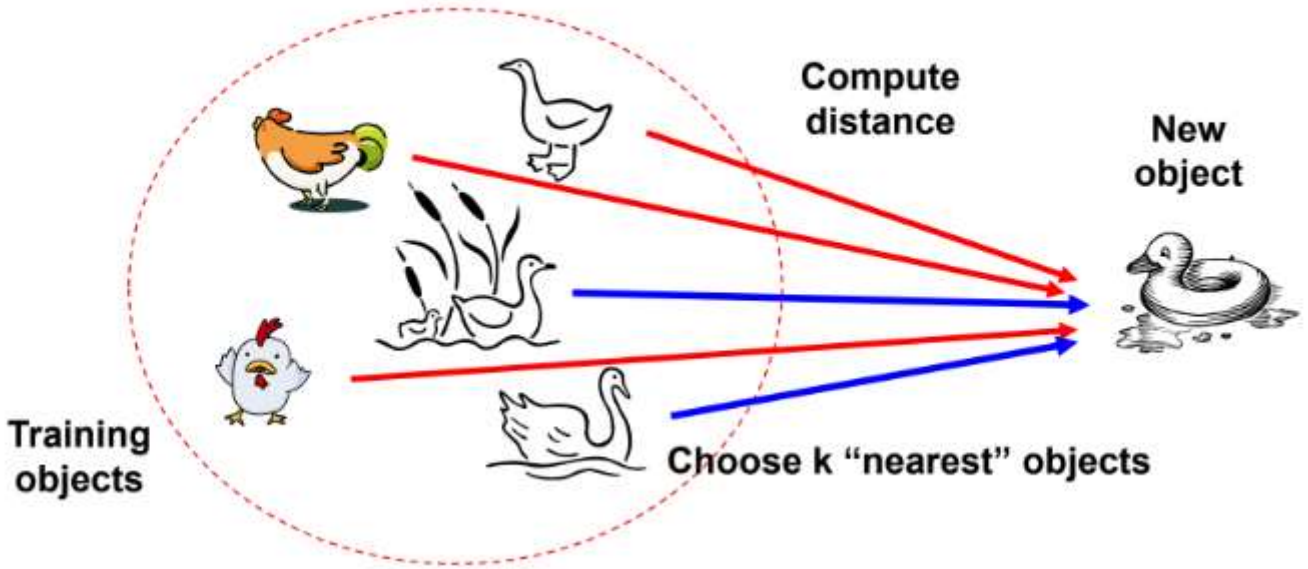
- 예측해야하는 값이 실수인가?
  - (Yes) 회귀(Regression )
  - (No) 분류(Classification)

- 정답을 알고있는 데이터인가?
  - (Yes) 지도학습(Supervised Learning)
  - (No) 비지도학습(Unsupervised Learning)

- 패러미터로 모델을 표현할 수 있는가?
  - (Yes) Parametric Method
  - (No) Non-parametric Method

# KNN - Idea

- **KNN: K Nearest Neighborhood**
- **If it walks like a duck, quacks like a duck, then it's probably a duck**

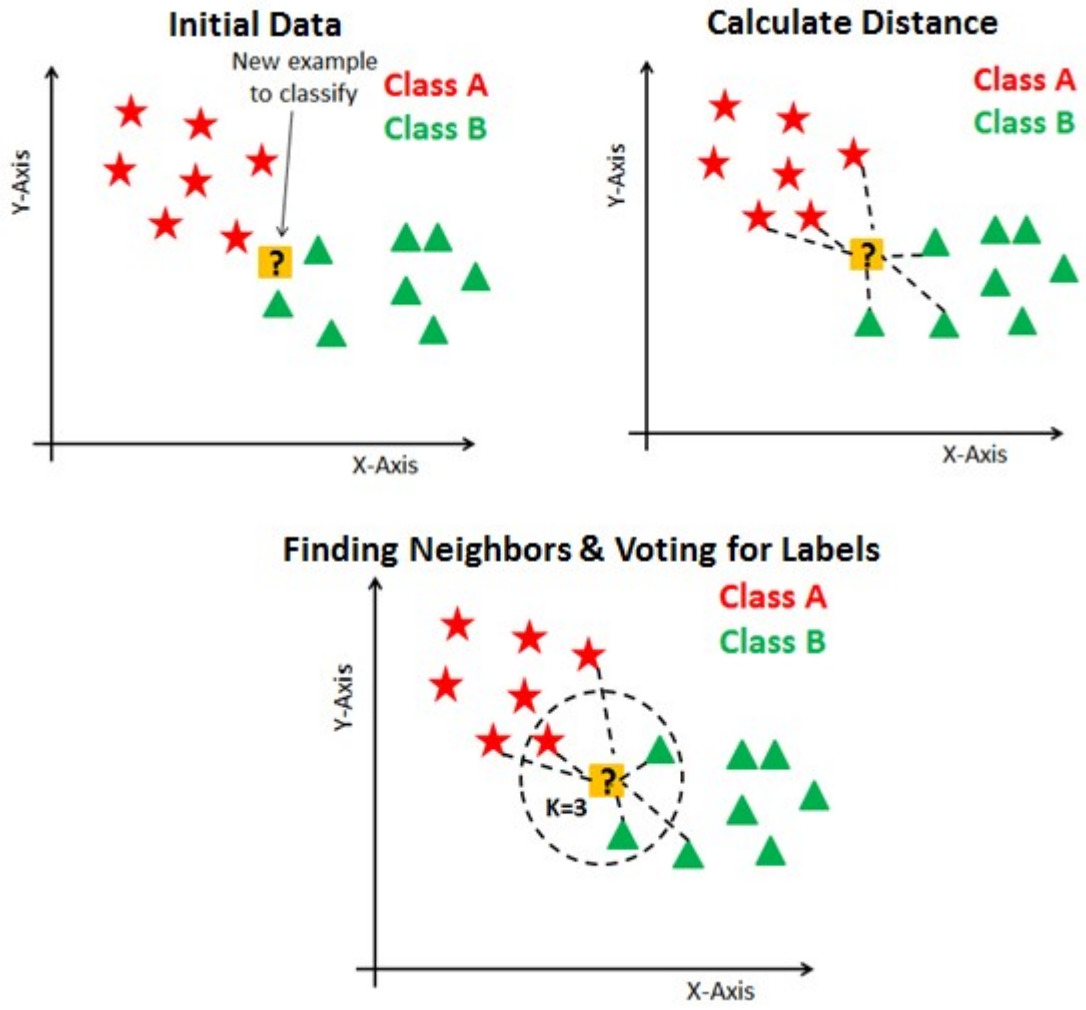


# KNN - Idea

- 모델의 학습
  - 학습 필요 없음, 단순히 데이터를 저장 → 실시간 업데이트
  - 하이퍼 패러미터 K를 설정

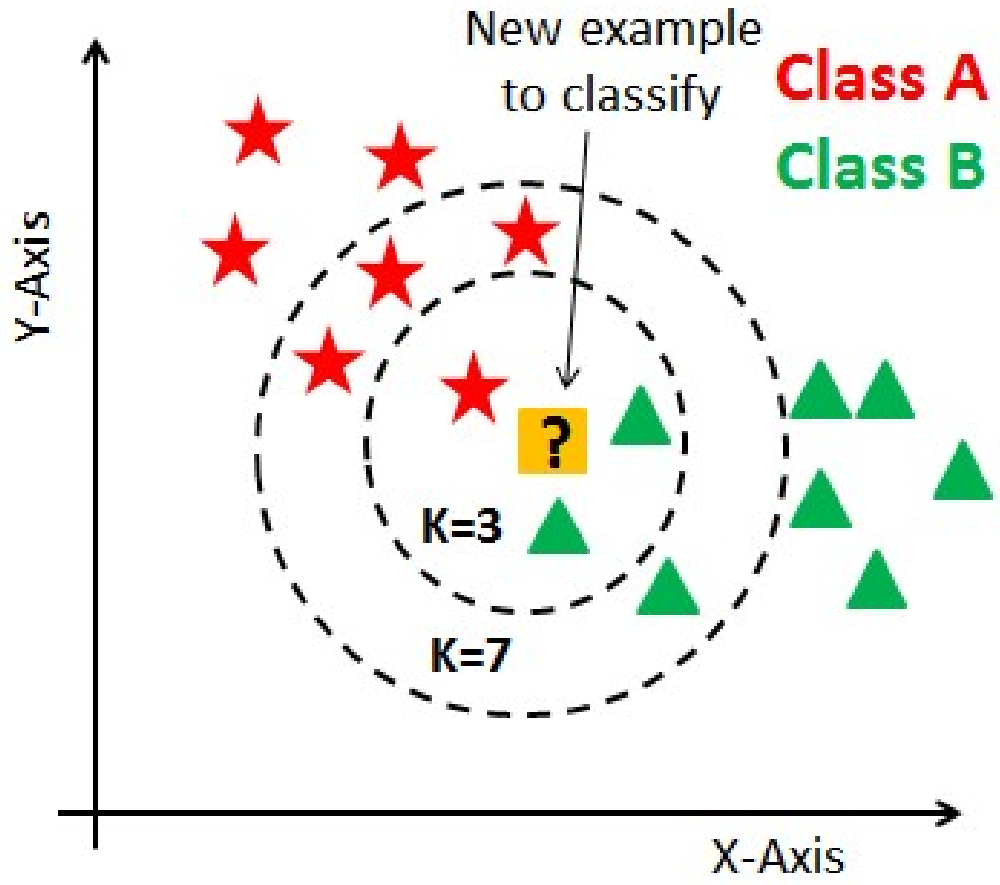
- 모델의 예측
  - 1) 예측하고자 하는 데이터 x와 과거 데이터 사이의 거리를 구한다.
  - 2) x와 가장 거리가 가까운 K개의 데이터를 찾는다.
  - 3) (분류) x와 근접한 K개의 데이터의 Label들 중 가장 많은 Label로 $\hat{y}$ 예측
  - 3) (회귀) x와 근접한 K개의 데이터의 Label의 평균값

# KNN의 계산과정

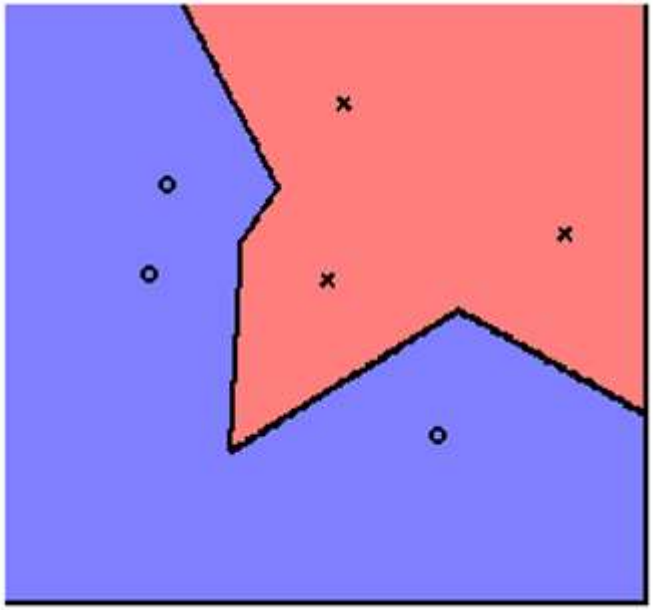

Initial Data
New example
to classify
Class A
Class B
Y-Axis
X-Axis
?
Calculate Distance
Class A
Class B
Y-Axis
X-Axis
?
Finding Neighbors & Voting for Labels
Class A
Class B
Y-Axis
X-Axis
?
K=3

# KNN의 하이퍼 패러미터: K

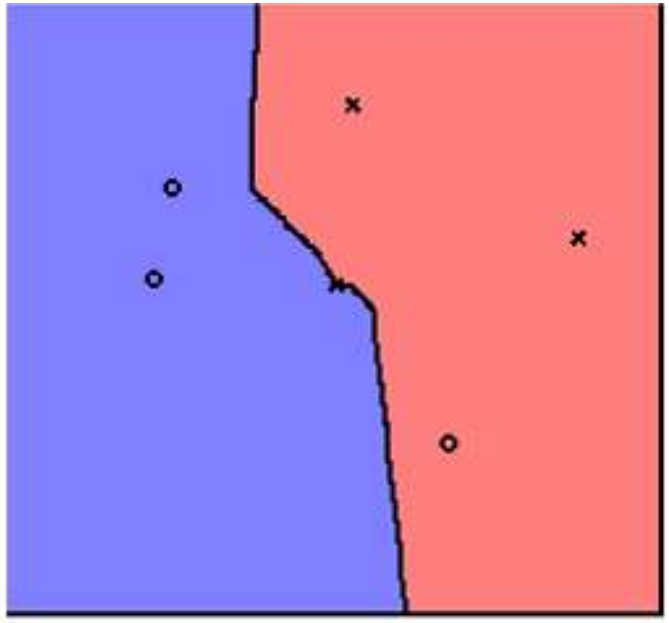
- 인접한 이웃의 개수 K에 따라 예측의 결과가 달라짐



# 분류문제에서 KNN의 경계선 (K의 영향)



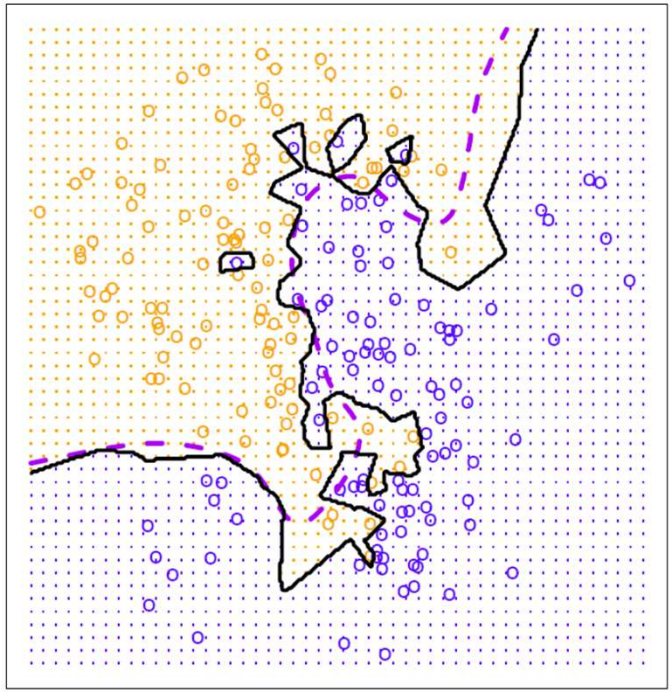
- K=1일 경우 두 카테고리사이의 경계선



- K=3일 경우 두 카테고리사이의 경계선

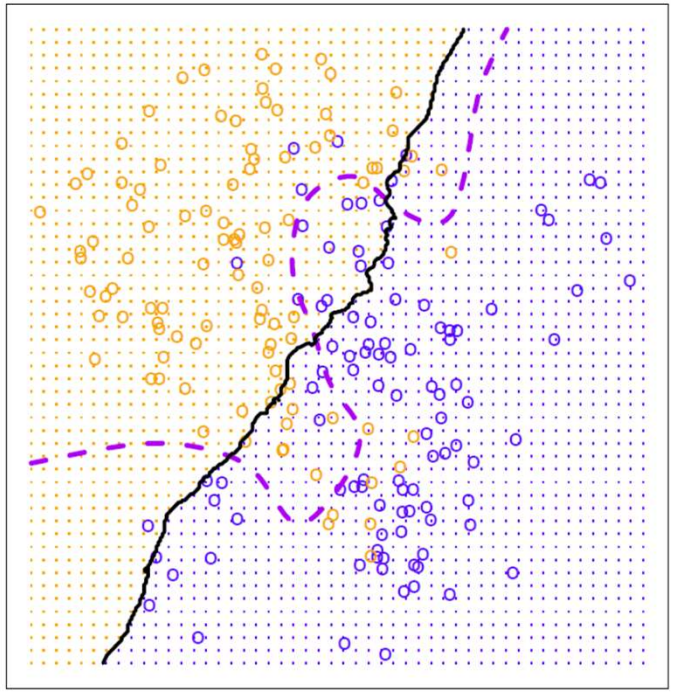
# KNN - Hyper Parameter

KNN: K=1



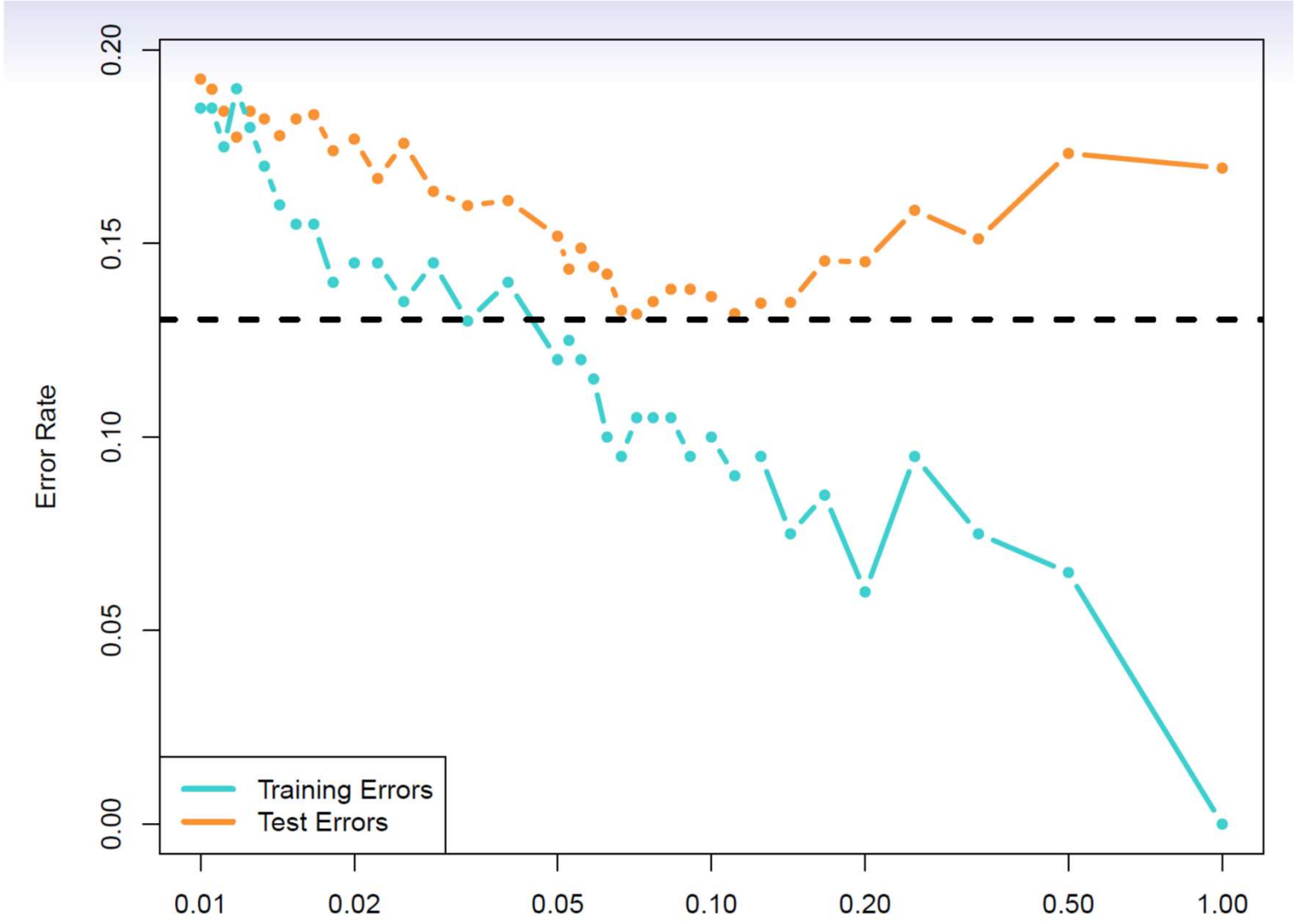
- 경계선이 근처 1개의 점에 의해 결정
- 경계선이 주변 점에 매우 민감 (Much Flexible)

KNN: K=100



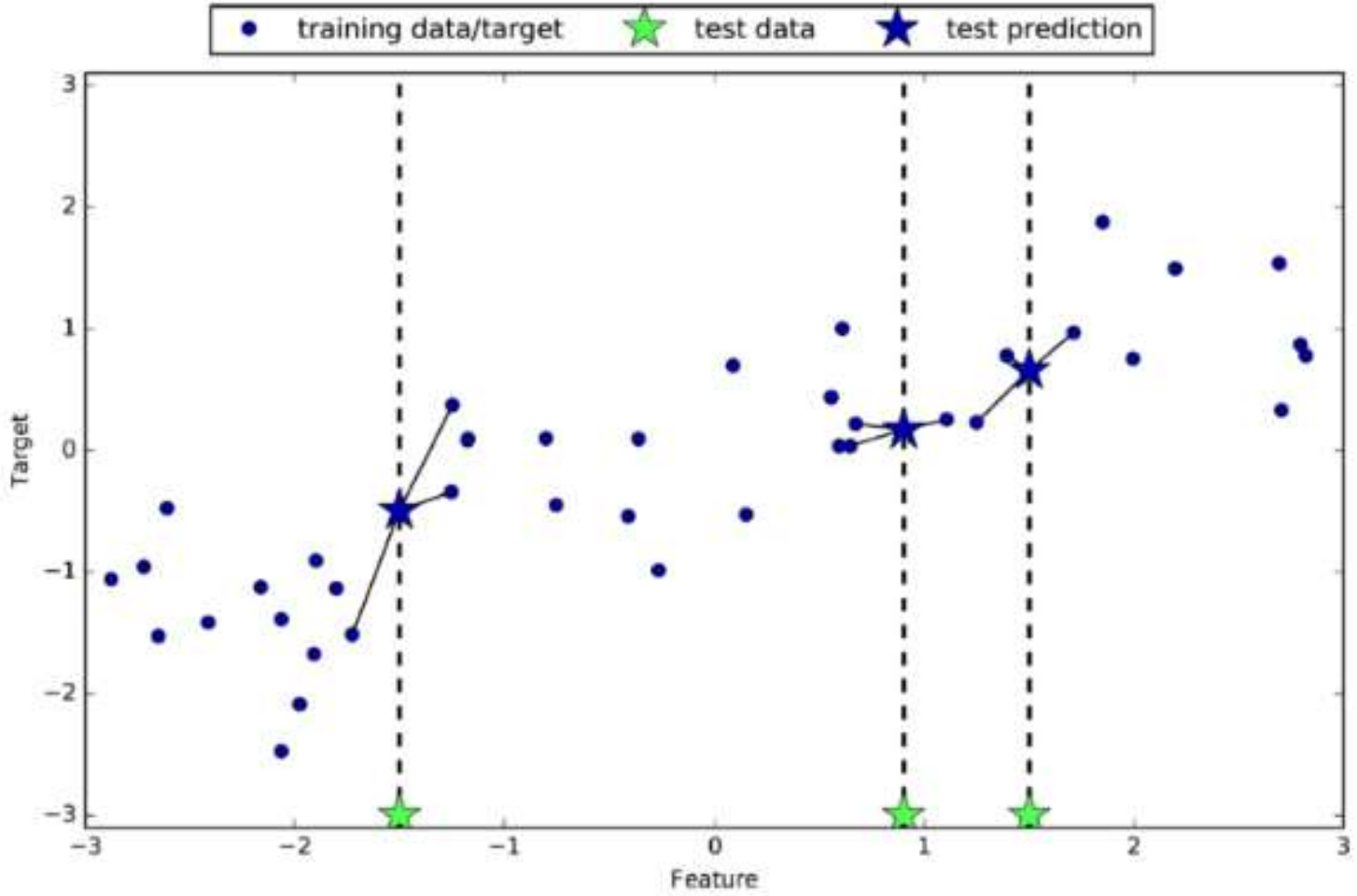
- 경계선이 근처 100개의 점에 의해 결정
- 경계선이 주변 점에 매우 둔감 (Less Flexible)

# KNN - Hyper Parameter Tuning

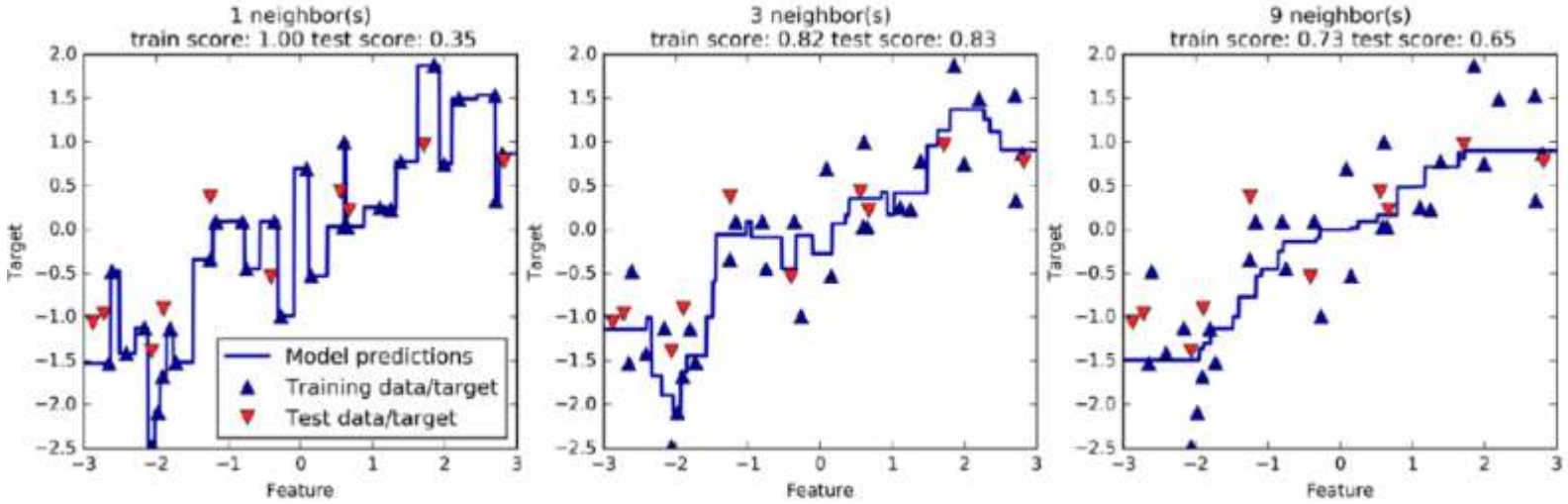


# KNN Regression

- K=3의 경우



# KNN Regression의 Hyperparameter 효과




- 회귀선이 주변 1개 선에 의해 결정
- Very Flexible (Bumpy line)

- 회귀선이 주변 9개 선에 의해 결정
- Less Flexible (Smoother line)

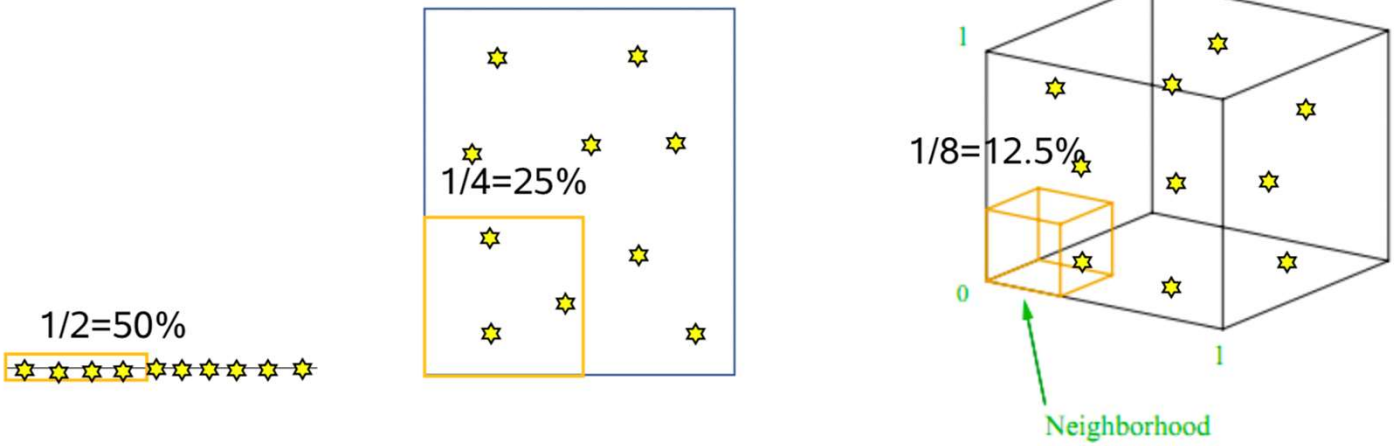
# KNN의 장점과 단점

- 장점
  - 예측이 주변 데이터에 의해 결정되므로 모형의 학습이 필요 없다.
    (Non Parametric and Instance-based method)
  - 데이터의 구조에 대한 사전 지식이 필요 없음
  - 하이퍼 패러미터가 K 한 개로 단순하며, 생각보다 성능이 좋은경우 존재

- 단점
  - 데이터의 수가 많다면, 예측에 걸리는 시간이 길어짐
    (왜일까요?)
  - 데이터 특성의 수가 커지면 성능이 저하됨 → 차원의 저주

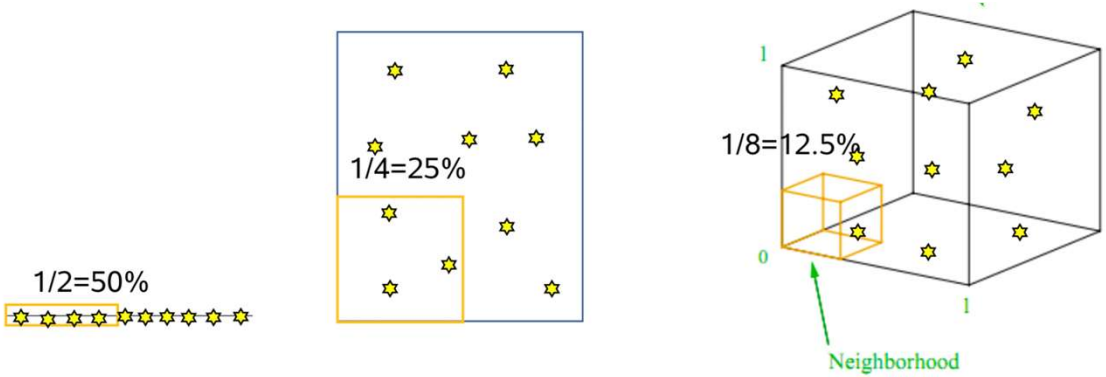
# 이슈1 - 차원의 저주

- 각 변수의 50%에 해당하는 자료를 가지고 있다고 할때,
  - 전체 자료의 얼마만큼을 확보할 수 있는가?



# 이슈1 - 차원의 저주

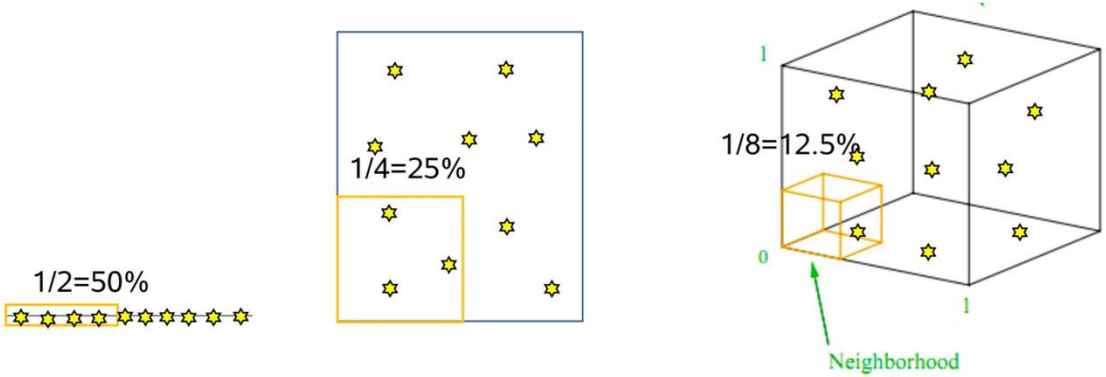
- 각 변수의 50%에 해당하는 자료를 가지고 있다고 할때,
  - 전체 자료의 얼마만큼을 확보할 수 있는가?



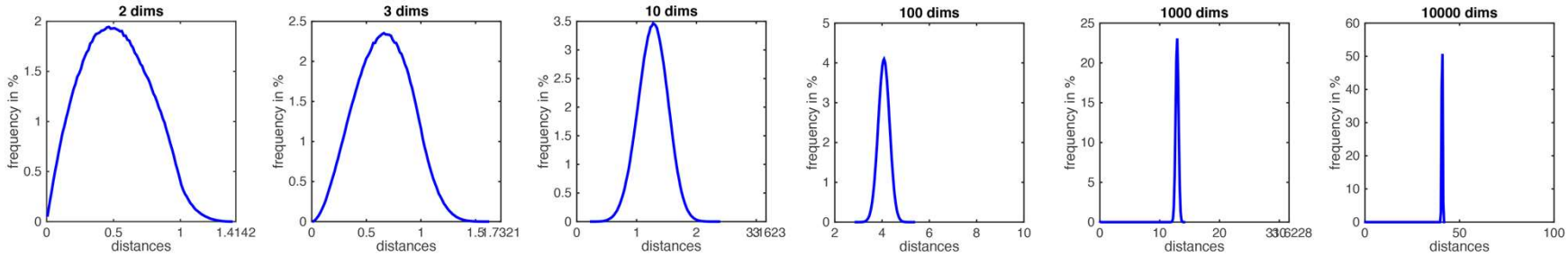

  - 차원이 커질수록 패턴을 설명하기 위해 필요한 데이터의 양이 기하급수적으로 증가

# 이슈1 - 차원의 저주

- 각 변수의 50%에 해당하는 자료를 가지고 있다고 할때,
  - 전체 자료의 얼마만큼을 확보할 수 있는가?




