

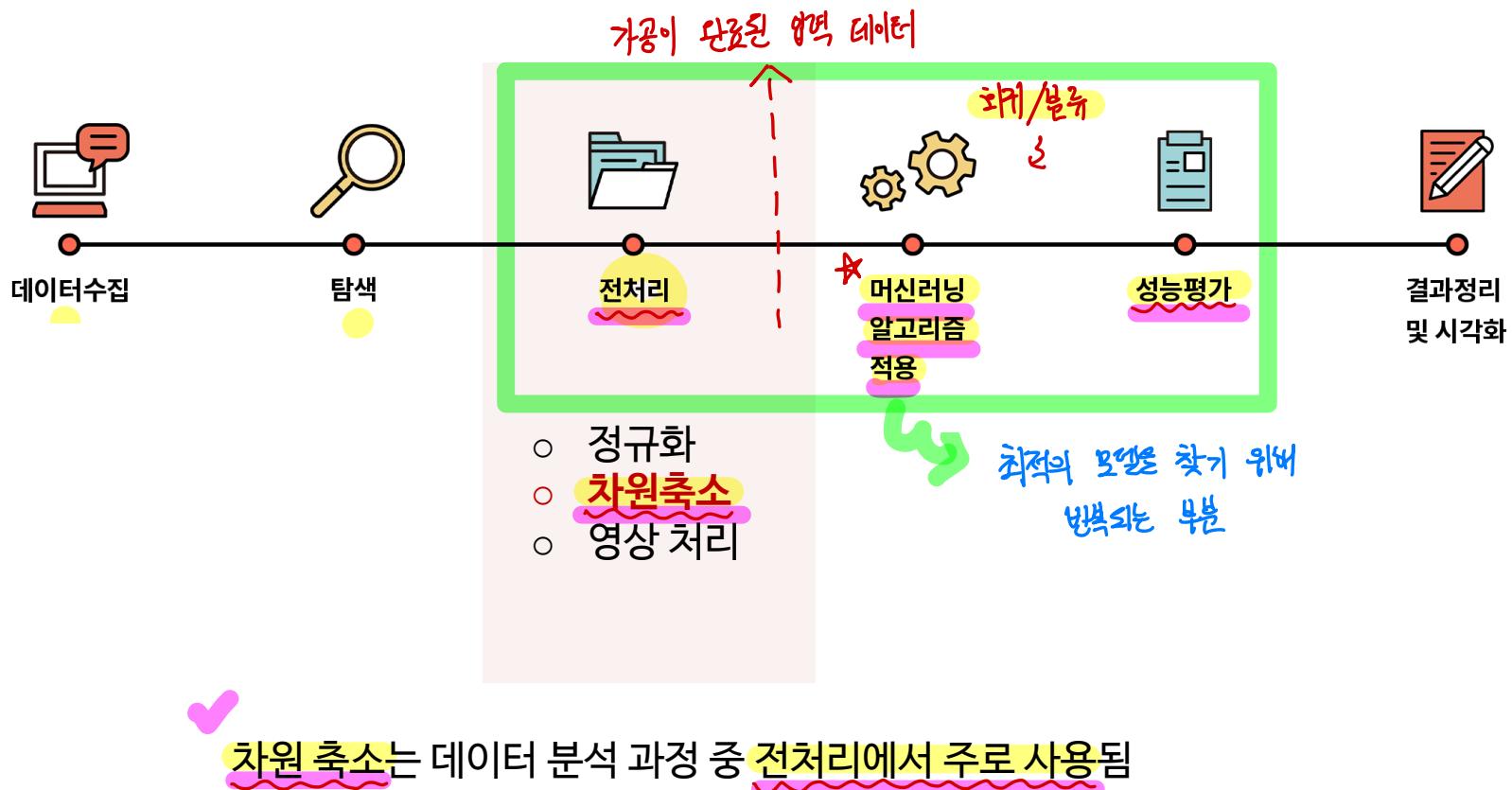
# 차원 축소

---

Dimension Reduction

# 데이터 분석 과정

## ■ 머신러닝을 이용한 데이터 분석 과정



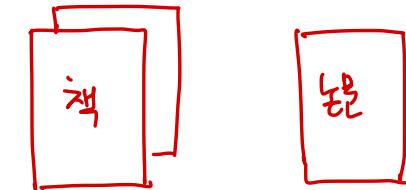
# 차원 축소의 필요성

~> 목적: 고차원  $\Rightarrow$  저차원

- 고차원 데이터 (High dimensional data)의 예시

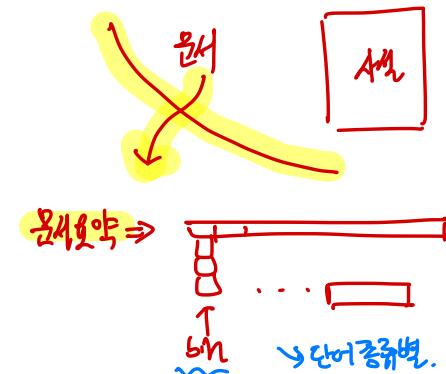
- 문서 요약 (데이터 예시)

- 예) 한 문서의 크기  $\rightarrow$  한 언어의 단어 수로 표현
- Billions of documents \* bag of words



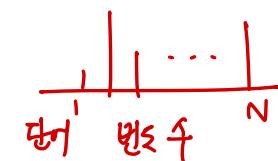
- 추천 시스템

- 예) 사용자 \* 총 영화 수 matrix로 표현
- 480,189 users \* 17,770 movie



- 유전자 군집화

- 예) 유전자 수 \* 유전자의 컨디션
- $10,000 \text{ genes} * 1,000 \text{ conditions} \Rightarrow 10^7$



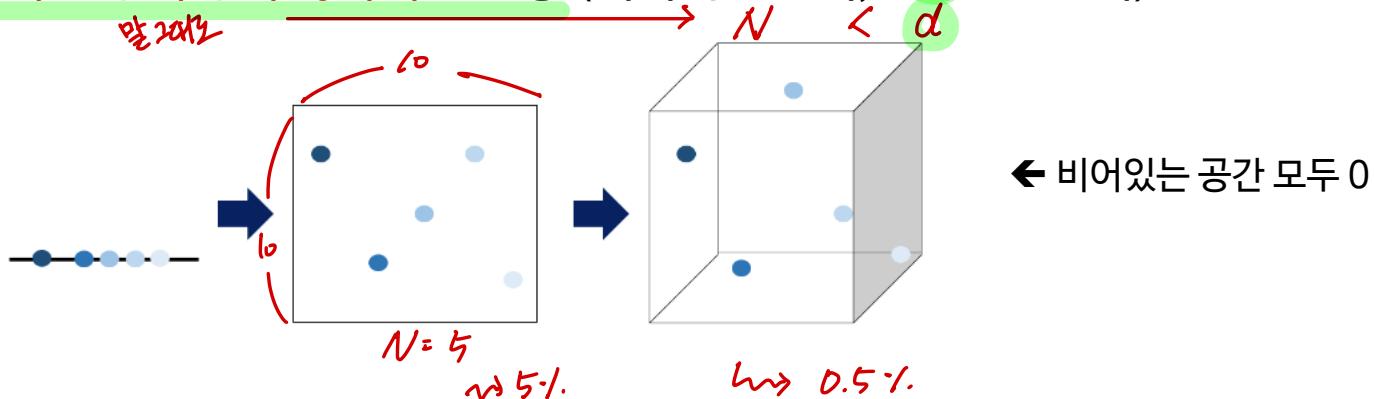
~> 이제 우리가 다룬 데이터의 차원이  
크기가 축소할 필요!!