  - 차원이 커질수록 데이터 사이의 간격이 멀어지며, 데이터 간의 변별력도 줄어들게 됨



# 이슈1 - 차원의 저주

- KNN 뿐만 아니라 대부분의 머신러닝 모형에 적용되는 현상

- 데이터의 차원이 지나치게 많은 경우 모형의 성능에 악영향을 미침
  - 학습속도의 저하
  - Generalization 실패 → 오버피팅 발생

- 차원의 저주 방지 전략
  - 특성 선택: 중요 변수만을 선택
  - 비지도 학습 기반 차원축소 기법 사용
    - PCA
    - Manifolds Methods

# 이슈2 - 변수의 Scaling

- 변수들의 단위(metric)가 같고, 서로 비교 가능할 때
  - X1: 위도, X2: 경도
  - 원 데이터를 그대로 사용하는것이 바람직
  - Scaling (X)

- 변수들 간의 단위가 다르고, 서로 비교할 수 없을 때
  - X1: 키, X2: 몸무게
  - 데이터에서 단위를 없애고 비교 가능하도록 Scaling 하는것이 필요
  - Scaling 방법
    - Z-normalization: 모든 변수들이 평균 0, 표준편차 1이 되도록 변환

$$\frac{x_i - \mu_i}{\sigma_i}$$

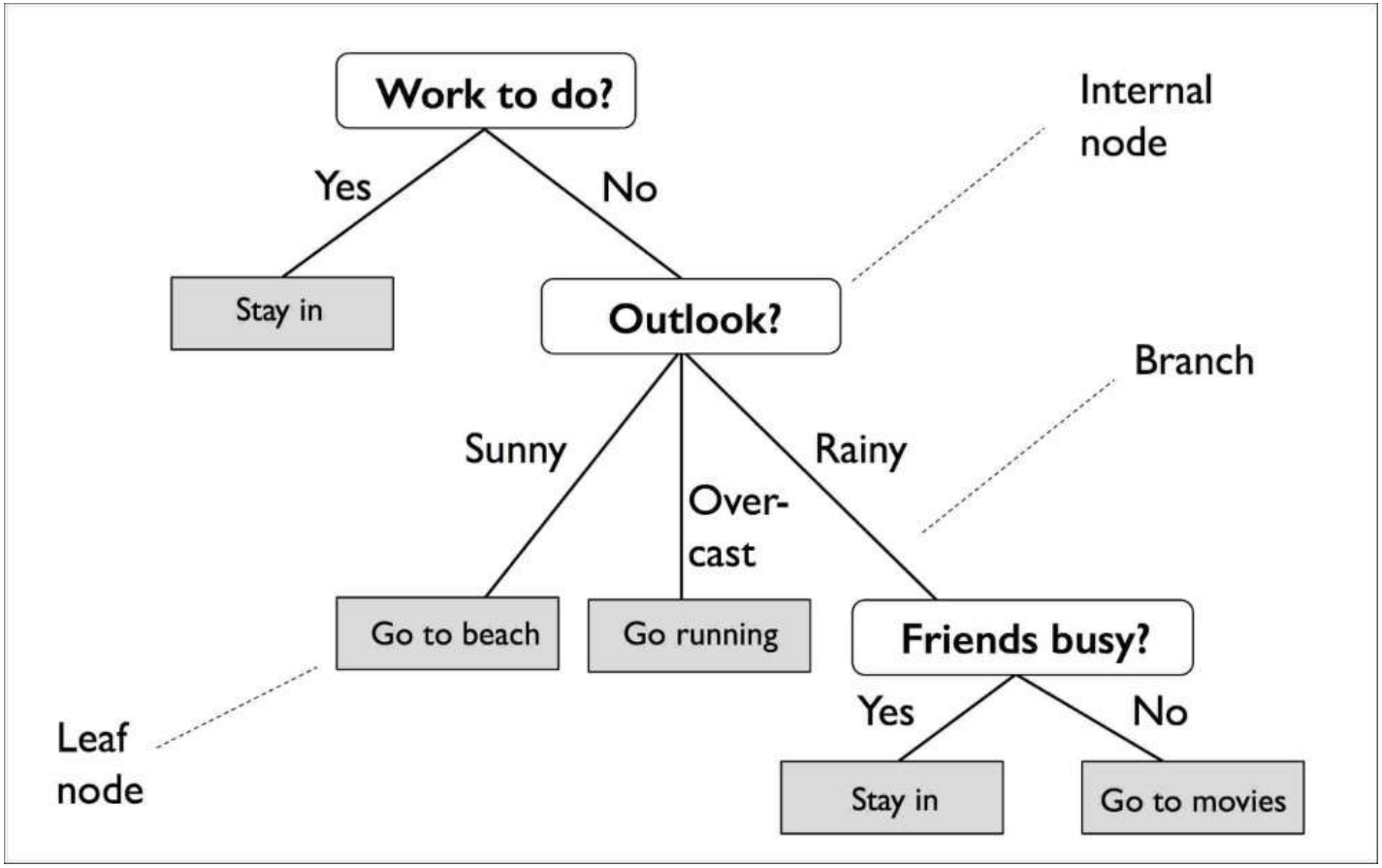
    - Min-max Scaling: 모든 변수들이 최대값 1, 최소값 0 되도록 변환

# 선형회귀모형을 넘어 (Decision Tree)

시스템경영공학부
이지환 교수

# Decision Tree

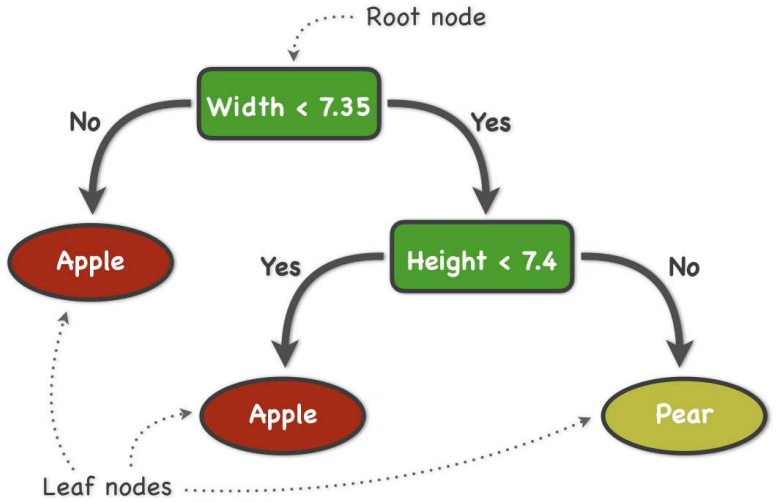
- Making a decision based on asking a series of questions!
- Strength: Easy to explain



# 데이터로부터 Decision Tree 만들기

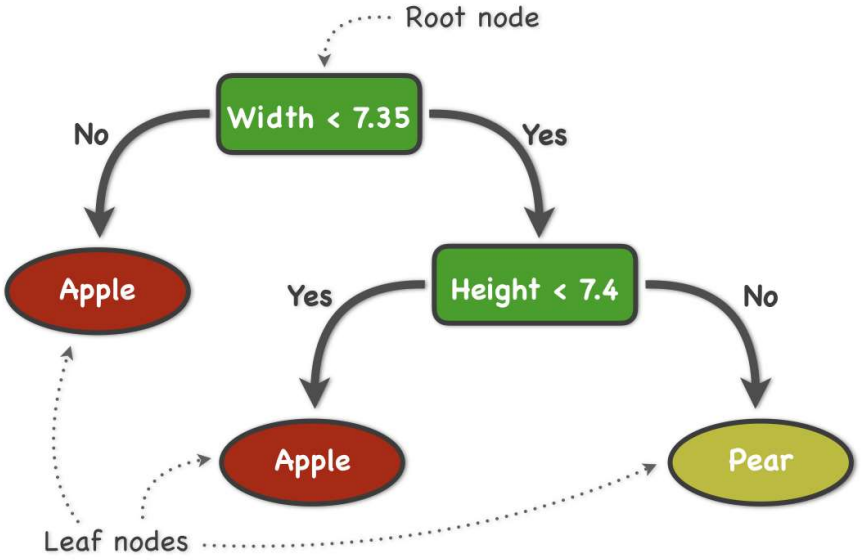
- 너비와 높이를 통해 사과인지 배인지 예측하는 모형을 Decision Tree로 표현하면 다음과 같을 것이다.

| Width | Height | Fruit |
|---|---|---|
| 7.1 | 7.3 | Apple |
| 7.9 | 7.5 | Apple |
| 7.4 | 7.0 | Apple |
| 8.2 | 7.3 | Apple |
| 7.6 | 6.9 | Apple |
| 7.8 | 8.0 | Apple |
| 7.0 | 7.5 | Pear |
| 7.1 | 7.9 | Pear |
| 6.8 | 8.0 | Pear |
| 6.6 | 7.7 | Pear |
| 7.3 | 8.2 | Pear |
| 7.2 | 7.9 | Pear |

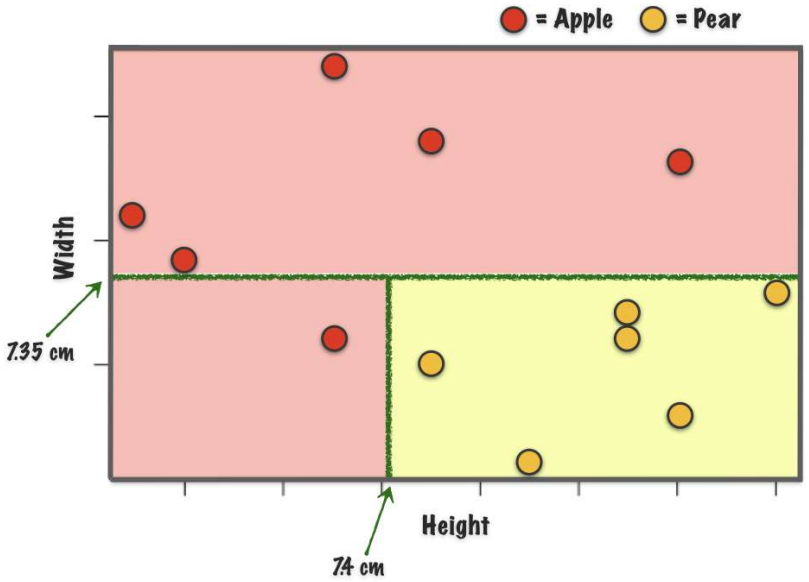


# Decision Tree의 원리

- 사과/배를 분류할 수 있는 Decision Tree

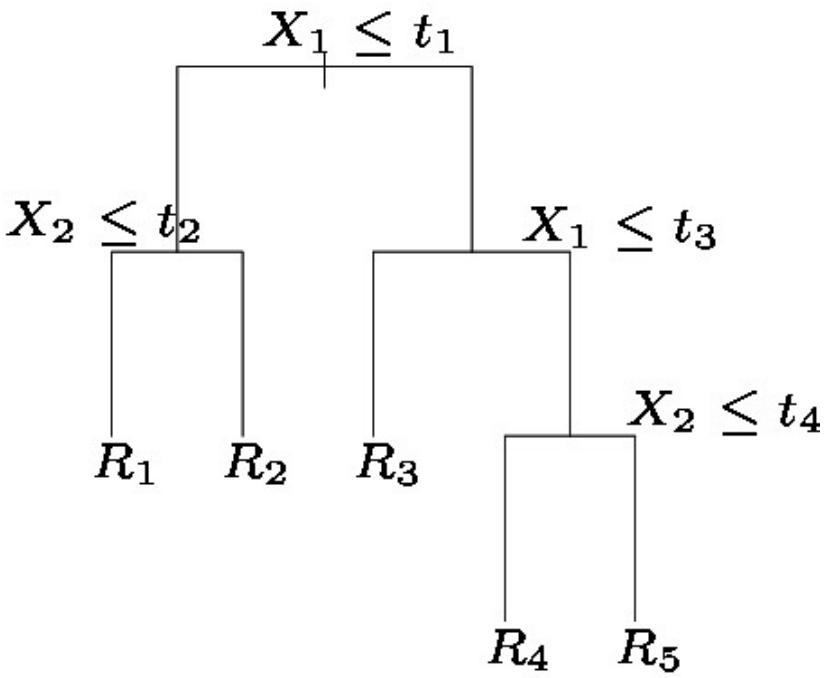



- Decision Tree에 의해 분할된 공간

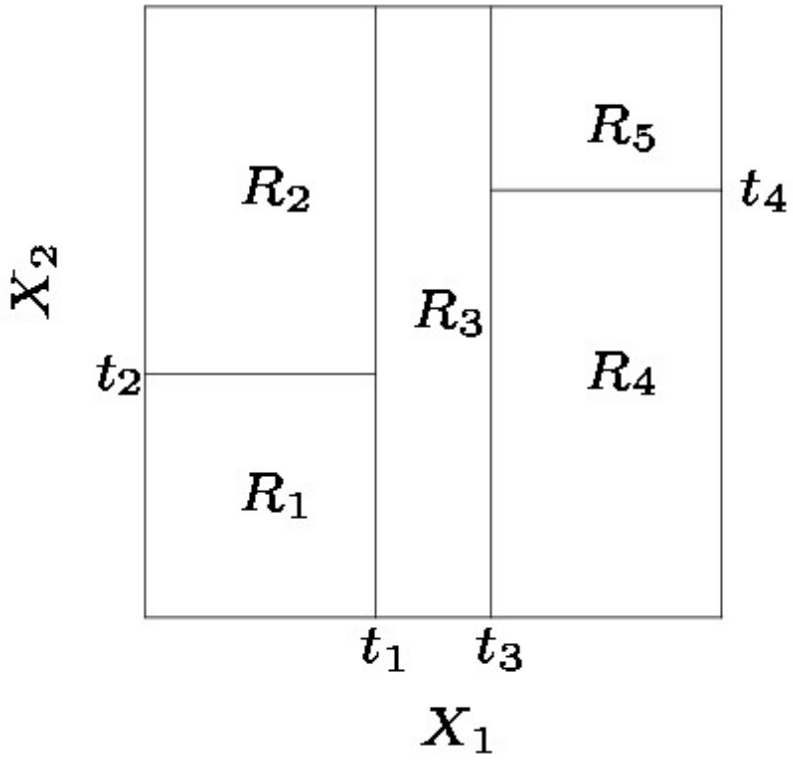



- 차원에 따라 공간을 쪼개나는 것 → Decision Tree에 질문을 추가하는 것
- 두 Label이 잘 분리되도록 공간을 쪼개면 → 좋은 Decision Tree

# Decision Tree와 공간의 분할



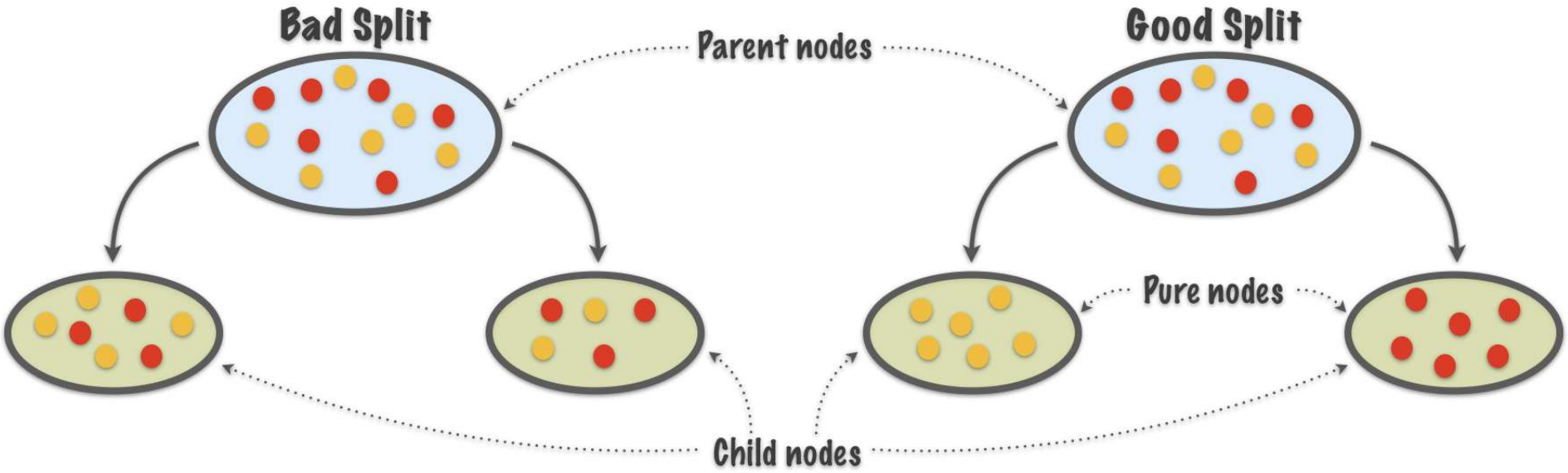

Decision Tree




분할된 공간

# 어떤 분할이 좋은 분할인가?

- Split의 결과로 두 Class가 깨끗하게 나뉘어질 수록 좋은 분할 일 것이다.

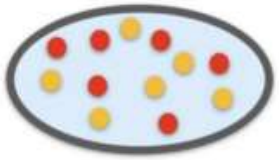


# 엔트로피 (entropy)

- 무질서 함의 정도를 측정하는 지표 → 1에 가까울수록 무질서
- 무질서: 두 Label이 동등히 섞였을때 무질서함이 가장 크다

$$H(X) = -\sum_{i=1}^{n} P(x_i)\log_2 P(x_i)$$

**Calculate Entropy**



$$H(X) = \sum_{i=1}^{n} P(x_i)log_2P(x_i)$$

$$= -\frac{6}{12}\cdot log_2(\frac{6}{12}) - \frac{6}{12}\cdot log_2(\frac{6}{12})$$

$$= \frac{1}{2} + \frac{1}{2} = 1$$

$$H(X) = -\overbrace{\frac{4}{7}\cdot log_2(\frac{4}{7})}^{\text{●}} - \overbrace{\frac{3}{7}\cdot log_2(\frac{3}{7})}^{\text{●}}$$

$$= 0.4613 + 0.5239 = 0.9852$$

$$H(X) = -\overbrace{\frac{3}{5}\cdot log_2(\frac{3}{5})}^{\text{●}} - \overbrace{\frac{2}{5}\cdot log_2(\frac{2}{5})}^{\text{●}}$$

$$= 0.4422 + 0.5288 = 0.9710$$

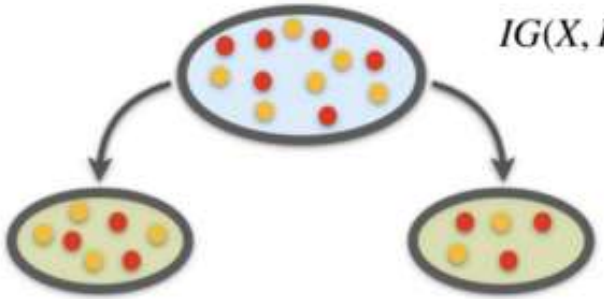
$$H(X) = -\frac{6}{6}\cdot log_2(\frac{6}{6}) = 0$$

$$H(X) = -\frac{6}{6}\cdot log_2(\frac{6}{6}) = 0$$

# Information Gain

- 공간을 쪼갬으로 인해서 얻어지게는 무질서함의 감소정도

**Calculate Information Gain**

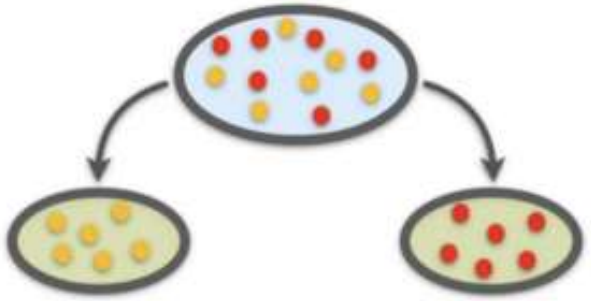


$$IG(X,F) = (original\ entropy) - (entropy\ after\ split)$$

$$= H(X) - \sum_{i=1}^{n} \frac{|x_i|}{X} H(x_i)$$

$$= 1 - \frac{7}{12} \cdot (0.9852) - \frac{5}{12} \cdot (0.9710) = 0.0213$$

$$IG(X,F) = 1 - \frac{6}{12} \cdot (0) - \frac{6}{12} \cdot (0) = 1$$



# Decision Tree Algorithm

## ID3 algorithm

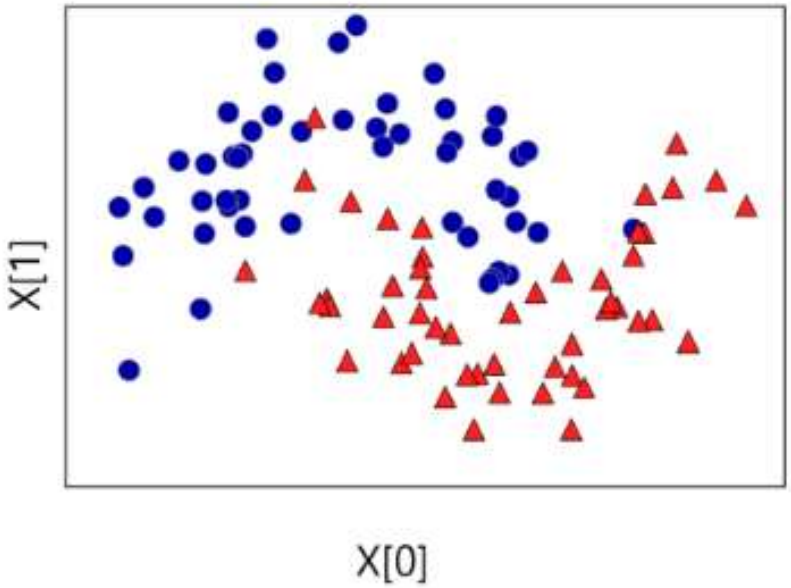
- Split (node, {examples} ):
  1. A ← the best attribute for splitting the {examples}
  2. Decision attribute for this node ← A
  3. For each value of A, create new child node
  4. Split training {examples} to child nodes
  5. For each child node / subset:

     if subset is pure: STOP

     else: Split (child_node, {subset} )

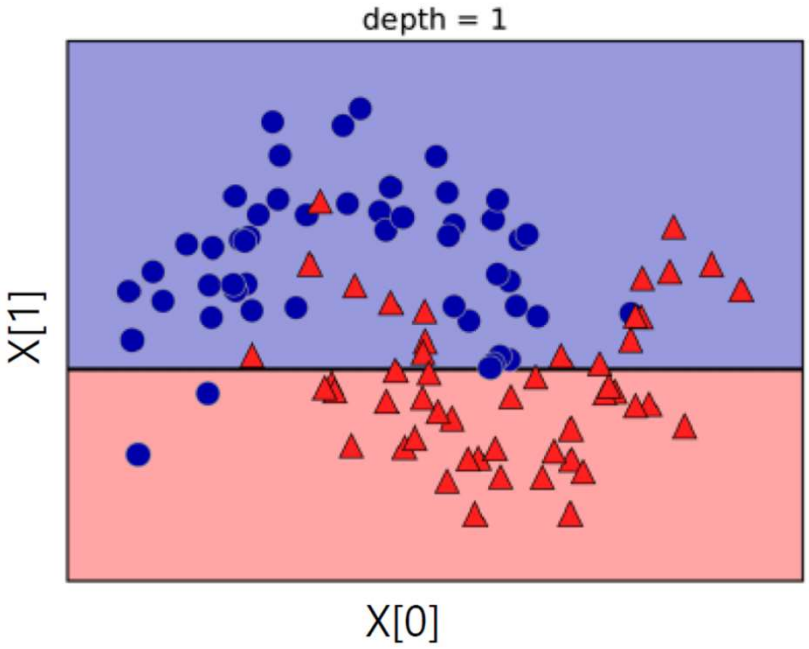
재귀 알고리즘!!

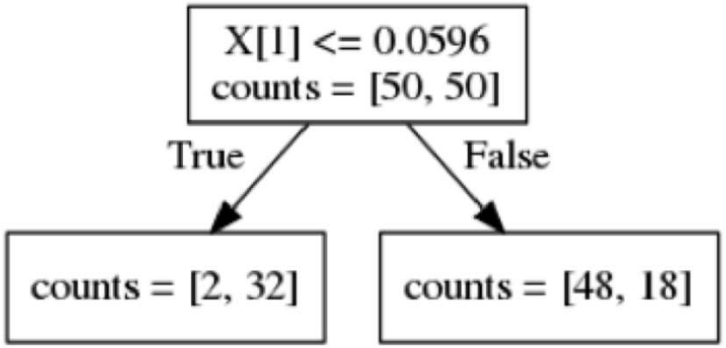
# Decision Tree Algorithm



# Decision Tree Algorithm

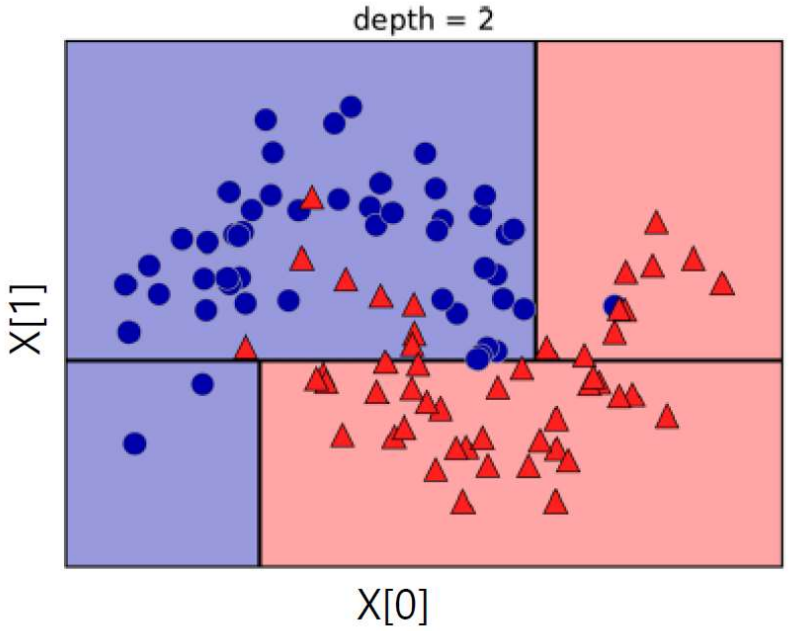
- Information Gain을 높이는 가장 좋은 Feature 고르기 →
  X1 <= 0.0596
- 이 값에 따라 두개의 Child Node 생성

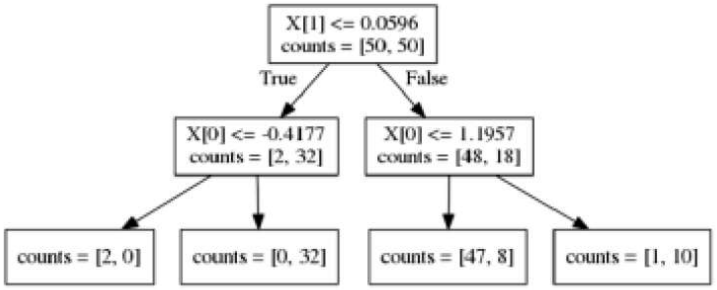


# Decision Tree Algorithm

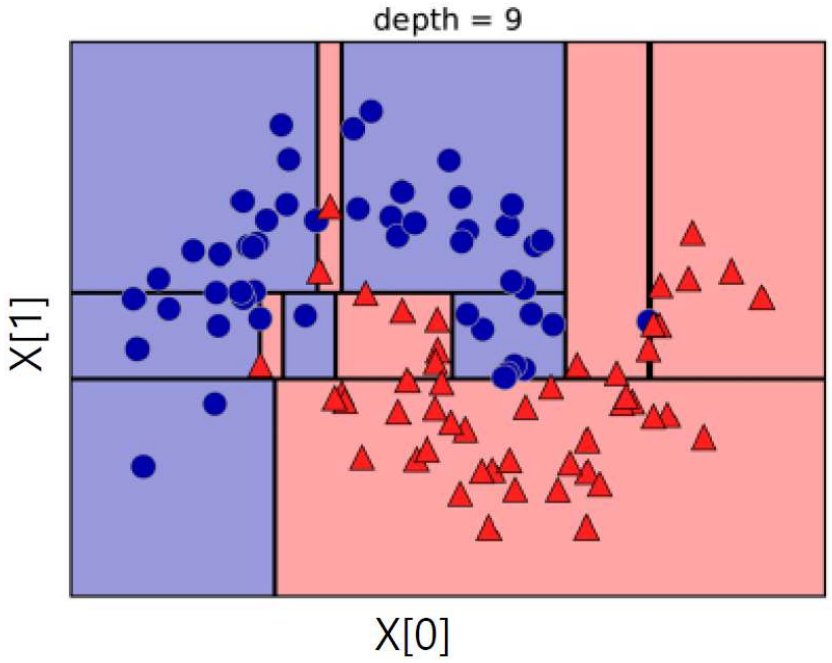
- 각각의 Child node에 대하여 똑같은 과정을 수행 (recursive partitioning)

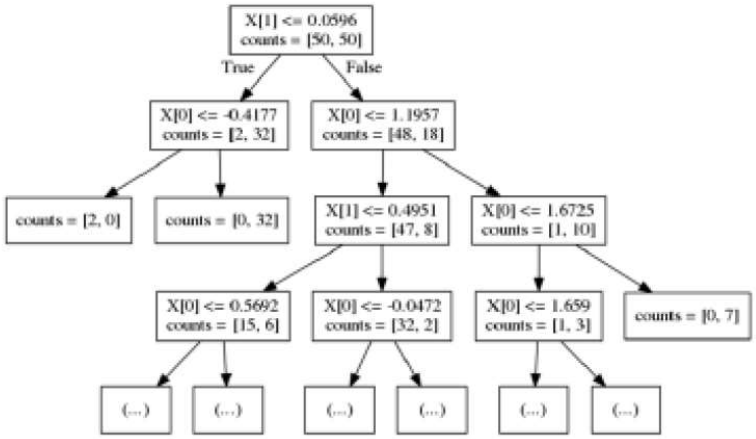


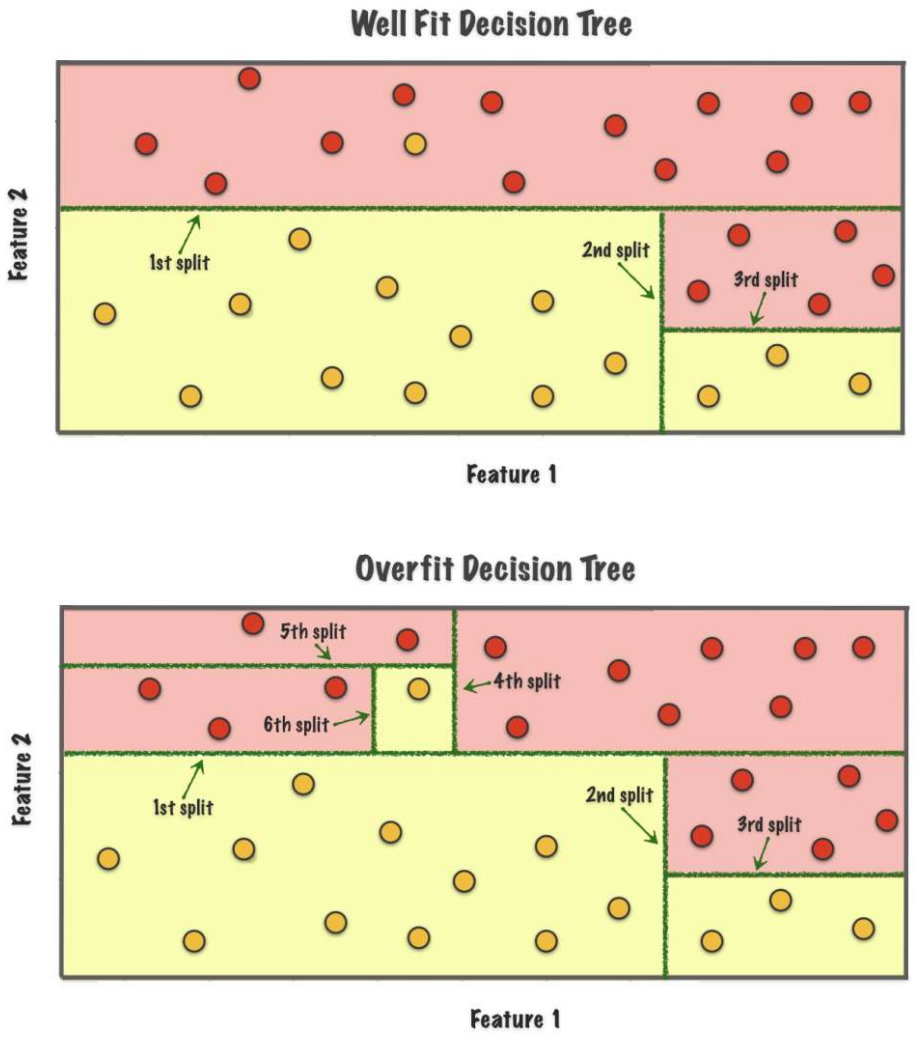
# Decision Tree Algorithm

- 모든 Child Node가 Pure 할 때까지 같은 과정을 반복



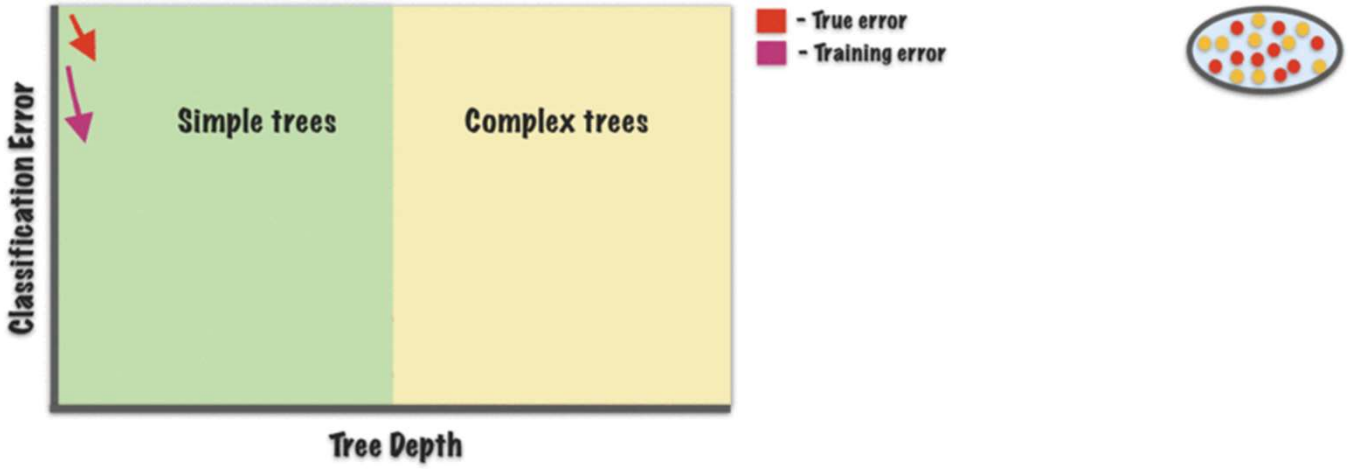
# Decision Tree의 오버피팅




- Tree의 깊이가 깊어질수록 개별적인 데이터를 잘 분류할 수 있게 됨 (Flexible)
- 하지만 지나치게 Tree의 길이가 깊어지면 Overfitting이 발생할 수 있음

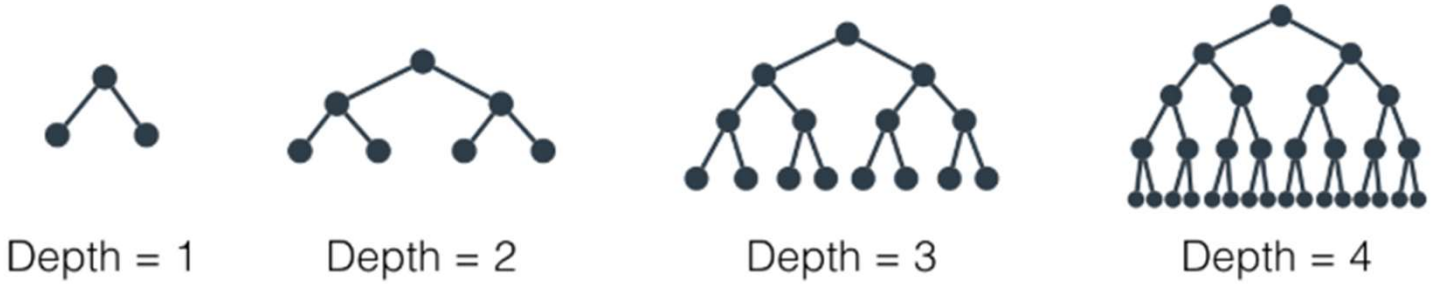
# Decision Tree의 오버피팅

- Tree의 깊이가 깊어질수록 개별적인 데이터를 잘 분류할 수 있게 됨 (Flexible)
- 하지만 지나치게 Tree의 길이가 깊어지면 Overfitting이 발생할 수 있음

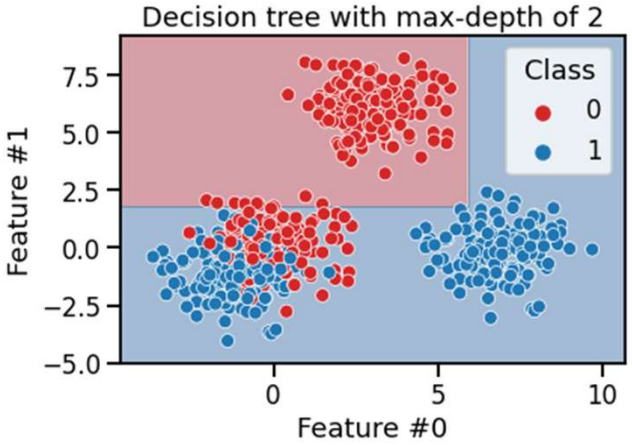


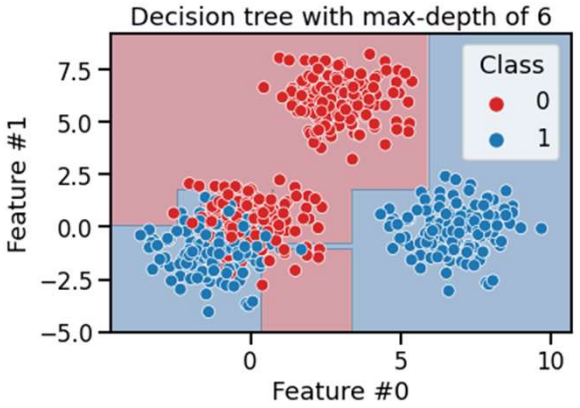
# Decision Tree의 하이퍼 패러미터

- 트리의 최대 깊이 (Max Depth)



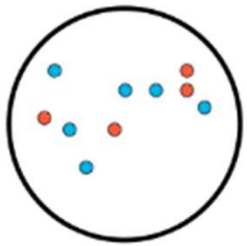

Maximum depth of a decision tree



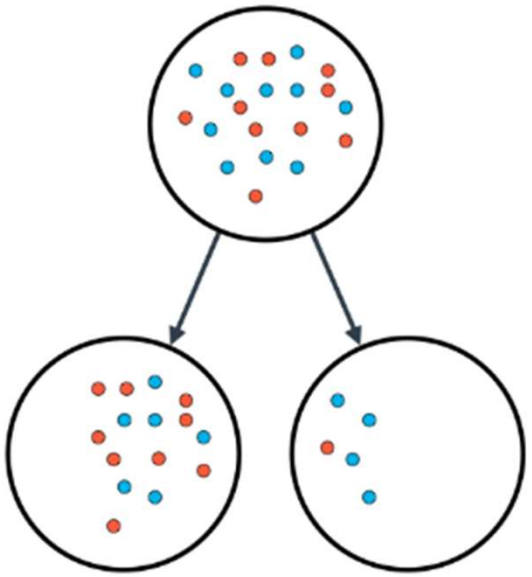



# Decision Tree의 하이퍼 패러미터

- Minimum number of samples to split
- Split을 멈추는 노드안의 최소 데이터 개수
  - 작을수록: 유연한 모델이 될 것이다.