# 차원 축소 개요

## • 차원의 저주 (Curse of dimensionality) 차원이 증가 ~ 모델 성능 저하

- ( 차원이 증가할 수록 동일 정보량을 표현하기 위해 필요한 데이터의 수는 지수적으로 증가한다는 의미 )

- 데이터 학습을 위해 차원이 증가하면서 학습 데이터 수가 차원 수보다 적어져 모델의 성능이 저하되는 현상
- ( 데이터 차원이 증가할 수록 ) 개별 차원 내 학습할 데이터 수가 적어지는 (sparse) 현상 발생
- 무조건 변수의 수가 증가한다고 해서 차원의 저주 문제가 있는 것은 아니며, 데이터의 수 보다 변수의 수가 많아지면 발생 (데이터 200개, 변수 7000개)



# 차원 축소 개요

16 pixels  
x  
16 pixels  
 $\hookrightarrow 16 \times 16 = 256$

## ■ 차원의 저주 (Curse of dimensionality)

- 일반적으로 intrinsic dimension은 original dimension 보다 상대적으로 작음

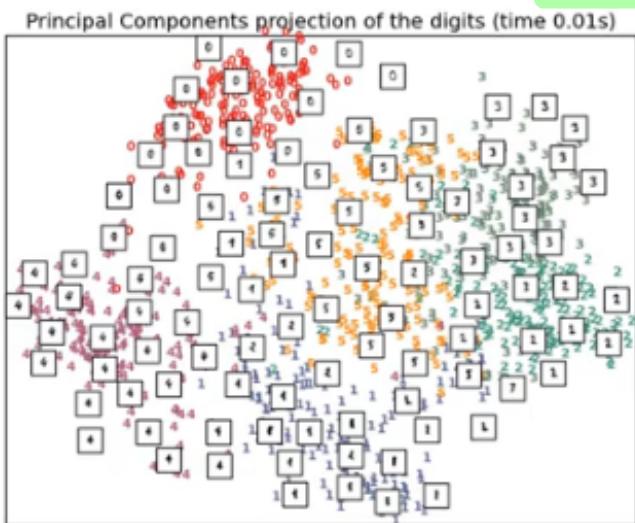
- 예시) 고유차원  $\rightsquigarrow$  표면적을 살피자 X 차원의 차원

- MNIST 16x16 (256 dimensions) 데이터  $\rightsquigarrow$  2차원 데이터

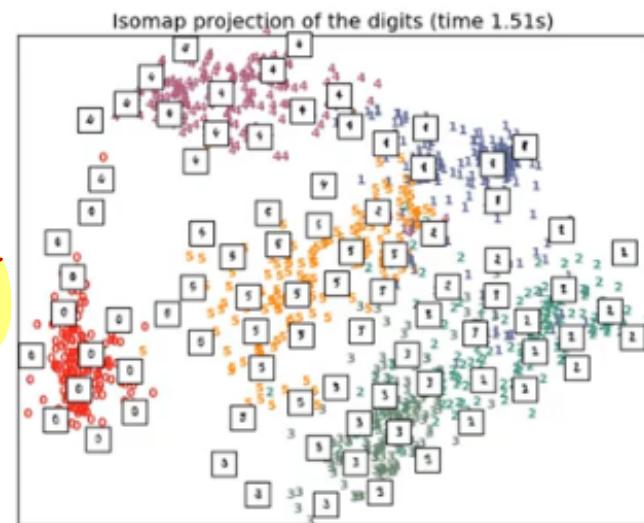
- ~~IPCA~~와 ISOMAP을 통한 2차원 데이터로 차원 축소

PCA

256  $\rightarrow$  2차원



→  
같은 클래스  
끼리 잘  
뭉쳐있다.



2차원이지만 대략적으로 MNIST 클래스의 형태가 유지됨

# 차원 축소 개요

## ■ 차원의 저주 (Curse of dimensionality)

### ▪ 차원이 높을 수록 발생하는 문제

- 데이터에 포함될 노이즈의 비율도 높아짐

- 성능 감소를 야기함



- 모델 학습과 추론의 계산 복잡도가 높아짐 방법의 + - \* /

- 동일한 성능을 얻기 위해 더 많은 데이터의 수가 필요함

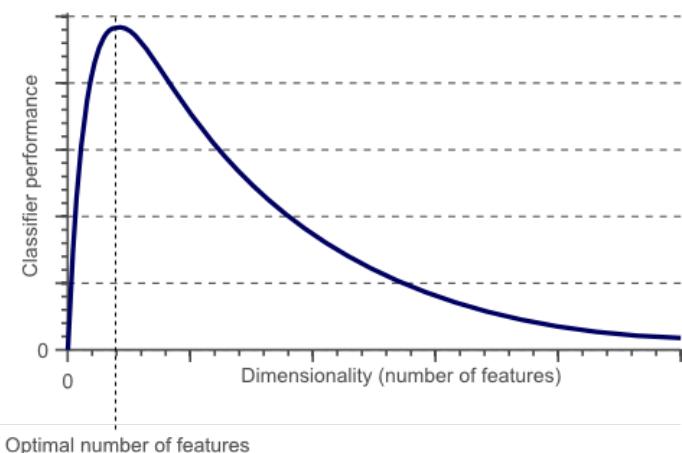
$$d \uparrow \sim N \uparrow$$

### ▪ 차원의 저주 해결 법

- 도메인 지식을 이용 → 중요한 특성만 사용

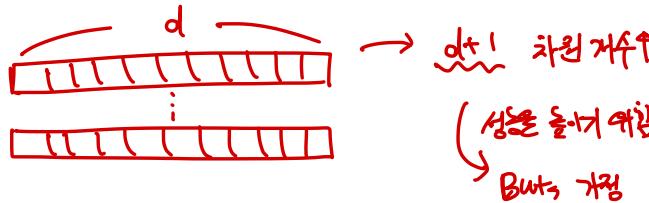
- 목적함수에 Regularization term 추가

- 차원 축소 기술을 전처리로 사용



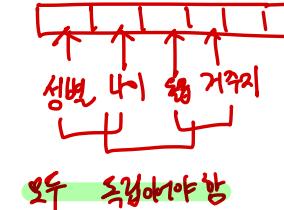
# 차원 축소 개요

N=10



## ■ 차원 축소 배경

- 이론적으로, 차원의 증가는 모델 성능을 향상 시킴
- ~~But~~ 가정: 모든 변수가 서로 독립일 경우 →  
실제로, 차원의 증가는 모델 성능 저하를 가져옴
- 모든 변수는 서로 상관관계가 있고, 노이즈가 존재함  
★  
가정이 틀림.



## ■ 차원 축소 목적

- 모델의 성능을 최대로 해주는 변수의 일부 셋을 찾는 것

## ■ 차원 축소 효과

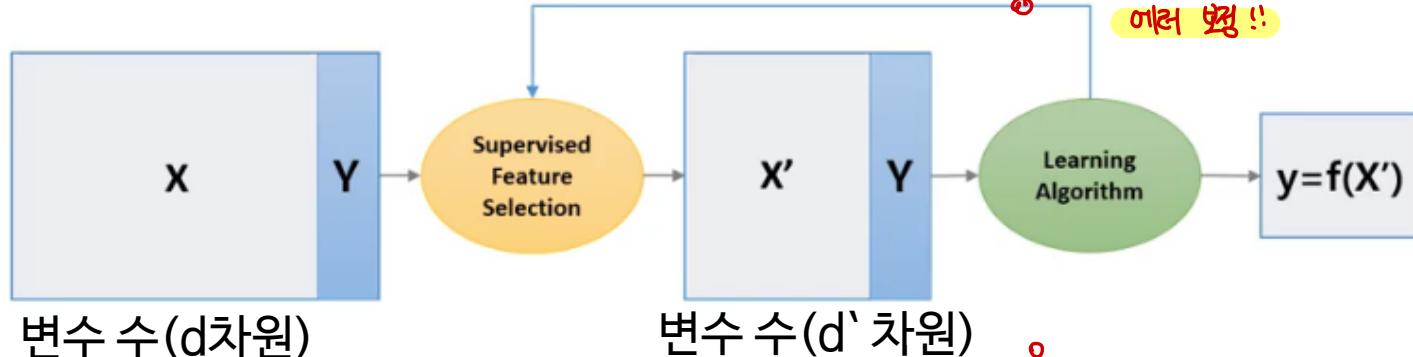
- 변수간 상관 관계(correlations) 제거
- 단순한 후처리 (Post-processing)
- 적절한 정보를 유지하면서 중복되거나 불필요한 변수를 제거
- 시각화가 가능

# 차원 축소 개요

## ■ 지도학습 기반 차원 축소

- 학습결과가 피드백 되어 Feature Selection 을 반복함

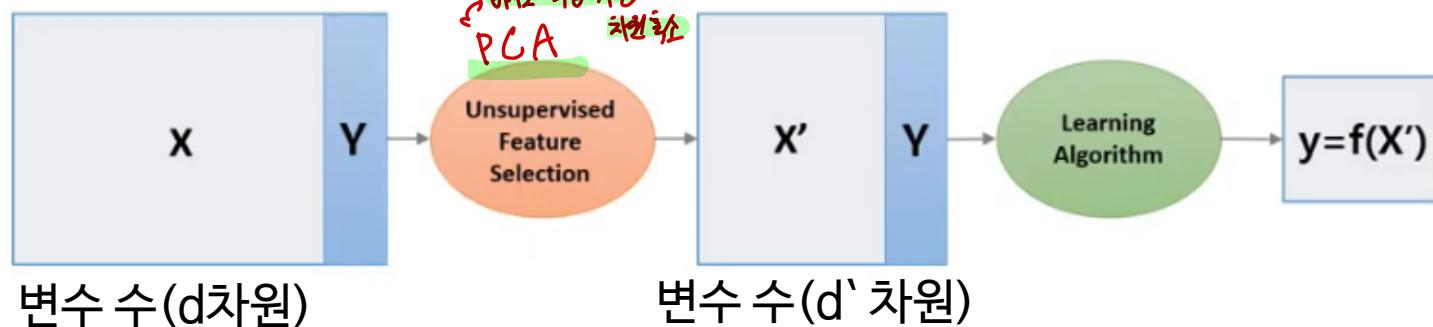
데이터 수



## ■ 비지도학습 기반 차원 축소

- 지도학습 처럼 피드백을 통한 Feature Selection 반복 없음

데이터 수



# 차원 축소 개요

## ■ 차원 축소 방법

### ▪ 변수/피쳐 선택(Feature selection): 유의미한 변수만 선택

- 장점: 선택한 변수 해석 용이
- 단점: 변수간 상관관계 고려의 어려움

$$x_1, x_2, \dots, x_{100} \rightarrow x_1, x_5$$

상관없는  
많은  
A&B

변환 변수 필요.

### ▪ 변수/피쳐 추출(Feature extraction): 예측 변수의 변환을 통해 새로운 변수 추출

- 변수/피쳐 생성(Feature construction)이라고도 함
- 장점: 변수간 상관관계 고려, 변수의 개수를 “많이” 줄일 수 있음
- 단점: 추출된 변수의 해석이 어려움

$$z = f(x_1, x_2, \dots, x_{100})$$

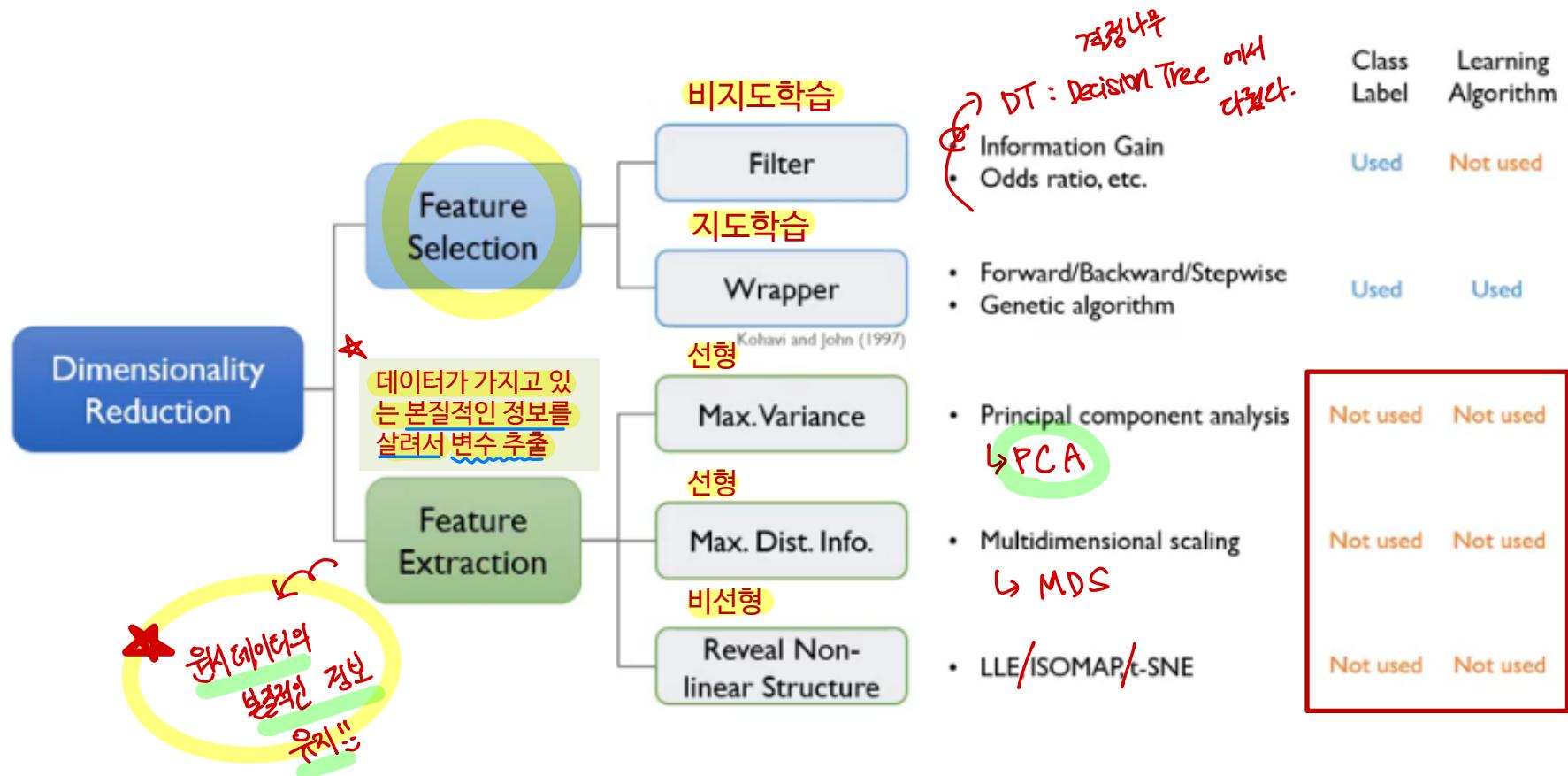
새로운 변수

선형결합

$$\begin{aligned} z_1 &= d_{1,1}x_1 + d_{1,2}x_2 + d_{1,3}x_3 \cdots d_{1,100}x_{100} \\ z_2 &= d_{2,1}x_1 + d_{2,2}x_2 + d_{2,3}x_3 \cdots d_{2,100}x_{100} \\ &\vdots \end{aligned}$$