No split!



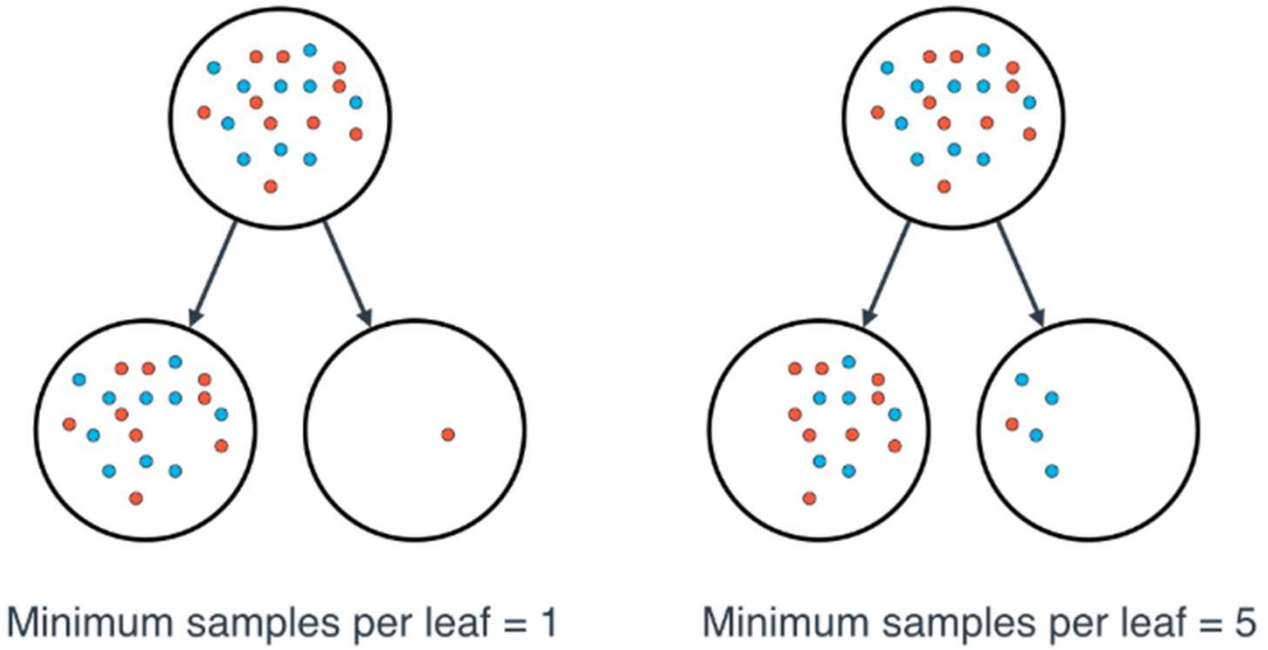
Minimum number of samples to split = 11

Minimum number of samples to split = 11

Minimum number of samples to split

# Decision Tree의 하이퍼 패러미터

- Minimum number of samples per leaf
  - 작을수록: 유연한 모델이 될것이다.
  - 또한 Decision Tree의 비대칭성이 강화된다. (왜?)



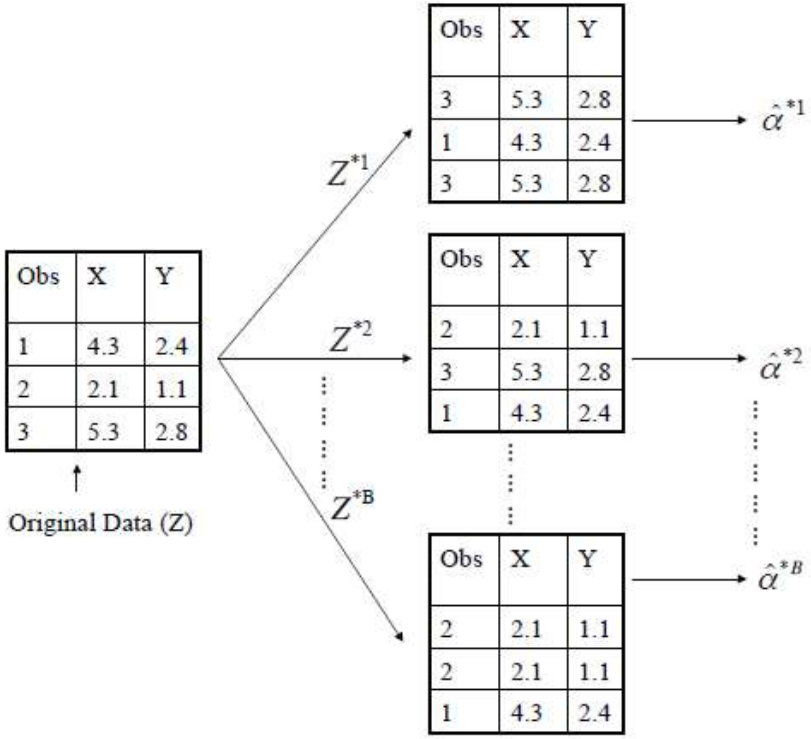
# Decision Tree의 장단점

- 장점
  - 이해하기 쉬운 규칙 : If ~ Then 형식
  - 분류예측에 유용
  - 연속형, 범주형 모두 취급 가능
  - 변수의 중요성 비교 가능
  - 비교적 빠른 속도

- 단점
  - 연속형 변수값을 예측할 때 적당하지 않음
  - 회귀모형에서 예측력이 떨어짐
  - 트리 모형이 복잡하면 예측력 저하, 해석 어려움
  - 데이터 변형에 민감하여 안정적이지 않음 (오버피팅이 발생하기 쉬움)

# 부트스트래핑 (Bootstraping)

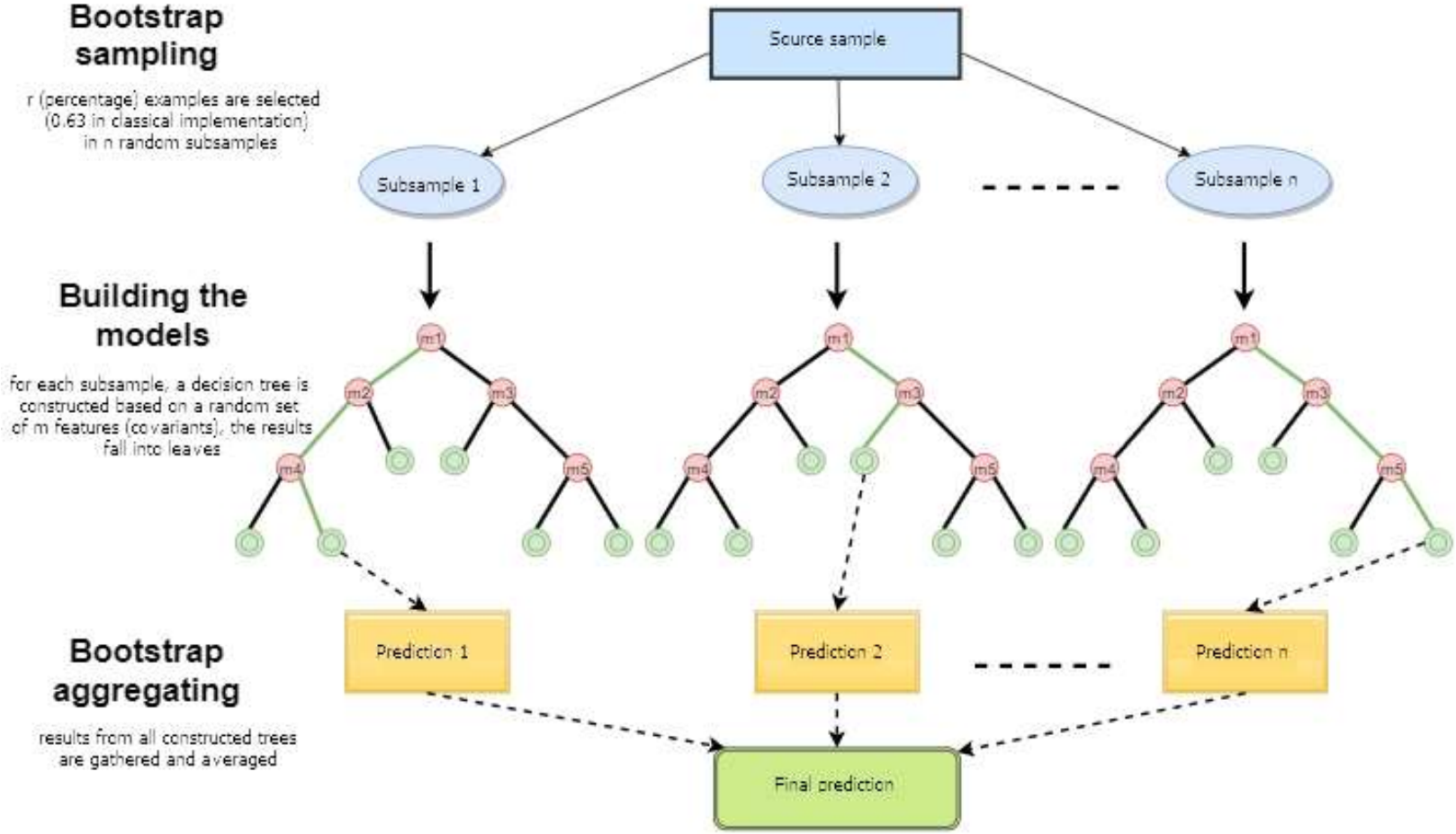
- 특정 통계량을 얻기 위해 다수의 중복된 데이터셋을 다수 만들고, 이들의 결과를 평균내는 기법
  - 각 데이터셋은 전체 데이터셋의 일부가 무작위로 추출됨
  - 일부 중복된 데이터를 가지고 있음




- 데이터의 일반화 능력 향상 가능
  - 데이터양의 증가
  - Variance의 감소

# 랜덤 포레스트(Random Forest)

- 부트스트래핑을 통해 여러 개의 데이터셋을 만들고
- 각자 Decision Tree를 학습 --> 학습된 결과를 취합하여 예측



# 랜덤 포레스트 (random forest) 장단점

- 장점
  - 일반화 능력이 좋으면서도 예측력이 좋음
  - 중간 사이즈의 데이터에서 매우 많이 사용되는 실용적인 기법

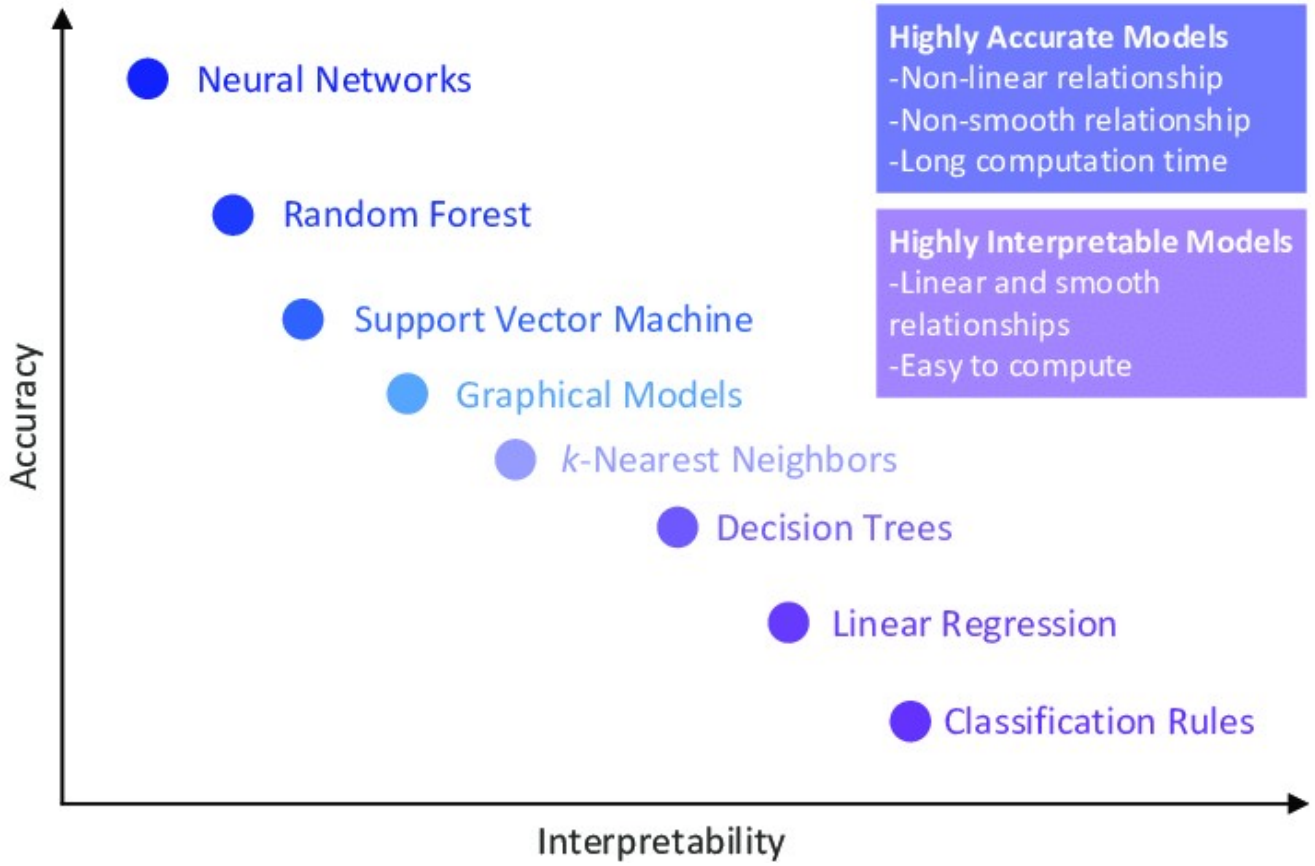
- 단점
  - 여러 개의 Decision Tree가 중첩되어 있으므로 설명하기가 어려움
  - 단일 트리에 비해 계산이 많이 소요됨 (컴퓨팅 파워로 극복 가능)

# 다루지 않은 방법론

- SVM (Support Vector Machine)
- ANN (Artificial Neural Network)
  - Deep Neural Network
  - 곧 다룰 예정
- Random Forest의 개량버전
  - XGBoost (높은 정확도로 유명)
  - LGBM (병렬 계산 가능)
  - ..
- …

# Model Trade-off

- 무조건 예측력이 좋다고 해서 좋은 모델은 아님.
- 설명력-정확도의 Trade-off를 잘 따져서 문제에 맞는 문제를 풀어야 함



# 비지도학습
# (Unsupervised Learning)

시스템경영공학부
이지환 교수

# 머신러닝 모형의 분류

- 예측해야하는 값이 실수인가?
  - (Yes) 회귀(Regression )
  - (No) 분류(Classification)

- 정답을 알고있는 데이터인가?
  - (Yes) 지도학습(Supervised Learning)
  - (No) 비지도학습(Unsupervised Learning)

- 패러미터로 모델을 표현할 수 있는가?
  - (Yes) Parametric Method
  - (No) Non-parametric Method

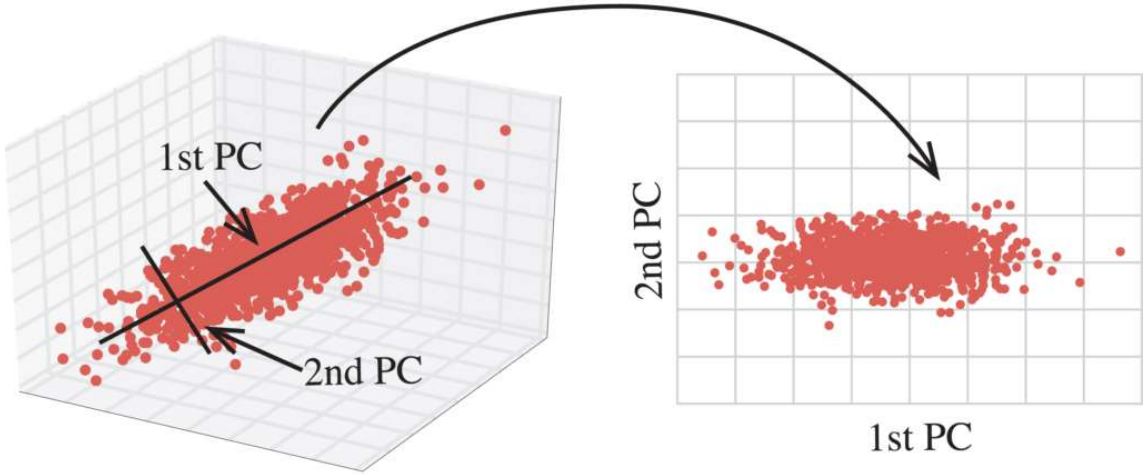
# 지도학습 vs 비지도 학습

- 지도학습
  (Supervised Learning)
- 데이터: (X,y)
  - X 데이터
  - y 레이블(정답)
- 목표
  - X → y의 관계를 맵핑하는 함수를 찾는 것
- 예시
  - 회귀, 분류, CNN, 객체탐지, Sequence 모델

- 비지도학습
  (Unsupervised Learning)
- 데이터: X
  - X 데이터
  - y는 존재하지 않음
- 목표
  - 데이터를 설명하는 내재적인 구조를 학습
- 예시
  - 군집분석(Clustering)
  - 차원축소 (Dimensionality Reduction)
  - 이상치 탐지

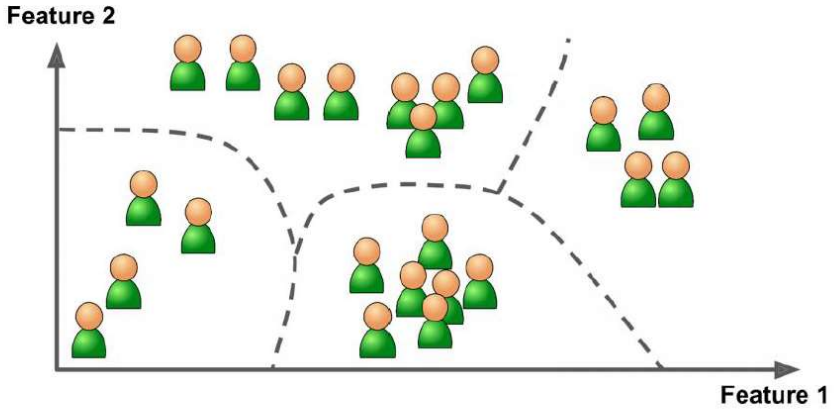
# 비지도학습의 대표 application

- 주어진 데이터의 정보를 최대한 보존하면서 적은수의 변수로 표현

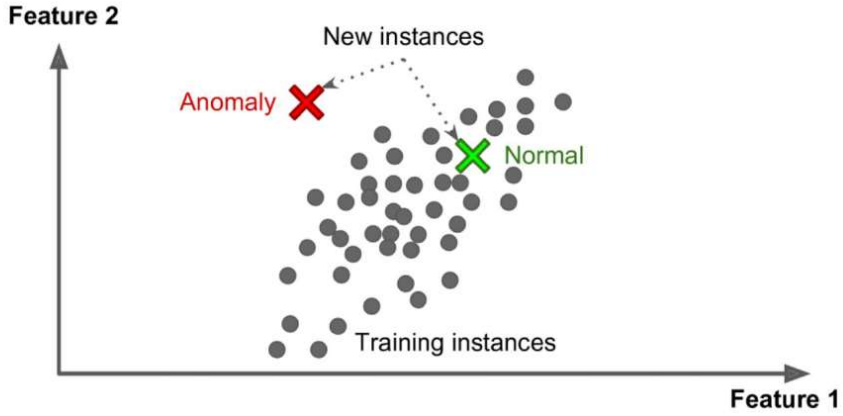


# 비지도학습의 대표 application

주어진 데이터로부터 객체들을 유사한 특성을 갖는 몇 개의 군집으로 분할




주어진 데이터로부터 대세 경향이나 분포를 따르지 않는 이상 객체를 탐지



# 차원축소 (Dimensionality Reduction)

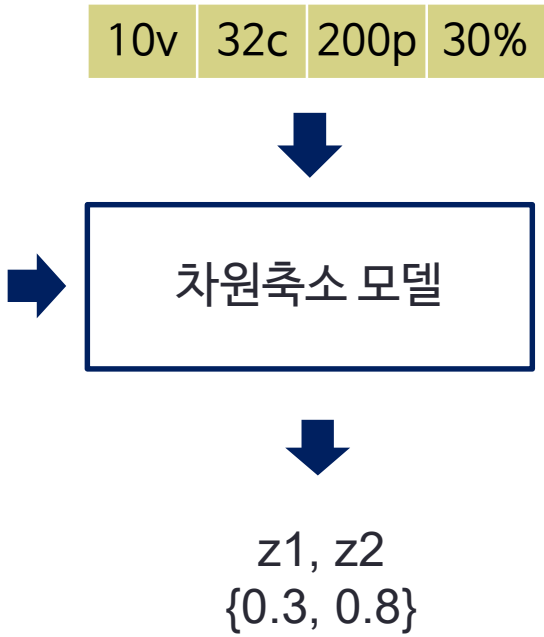
시스템경영공학부
이지환 교수

# 비지도학습 (Unsupervised Learning)

- 학습하는 데이터내에 예측하고자 하는 값이 없는 경우
- 예시) 시간에 따라 기록된 5,000개의 공정 데이터가 존재한다. 데이터의 {정보의 손실을 최소화 하면서} 데이터의 차원을 {2개로 축소하여 표현}하고 싶다.

- 차원 축소(dimensionality reduction)

| | 전압 | 온도 | 압력 | 습도 | z1 | z2 |
|---|---|---|---|---|---|---|
| 1 | xx | xx | xx | xx | 0.2 | 0.3 |
| 2 | xx | xx | xx | xx | 0.5 | 0.8 |
| 3 | xx | xx | xx | xx | 0.2 | 0.4 |
| | | | | | | |
| 4999 | xx | xx | xx | xx | -3.5 | 1.2 |
| 5000 | xx | xx | xx | xx | -2.5 | 1.5 |



# 차원 축소의 이점 - 시각화

- 50개의 특성으로 이루어진 데이터

**Data Visualization**

$x \in \mathbb{R}^{50}$

| Country | $x_1$ GDP (trillions of US$) | $x_2$ Per capita GDP (thousands of intl. $) | $x_3$ Human Develop-ment Index | $x_4$ Life expectancy | $x_5$ Poverty Index (Gini as percentage) | $x_6$ Mean household income (thousands of US$) | ... |
|---|---|---|---|---|---|---|---|
| Canada | 1.577 | 39.17 | 0.908 | 80.7 | 32.6 | 67.293 | ... |
| China | 5.878 | 7.54 | 0.687 | 73 | 46.9 | 10.22 | ... |
| India | 1.632 | 3.41 | 0.547 | 64.7 | 36.8 | 0.735 | ... |
| Russia | 1.48 | 19.84 | 0.755 | 65.5 | 39.9 | 0.72 | ... |
| Singapore | 0.223 | 56.69 | 0.866 | 80 | 42.5 | 67.1 | ... |
| USA | 14.527 | 46.86 | 0.91 | 78.3 | 40.8 | 84.3 | ... |
| ... | ... | ... | ... | ... | ... | ... | |

# 차원 축소의 이점 - 시각화

- 2개의 차원으로 축소 시행

**Data Visualization**

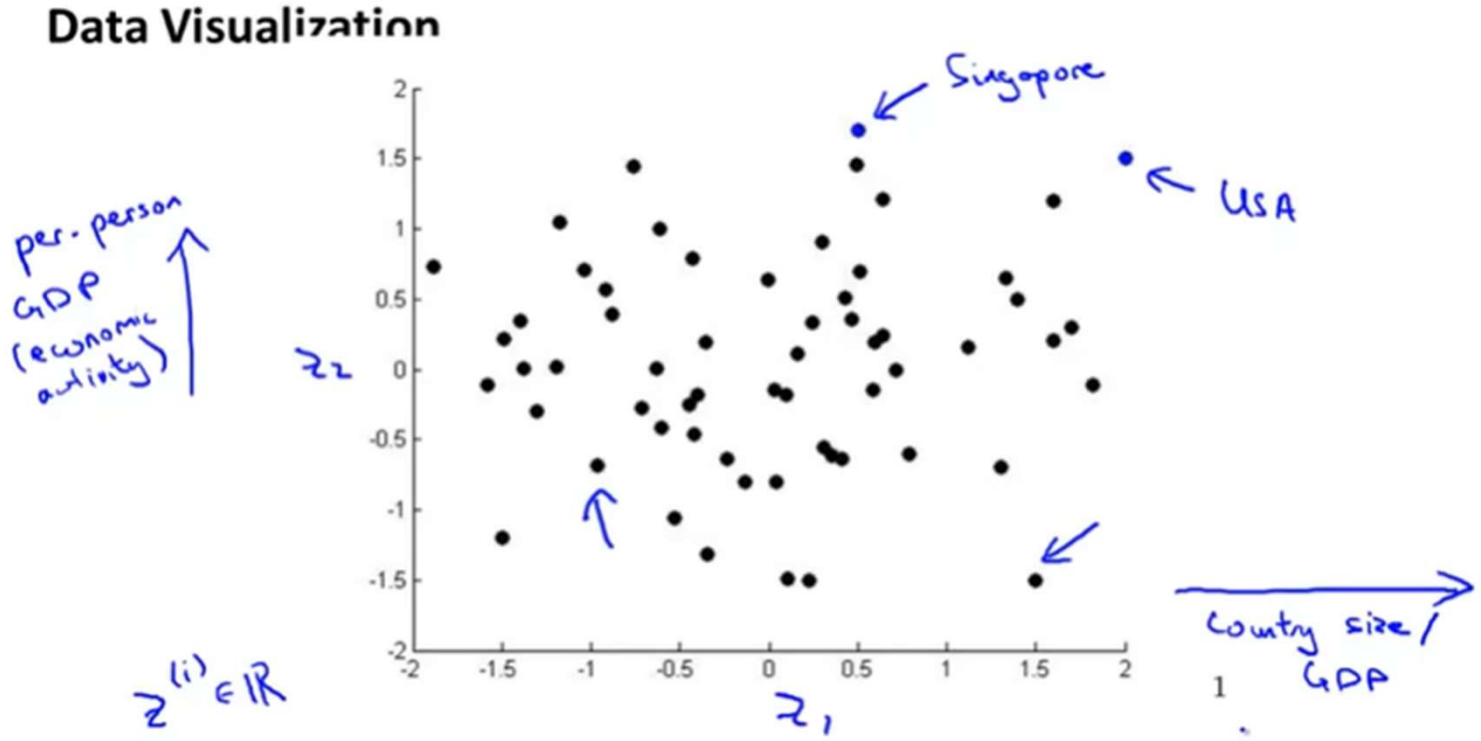
| Country | $z_1$ | $z_2$ |
|---|---|---|
| Canada | 1.6 | 1.2 |
| China | 1.7 | 0.3 |
| India | 1.6 | 0.2 |
| Russia | 1.4 | 0.5 |
| Singapore | 0.5 | 1.7 |
| USA | 2 | 1.5 |
| ... | ... | ... |

$z^{(i)} \in \mathbb{R}^2$

Reduce data from 50D to 2D

# 차원 축소의 이점 - 시각화

- 데이터 간의 상대적 위치를 직관적으로 시각화 할 수 있음
  - 단 각 차원에 대한 의미부여 필요



# 차원 축소의 이점 - 차원의 저주 극복 가능

- KNN 뿐만 아니라 대부분의 머신러닝 모형에 적용되는 현상

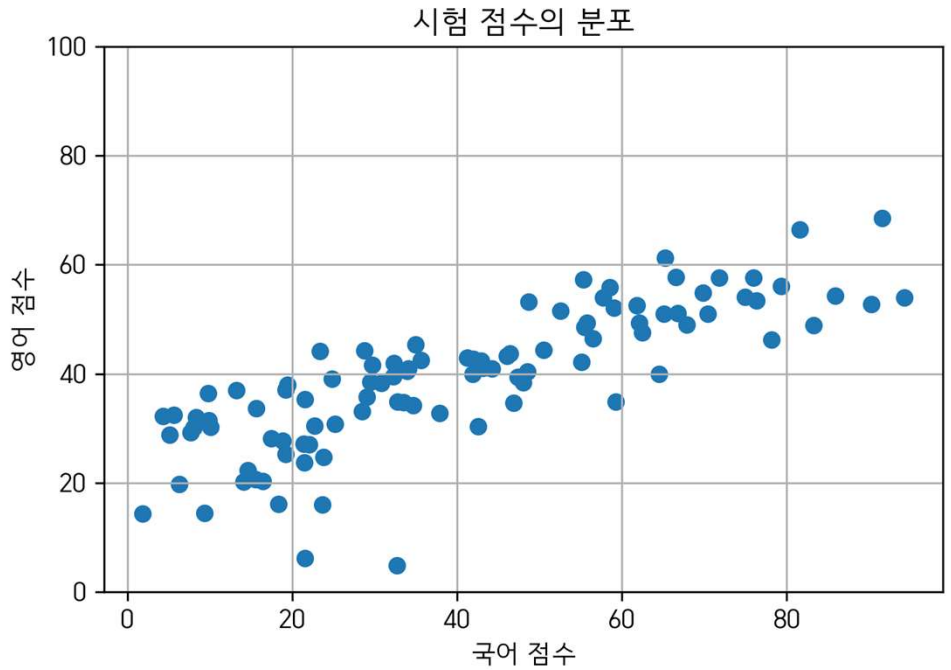
- 데이터의 차원이 지나치게 많은 경우 모형의 성능에 악영향을 미침
  - 학습속도의 저하
  - Overfitting: Generalization 실패 (특성 대비 데이터 수의 부족)

- 차원의 저주 방지 전략
  - 특성 선택: 중요 변수만을 선택
  - 비지도 학습 기반 차원축소 기법 사용
    - PCA
    - Manifolds Methods

# 아이디어

- 국어 점수와 영어 점수를 하나의 점수로 환원하여 계산하여야 한다면 어떻게 취합해야 데이터를 가장 잘 설명할 수 있을까?

| 국어 점수 | 영어 점수 |
|---|---|
| 100 | 83 |
| 70 | 50 |
| 30 | 25 |
| 45 | 30 |
| ⋮ | ⋮ |
| 80 | 60 |

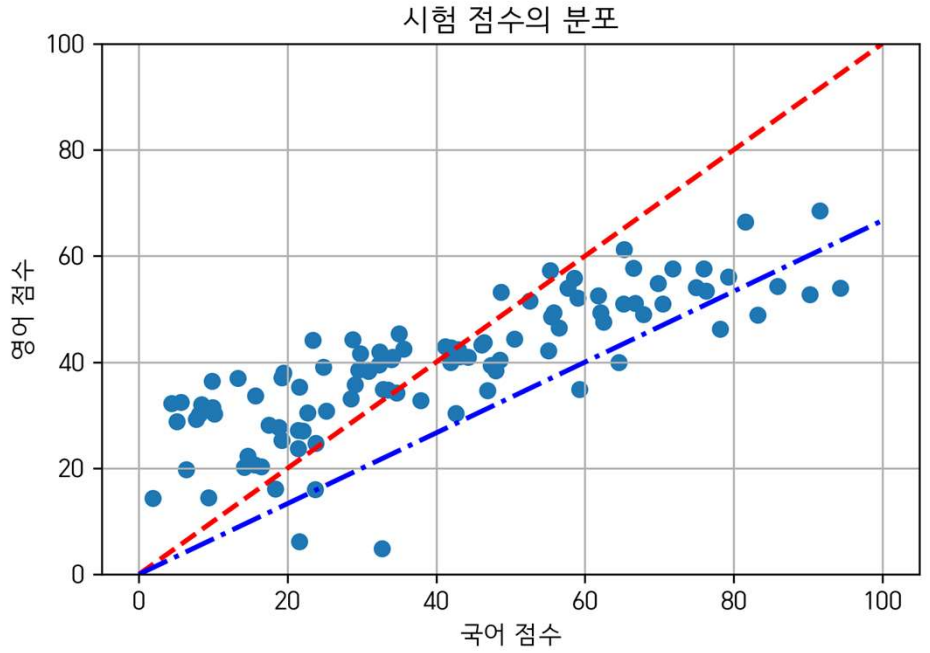


# 아이디어

- 국어 점수와 영어 점수를 하나의 점수로 환원하여 계산하여야 한다면 어떻게 취합해야 데이터를 가장 잘 설명할 수 있을까?