# 차원 축소 (Dimension Reduction)

## ■ 차원 축소 방법 정리



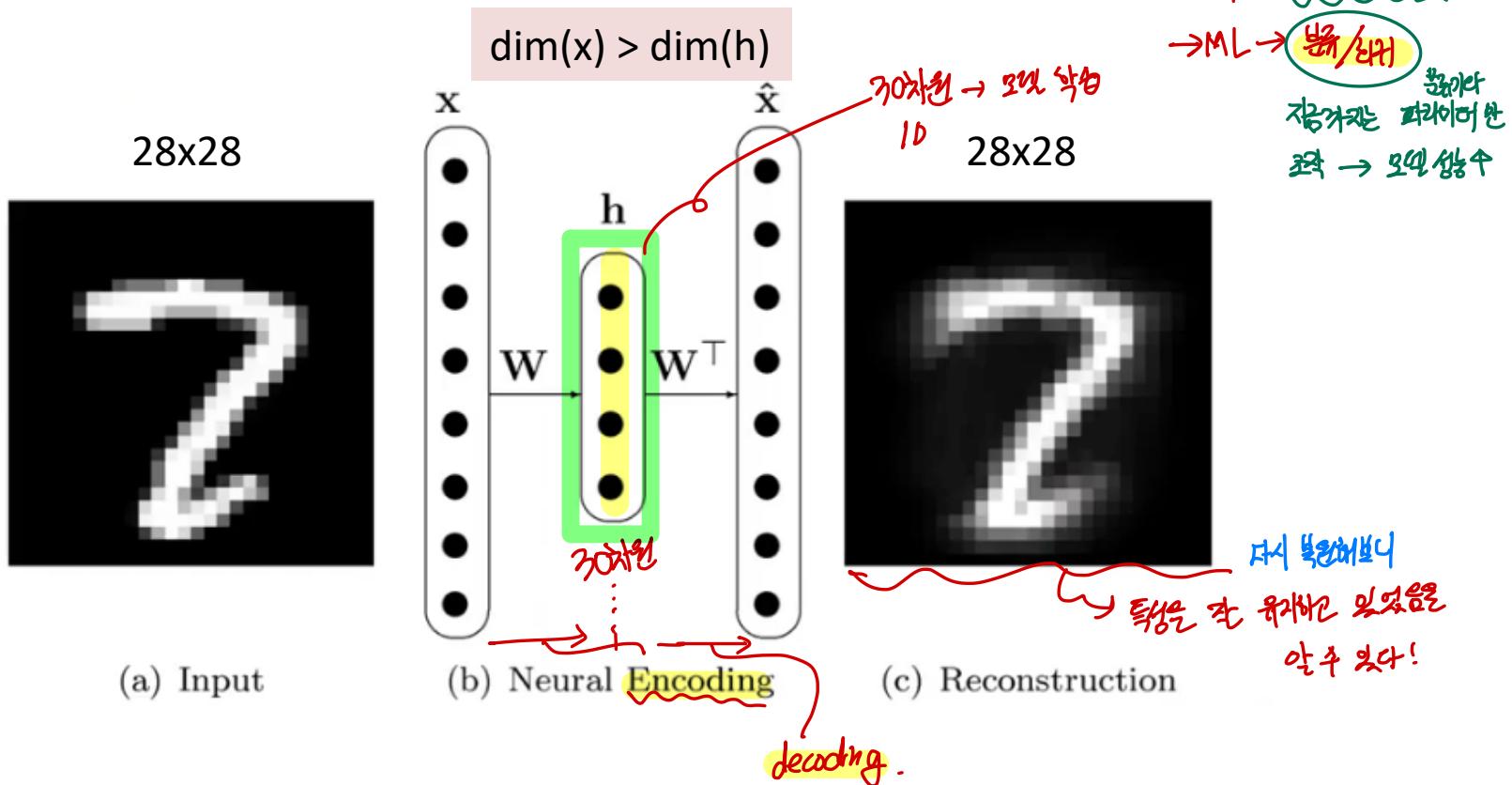
# 차원 축소 (Dimension Reduction)

참고

- 최근 동향: Representation learning: Deep Auto-Encoder

- 인공 신경망 방법을 이용하여 차원을 축소해보자

- Bottleneck layer의 차원이 고차원 데이터를 잘 표현 할 수 있도록 축소 한 것

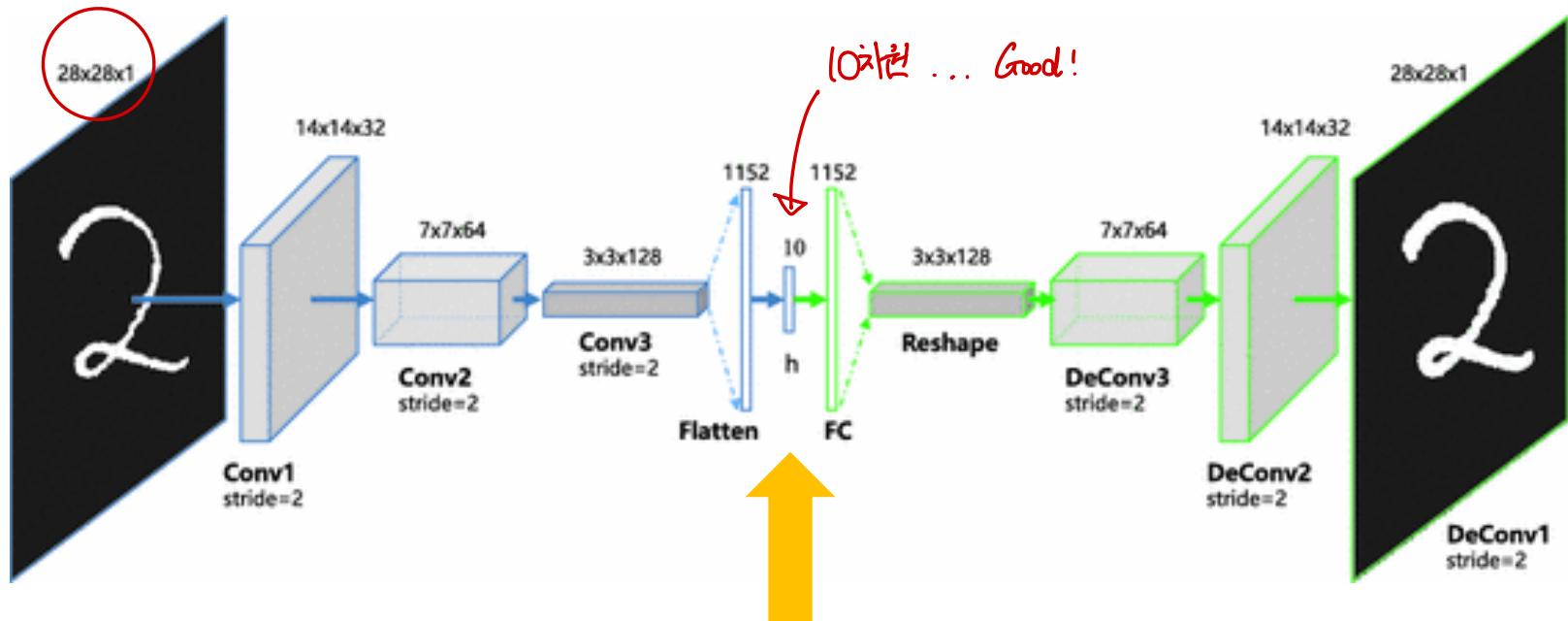


# 차원 축소 (Dimension Reduction)

- 최근 동향: Representation learning: Convolutional Neural Network
  - 인공 신경망 방법을 이용하여 차원을 축소해보자
  - Bottleneck layer의 차원이 고차원 데이터를 잘 표현 할 수 있도록 축소 한 것

→ CNN

최근 인공 신경망으로 차원축소 연구 활발히 진행!



## ★ 핵심

데이터들은 정사영 시켜 차원을 낮출다면,  
어떤 벡터에 데이터들을 정사영 시켜야 원래의 데이터 구조를  
제일 잘 유지할까?