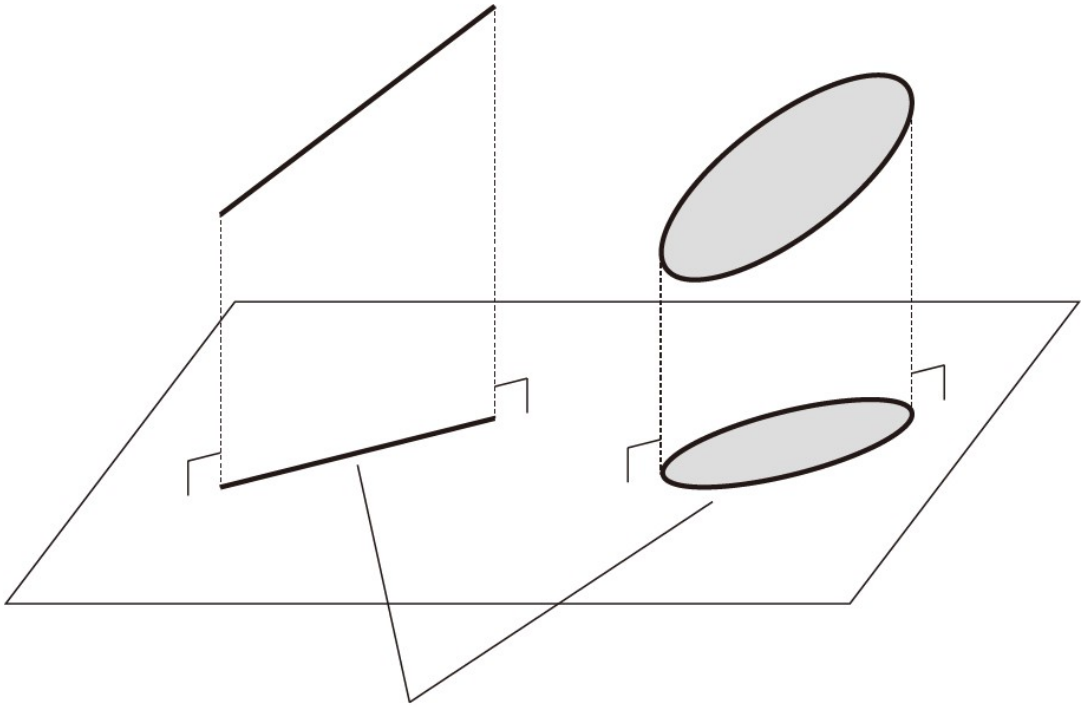
국어×0.5+영어×0.5 (5:5 결합)
국어×0.6+영어×0.4 (6:4 결합)

| 국어 점수 | 영어 점수 |
|---|---|
| 100 | 83 |
| 70 | 50 |
| 30 | 25 |
| 45 | 30 |
| ⋮ | ⋮ |
| 80 | 60 |



# 정사영 (Orthogonal Projection)

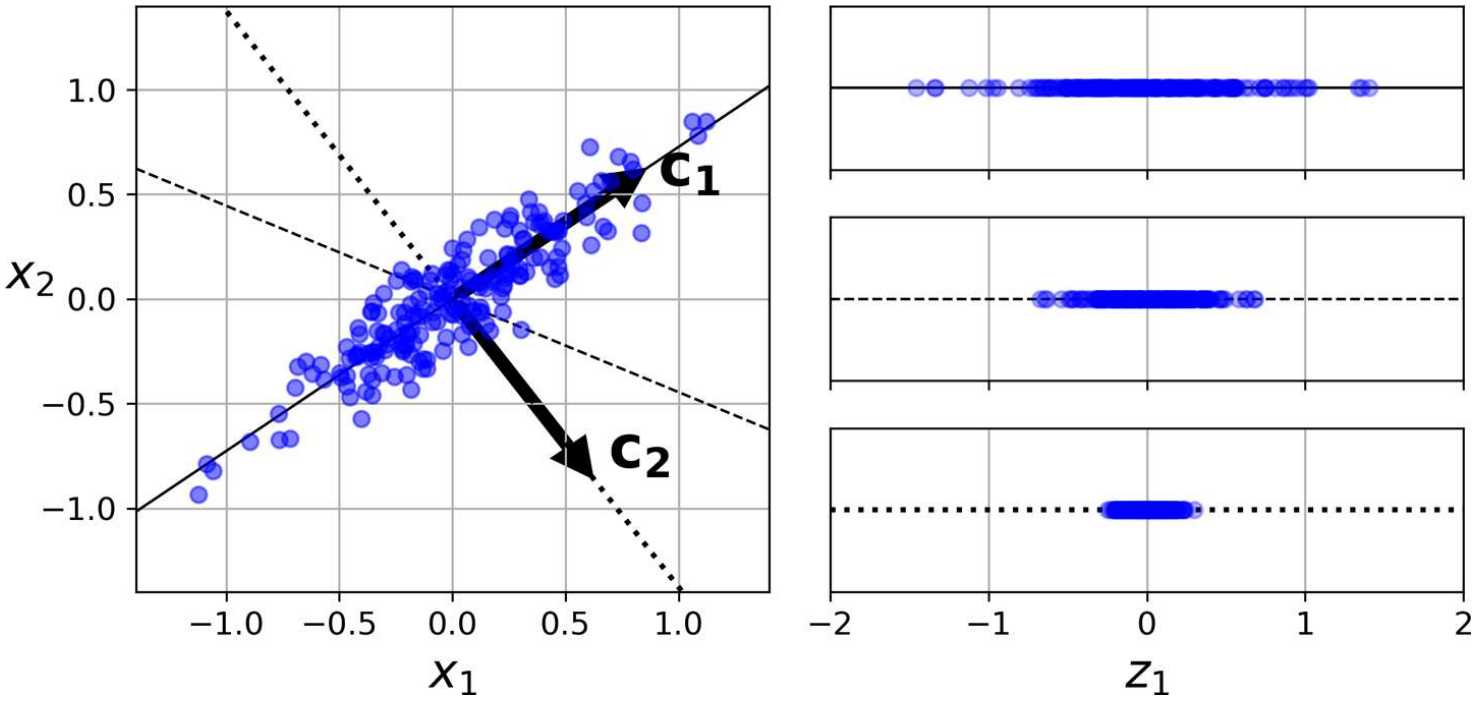
- 평면에 수직으로 빛이 비추어서 만들어진 그림자



Orthographic projection

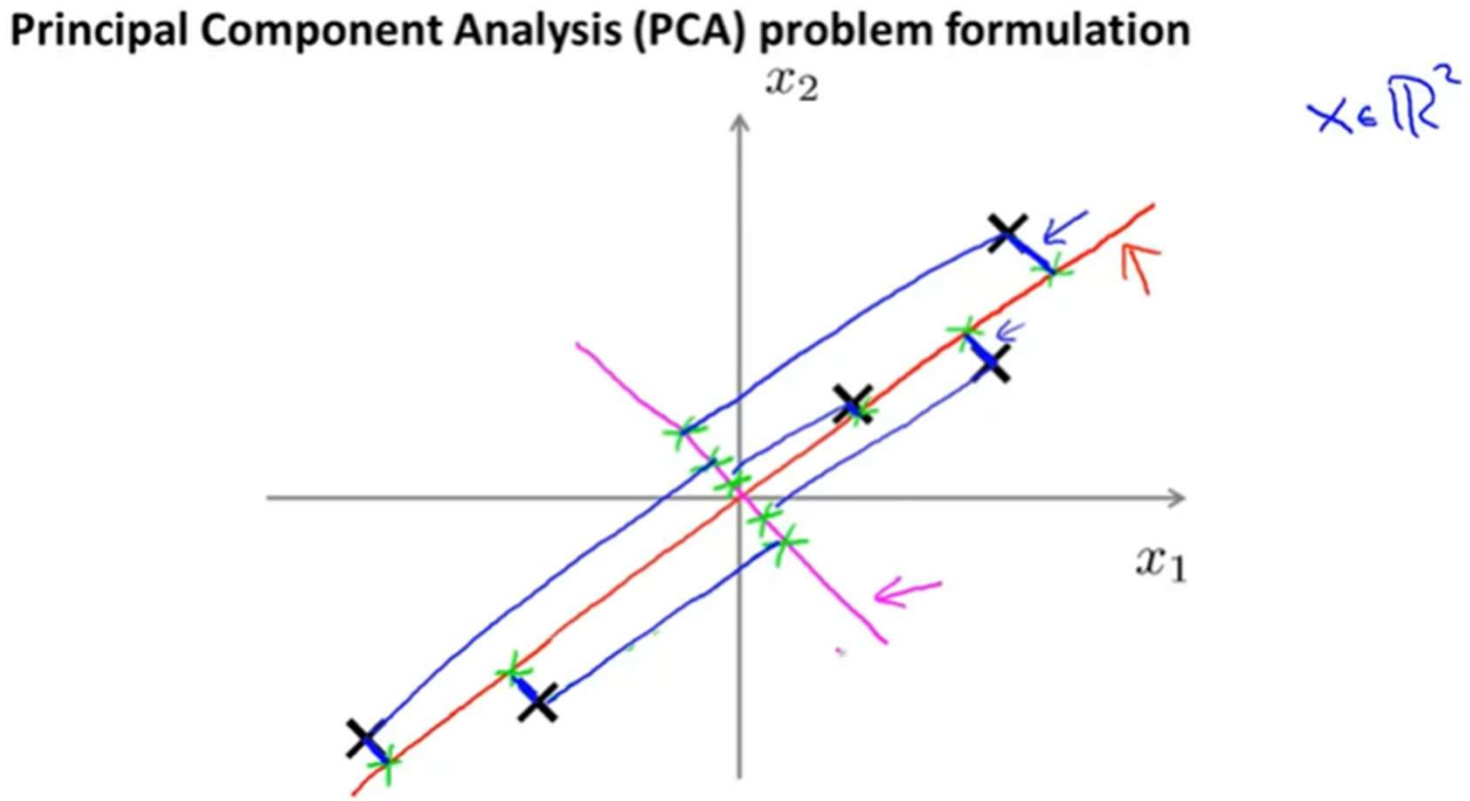
# PCA (Principal Component Analysis)

- 어떤 방향으로 정사영해야 데이터의 정보가 최대한 보존될까?
  - 다른 데이터 간의 차이를 최대한 벌려주는 것이 좋을 것이다. → 사영후 데이터 간의 분산이 커야 한다.
  - C1방향으로 사영했을 때, 데이터 사이의 {분산}이 가장 크다.



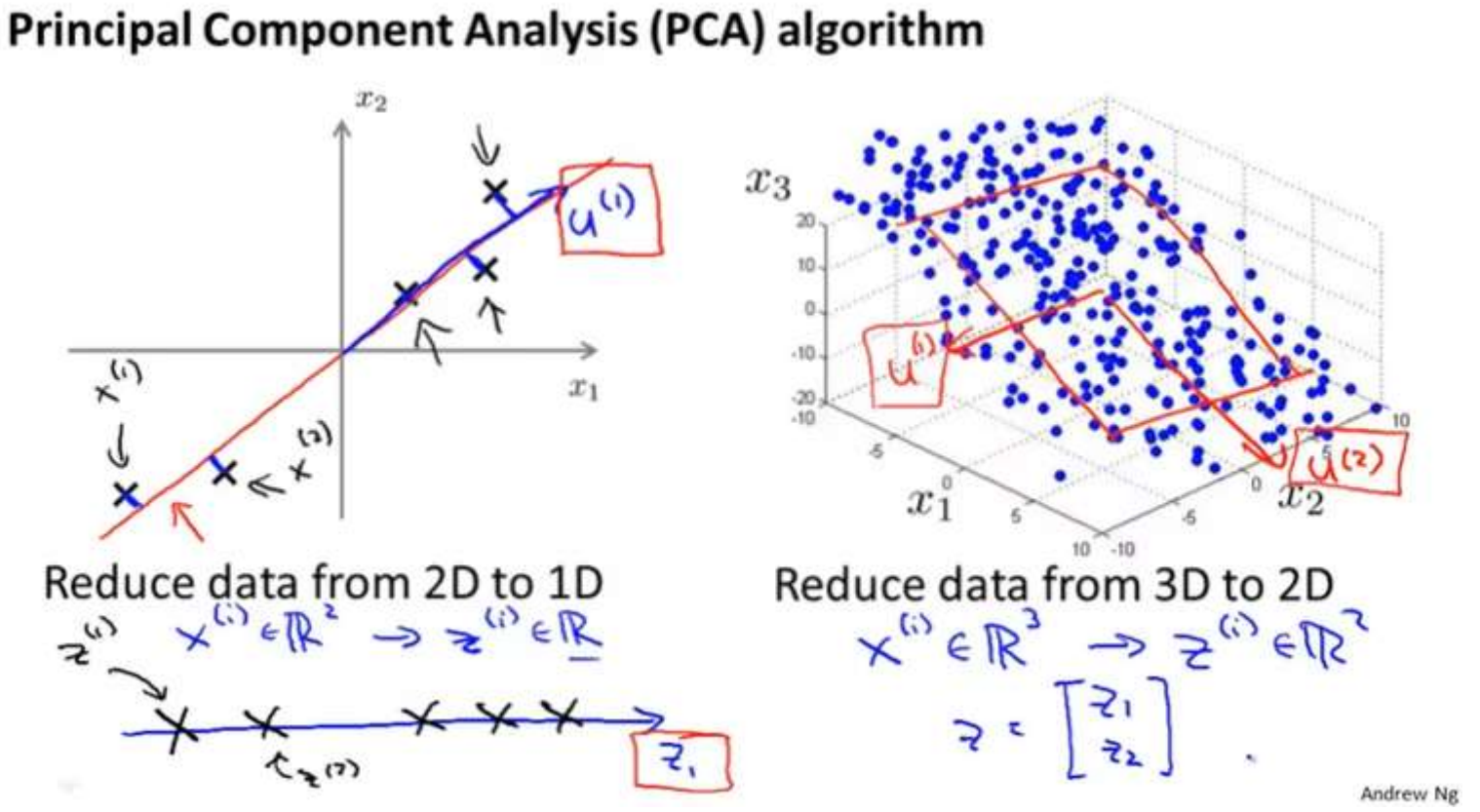
# PCA (Principal Component Analysis)

- 주성분 분석: 전체 p개의 차원에서 데이터의 분산을 최대한 보존할 수 있는 k개 (p<k)의 직교하는 차원을 찾는 방법



# PCA (Principal Component Analysis)

- 주성분 분석: 전체 p개의 차원에서 데이터의 분산을 최대한 보존할 수 있는 최소화 시키는 k개 (p<k)의 직교하는 차원을 찾는 방법



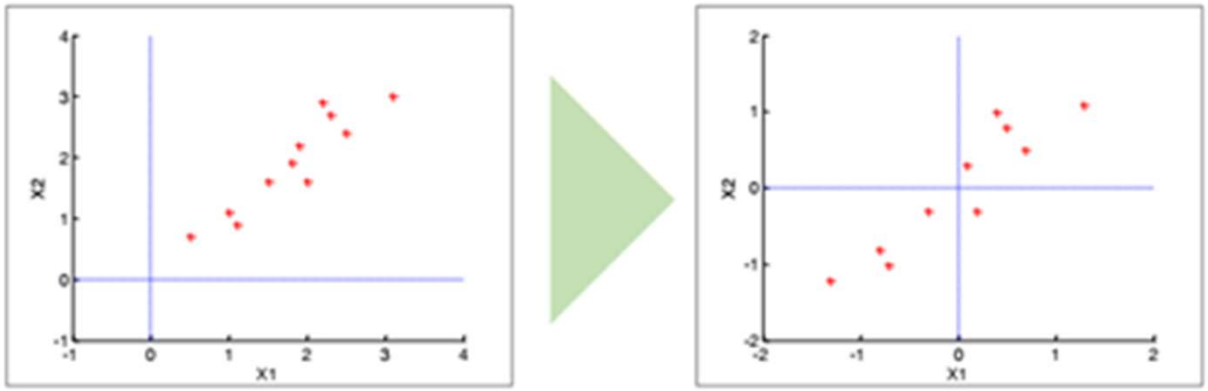
# PCA 알고리즘

## 1단계: 데이터 센터링 (Data Centering)

데이터의 평균을 0으로 변경한다. Standardization을 수행하면 된다.

$\mu$는 데이터의 평균, $\sigma$는 표준편차이다.

$$X' = \frac{X - \mu}{\sigma}$$



# PCA 알고리즘

## 2단계: 최적화 문제 정의

데이터 $X$를 벡터 $W$에 사영했을 때, 분산 $V$를 최대화해야 한다.

($S$는 $X$의 공분산 행렬이다.)

$$V = \frac{1}{n}(w^T X)(w^T X)^T = \frac{1}{n} w^T X X^T w = w^T S w$$

$$\text{maximize } w^T S w$$
$$\text{s.t. } w^T w = 1 (w \text{ is unit vector})$$

# PCA 알고리즘

## 3단계: 최적화 문제 해결

라그랑지 멀티플라이어를 적용한다.

$$L = w^T S w - \lambda(w^T w - 1)$$

$$\frac{\partial L}{\partial w} = 0 \rightarrow Sw - \lambda w = 0 \rightarrow (S - \lambda I)w = 0$$

이때, $W$가 $S$의 고유 벡터가 되고, $\lambda$가 $S$의 고유값이 된다.

# PCA 알고리즘 결과 - 고유벡터 및 고유값

- 고유 벡터: 사영하는 축의 방향
- 고유값: 해당 고유 벡터가 설명하는 분산의 값

$$\mathbf{S} = \begin{bmatrix} 0.6779 & -0.7352 \\ 0.7352 & 0.6779 \end{bmatrix} \qquad \lambda = \begin{bmatrix} 1.2840 & 0.0491 \end{bmatrix}$$

$w_1$ $\lambda_1$

모든 2차원 데이터를
[0.6779, 0.7352]
방향으로 변환

$$w_1^T X = z_1$$

전체 데이터의 분산의 96% 보존
= 1.284/(1.284+0.0491)

# PCA 알고리즘 - 고유벡터 및 고유값

- 고유 벡터: 사영하는 축의 방향
- 고유값: 해당 고유 벡터가 설명하는 분산의 값

$$\mathbf{S} = \begin{bmatrix} 0.6779 & -0.7352 \\ 0.7352 & 0.6779 \end{bmatrix} \qquad \lambda = \begin{bmatrix} 1.2840 & 0.0491 \end{bmatrix}$$

$w_2$ $\lambda_2$

모든 2차원 데이터를
[-0.7352, 0.6799] 방향으로 변환

전체 데이터의 분산의 4% 보존 =
0.0491/(1.284+0.0491)

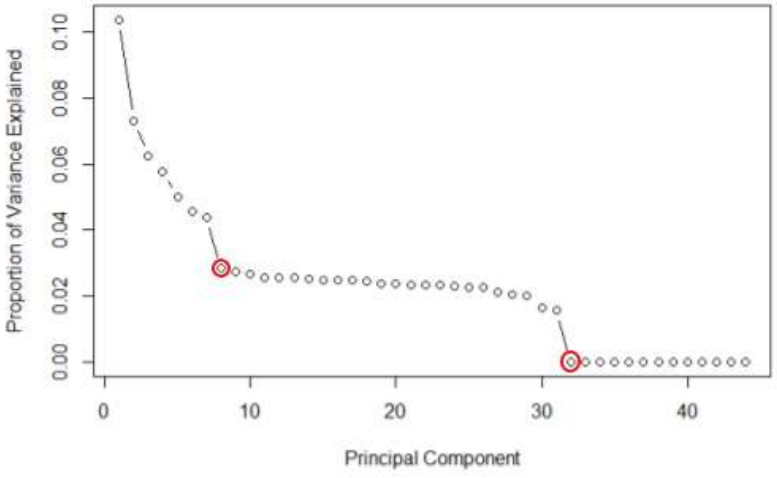
$$w_2^T X = z_2$$

# 주성분의 개수 선정하기

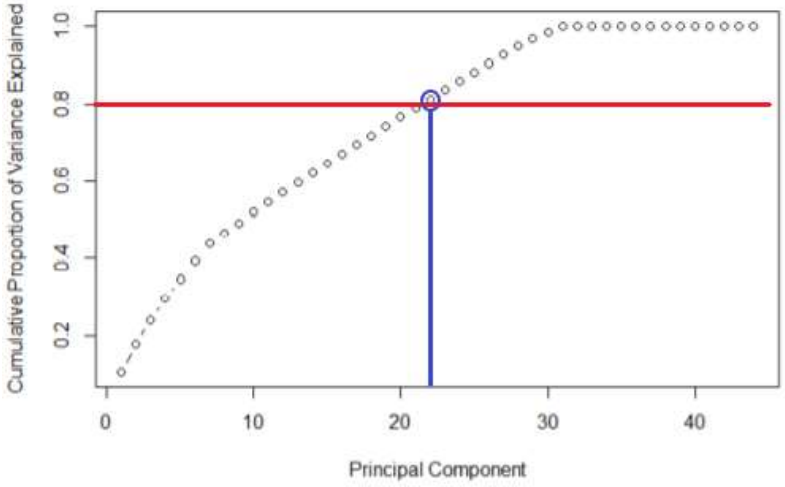
## 주성분의 개수 선정하기

그렇다면, $n$차원의 데이터를 얼마나 축소해야할까? 크게 두 가지 방법으로 축소할 차원을 선정한다.

- 고유 값 감소율이 유의미하게 낮아지는 Elbow Point에 해당하는 주성분 선택하기
- 일정 수준 이상의 분산 비를 보존하는 최소한의 주성분 선택하기 (보통 70%이상)



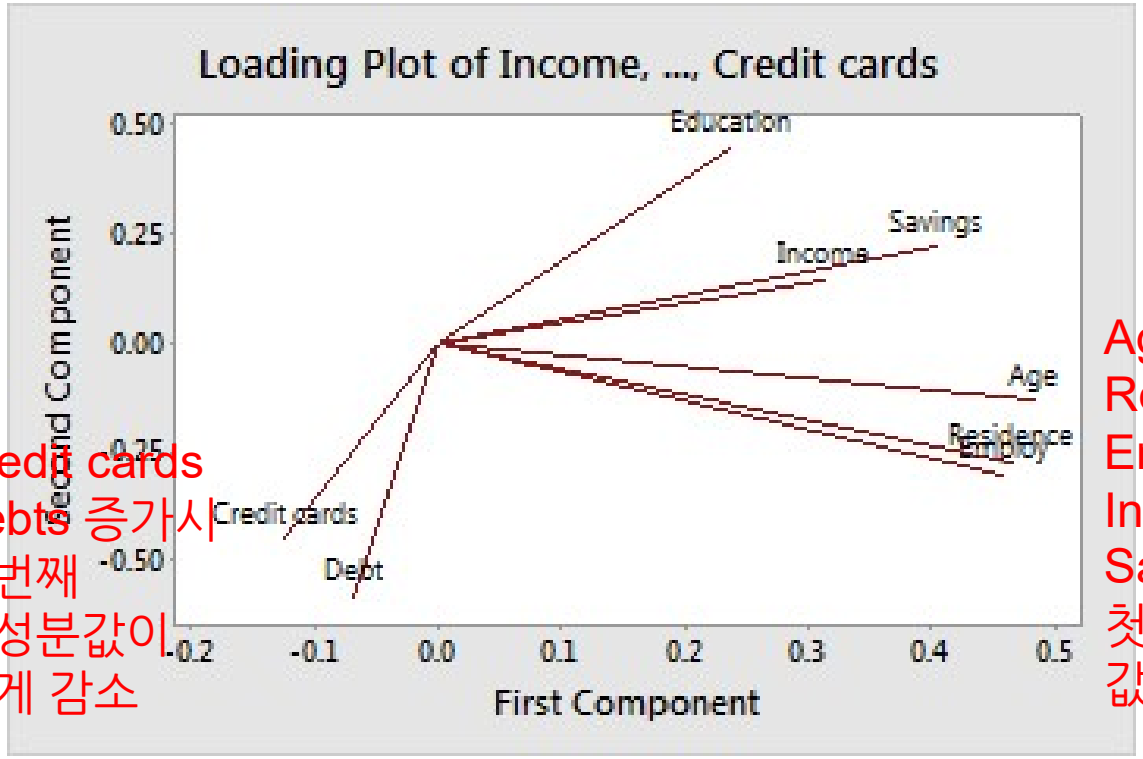

K 번째 주성분의
고유값




K번째 주성분이
표현하는 분산 비

# 주성분 해석하기

- 각 변수의 값을 증가시킬때 주성분이 어떤 영향을 받는지 시각화

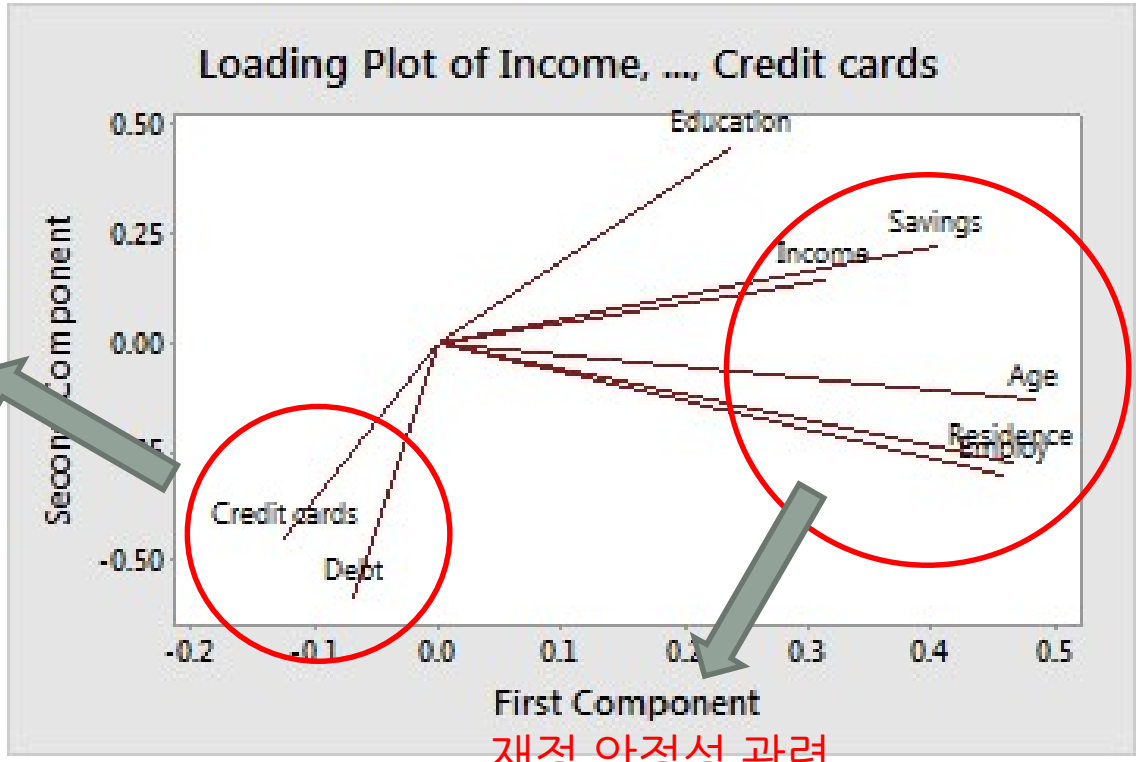



Credit cards
Debts 증가시
두번째
주성분값이
크게 감소

Age,
Residence,
Employee,
Income,
Saving 증가시
첫번째 주성분
값이 크게 증가

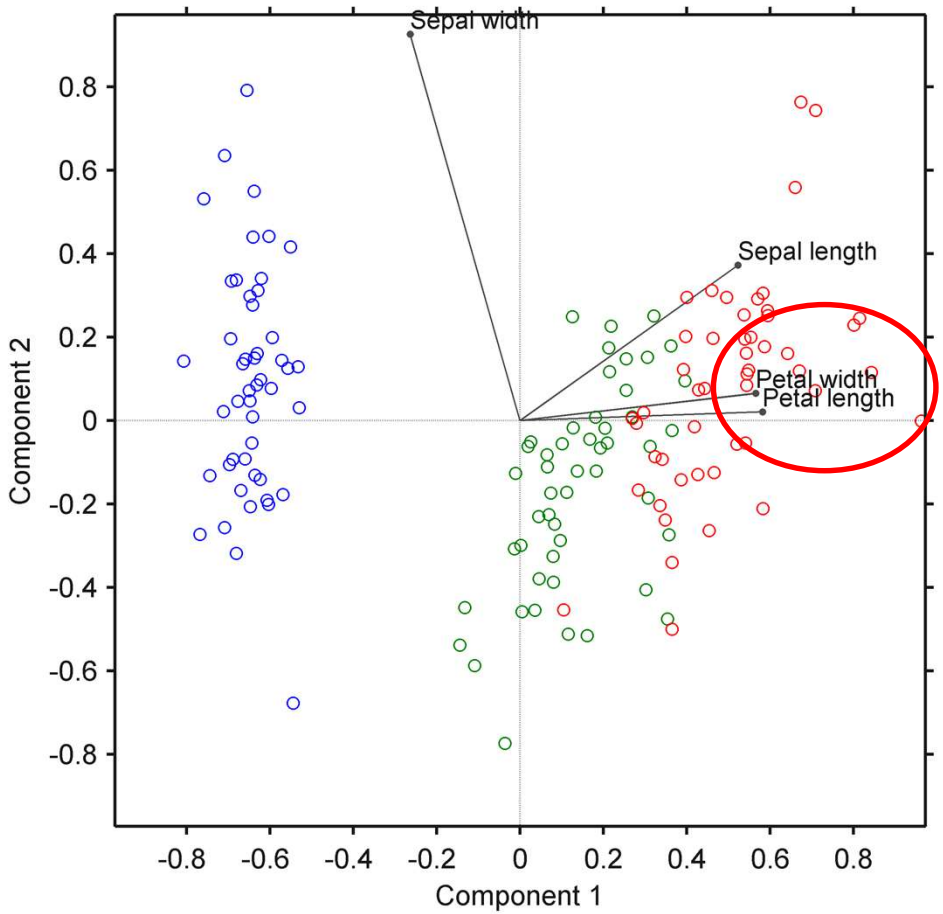
# 주성분 해석하기

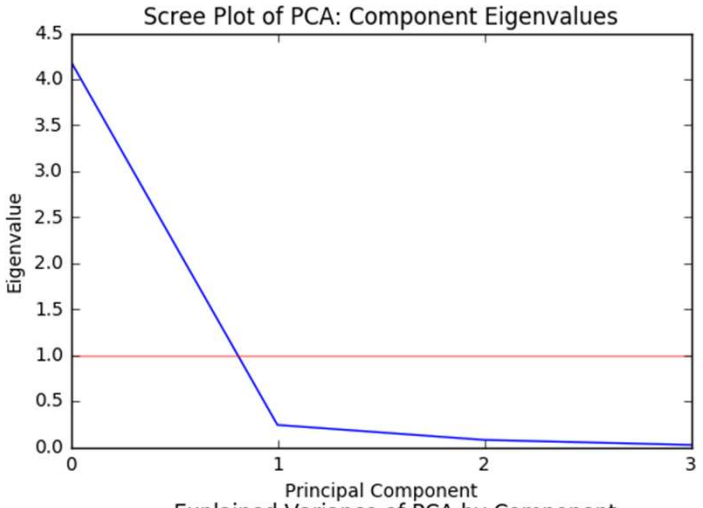
채무 관련 변수

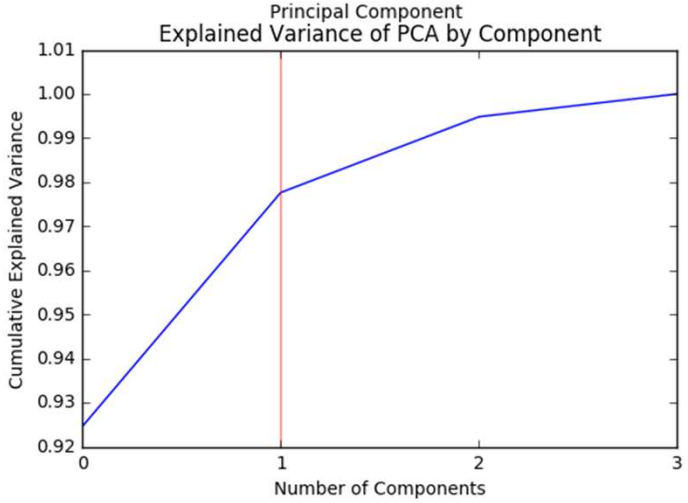



재정 안정성 관련 변수

# 주성분 해석하기 - Loading plot



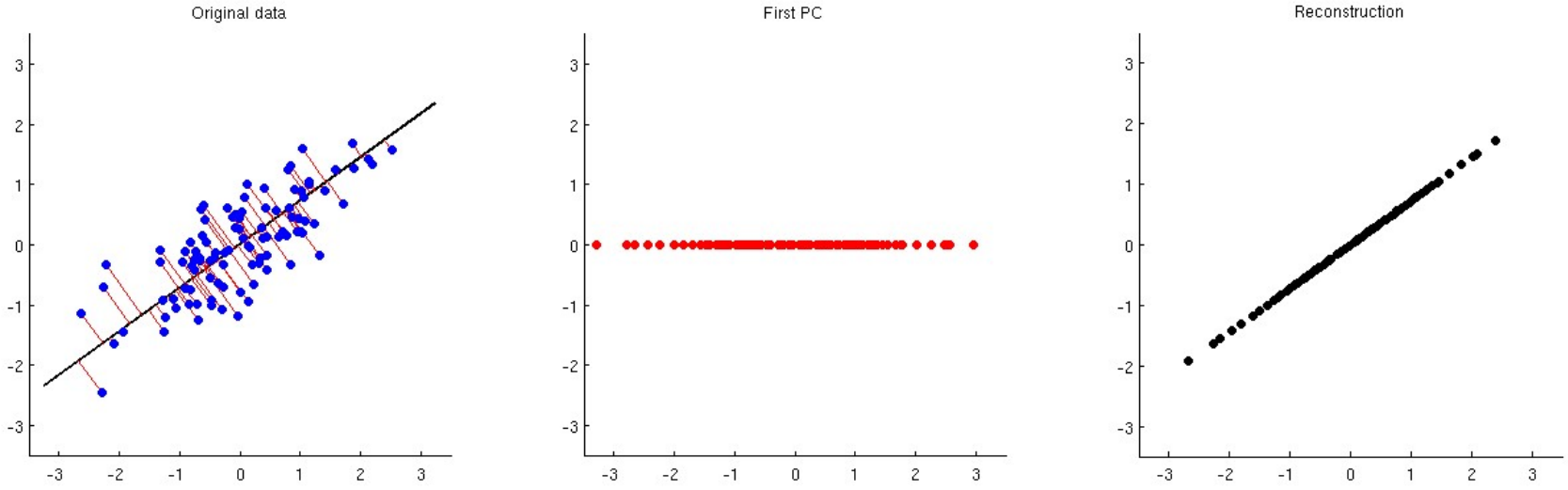



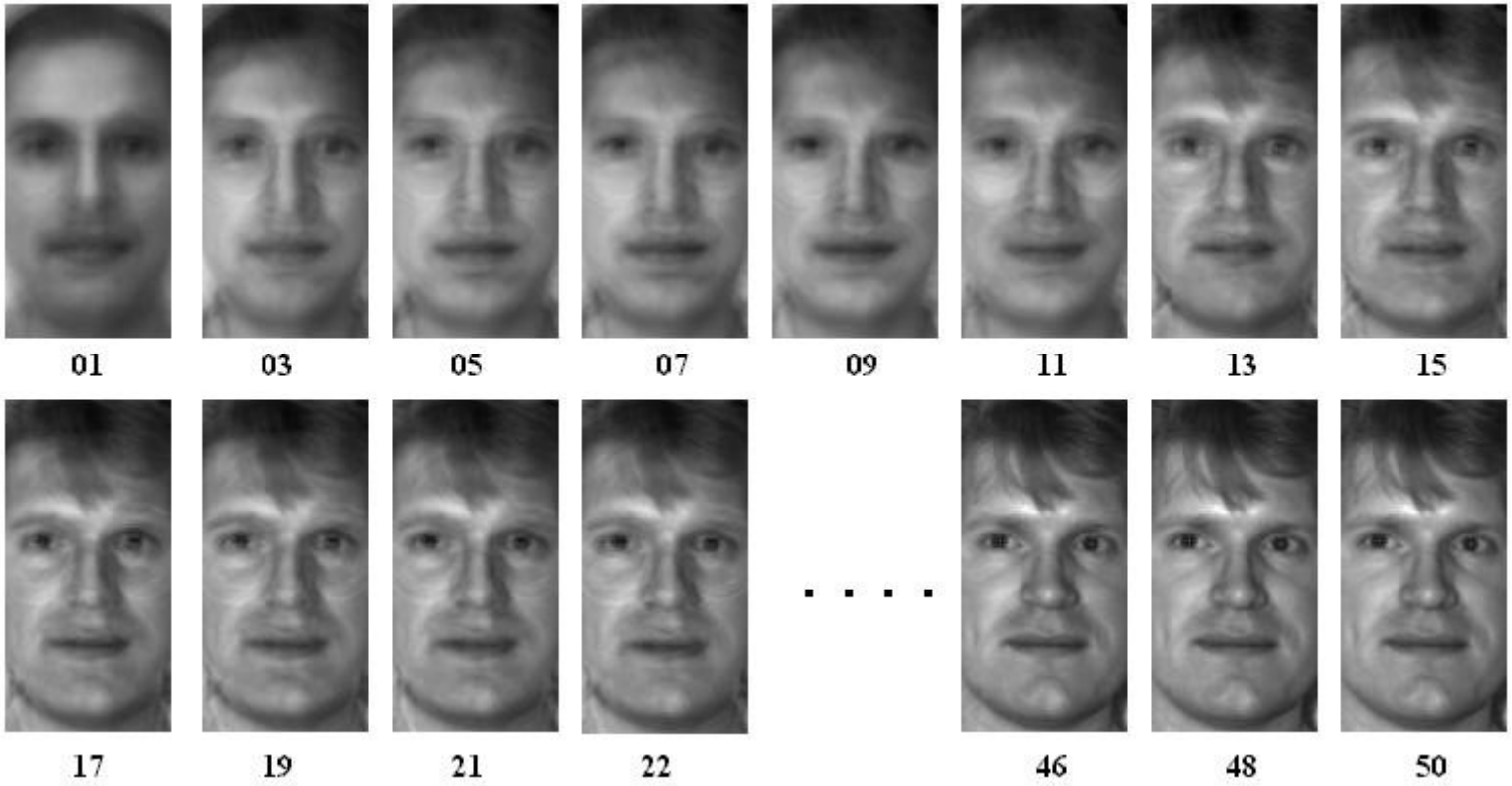
# 주성분에서 원데이터로 복구

- z: 차원이 감소된 원데이터
- 주성분 벡터를 활용하여 원래 차원의 데이터로 복귀 가능
- 하지만 주성분의 차원이 적다면, 원래 데이터로 완전한 복귀는 불가능

$$\text{PCA reconstruction} = \text{PC scores} \cdot \text{Eigenvectors}^{\top} + \text{Mean}$$



# 주성분에서 원데이터로 복구



# PCA의 주 용도

- 압축
  - 저장되는 데이터의 메모리/디스크 감소
  - 학습 알고리즘의 속도 증가

- 시각화
  - 차원을 낮추어 데이터의 전체적인 분포에 대한 인사이트 도출
  - 이상치/군집가능성 파악

# PCA의 올바른 사용

- PCA는 정보의 양을 감소시키고, 특히 예측값에 상관없이 차원을 축소하기 때문에 꼭 필요한 정보를 잃을 수도 있다.
- 따라서, overfitting 문제를 해결하려면 regularzation을 사용하는 것이 권장된다.

$$\min_{\theta} \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2$$

# PCA의 올바른 사용

- 전체 데이터를 모두 사용하여 학습을 수행한다
  - Regularization등 포함

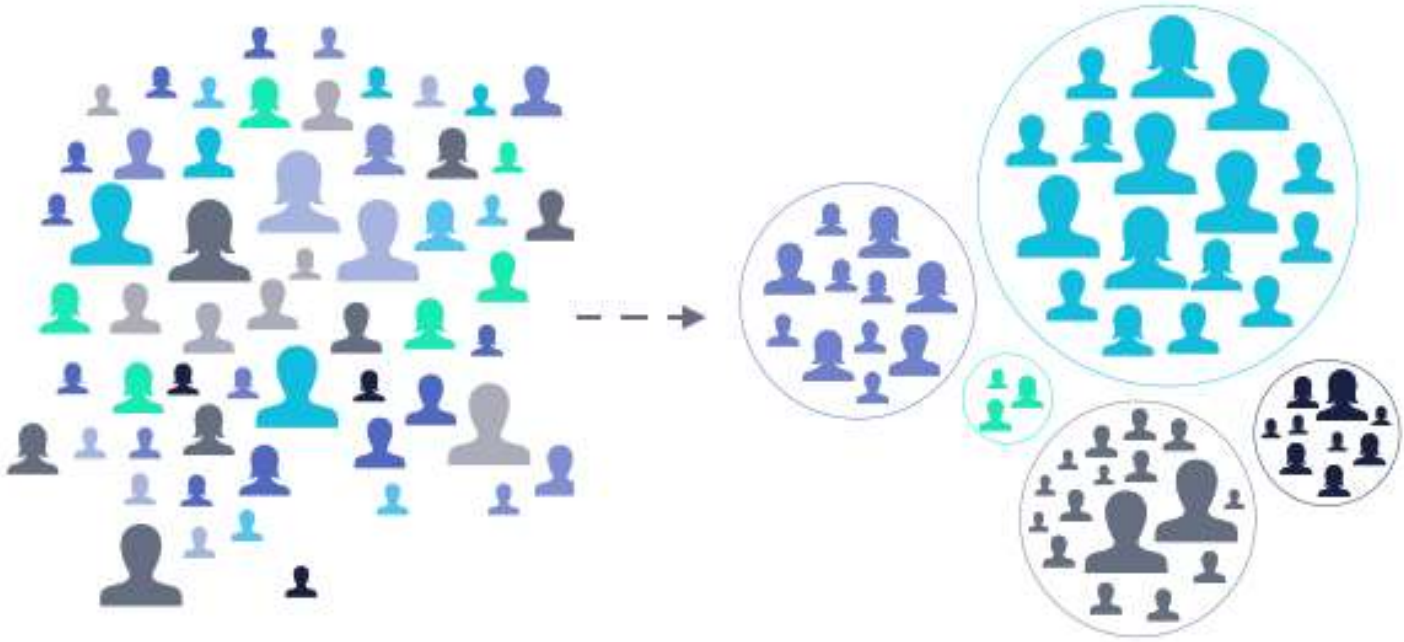
- PCA는 위 절차가 만족스럽지 않을 때에만 사용한다.
- 즉, PCA를 우선적으로 사용하면 안된다.