방법, 이런 선형 변환이고 하나의 벡터 공간을  
선형적으로 다른 공간으로 mapping 하는 기능을  
가짐 //

## 차원 축소: PCA

---

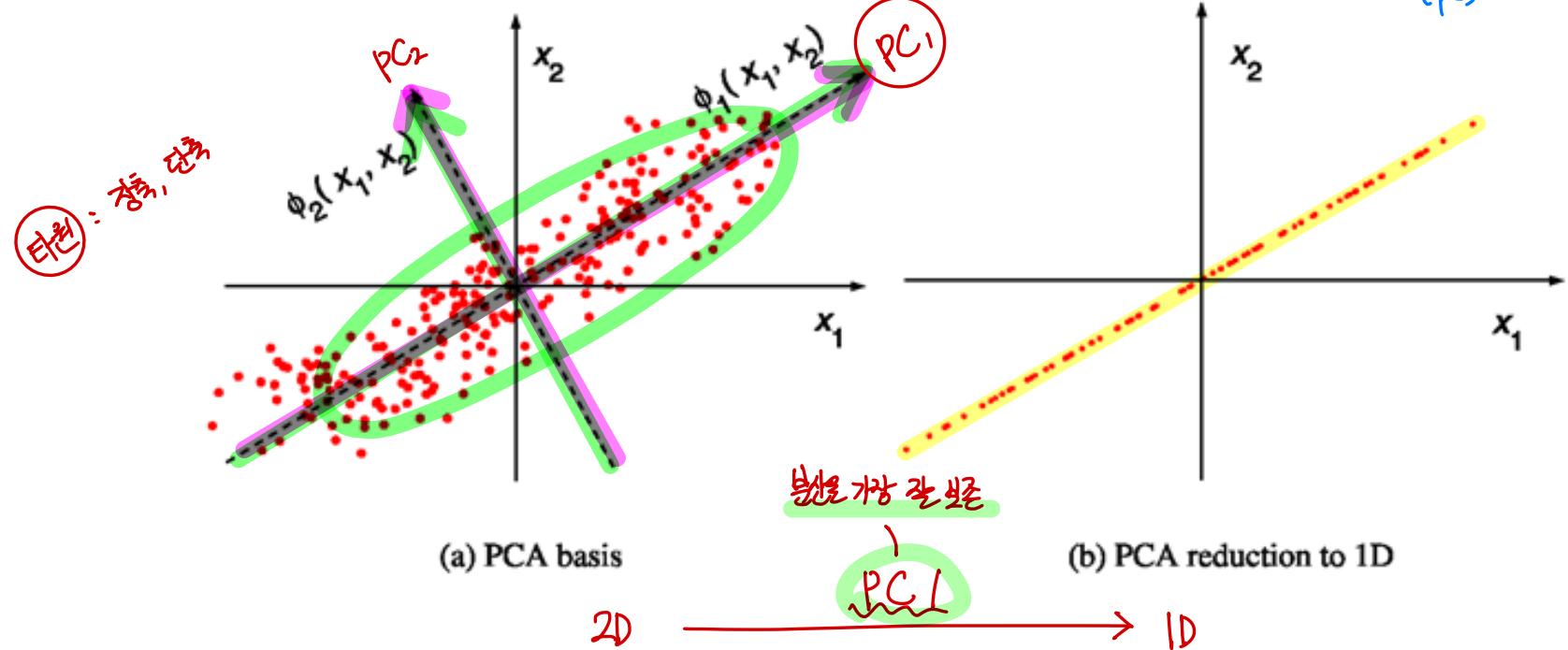
Dimension Reduction

# 주성분 분석(PCA)

## ■ 주성분 분석의 목적

- 차원을 줄이는 비지도 학습 방법 중 한가지
- 사영 후 원 데이터의 **분산(variance)**을 최대한 보존할 수 있는 **기저를** 찾아 차원을 줄이는 방법

★  
기저를 찾아 차원을  
줄이는 방법  
축, principal Axis  
principal Components  
(pc)



# 주성분 분석(PCA)

- MNIST의 예시



MNIST

PCA

PCA 측

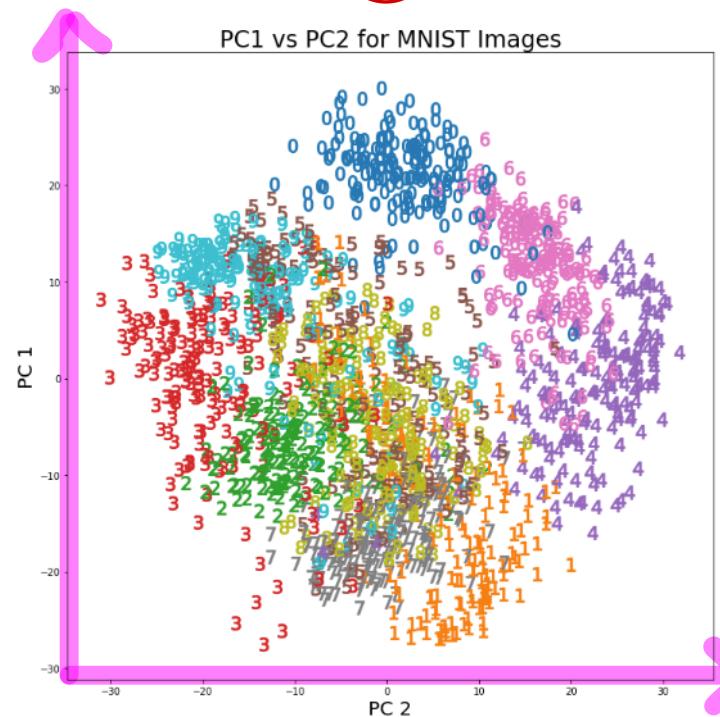
Feature

I/O Vector

3

2D, 3D 만드는 시각화

t-SNE

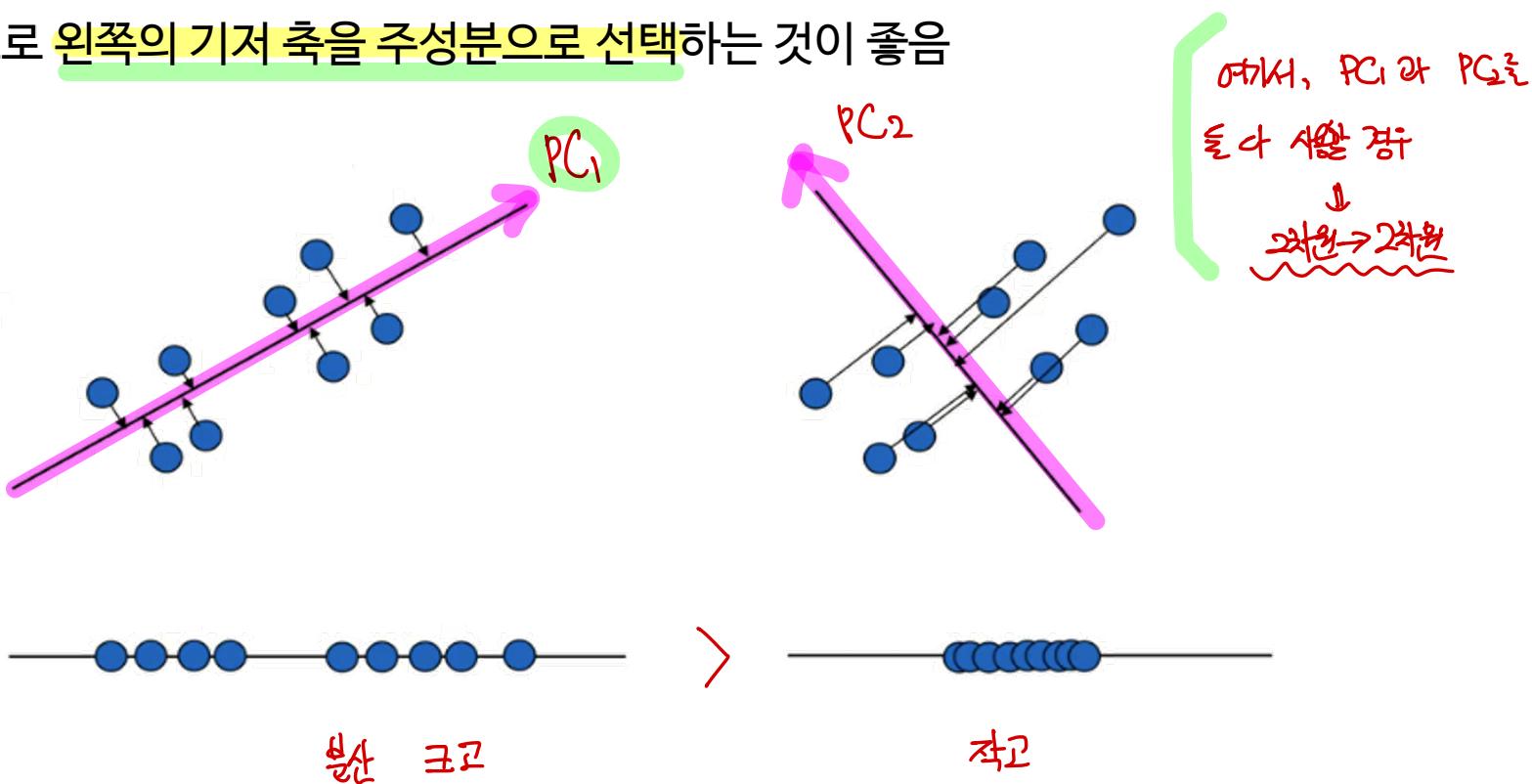


PCA

# 주성분 분석(PCA)

## ▪ 주성분 분석

- 데이터를 사영(projection) 시킬 경우 손실되는 정보의 양이 적은 쪽의 기저(축)를 선택
- 아래 예시의 경우 왼쪽 기저(축)가 오른쪽 기저 보다 원 데이터의 분산을 최대로 유지하므로 왼쪽의 기저 축을 주성분으로 선택하는 것이 좋음