# 군집분석
# (Clustering Analysis)

시스템경영공학부
이지환 교수

# 군집분석(Clustering Analysis)

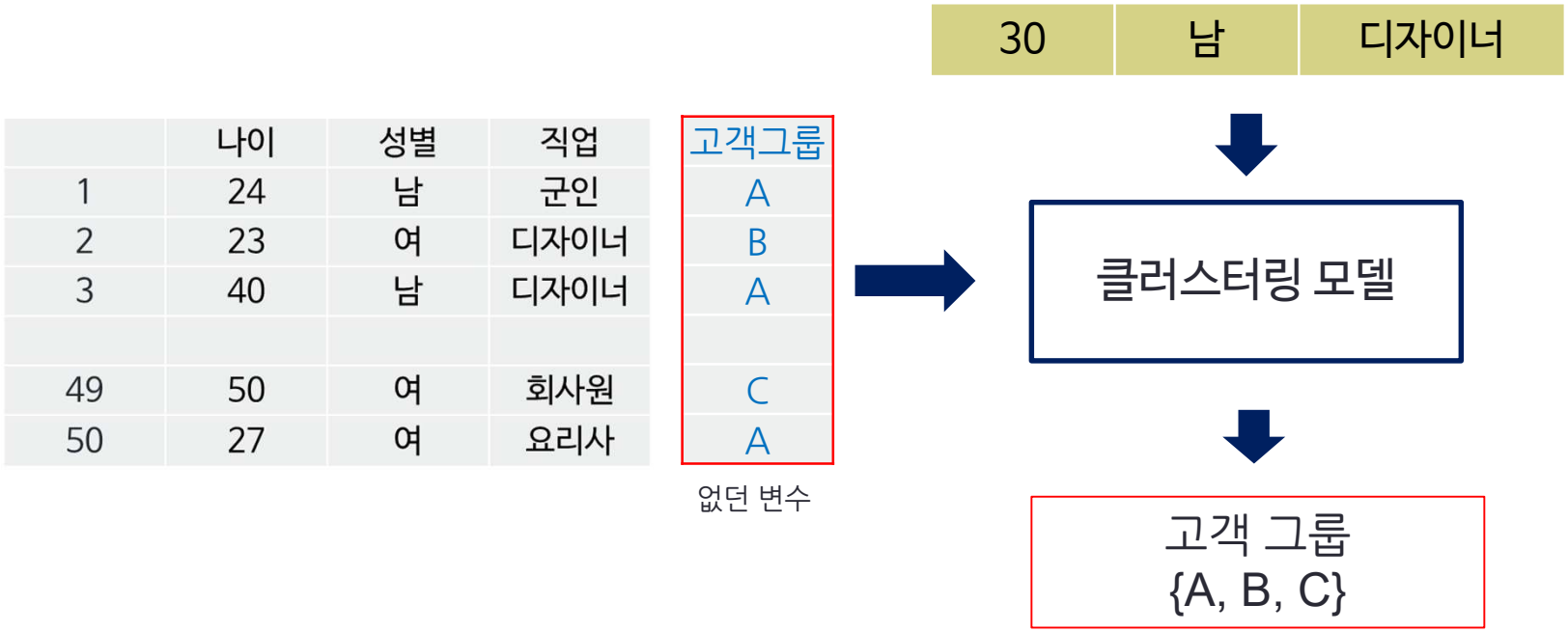
- 서로 유사한 데이터 객체들 끼리 묶어 군집을 만드는 기법



# 비지도학습 (Unsupervised Learning)

- 학습하는 데이터내에 예측하고자 하는 값이 없는 경우
- 예시) 나이, 성별, 직업의 유사도에 따라 3개의 그룹으로 나누고 싶다.
  - 데이터를 설명하는 내재적인 구조를 학습해야 함

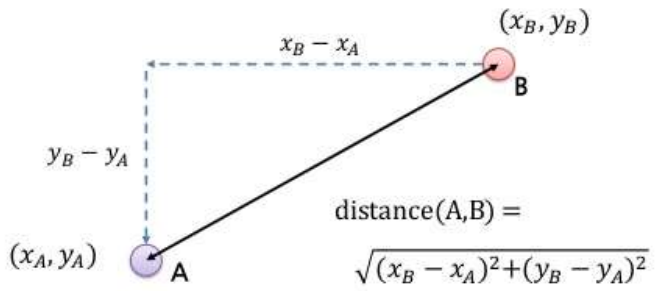
- 군집분석(Clustering Analysis)



# 데이터 객체간의 유사도 측정

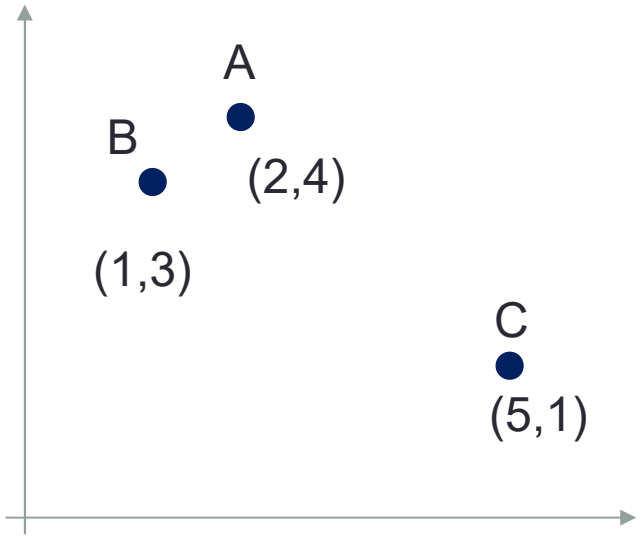
- 두 데이터의 특성이 유사함을 어떻게 측정해야 할까?
- 유클리디안 거리

**유사도 매칭**
**유클리디안 거리(Euclidean distance)**



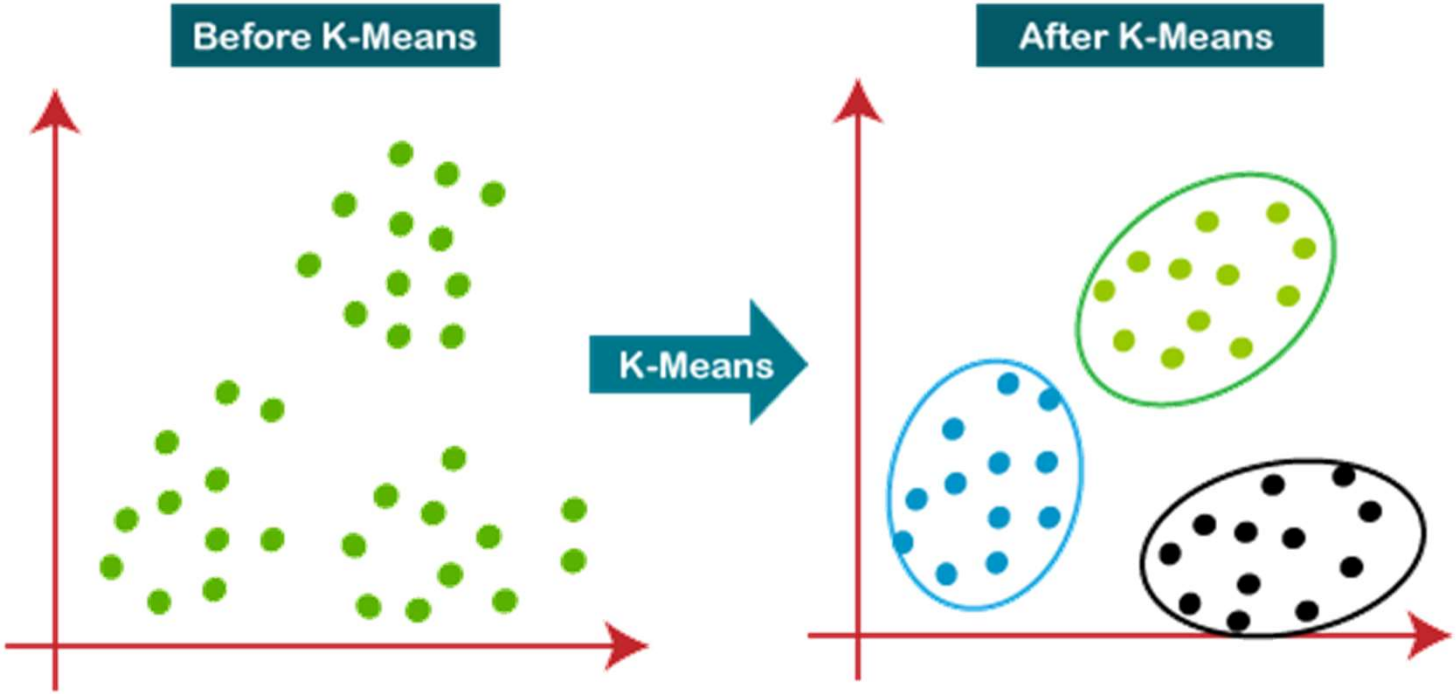

m차원 공간

$$\text{distance(A,B)} = \sqrt{(d_{1,A} - d_{1,B})^2 + (d_{2,A} - d_{2,B})^2 + (d_{3,A} - d_{3,B})^2 + \cdots + (d_{m,A} - d_{m,B})^2}$$



# K-means 클러스터링

- 주어진 데이터를 k개의 클러스터로 묶는 알고리즘
- 정해진 중점을 기준으로 가까운 점들을 묶는 과정을 반복하여 군집 파악



# K-means 클러스터링 알고리즘

- 입력값
  - K (하이퍼 패러미터) : 클러스터의 개수
  - 데이터셋 (비지도학습용)

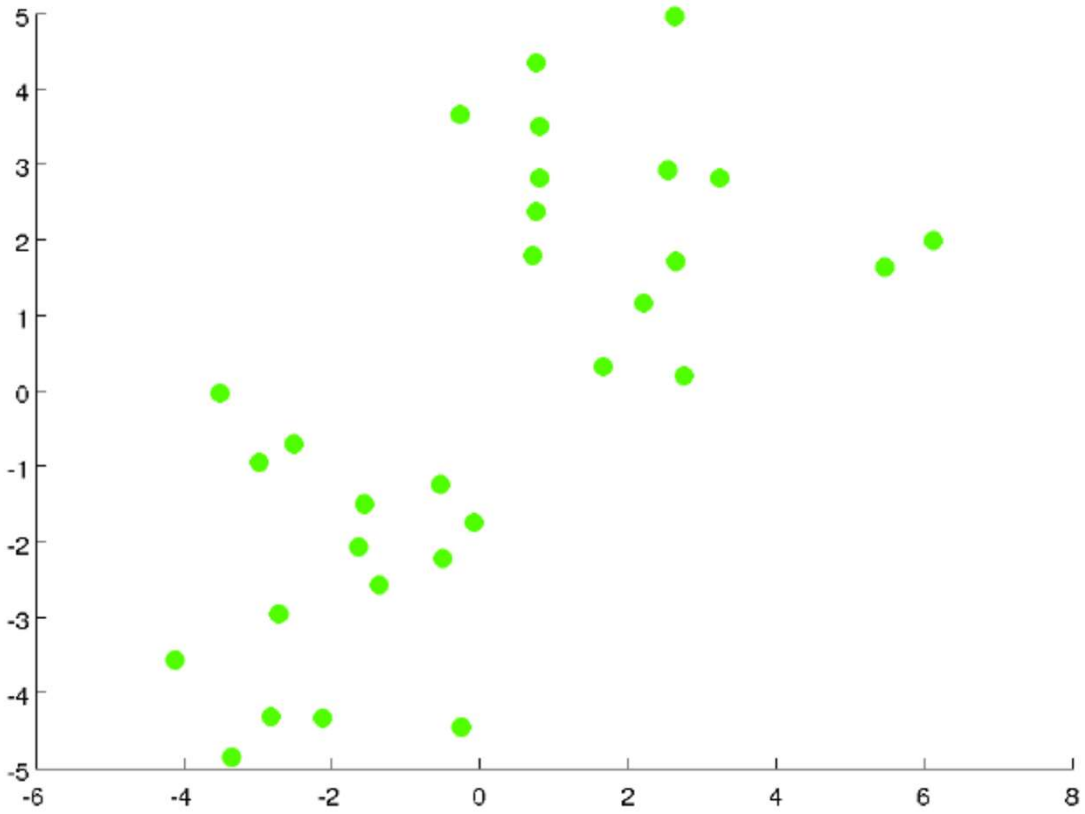
- notation
  - $\mu_k$: k번째 군집의 중점
  - $X^{(i)}$ : i번째 데이터
  - $c^{(i)}$ : i번째 데이터가 소속된 군집

# K-means 클러스터링 알고리즘

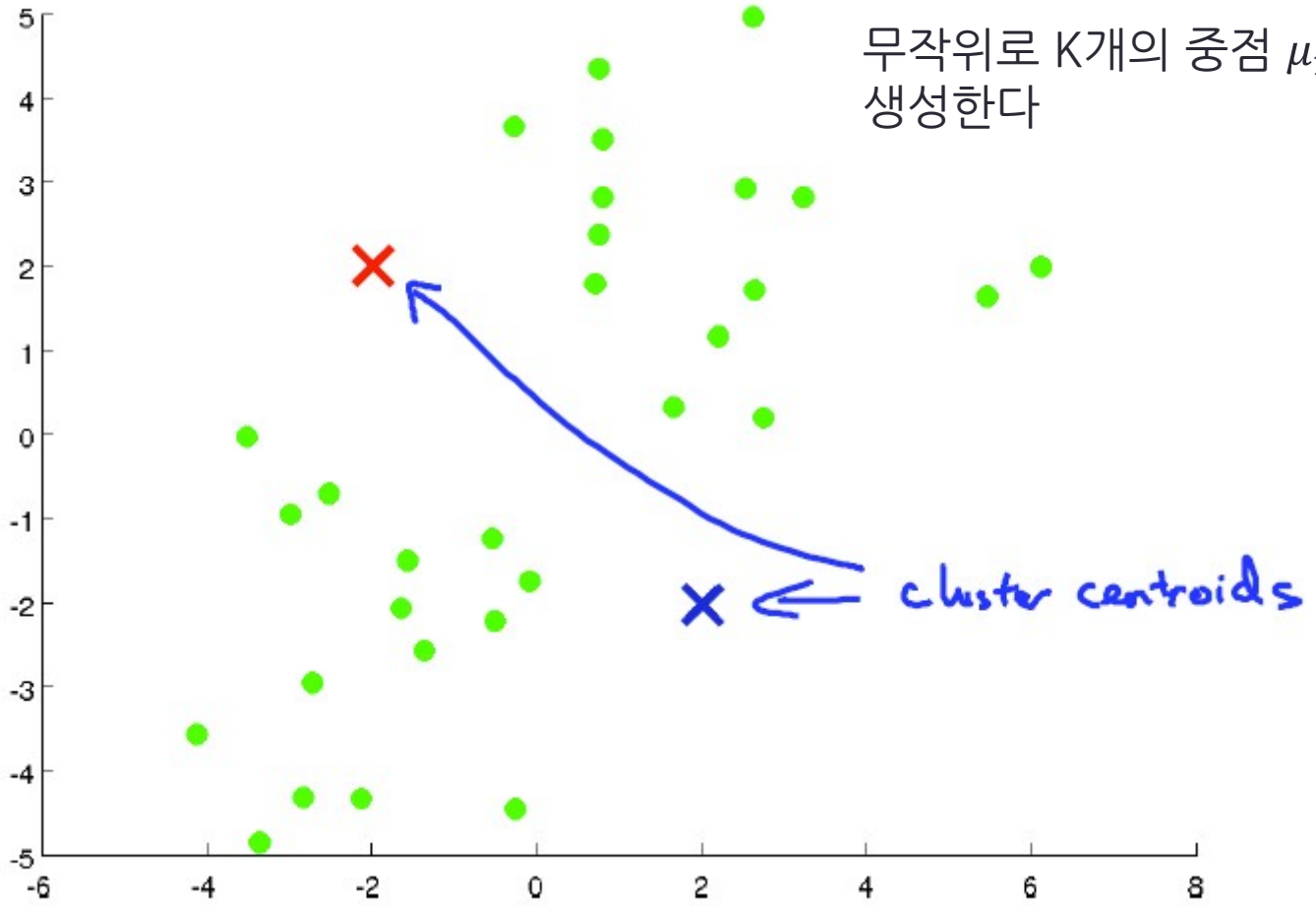
- 무작위로 K개의 중점 $\mu_1, \mu_2, \ldots, \mu_K$을 생성한다
- Step 1-2를 반복
  - Step 1) 데이터 클러스터 할당
    - 모든 데이터 $X^{(1)}, \ldots, X^{(m)}$에 대하여 다음을 반복한다
    - $c^{(i)} := \mu_1, \mu_2, \ldots, \mu_K$와 거리를 비교하여 가장 가까운 거리에 있는 클러스터의 인덱스로 갱신
  - Step 2) 중점 갱신
    - 모든 클러스터 k=1부터 K까지 다음을 반복한다
    - $\mu_k$:= 클러스터 k에 포함되어 있는 모든 점들의 무게중심으로 갱신

# 계산 예시

- 원 데이터
- K=2

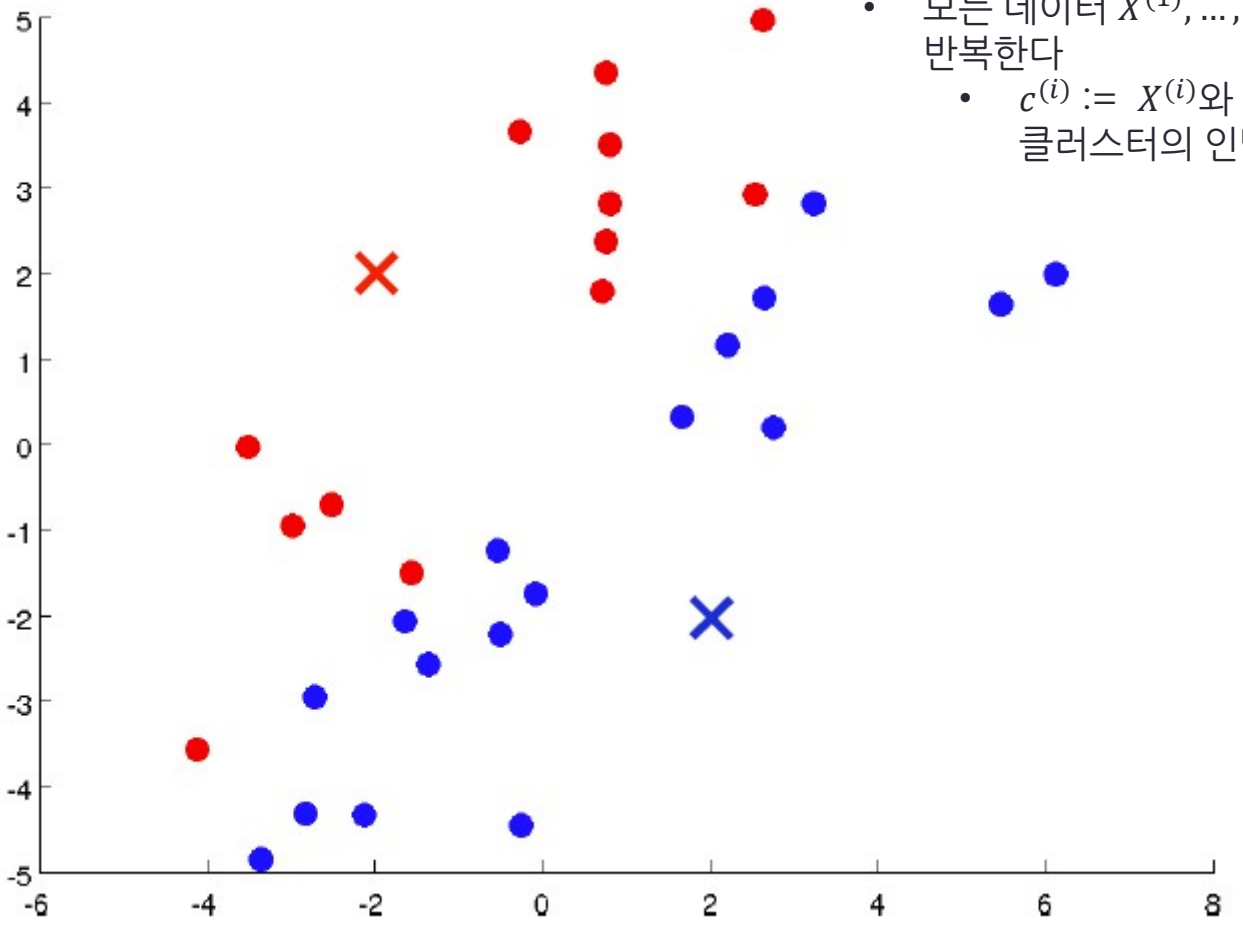


# 계산 예시 - 중점 생성



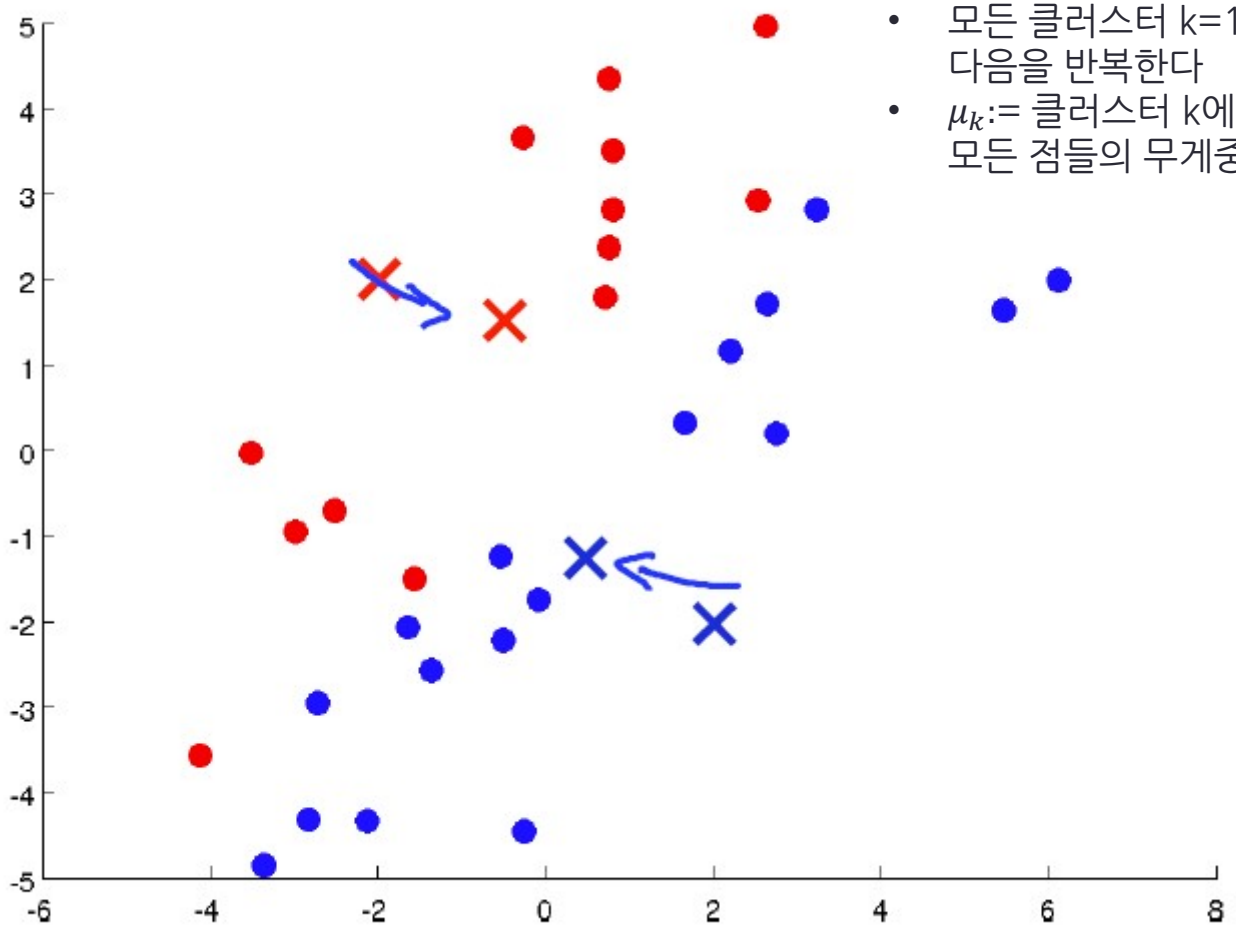

무작위로 K개의 중점 $\mu_1, \mu_2, \ldots, \mu_K$을 생성한다

# 계산 예시 - 데이터 클러스터 할당



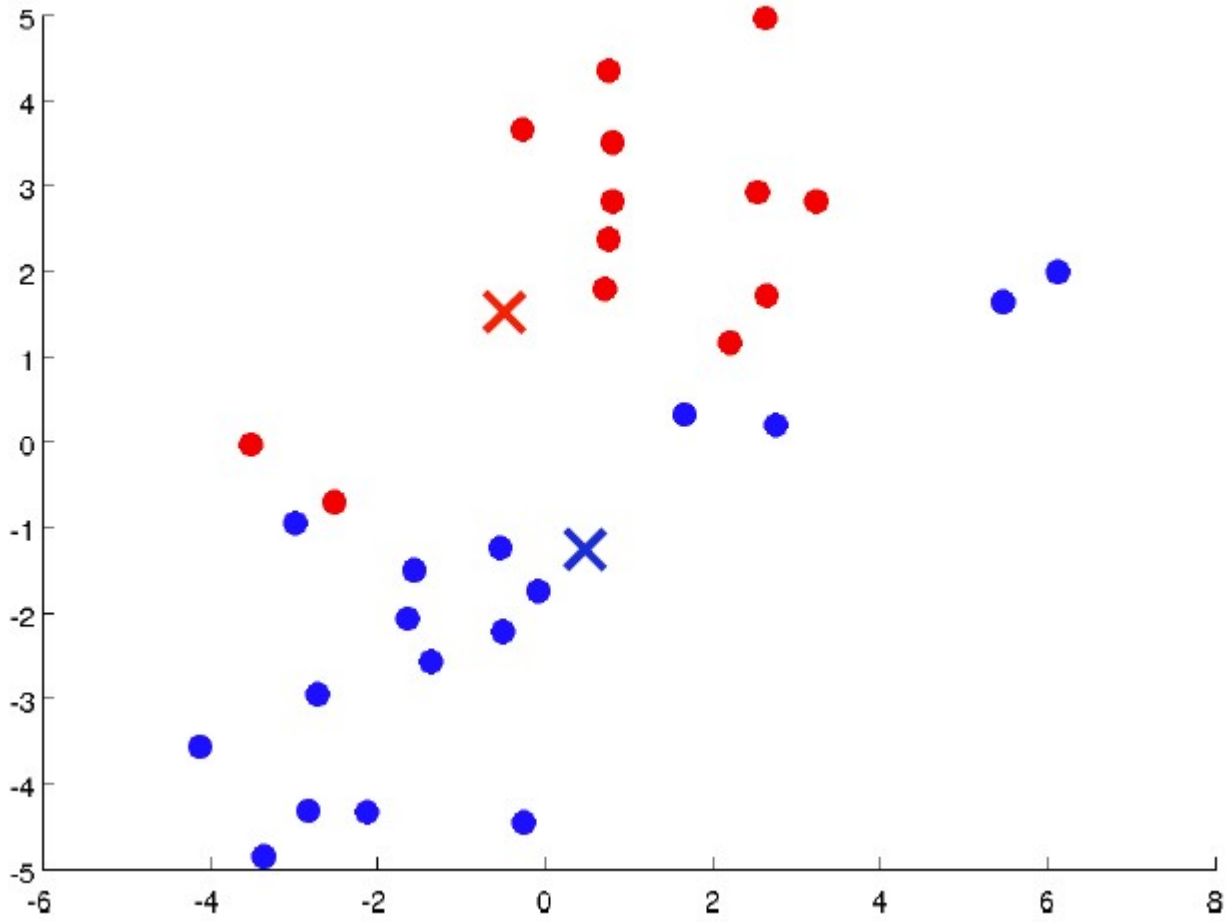

- 모든 데이터 $X^{(1)}, \ldots, X^{(m)}$에 대하여 다음을 반복한다
  - $c^{(i)} := X^{(i)}$와 가장 가까운 거리에 있는 클러스터의 인덱스로 갱신

# 계산 예시 - 무게중심 갱신

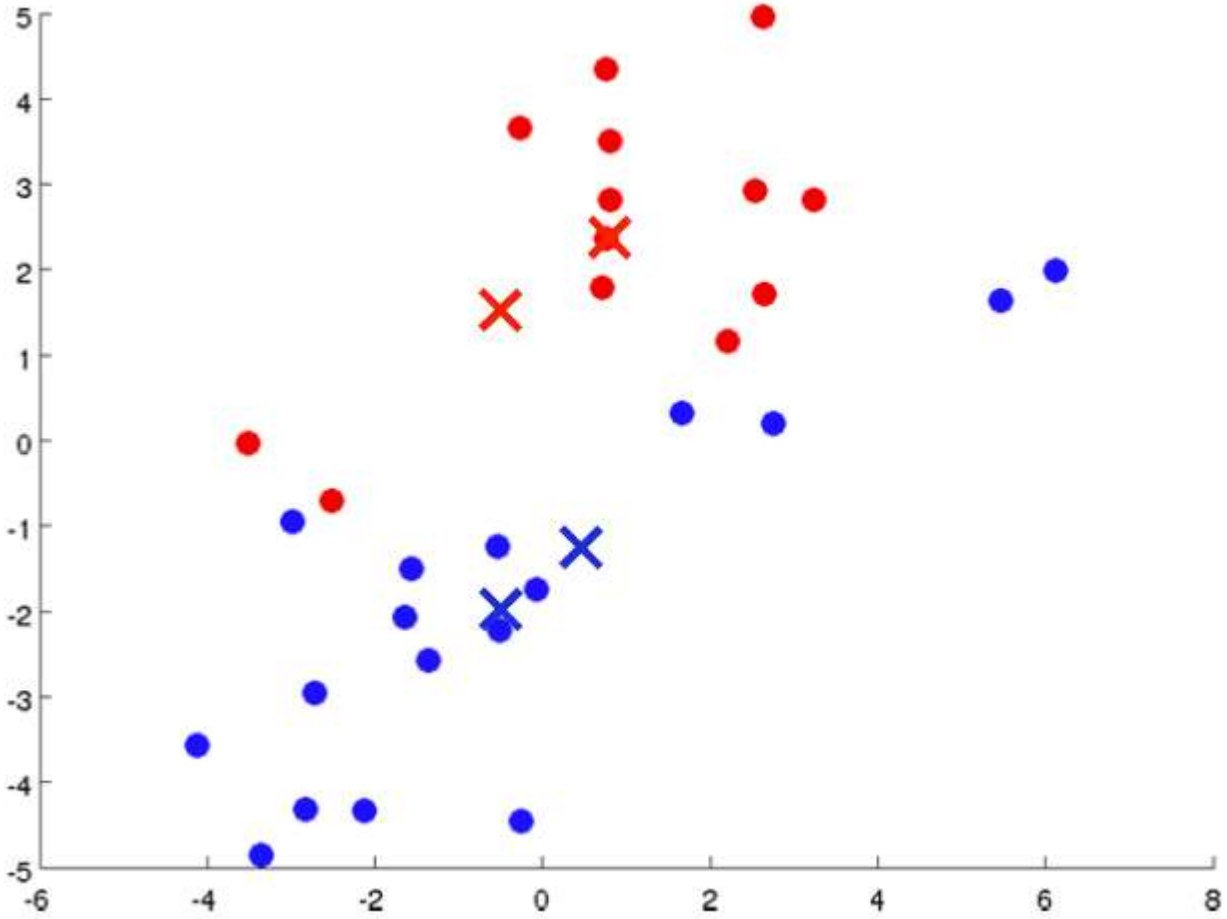



- 모든 클러스터 k=1부터 K까지 다음을 반복한다
- $\mu_k$:= 클러스터 k에 포함되어 있는 모든 점들의 무게중심으로 갱신

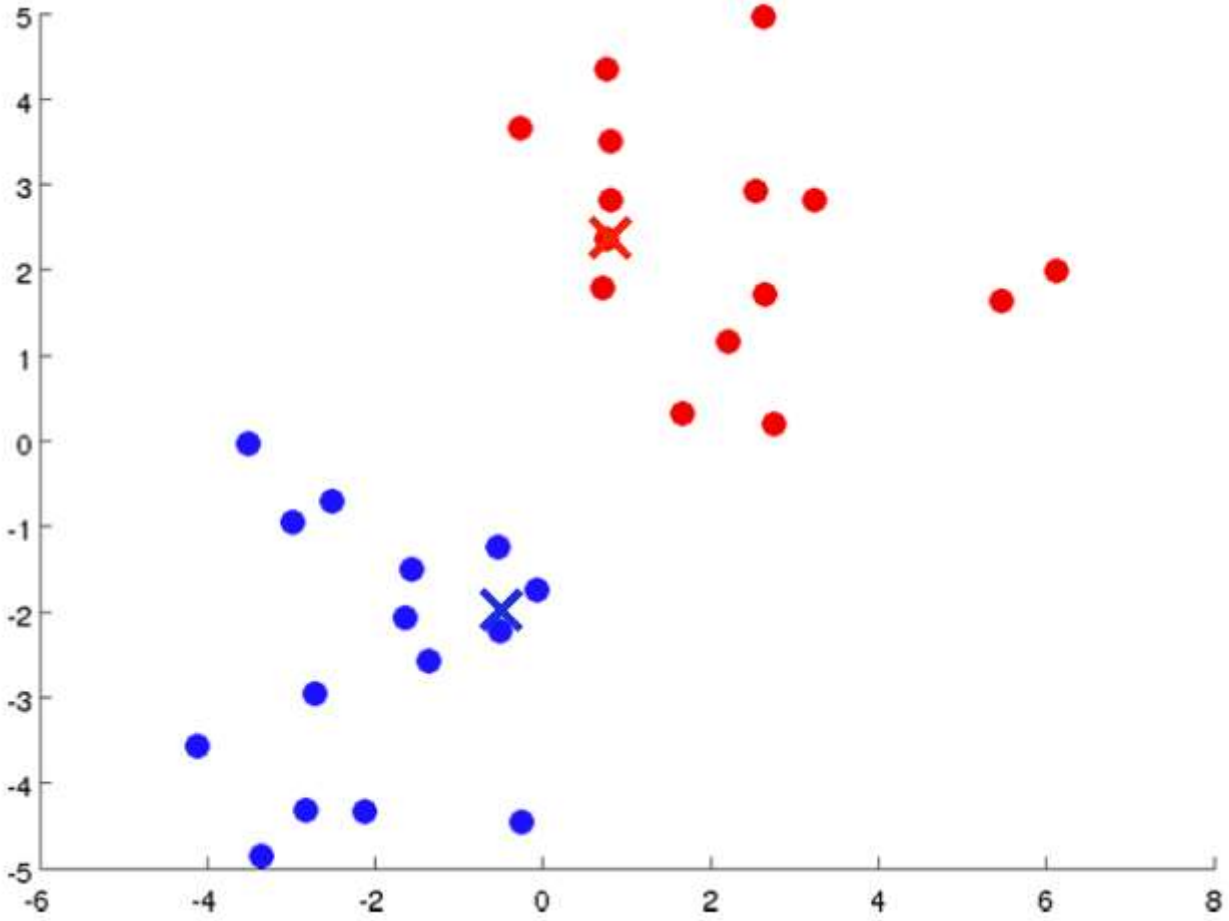
# 계산 예시 - 데이터 클러스터 할당



# 계산 예시 - 무게중심 갱신



# 계산 예시 - 데이터 클러스터 할당



# 클러스터링 분석의 평가

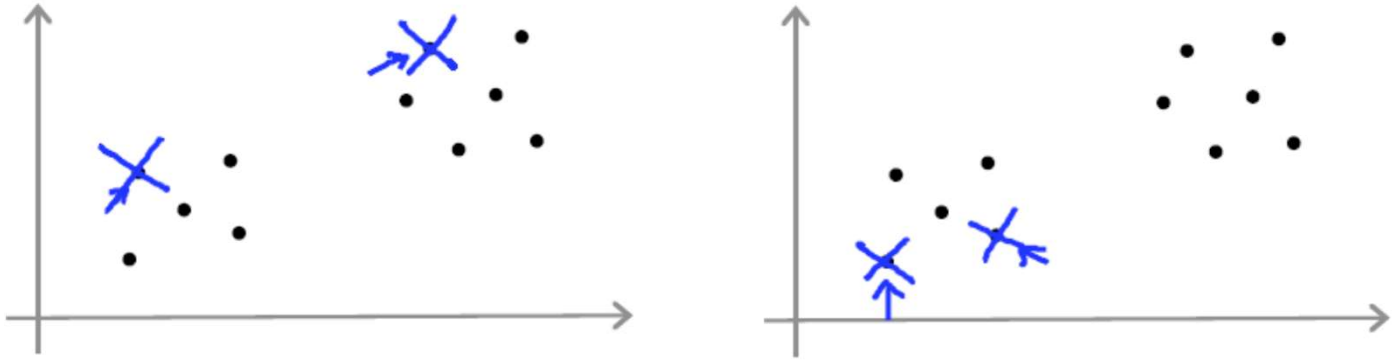
- notation
  - $\mu_k$: k번째 군집의 중점
  - $X^{(i)}$ : i번째 데이터
  - $c^{(i)}$ : i번째 데이터가 소속된 군집

- 클러스터 분석의 비용함수

$$J(c^{(1)},\ldots,c^{(m)},\mu_1,\ldots,\mu_K)=\frac{1}{m}\sum_{i=1}^{m}\left\|x^{(i)}-\mu_{c^{(i)}}\right\|^2$$

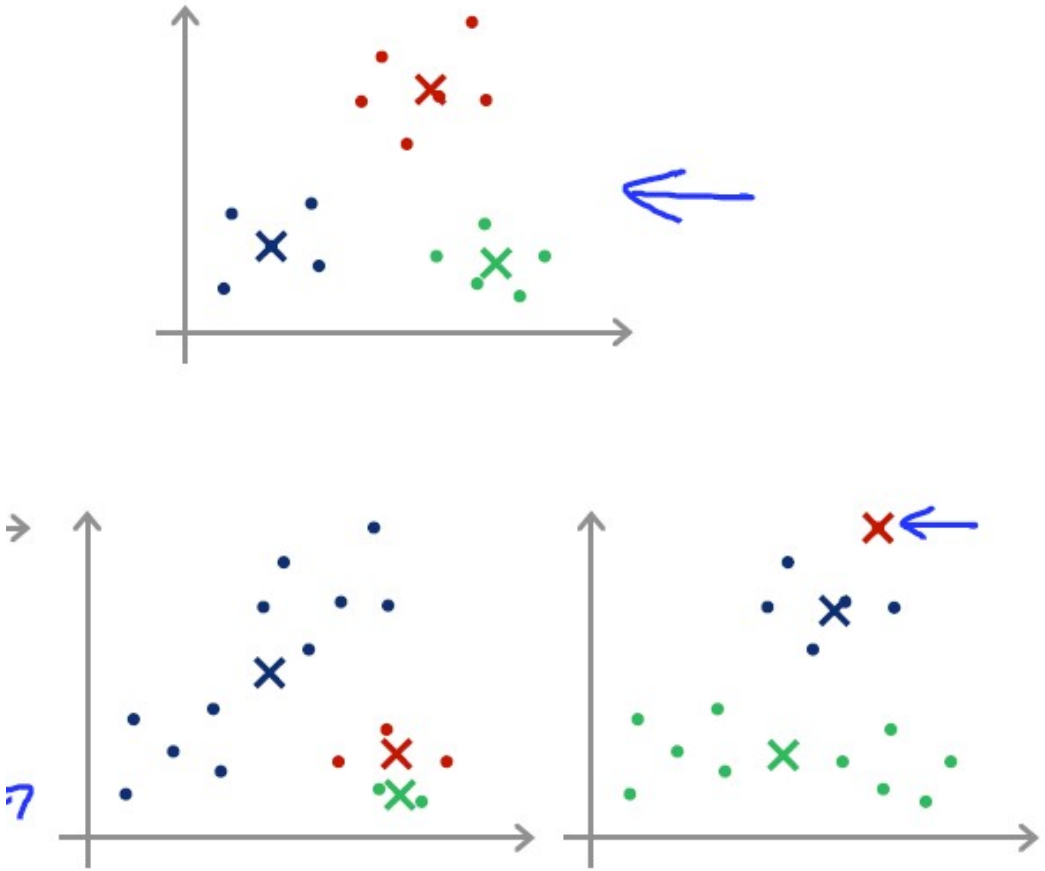
# K-means 알고리즘의 약점

- 클러스터 분석의 성능은 무작위로 생성된 첫 클러스터 중점에 크게 영향을 받는다



# K-means 알고리즘의 약점 2

- 클러스터링 알고리즘은 때때로 Local Optimal한 정답을 내놓는 경우가 있다



# K-means의 약점을 극복하기 위한 방안

- 같은 데이터에 대하여 클러스터링 기법을 여러 번 수행하여, 수행결과를 취합하여 사용 (앙상블)

**Random initialization**

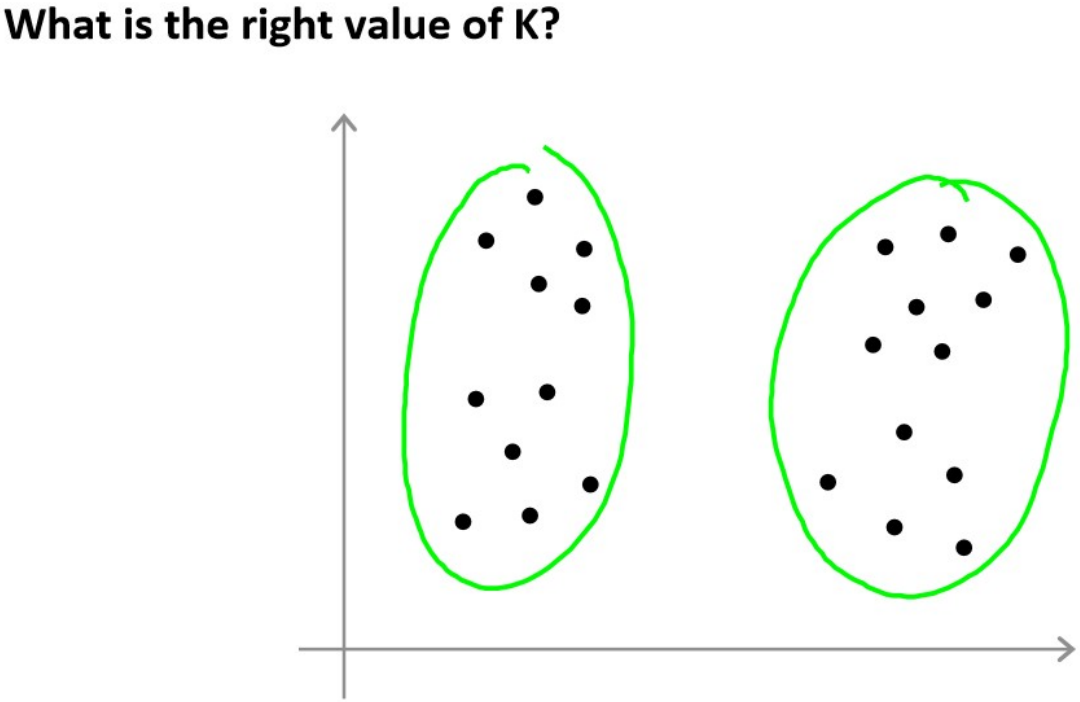
For i = 1 to 100 {

Randomly initialize K-means.
Run K-means. Get $c^{(1)}, \ldots, c^{(m)}, \mu_1, \ldots, \mu_K$.
Compute cost function (distortion)
$J(c^{(1)}, \ldots, c^{(m)}, \mu_1, \ldots, \mu_K)$
}

Pick clustering that gave lowest cost $J(c^{(1)}, \ldots, c^{(m)}, \mu_1, \ldots, \mu_K)$

# K-means 하이퍼 패러미터 조정

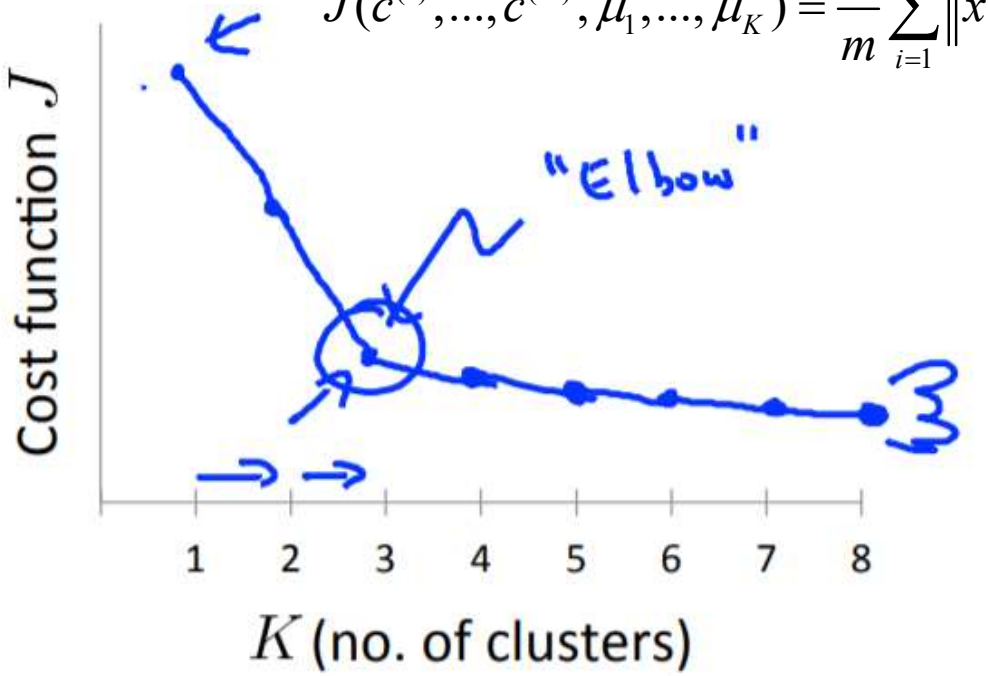
- 하이퍼 패러미터: 클러스터의 개수 K
- K의 개수에 따라 클러스터의 결과가 달라짐
- K값에 대한 실험적인 탐색 필요



# 하이퍼 패러미터 선택방법

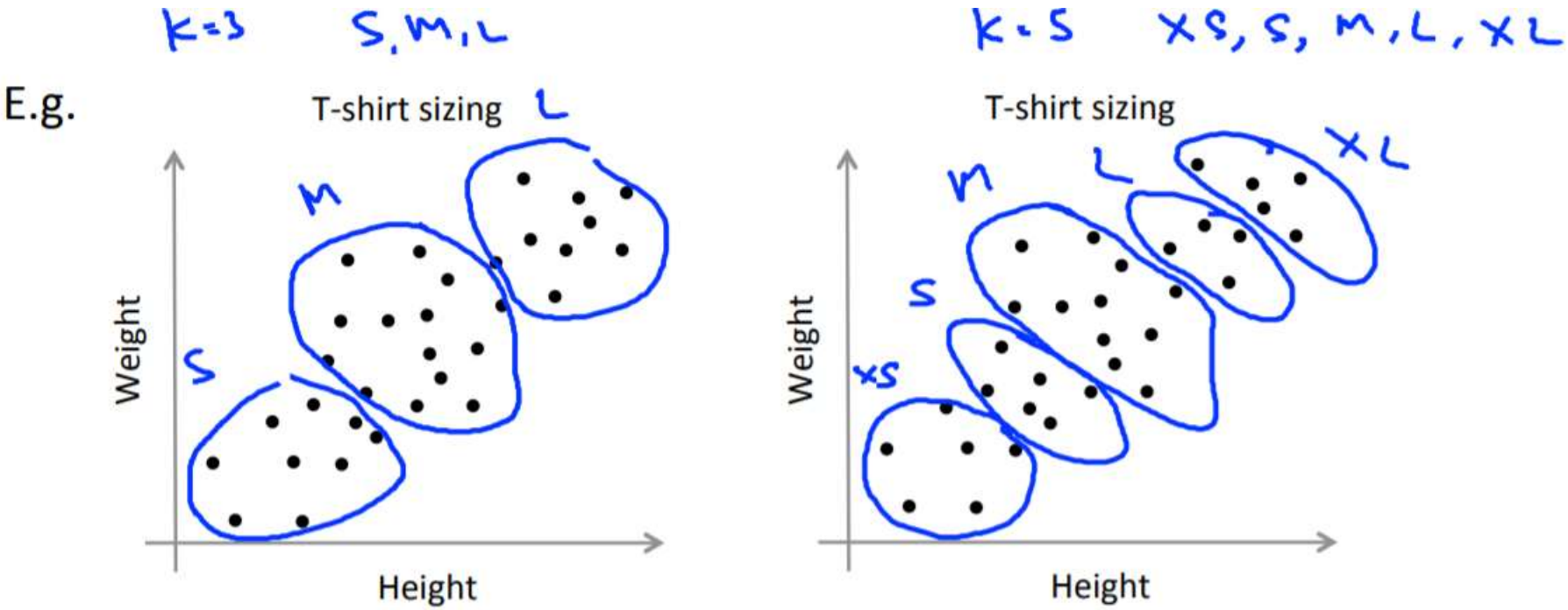
- 클러스터의 개수 K의 값을 변경시켜가며, 비용함수 J의 값 추적
- 비용함수의 감소가 급격히 완만해지는 지점 (elbow point)을 최적의 클러스터로 선택

$$J(c^{(1)},\ldots,c^{(m)},\mu_1,\ldots,\mu_K)=\frac{1}{m}\sum_{i=1}^{m}\left\|x^{(i)}-\mu_{c^{(i)}}\right\|^2$$



# 하이퍼 패러미터 선택방법

- 확인하고자 하는 클러스터의 개수가 이미 정해져있는경우
- 도메인 지식에 의존



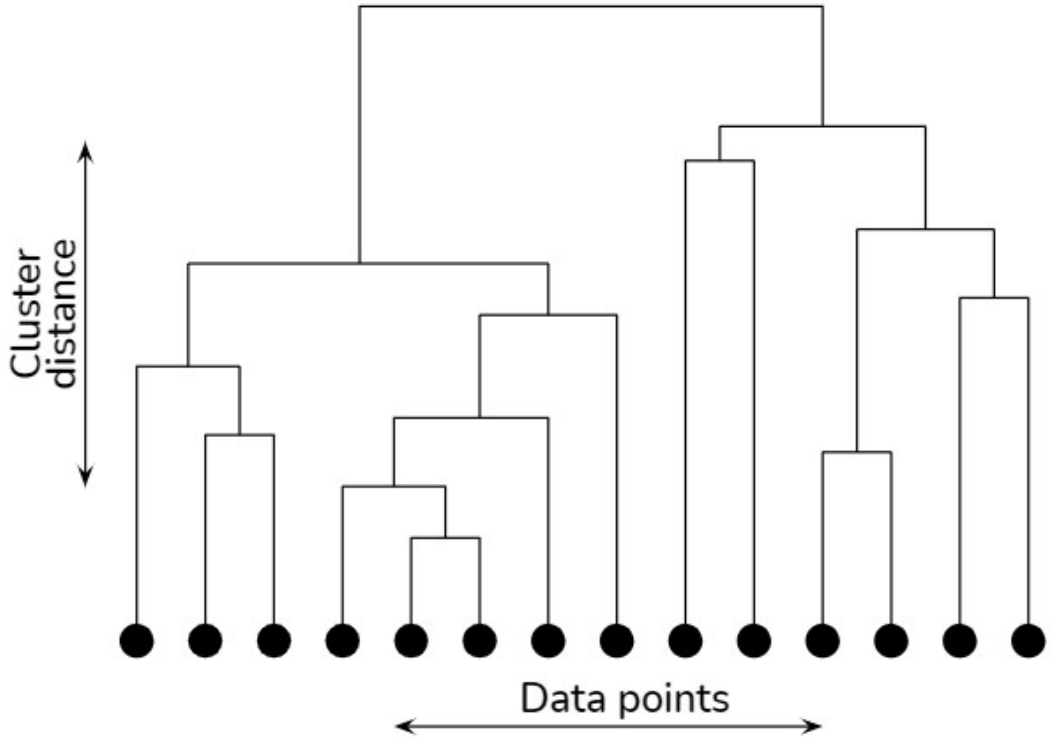
# K-means 클러스터링 장단점

- 장점
  - 구현이 간단
  - 계산량이 적음

- 단점
  - 하이퍼 패러미터 튜닝 필요 (K)
  - 초기 랜덤하게 생성된 중점에 영향을 받음
  - 데이터의 스케일에 민감 → 거리 계산전 표준화 등의 전처리 필요

# 계층적 클러스터링 (Hierarchical Clustering)

- 데이터간의 Dendrogram(계통도)를 그리고, 이를 바탕으로 유사한 데이터를 군집화 하는 기법
  - 가로축: 개별 데이터 객체
  - 세로축: 클러스터 사이의 거리



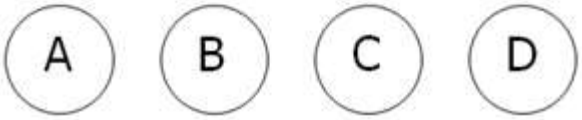
# 계층적 클러스터링 알고리즘

- (시작) 개별 데이터 객체 $X^{(1)}, \ldots, X^{(m)}$를 각각의 그룹(클러스터로 지정)
- (다음을 반복)
  - 가장 가까운 두개의 클러스터를 찾아 합침
  - 만약 모든 데이터가 하나의 클러스터로 합쳐지면 종료

# 계층적 클러스터링 알고리즘

- (시작) 개별 데이터 객체 $X^{(1)}, \ldots, X^{(m)}$를 각각의 그룹(클러스터로 지정)
- (다음을 반복)
  - 가장 가까운 두개의 클러스터를 찾아 합침
  - 만약 모든 데이터가 하나의 클러스터로 합쳐지면 종료

Initial Data Items



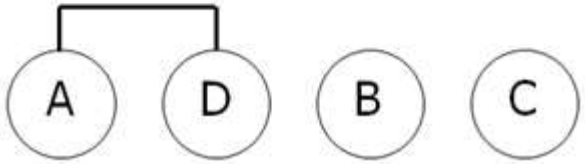

Distance Matrix

| Dist | A | B | C | D |
|---|---|---|---|---|
| A | | 20 | 7 | 2 |
| B | | | 10 | 25 |
| C | | | | 3 |
| D | | | | |

# 계층적 클러스터링 알고리즘

- (시작) 개별 데이터 객체 $X^{(1)}, \ldots, X^{(m)}$를 각각의 그룹(클러스터로 지정)
- (다음을 반복)
  - 가장 가까운 두개의 클러스터를 찾아 합침
  - 만약 모든 데이터가 하나의 클러스터로 합쳐지면 종료

Current Clusters



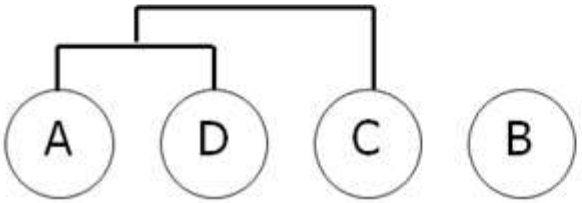

Distance Matrix

| Dist | AD | B | C | |
|---|---|---|---|---|
| AD | | 20 | 3 | |
| B | | | 10 | |
| C | | | | |
| | | | | |

# 계층적 클러스터링 알고리즘

- (시작) 개별 데이터 객체 $X^{(1)}, \ldots, X^{(m)}$를 각각의 그룹(클러스터로 지정)
- (다음을 반복)
  - 가장 가까운 두개의 클러스터를 찾아 합침
  - 만약 모든 데이터가 하나의 클러스터로 합쳐지면 종료

Current Clusters

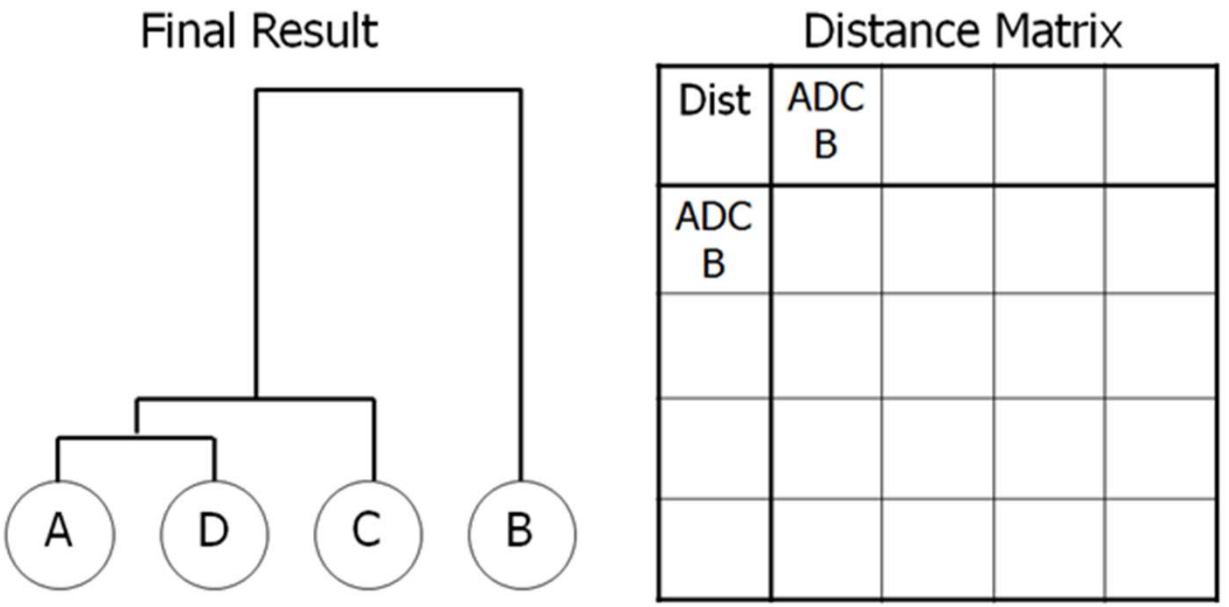



Distance Matrix

| Dist | ADC | B | | |
|---|---|---|---|---|
| ADC | | 10 | | |
| B | | | | |
| | | | | |
| | | | | |

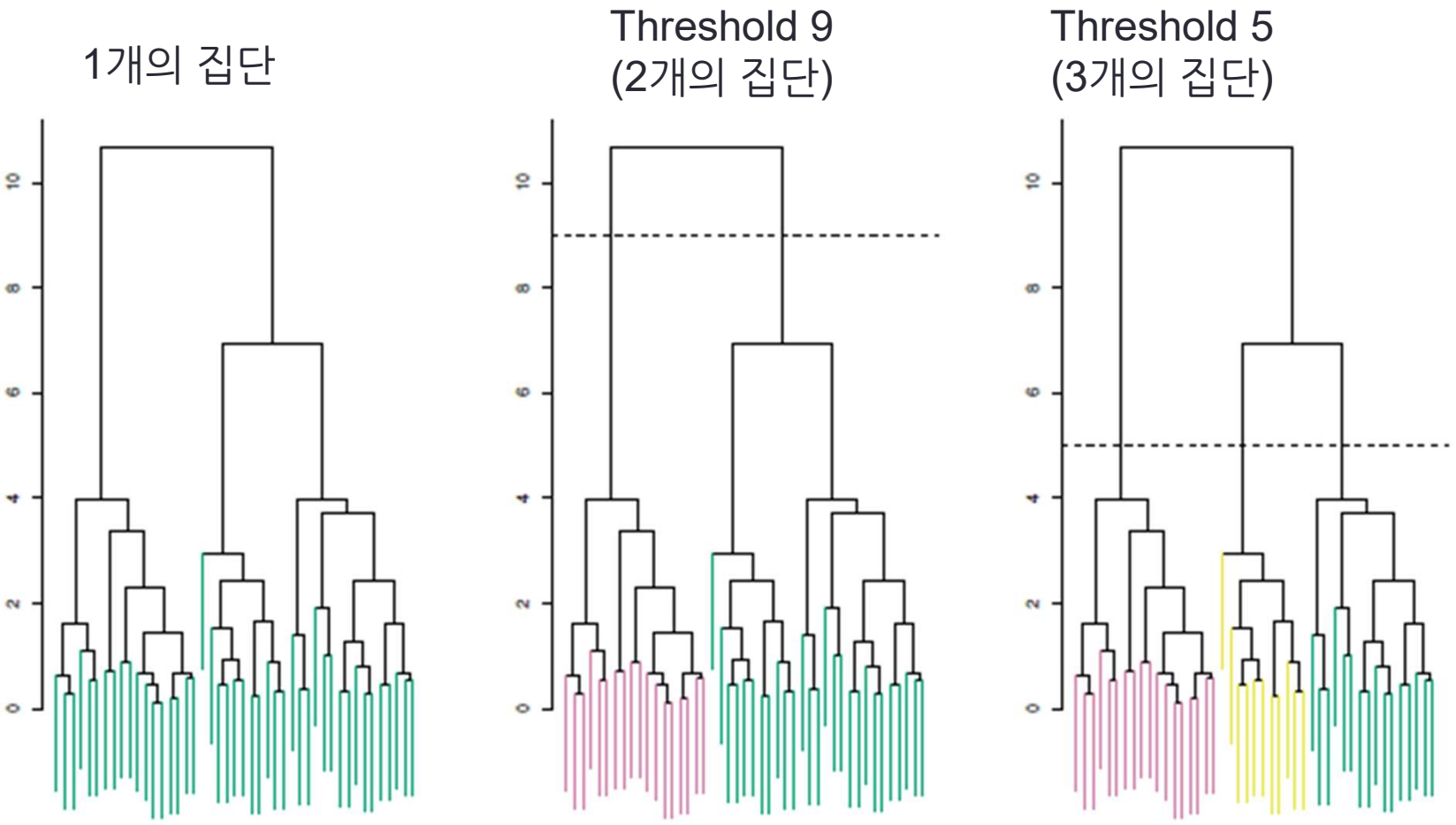
# 계층적 클러스터링 알고리즘

- (시작) 개별 데이터 객체 $X^{(1)}, \ldots, X^{(m)}$를 각각의 그룹(클러스터로 지정)
- (다음을 반복)
  - 가장 가까운 두개의 클러스터를 찾아 합침
  - 만약 모든 데이터가 하나의 클러스터로 합쳐지면 종료



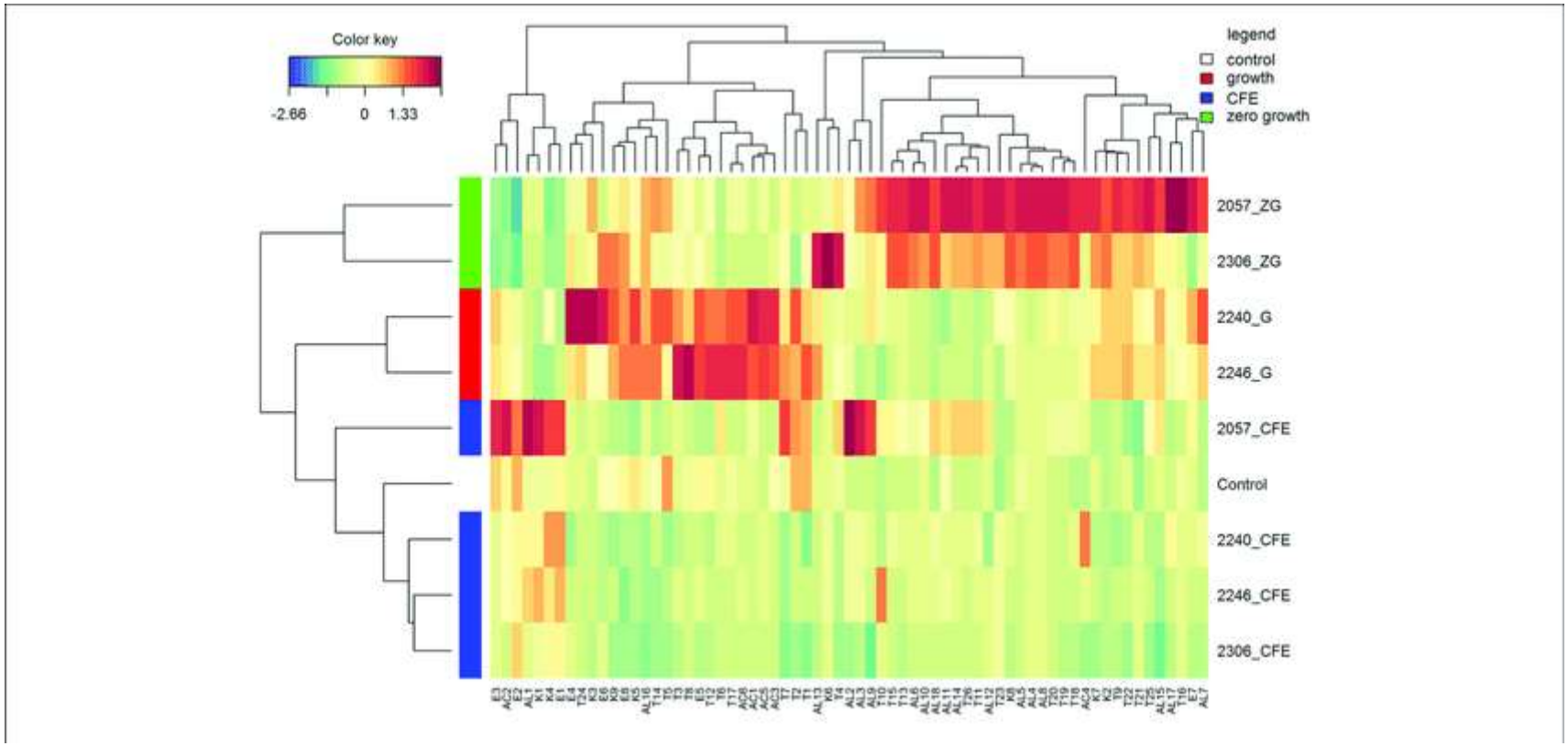
# 계통도에서 클러스터 탐지

- 계통도 완성후, 클러스터 거리에 따라 유동적으로 군집 파악 가능



# 계층적 클러스터링

- Heatmap등의 시각화 도구와 결합 가능
- 각 군집의 특성을 시각화 하여 파악할 수 있음



# 계층적 클러스터링 (장단점)

- 장점
  - 하이퍼 패러미터 필요 없음
  - 데이터들간의 관계에 대한 시각화가 가능

- 단점
  - K-means에 비해 계산이 복잡 (큰 규모의 데이터에 적합하지 않음)
    - 왜그럴까?
  - 데이터의 스케일에 민감 → 거리 계산전 표준화 등의 전처리 필요
  - 데이터의 수가 많아지면, 적절한 클러스터 기준을 찾는데 어려움 존재

# 클러스터링 이슈1 - 차원의 저주

- KNN 뿐만 아니라 대부분의 머신러닝 모형에 적용되는 현상

- 데이터의 차원이 지나치게 많은 경우 모형의 성능에 악영향을 미침
  - 학습속도의 저하
  - Generalization 실패 → 오버피팅

- 차원의 저주 방지 전략
  - 특성 선택: 중요 변수만을 선택
  - 비지도 학습 기반 차원축소 기법 사용
    - PCA
    - Manifolds Methods

# 클러스터링 이슈2 - 변수의 Scaling

- 변수들의 단위(metric)가 같고, 서로 비교 가능할 때
  - X1: 위도, X2: 경도
  - 원 데이터를 그대로 사용하는것이 바람직
  - Scaling (X)

- 변수들 간의 단위가 다르고, 서로 비교할 수 없을 때
  - X1: 키, X2: 몸무게
  - 데이터에서 단위를 없애고 비교 가능하도록 Scaling 하는것이 필요
  - Scaling 방법
    - Z-normalization: 모든 변수들이 평균 0, 표준편차 1이 되도록 변환

$$\frac{x_i - \mu_i}{\sigma_i}$$

    - Min-max Scaling: 모든 변수들이 최대값 1, 최소값 0 되도록 변환

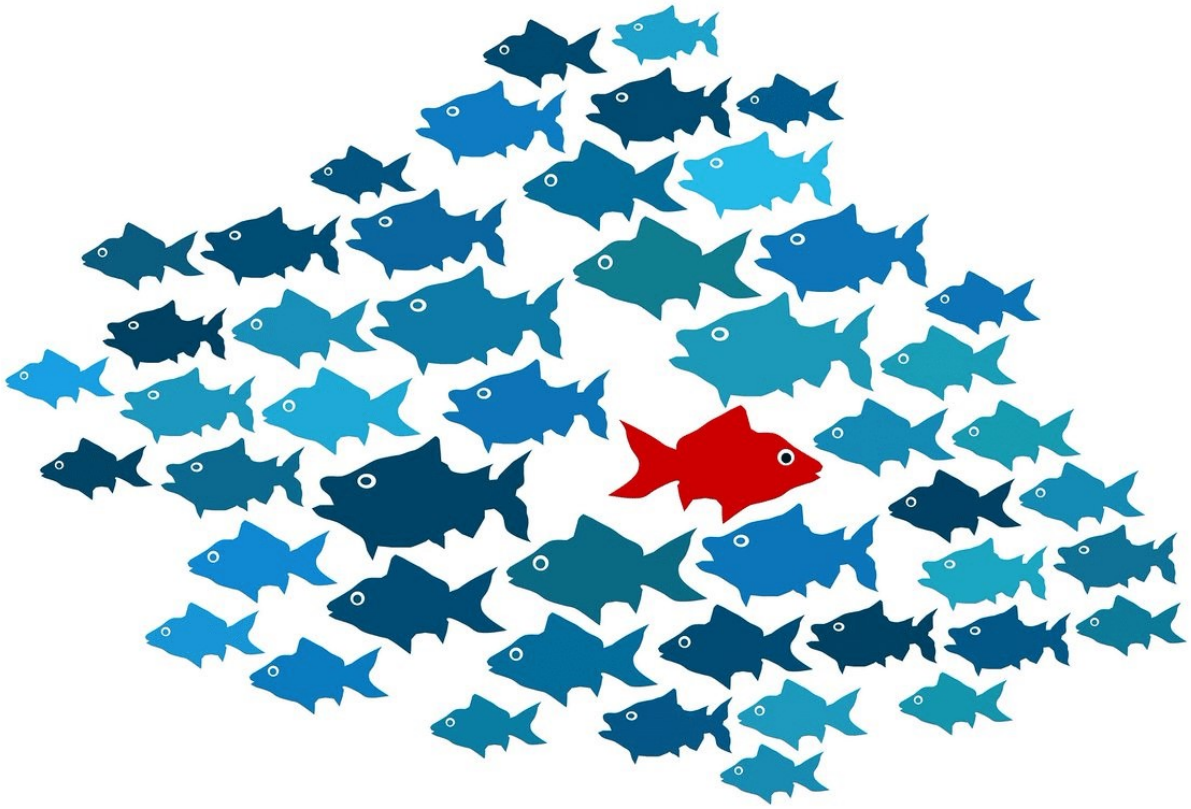
# PCA + 클러스터링

# 이상치 탐지 (Anomaly Detection)

시스템경영공학부
이지환 교수

# 이상치 탐지문제

- 주어진 데이터로부터 대세 경향이나 분포를 따르지 않는 이상객체를 탐지하는 문제

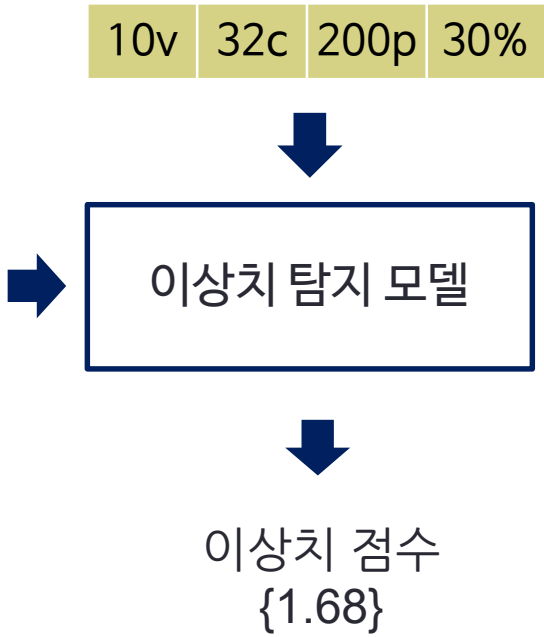


# 비지도학습 (Unsupervised Learning)

- 학습하는 데이터내에 예측하고자 하는 값이 없는 경우
- 예시) 시간에 따라 기록된 5,000개의 공정 데이터가 존재한다. 새로운 데이터가 기존의 데이터에서 얼마나 벗어나 있는지 측정하고자 한다.