# 주성분 분석(PCA): 수리적 배경

- 주성분 분석: 선형 결합

- “데이터( $X$ ) 사영 변환 후( $Z$ )에도 분산이 보존하는 기저( $a$ )을 찾는 것”

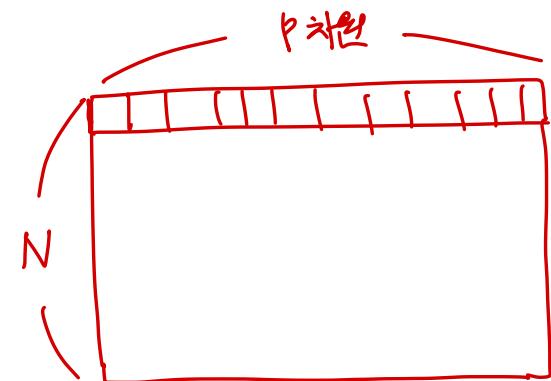
☞  
☞

$$Z_1 = \alpha_1^T X = \alpha_{11}X_1 + \alpha_{12}X_2 + \dots + \alpha_{1p}X_p$$

$$Z_2 = \alpha_2^T X = \alpha_{21}X_1 + \alpha_{22}X_2 + \dots + \alpha_{2p}X_p$$

⋮

$$Z_p = \alpha_p^T X = \alpha_{p1}X_1 + \alpha_{p2}X_2 + \dots + \alpha_{pp}X_p$$



( )  $X_1, X_2, \dots, X_p$ : 원 데이터  $P$  개 변수

( )  $\alpha_i = [\alpha_{i1}, \alpha_{i2}, \dots, \alpha_{ip}]$ :  $i$  번째 기저(basis) 또는 계수/loading)

( )  $Z_1, Z_2, \dots, Z_p$ : 각 기저로 사영 변환 후 변수 (주성분)

# 주성분 분석(PCA): 수리적 배경

- 공분산 (Covariance) : 변수의 상관 정도

- $X$  : 입력 데이터 ( $n$  개의 데이터,  $d$  개의 변수)

$$Cov(X) = \frac{1}{n} (X - \bar{X})(X - \bar{X})^T$$

$[d \times d]$        $[d \times n]$        $[n \times d]$

↑  $X$ 의 평균

- 데이터 셋의 전체 분산(Total variance)

$$\rightarrow \text{tr}[\text{Conv}(X)] = \text{Conv}(X)_{11} + \text{Conv}(X)_{22} + \dots + \text{Conv}(X)_{dd}$$

예시)

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 5 & 4 & 1 \\ 3 & 8 & 6 \end{bmatrix}, \quad \text{Conv}(A) = \begin{bmatrix} 2.67 & 0.67 & -2.67 \\ 0.67 & 4.67 & 2.33 \\ -2.67 & 2.33 & 4.67 \end{bmatrix}$$

1과 2의 상관관계

3

대각을 다 더하면

데이터셋의 전체 분산

# 주성분 분석(PCA): 수리적 배경

- 사영 (Projection)



$$(\vec{b} - p\vec{a})^T \vec{a} = 0 \Rightarrow \vec{b}^T \vec{a} - p\vec{a}^T \vec{a} = 0 \Rightarrow p = \frac{\vec{b}^T \vec{a}}{\vec{a}^T \vec{a}}$$

$$\vec{x} = p\vec{a} = \frac{\vec{b}^T \vec{a}}{\vec{a}^T \vec{a}} \vec{a}$$

If  $\vec{a}$  is unit vector

$$p = \vec{b}^T \vec{a} \Rightarrow \vec{x} = p\vec{a} = (\underbrace{\vec{b}^T \vec{a}}_{\text{scalar}}) \vec{a} \quad \vec{x} = (\vec{b}^T \vec{a}) \vec{a}$$



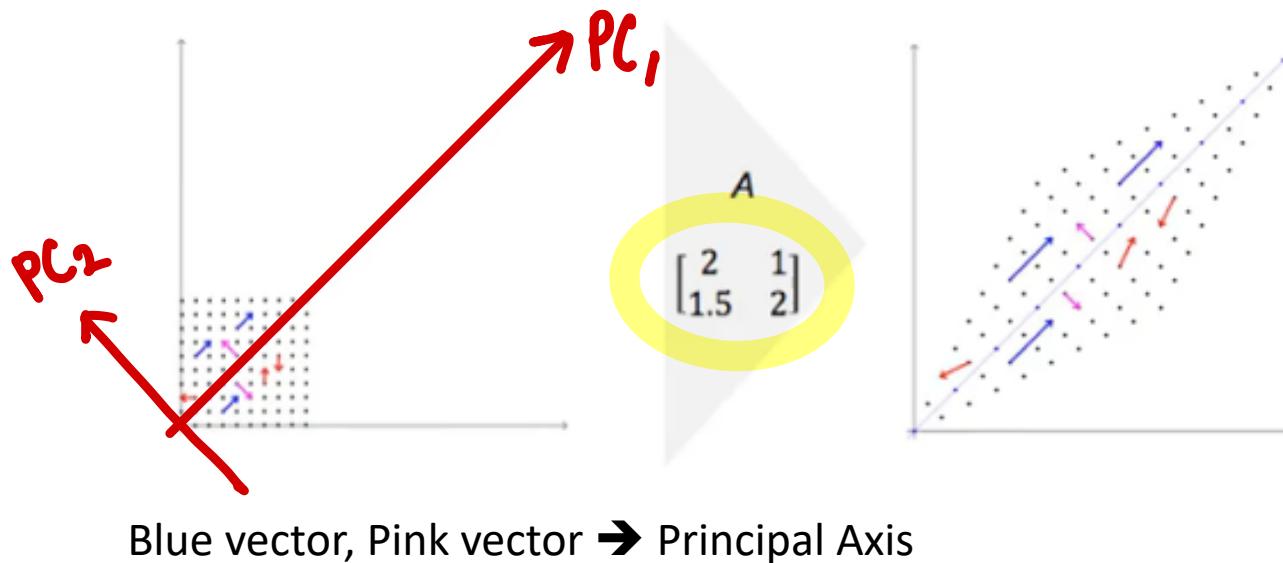
x: 사영후 벡터, p: 직교 사영을 위한 스케일러  
b = 데이터, a = 기저축 (PC)

# 주성분 분석(PCA): 수리적 배경

- 고유값(eigenvalue)과 고유벡터(eigenvector)

$$Ax = \lambda x \rightarrow (A - \lambda I)x = 0$$

- \* 벡터에 행렬을 곱하는 것은 선형 변환의 의미를 가짐
  - 고유벡터는 변화에 의해 방향 변화가 발생하지 않음
  - 고유벡터의 크기 변화는  $\lambda$  만큼



# 주성분 분석(PCA): 수리적 배경

- 고유값(eigenvalue)과 고유벡터(eigenvector)

$$\mathbf{Ax} = \lambda \mathbf{x} \rightarrow (\mathbf{A} - \lambda \mathbf{I})\mathbf{x} = 0$$

- 행렬  $\mathbf{A}$  가 Non-singular 하다면,  $d$ 개의 고유값과 고유벡터가 존재함
- ✓ 고유벡터는 서로 직교함(orthogonal) 기여.
- $\text{tr}(\mathbf{A}) = \lambda_1 + \lambda_2 + \dots + \lambda_d$ ,   
  $\downarrow$   
  $\text{PC1}, \text{PC2}$   
 $\Rightarrow \text{PCA 주들은 서로 직교}$ ,