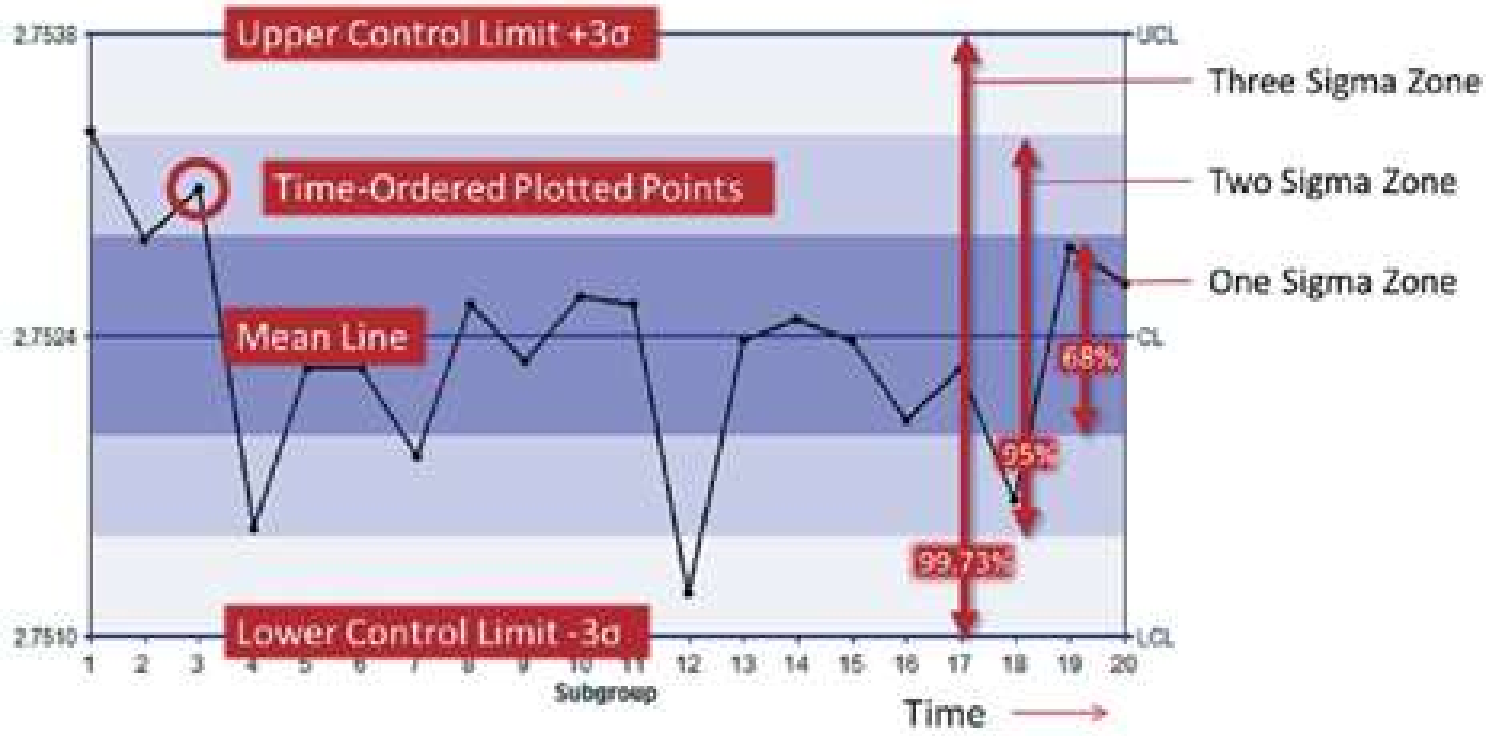
- 이상치 탐지(anomaly detection)

| | 전압 | 온도 | 압력 | 습도 | 이상치 점수 |
|---|---|---|---|---|---|
| 1 | xx | xx | xx | xx | 0.05 |
| 2 | xx | xx | xx | xx | 0.04 |
| 3 | xx | xx | xx | xx | 0.2 |
| | | | | | |
| 4999 | xx | xx | xx | xx | 0.01 |
| 5000 | xx | xx | xx | xx | 0.2 |



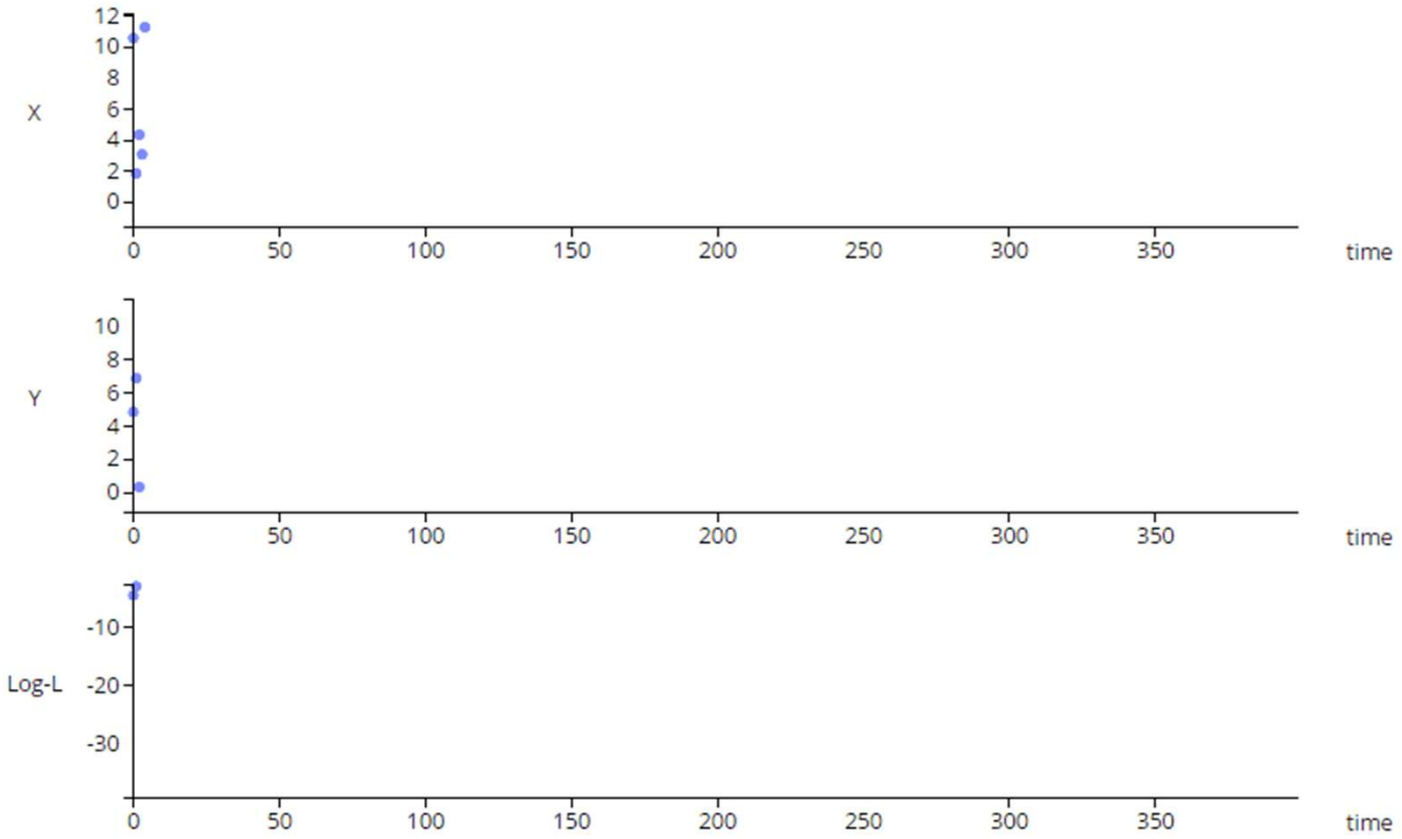
# 단일 변수의 이상치 탐지

- SPC (Statistical Process Control)
  - 공정관리도/품질관리에서 전통적으로 사용되던 기법
  - 공정에서 발생하는 단일 품질요소를 시간에 따라 수집하고, 특정 관측치가 통계적으로 유의미 (표준편차의 x배) 하게 벗어나는 경우 이상치로 판단



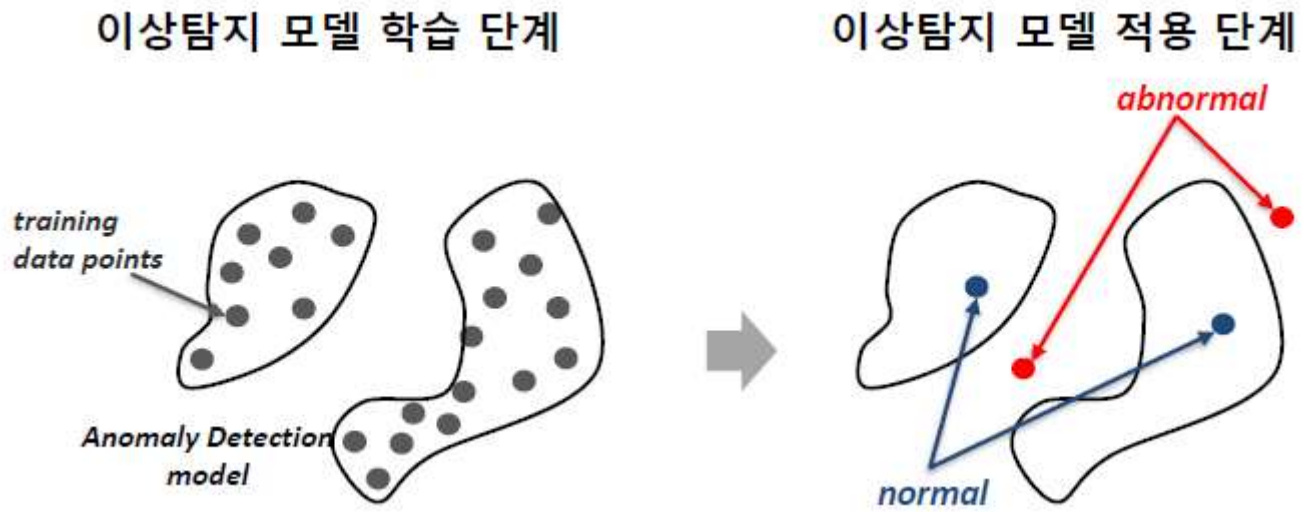
# 다중 특성

- 변수가 여러 개일 경우
  - 단일 변수에서는 이상치가 아니지만 여러 변수의 종합하여 판단하였을때 이상치인 경우 발생



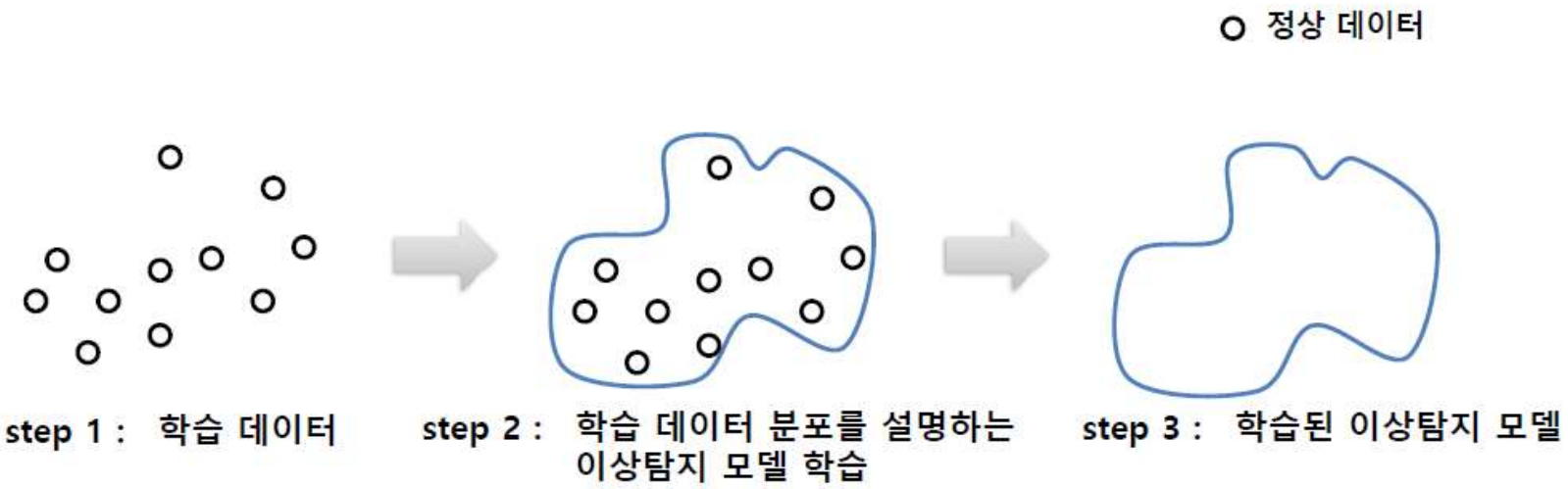
# 이상탐지 방법론 - 학습

- (가정) 주어진 데이터는 모두 정상이다.
- (학습) 정상 데이터의 분포/패턴을 설명하는 모형 학습
  - One-class classification
- (판단)
  - 학습된 모형이 설명하지 못하는 객체를 이상으로 판단
  - 이상치로 판단된 데이터에 대해 추가분석 수행



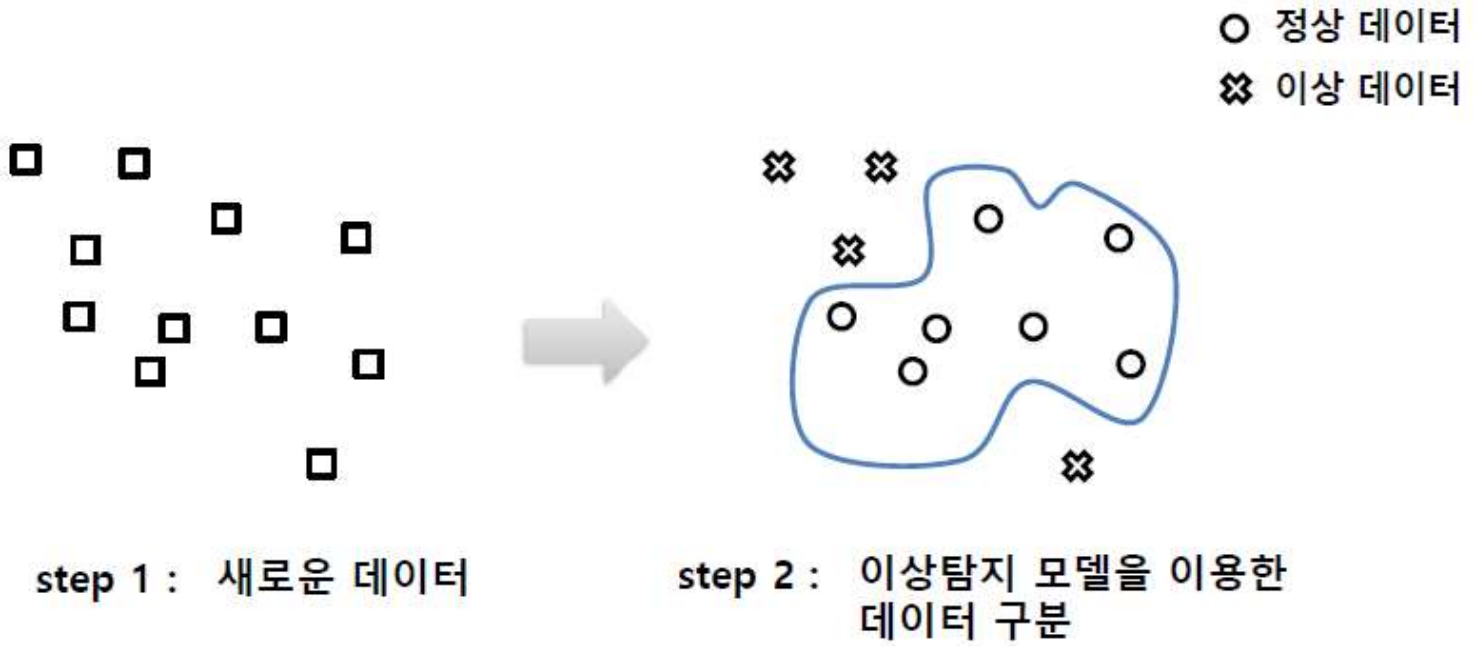
# 이상탐지 방법론 - 학습

- 정상 데이터의 분포를 설명하는 모형 학습



# 이상탐지 방법론 - 예측

- 이상탐지 모형이 설명하지 못하는 객체를 이상으로 판단

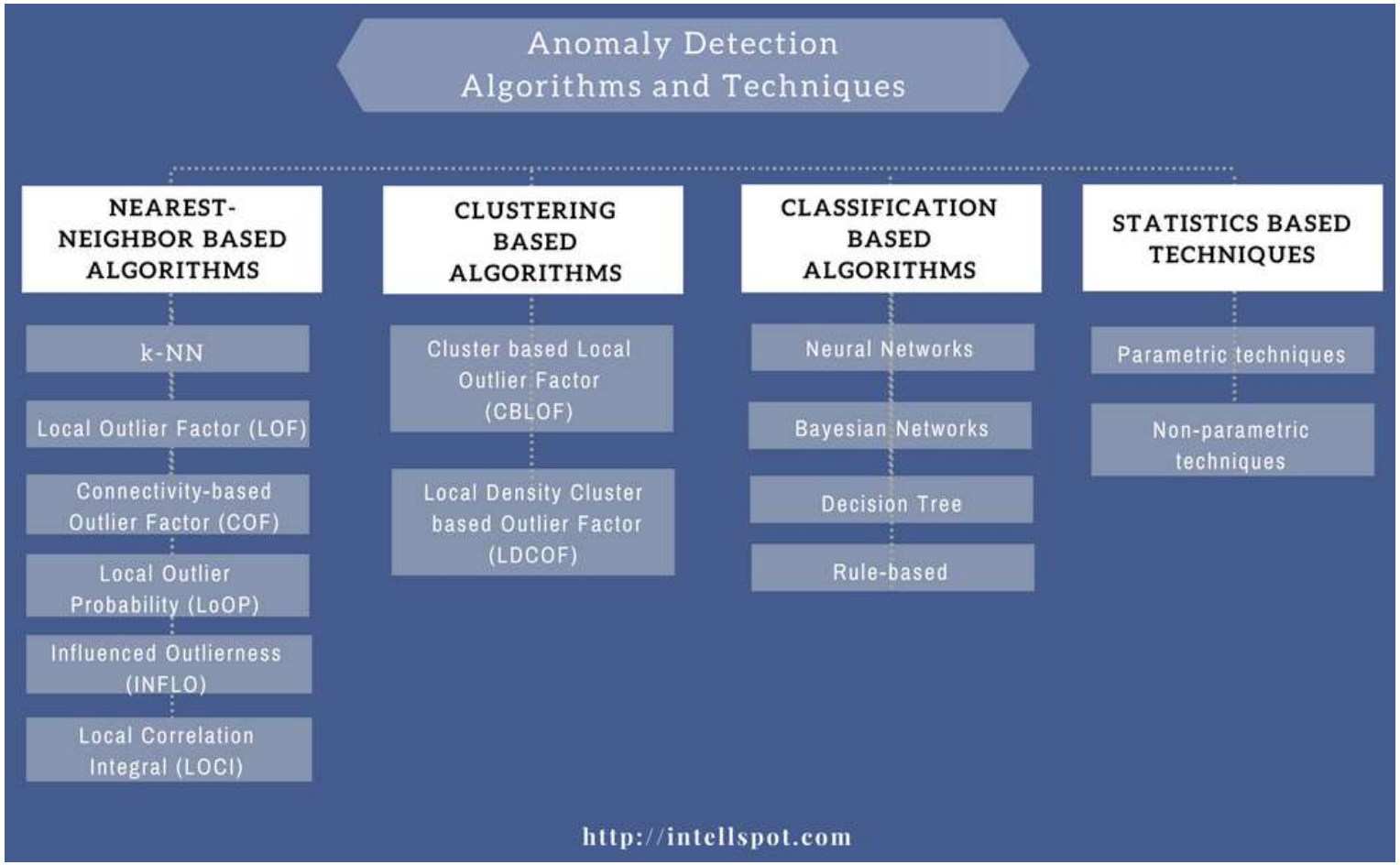


# 분류와 이상탐지문제의 구분

- 이상탐지문제가 분류 문제일경우
  - 데이터에 정상과 이상데이터가 존재
  - 해결해야할 문제
    - 두 정상/비정상 간의 불균형 문제 발생
    - 예시) 정상 99%, 비정상 1%
      - 모든 데이터에 대해 정상으로 예측만 해도 99%의 정확도 달성
      - Over sampling/Under sampling 등의 기법을 사용하여 레이블 불균형 해소필요
      - Accuracy외 다른 지표들을 종합적으로 고려 (Recall)

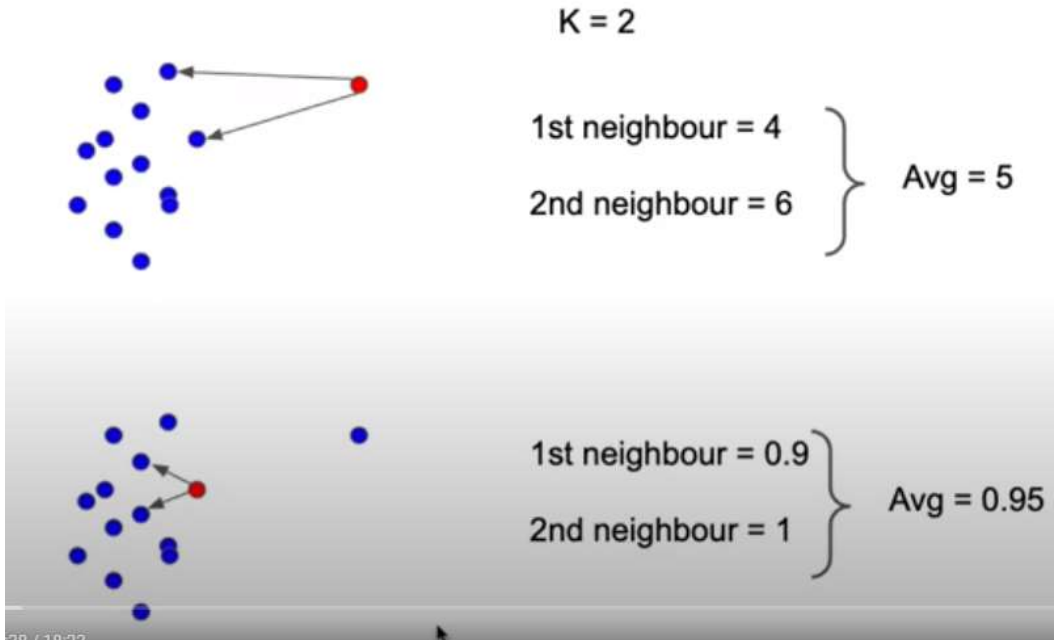
- 이상탐지문제가 비지도학습인 경우
  - 데이터에 정상데이터만 존재한다고 가정
  - 정상 데이터의 분포/패턴을 표현하는 모형을 학습

# 이상치 탐지모형의 분류


Anomaly Detection
Algorithms and Techniques
NEAREST-NEIGHBOR BASED ALGORITHMS
k-NN
Local Outlier Factor (LOF)
Connectivity-based Outlier Factor (COF)
Local Outlier Probability (LoOP)
Influenced Outlierness (INFLO)
Local Correlation Integral (LOCI)
CLUSTERING BASED ALGORITHMS
Cluster based Local Outlier Factor (CBLOF)
Local Density Cluster based Outlier Factor (LDCOF)
CLASSIFICATION BASED ALGORITHMS
Neural Networks
Bayesian Networks
Decision Tree
Rule-based
STATISTICS BASED TECHNIQUES
Parametric techniques
Non-parametric techniques
http://intellspot.com

# KNN을 이용한 이상탐지

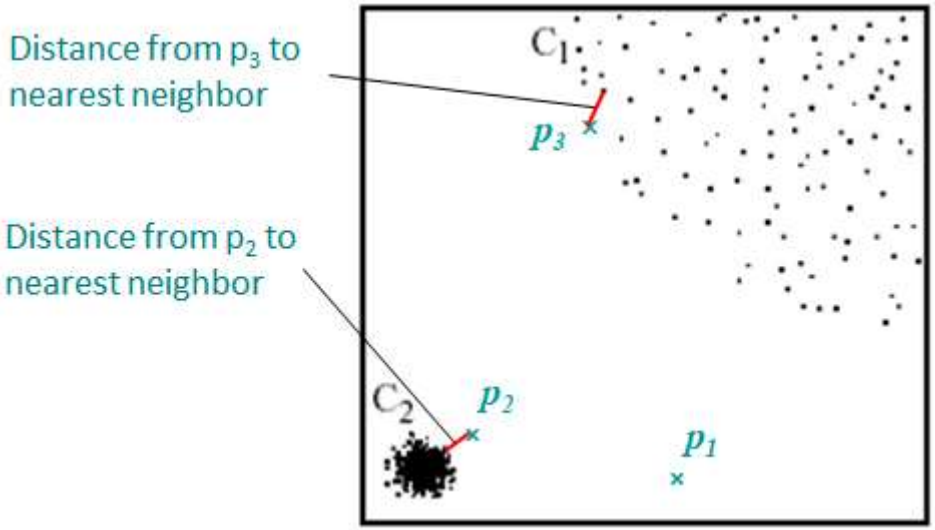
- 주변 K개의 데이터와의 평균 거리 --> 이상치의 정도
- 거리가 멀리 떨어져 있을수록 이상객체라고 판단할 수 있다.




- 단점
  - 최적의 K를 판단해야 함 (알수 없음)
  - 상대적인 데이터들의 밀도를 고려할 수 없음

# LOF (Local Outlier Factor)

- KNN의 단점
  - 데이터 주변의 상대적 밀도를 고려할 수 없음
    - P2 → C2 주변의 데이터들의 비해 멀리 떨어져 있음
    - P3 → C1 주변의 데이터들과 비슷하게 떨어져 있음
    - 하지만 {P2, P3} 모두 유사한 KNN-distance를 가질 것임



# LOF (Local Outlier Factor) - K-distance

### 2_1. the distance between objects p and q

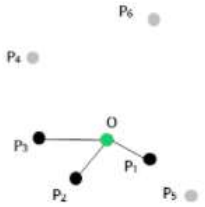
$d(p, q)$ : 관측치 p와 q의 거리

### 2_2. $k$-distance of an object p

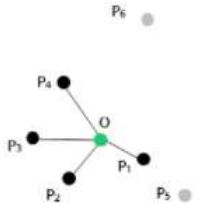
$k\text{-}distance(p)$ : 관측치 p와 가장 가까운 데이터 k개에 대한 거리의 평균

### 2_3. $k$-distance neighborhood of an object p

$N_k(p)$ : 관측치 p의 $k\text{-}distance(p)$보다 가까운 이웃의 집합
(= $k\text{-}distance(p)$를 계산할 때 포함된 이웃의 집합)




[예시 1]




[예시 2]

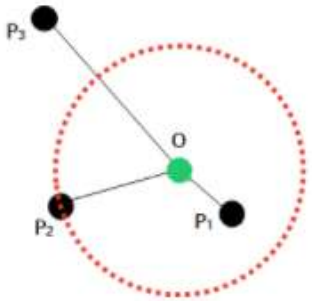
| | $d(O, P_1)$ | $d(O, P_2)$ | $d(O, P_3)$ | $d(O, P_4)$ | $d(O, P_5)$ | $d(O, P_6)$ |
|---|---|---|---|---|---|---|
| 예시 1 | 1 | 2 | 3 | 5 | 6 | 7 |
| 예시 2 | 1 | 2 | 3 | 3 | 6 | 7 |

| 3-distance(O) | $N_3(O)$ 개수 |
|---|---|
| 2 | 3 |
| 2.25 | 4 |

# LOF (Local Outlier Factor) - reachability distance

## 2_4. reachability distance of an object p w.r.t. object o

$$reach\text{-}dist_k(p,o) = max\{k\text{-}distance(o), d(o,p)\}$$




| | $P_1$ | $P_2$ | $P_3$ |
|---|---|---|---|
| $d(O, P_i)$ | 1 | 2 | 3 |
| reach-dist3($P_i$, O) | 2 | 2 | 3 |

3-distance(O) = 2

위의 그림으로 보면 직관적일 수 있습니다.

- 관측치 p가 o에서 멀다면 : 관측치 p와 o의 실제 거리
- 관측치 p가 o에서 가깝다면 : 관측치 o의 $k$-distance

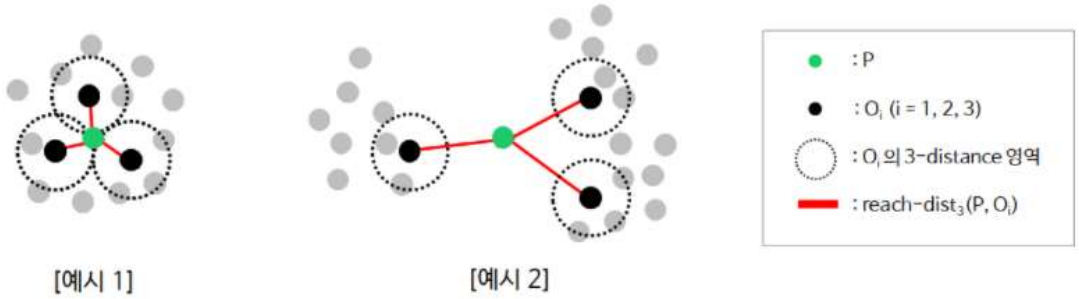
# LOF (Local Outlier Factor) - K-distance

### 2_5. local reachability density(lrd) of an object p

$$lrd_k(p) = \frac{|N_k(p)|}{\sum_{o \in N_k(p)} reach\text{-}dist_k(p, o)}$$

데이터 주변의 이웃의 평균 밀도

수식을 해석하자면, 관측치 p에 대한 k-이웃의 $reach\text{-}dist_k(p, o)$ 평균의 역수입니다. 즉, 관측치 p주변에 이웃이 얼마나 밀도있게 있는가를 대변할 수 있습니다. k=3 일때의 예시를 통해 쉽게 접근해보도록 하겠습니다.

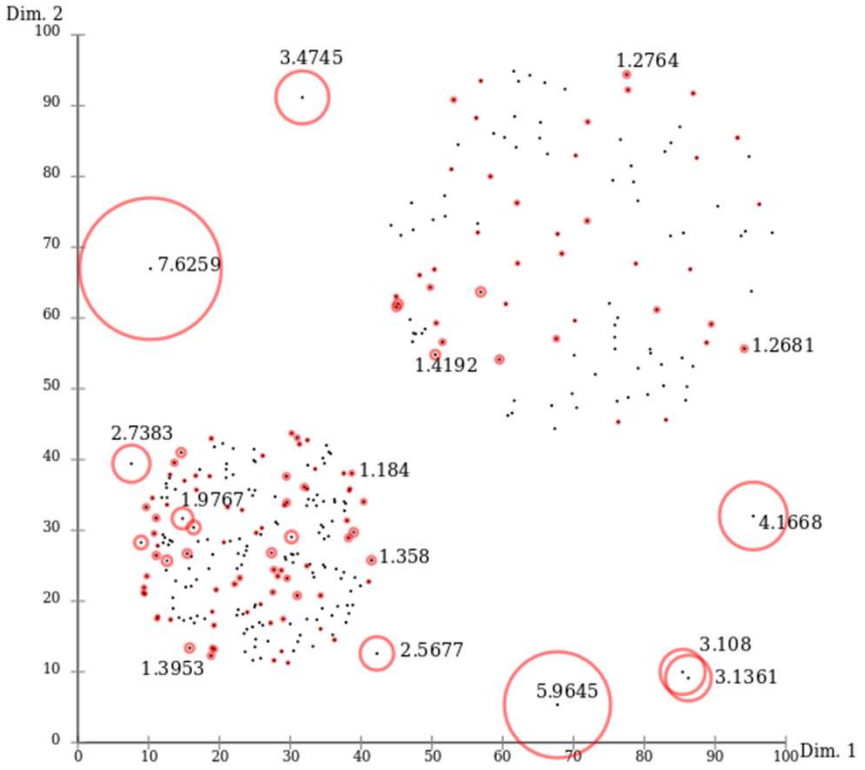



[예시 1] [예시 2]

- [예시1] : 관측치 p의 3-이웃이 가깝게 있고, 이웃 $O_i(i = 1, 2, 3)$의 $reach\text{-}dist_3(p, o_i)$도 작기 때문에 $lrd_3(p)$는 큰 값이 산출
- [예시2] : 관측치 p의 3-이웃이 멀게 있고, 이웃 $O_i(i = 1, 2, 3)$의 $reach\text{-}dist_3(p, o_i)$이 크기 때문에 $lrd_3(p)$는 작은 값이 산출

# LOF (Local Outlier Factor) - K-distance

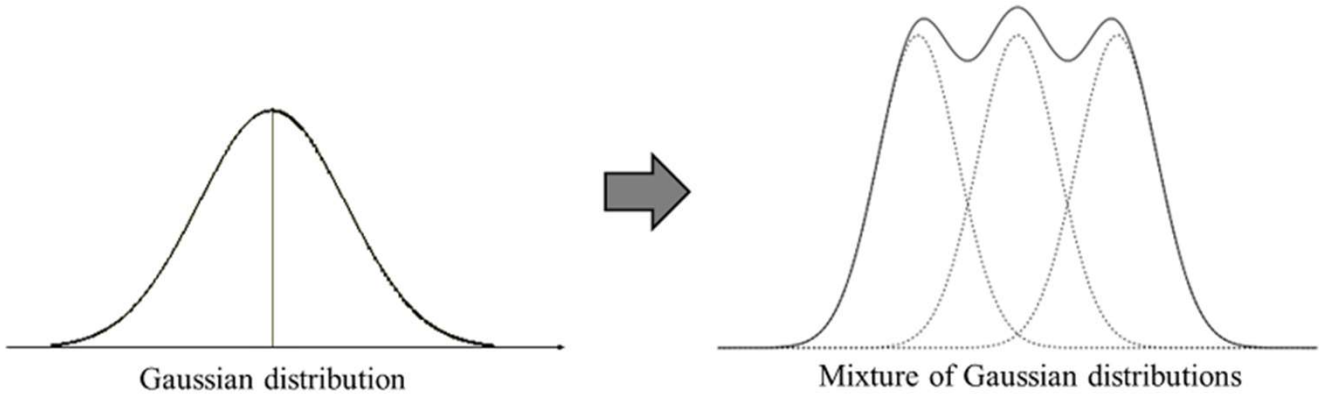
$$LOF_k(p) = \frac{\sum_{o \in N_k(p)} \frac{lrd_k(o)}{lrd_k(P)}}{|N_k(P)|} = \frac{\frac{1}{lrd_k(P)} \sum_{o \in N_k(p)} lrd_k(o)}{|N_k(P)|}$$




관측치 p의 밀도 대비 주변 k개 이웃 o들의 상대적 밀도 (lrd) 계산
→ 큰값 : p가 이상치
→ 작은값: p는 정상

# Gaussian Mixture Model

- Gaussain Mixture Model
  - 정규 분포 여러 개를 결합하여 데이터의 분포를 설명하는 기법



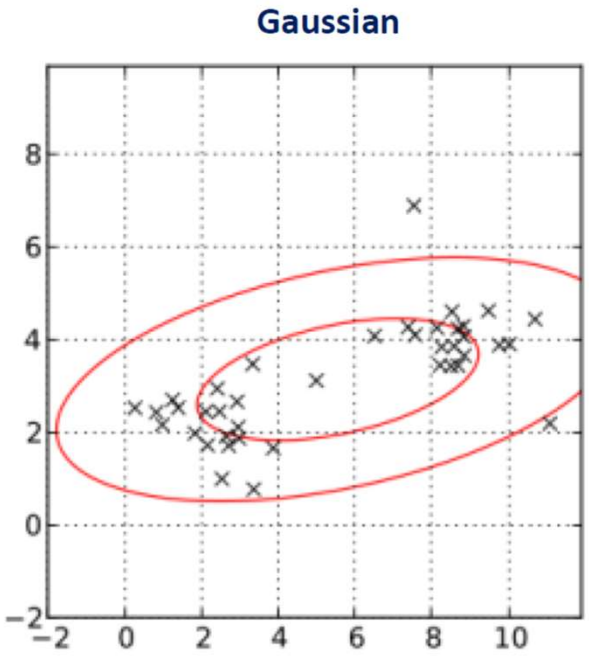

- x가 발생할 확률

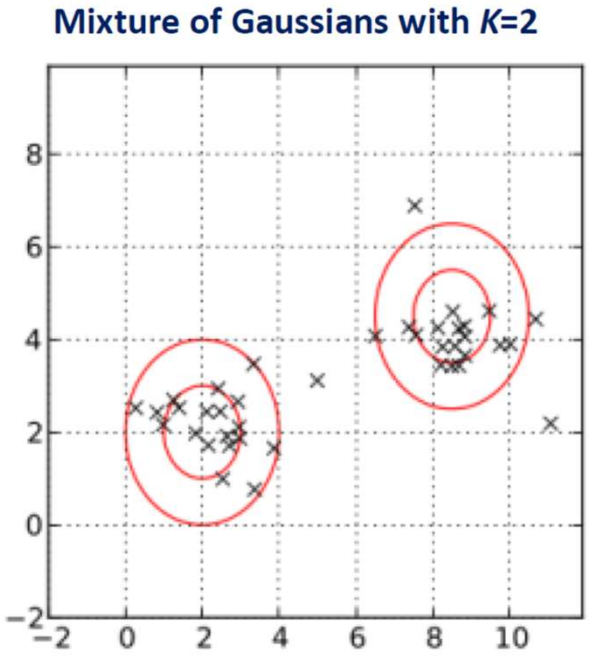
$$p(x) = \sum_{k=1}^{K} \pi_k N(x|\mu_k, \Sigma_k) \qquad (1)$$

K개 정규분포의 합

# GMM을 이용한 클러스터링

- 하이퍼 패러미터 K
  - 정규분포의 개수 → 클러스터의 개수

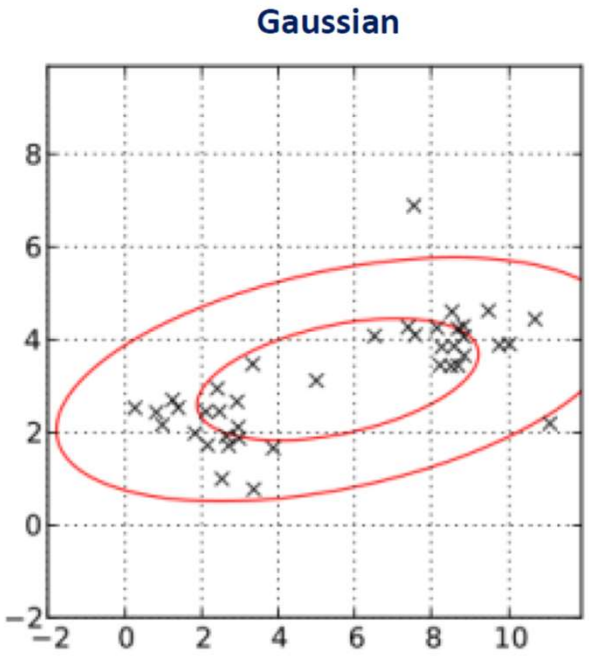


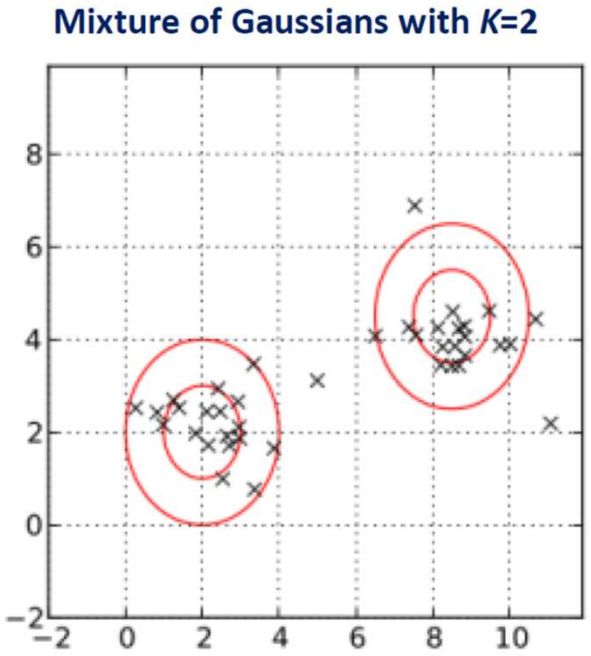




  - 각 정규분포의 평균과 분산을 학습
  - 각 데이터가 정규분포에 포함될 확률 계산 가능

# GMM을 이용한 이상치 탐지

- 하이퍼 패러미터 K
  - 정규분포의 개수 → 클러스터의 개수



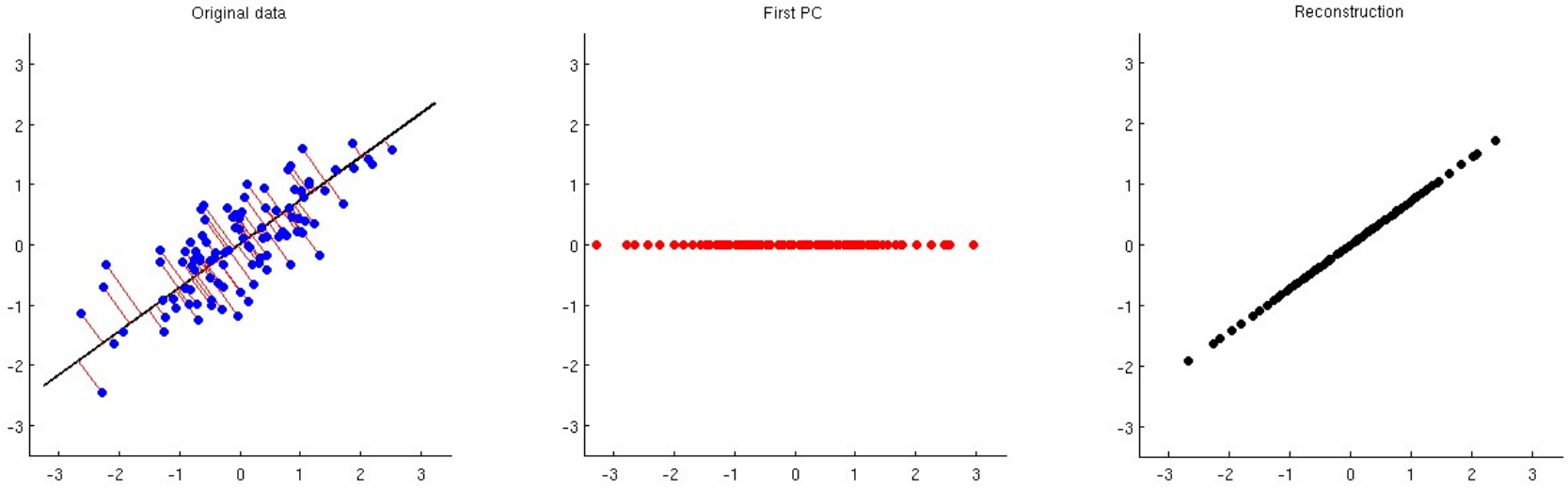




  - 각 정규분포의 평균과 분산을 학습
  - 각 데이터가 정규분포에 포함될 확률 계산 가능
  - 이상치: 모든 정규분포로부터 생성될 확률이 적은 데이터

# PCA를 이용한 이상치 탐지

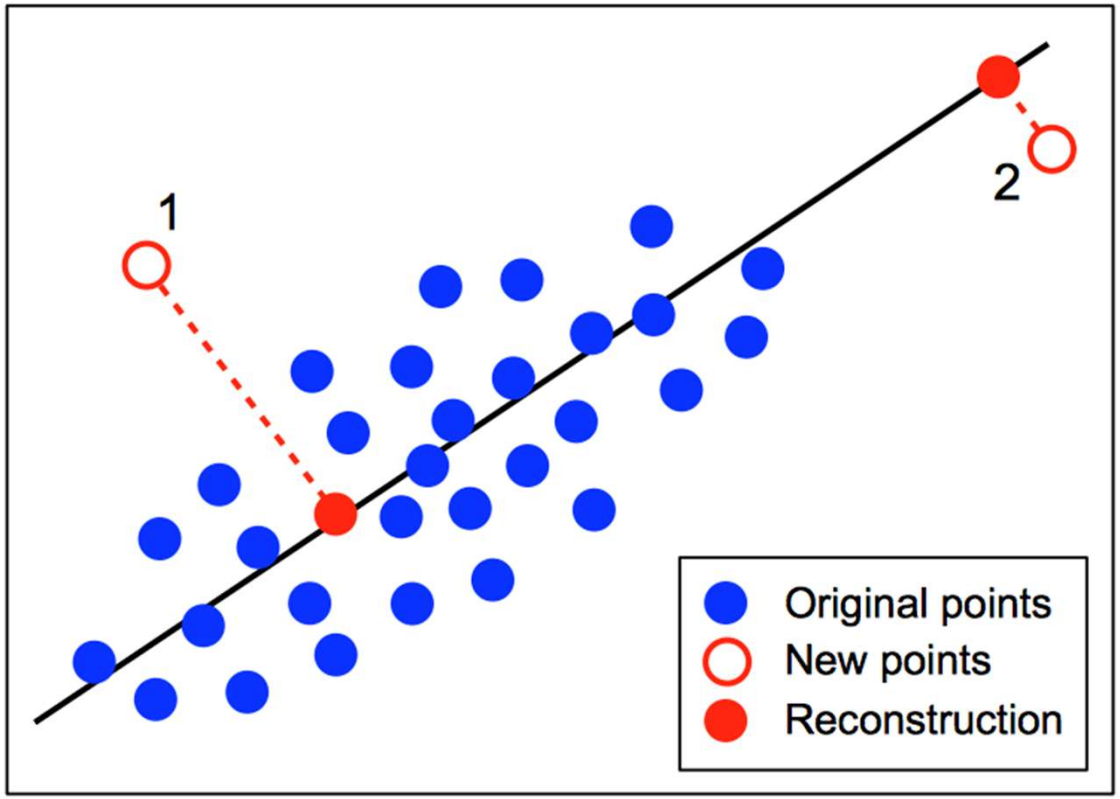
- z: 차원이 감소된 원데이터
- 주성분 벡터를 활용하여 원래 차원의 데이터로 복귀 가능
- 하지만 주성분의 차원이 적다면, 원래 데이터로 완전한 복귀는 불가능

$$\text{PCA reconstruction} = \text{PC scores} \cdot \text{Eigenvectors}^{\top} + \text{Mean}$$



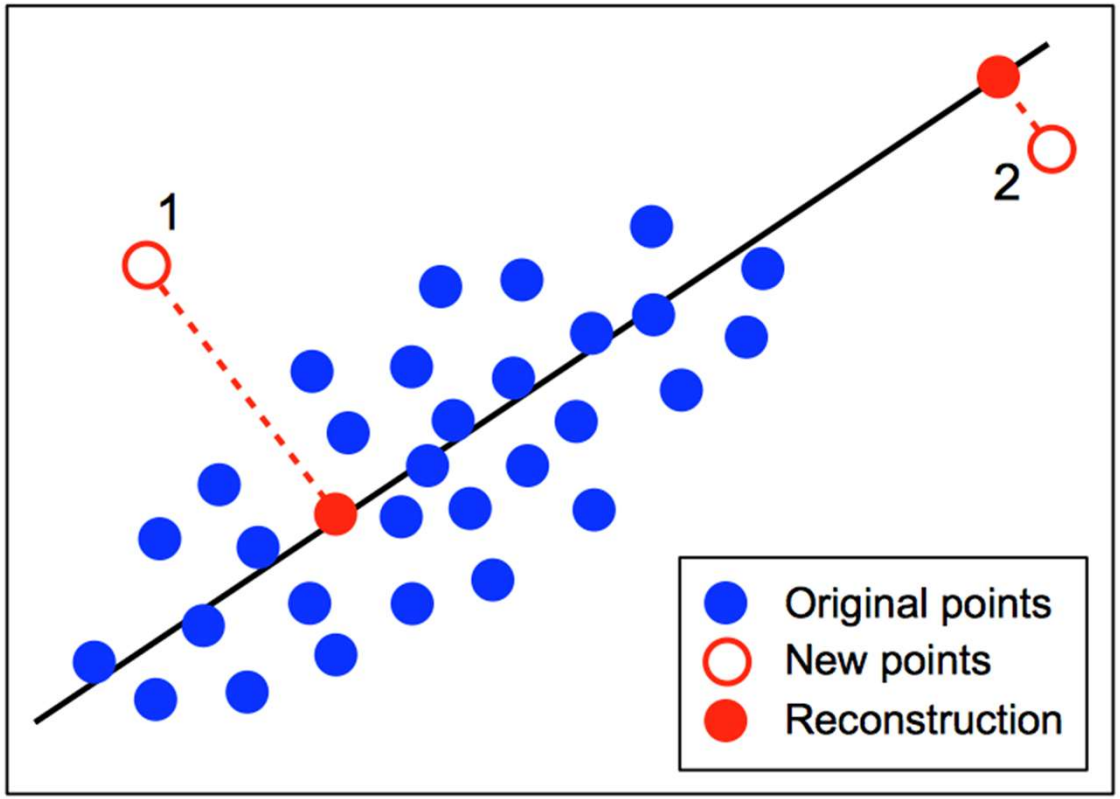
# PCA를 이용한 이상치 탐지

- 주성분이 데이터의 패턴을 잘 설명한다고 했을때,
- 실제 데이터와 PCA 사용후 복구 했을때 차이가 큰 데이터는 정상 분포를 따르지 않는 데이터라고 생각할 수 있음



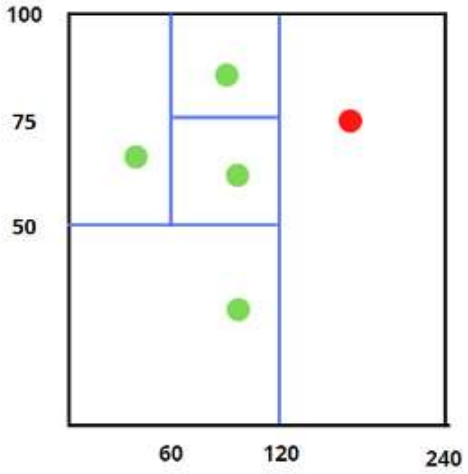
# PCA를 이용한 이상치 탐지

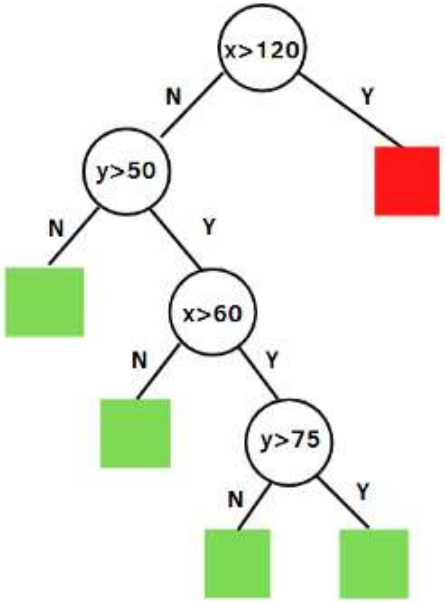
- 복구후 데이터와 원 데이터 사이의 거리 = 이상치 점수로 사용 가능



# Isolation Forest

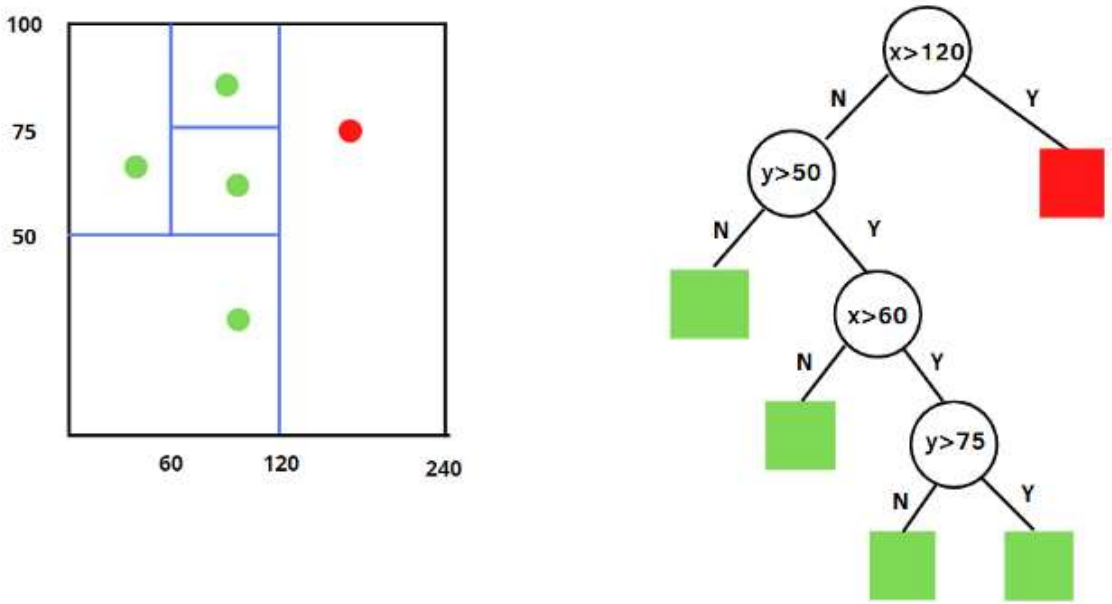
- Isolation Tree 알고리즘
  - 아래 과정을 반복
  - 데이터 X를 구성하는 특성중 하나를 무작위로 선정
  - 선택된 특성에서 무작위로 값을 추출(Uniform)하여 Split
  - 노드가 더 이상 나누어지지 않는다면 Stop.



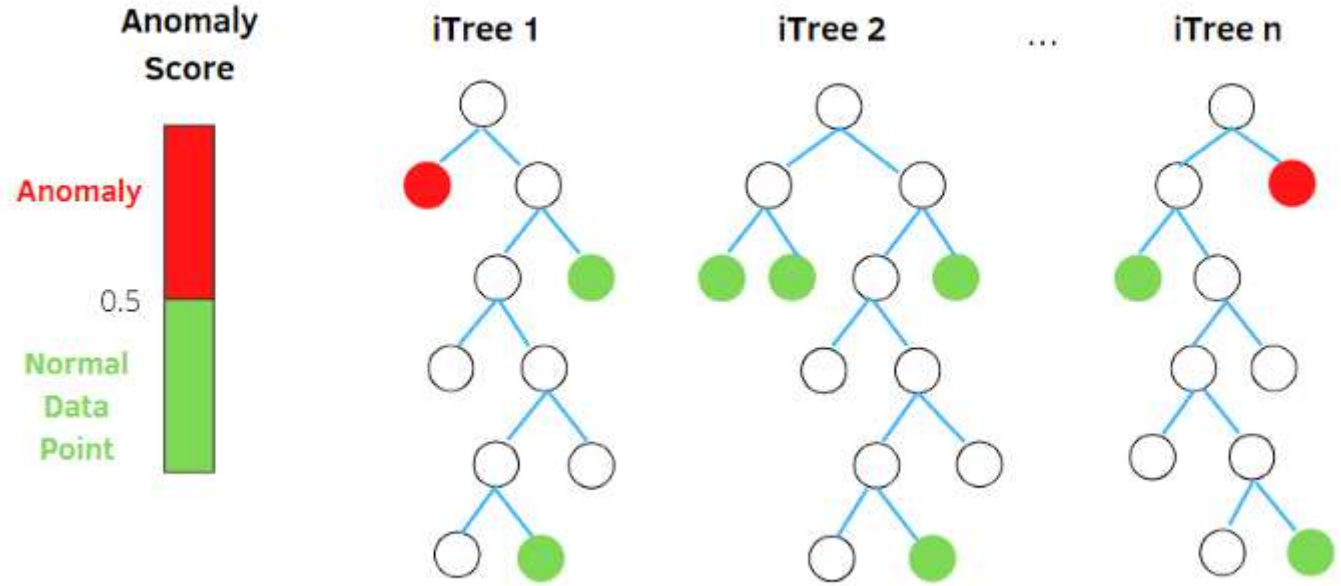
# Isolation Forest

- Isolation Tree의 특징
  - 정상데이터가 고립되려면 많은 수의 질문을 거쳐야 함 (depth가 커짐)
  - 비정상데이터는 적은수의 질문으로 고립 가능 (depth가 낮음)



# Isolation Forest

- 이상치의 측정
  - Random Forest처럼 앙상블 기법을 사용하여 여러 개의 Isolation Tree를 만듦
  - 각 데이터가 고립되는데까지 걸리 평균 거리가 짧을수록 이상치에 가까움



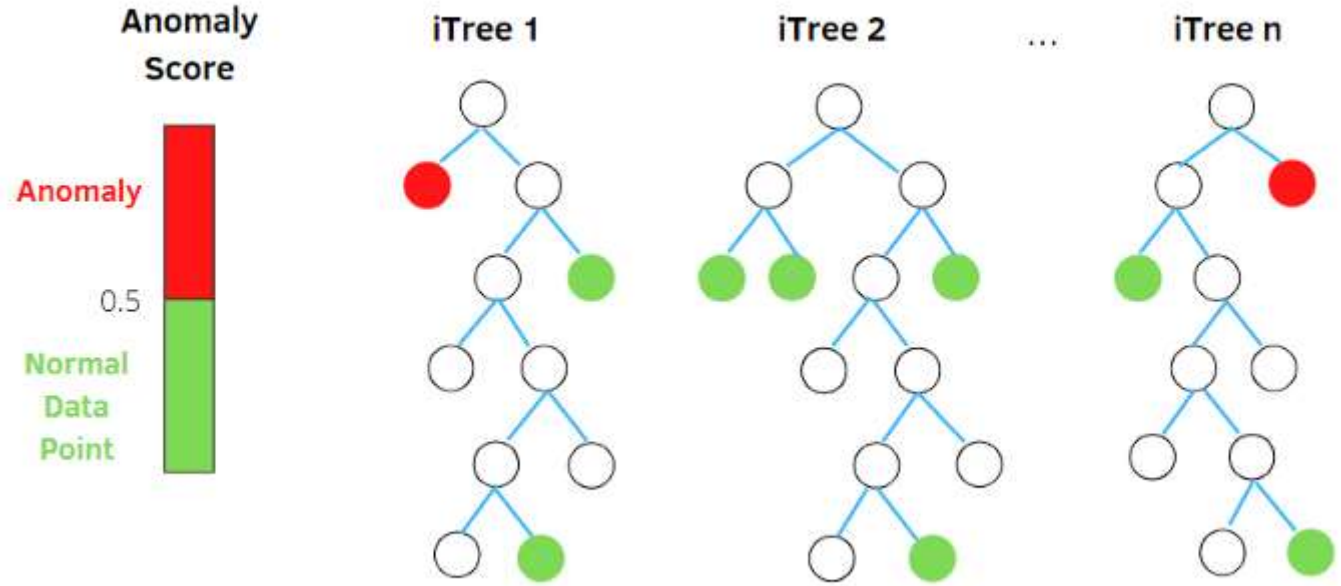
# Isolation Forest

- 이상치의 측정
  - Random Forest처럼 앙상블 기법을 사용하여 여러 개의 Isolation Tree를 만듦
  - 각 데이터가 고립되는데까지 걸리 평균 거리가 짧을수록 이상치에 가까움

$$s(x,n) = 2^{-\frac{E(h(x))}{c(n)}}$$

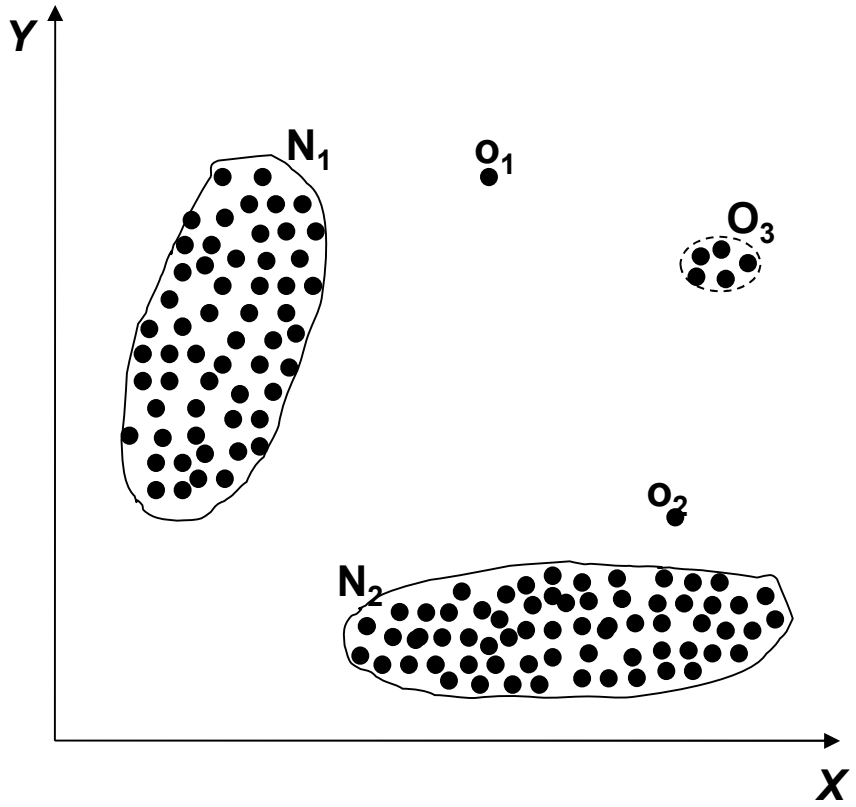
$E(h(x))$ 데이터까지 도달하기까지 평균 경로 길이

$c(n)$ 모든 데이터의 평균길이 (상수)



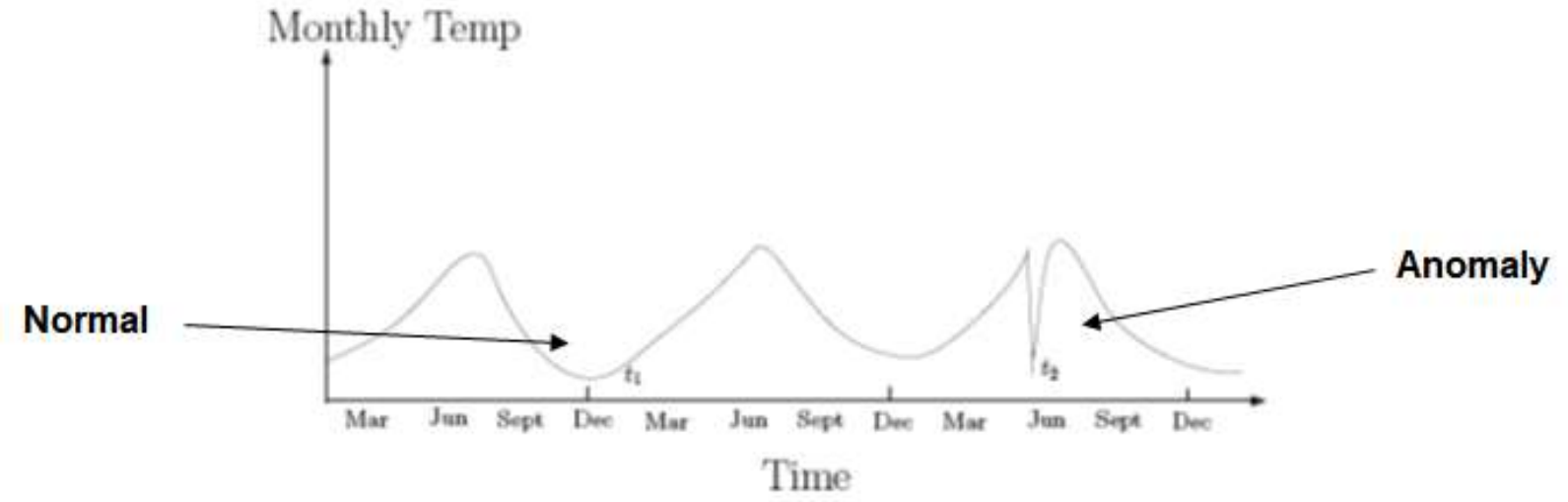
# 이상치의 유형 - Point Anomalies

- 이상치 판단의 단위: 개별 데이터
- 개별 데이터 객체가 다른 정상객체와 다른경우



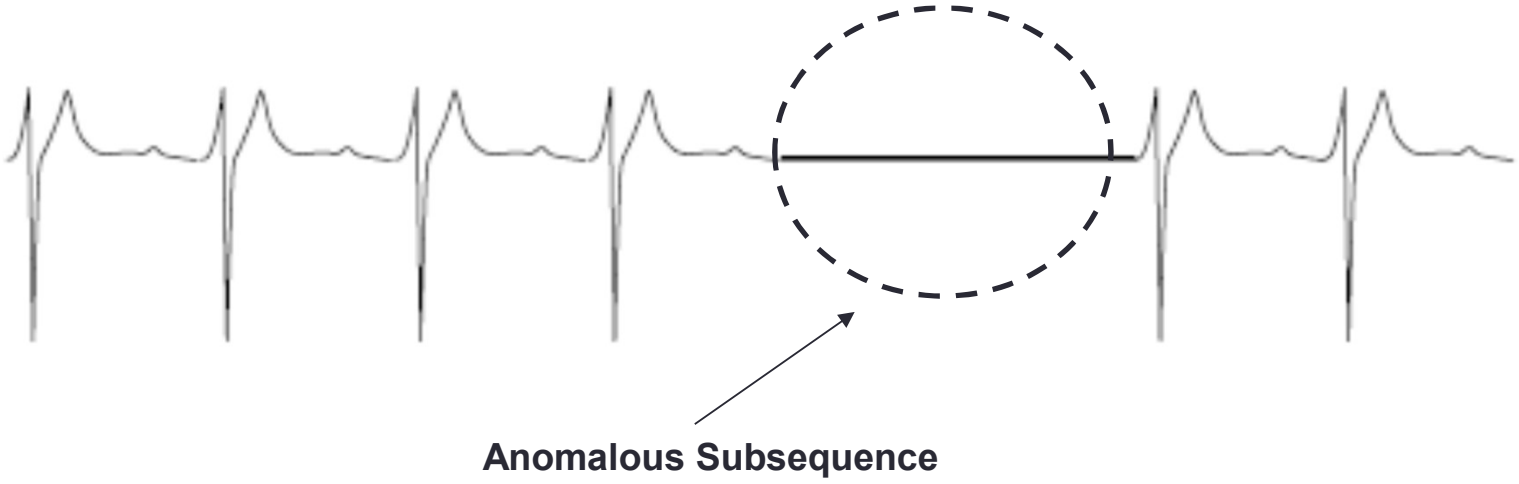
# 이상치의 유형 - Contextual Anomalies

- 개별 값은 정상적으로 보이지만, 주변 데이터와 비교해 보았을때 이상
- 컨텍스트에 대한 정의 필요
  - 공간적/시간적/…
- A.K.A conditional anomalies



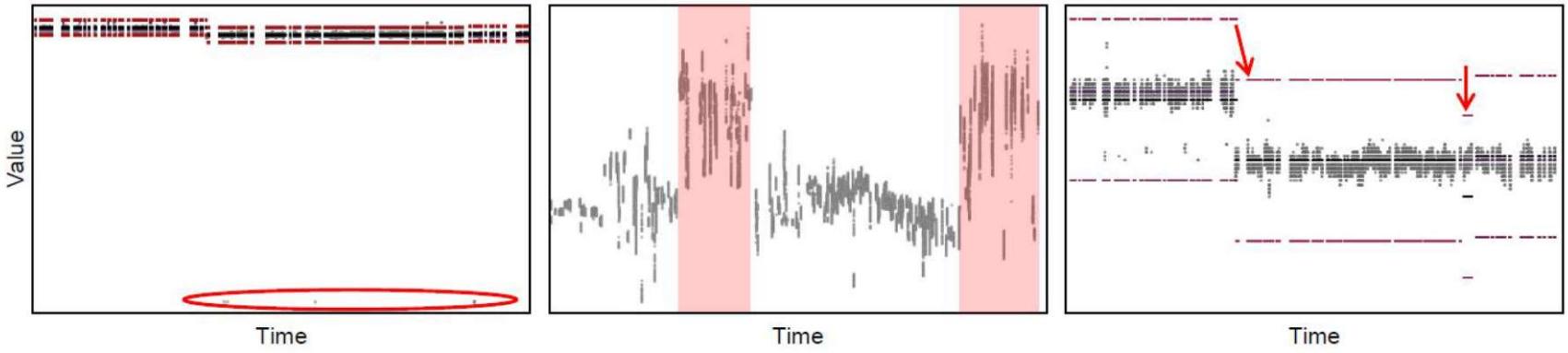
# 이상치의 유형 - Collective Anomalies

- 이상치 판단 범위: 데이터 집단
- 데이터 집단이 다른 집단과 다른 분포를 보일때



# 이상치 탐지의 응용분야

- **제조 공정에서 시간의 흐름에 따른 점진적/급진적 데이터 특성 변화를 탐지**
  - **이상치 –** 외부 충격, 불량 발생, 작업자 오류 (특히, 수기 데이터) 등
  - **점진적 데이터 특성 변화 –** 설비 노후화, Calibration, 점진적 내/외부 환경 변화 등
  - **급진적 데이터 특성 변화 –** 제품/설비 조건 변경, 설비 정비/유지보수 등




(a) Outliers (b) Seasonality (c) Process Drift