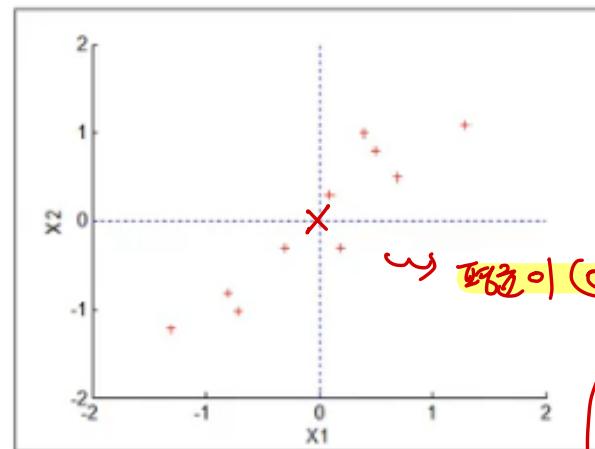
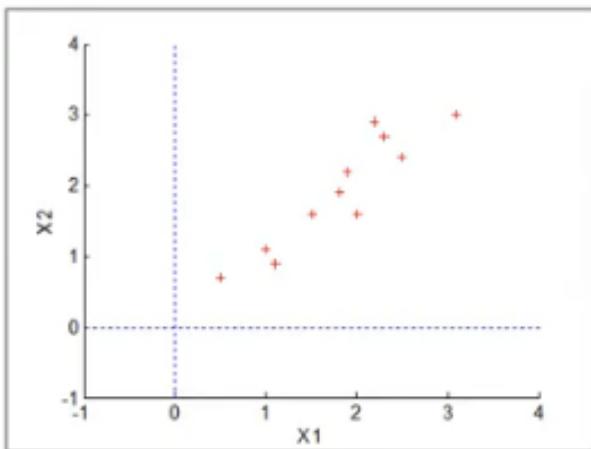
# 주성분 분석 알고리즘

PCA가 실제로 어떻게 동작하는지 알아보자!

- Step1: 데이터 센터링 (data centering)

- 데이터 평균을 0으로 변경

$x_1$	2.5	0.5	2.2	1.9	3.1	2.3	2	1	1.5	1.1
$x_2$	2.4	0.7	2.9	2.2	3	2.7	1.6	1.1	1.6	0.9
$x_1$	0.69	-1.31	0.39	0.09	1.29	0.49	0.19	-0.81	-0.31	-0.71
$x_2$	0.49	-1.21	0.99	0.29	1.09	0.79	-0.31	-0.81	-0.31	-1.01



→ 평균이 (0,0)이 되도록

부차적으로  
정규화한 폭의를  
찾는데 도움.

# 주성분 분석 알고리즘

PCA의 방향성  
→ 사영된 분산의 최대화  
데이터

## ■ Step2: 최적화 문제 정의

- 데이터  $X$ 를 기저 벡터  $W$ 에 사영(projection)하면, 사용 후 분산은 다음과 같음

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

- $S$ 는  $X$ 의 covariance matrix  $\rightarrow$  데이터의 구조를 설명해준며, 특히 특정 성분의 변동이 얼마나 높았는지를 나타냄.

- PCA의 목적은 사영 이후 분산  $V$ 를 최대화 하는 것

$$\max \mathbf{w}^T \mathbf{S} \mathbf{w}$$

$$\text{s.t. } \mathbf{w}^T \mathbf{w} = 1$$

$$\Rightarrow S = \begin{pmatrix} 0.6166 & 0.6154 \\ 0.6154 & 0.7166 \end{pmatrix}$$

# 주성분 분석 알고리즘

S는 X의 covariance matrix  
W는 S의 eigen vector  
람다는 S의 eigen value

- Step3: 최적화 문제 솔루션

- 라그랑지 멀티플라이어(Lagrangian multiplier) 적용

$$\begin{aligned} \max \quad & \mathbf{w}^T \mathbf{S} \mathbf{w} \\ \text{s.t.} \quad & \mathbf{w}^T \mathbf{w} = 1 \end{aligned}$$

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

이러한 형태가  
된다.

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

$\mathbf{w}$ 는 S의 eigen vector  
 $\lambda$ 는 S의 eigen value

의  $\mathbf{w}$

$$\text{Eigenvectors} = \begin{bmatrix} 0.6779 & -0.7352 \\ 0.7352 & 0.6779 \end{bmatrix}$$

의  $\lambda$

$$\text{Eigenvalues} = (1.2840 \ 0.0491)$$

# 주성분 분석 알고리즘

S는 X의 covariance matrix  
W는 S의 eigen vector  
람다는 S의 eigen value

## ■ Step4: 주축 정렬

- Eigenvalue에 해당되는 eigenvectors를 순서대로 정렬

$$\text{Eigenvectors} = \begin{bmatrix} 0.6779 & -0.7352 \\ 0.7352 & 0.6779 \end{bmatrix}$$

$$\text{Eigenvalues} = (1.2840 \quad 0.0491)$$

- $W_1$ 은 Eigen vector이고  $\lambda_1$ 은 대응되는 eigen value이다.
- (아래 식의 유도에 의하면)  $W_1$ 에 사영된 데이터의 분산은  $\lambda_1$ 이다.

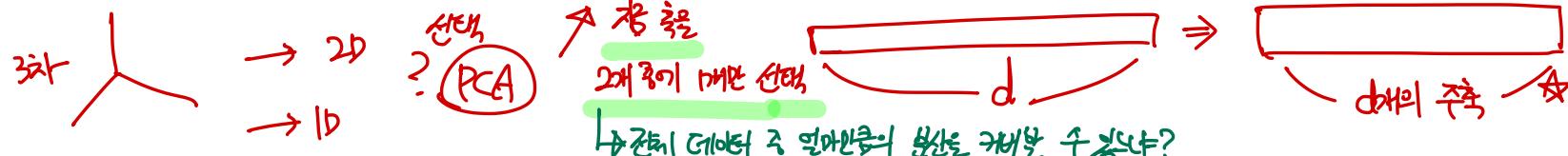
$$v = (w_1^T X)(w_1^T X)^T = w_1^T X X^T w_1 = w_1^T S w_1$$

$$\text{Since } S w_1 = \lambda_1 w_1, \quad w_1^T S w_1 = w_1^T \lambda_1 w_1 = \lambda_1 w_1^T w_1 = \lambda_1$$

첫번째 주성분으로 설명가능한 데이터 비율

$$= \frac{\lambda_1}{\lambda_1 + \lambda_2} = \frac{1.2840}{1.2840 + 0.0491} = 0.96$$

첫번째 주성분으로 설명 가능한 비율은 0.96입니다.  
여러 주축이 중요한 주축?  
나머지가 큰 것



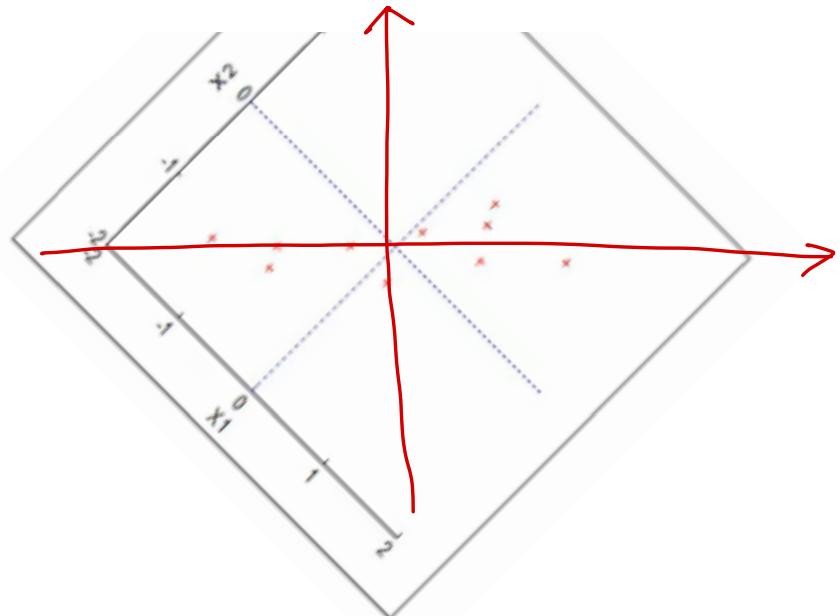
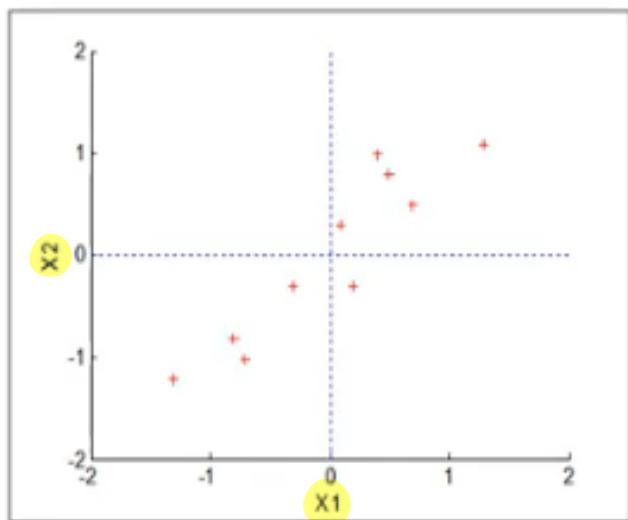
# 주성분 분석 알고리즘

## ■ Step5: PCA로 변환된 데이터

▪ Principle Component 1 =  $Z_1 = W_1^T X$

↳ 변환된 데이터

$x_1$	0.69	-1.31	0.39	0.09	1.29	0.49	0.19	-0.81	-0.31	-0.71
$x_2$	0.49	-1.21	0.99	0.29	1.09	0.79	-0.31	-0.81	-0.31	-1.01
$z_1$	0.83	-1.78	0.99	0.27	1.68	0.91	-0.10	-1.14	-0.44	-1.22



# 주성분 분석 알고리즘

- Step6: 원데이터로 복원  
2D 데이터  $\rightarrow$  1D PCA  $\rightarrow$  2D 데이터 복원(reconstruction)  
*Loss Angle!*

$$\begin{array}{ccc} \mathbf{X} & \xrightarrow{\text{Projection}} & \mathbf{w}^T \mathbf{X} \\ (\text{d by n}) & & (1 \text{ by d}) (\text{d by n}) \end{array} \quad \begin{array}{c} \xrightarrow{\text{Reconstruction}} \\ \mathbf{w} \mathbf{w}^T \mathbf{X} \\ (\text{d by 1})(\text{1 by d}) (\text{d by n}) \end{array}$$

$x_1$	0.69	-1.31	0.39	0.09	1.29	0.49	0.19	-0.81	-0.31	-0.71
$x_2$	0.49	-1.21	0.99	0.29	1.09	0.79	-0.31	-0.81	-0.31	-1.01
$z_1$	0.83	-1.78	0.99	0.27	1.68	0.91	-0.10	-1.14	-0.44	-1.22
$x'_1$	0.56	-1.21	0.67	0.19	1.14	0.62	-0.07	-0.78	-0.30	-0.83
$x'_2$	0.61	-1.31	0.73	0.20	1.23	0.67	-0.07	-0.84	-0.32	-0.90

# 주성분 분석 이슈

256개의 차원을 몇차원으로 줄여야 할까?



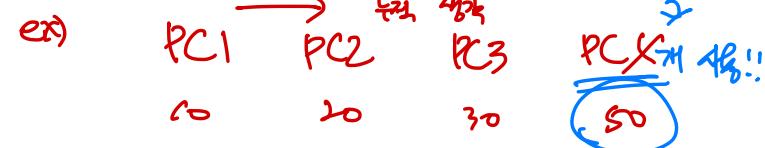
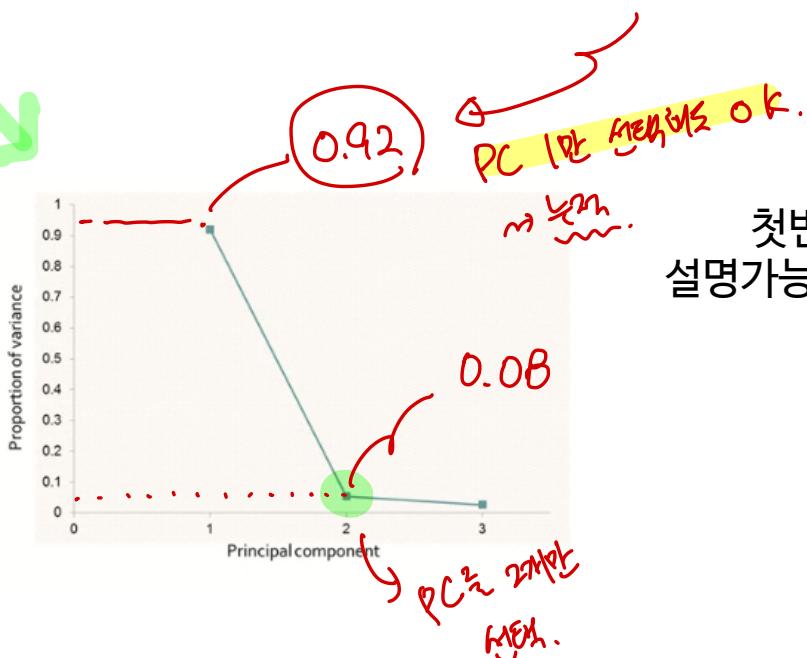
- 주성분 개수 선정 법 → 몇개의 주성분을 사용해야 할까?

- 선택 방법 #1

- 고유 값 감소율이 유의미하게 낮아지는 Elbow Point에 해당하는 주성분을 선택

- 선택 방법 #2

- 일정 수준 이상의 분산 비를 보존하는 최소의 주성분을 선택 (보통 70% 이상)



첫번째 주성분으로  
설명가능한 데이터 비율

$$\begin{aligned} &= \frac{\lambda_3}{\lambda_1 + \lambda_2 + \lambda_3} \\ &= \frac{2.7596}{0.0786 + 0.1618 + 2.7596} \\ &= 0.920 \end{aligned}$$

# 주성분 분석 이슈

- 몇개의 주성분을 사용해야 할까?  
  - 조건: 전체 분산의 80%를 포함해야 함  
*5개.*
- 솔루션  
  - (위) 원 데이터, (아래) PCA 결과
  - PCA 결과의 누적분산비를 살펴보면  
        PC1~PC5까지 선택해야 전체 데이터  
        분산의 82%에 도달할 수 있음

시리얼 이름	제조업체명	유형	칼로리	단백질	지방	나트륨	식이섬유	복합단수화물	설탕	칼륨	비타민
100% Bran	N C	70	4	1	130	10	5	6	280	25	
100% Natural Bran	Q C	120	3	5	15	2	8	8	135	0	
All-Bran	K C	70	4	1	260	9	7	5	320	25	
All-Bran with Extra Fiber	K C	50	4	0	140	14	8	0	330	25	
Almond Delight	R C	110	2	2	200	1	14	8	70	25	
Apple Cinnamon Cheerios	G C	110	2	2	180	1.5	10.5	10	70	25	
Apple Jacks	K C	110	2	0	125	1	11	14	30	25	
Basic 4	G C	130	3	2	210	2	18	8	100	25	
Bran Chex	R C	90	2	1	200	4	15	6	125	25	
Bran Flakes	P C	90	3	0	210	5	13	5	190	25	
Cap'n Crunch	Q C	120	1	2	220	0	12	12	35	25	
Cheerios	G C	110	6	2	290	2	17	1	105	25	
Cinnamon Toast Crunch	G C	120	1	3	210	0	13	9	45	25	
Clusters	G C	110	3	2	140	2	13	7	105	25	
Cocoa Puffs	G C	110	1	1	180	0	12	13	55	25	
Corn Chex	R C	110	2	0	280	0	22	3	25	25	
Corn Flakes	K C	100	2	0	290	1	21	2	35	25	
Corn Pops	K C	110	1	0	90	1	13	12	20	25	
Count Chocula	G C	110	1	1	180	0	12	13	65	25	
Cracklin' Oat Bran	K C	110	3	3	140	4	10	7	160	25	

변수이름	PC1	PC2	PC3	4	5	6	7
calories	0.2995424	0.39314792	0.11485746	0.20435865	0.20389892	-0.25590625	-0.02559552
protein	-0.30735639	0.16532333	0.27728197	0.30074316	0.319749	0.120752	0.28270504
fat	0.03991544	0.34572428	-0.20489009	0.18683317	0.58689332	0.34796733	-0.05115468
sodium	0.18339655	0.13722059	0.38943109	0.12033724	-0.38836424	0.66437215	-0.28370309
fiber	-0.45349041	0.17981192	0.0697604	0.03917367	-0.255119	0.0642436	0.11232537
carbo	0.19244903	-0.14944831	0.56245244	0.0878355	0.18274252	-0.32639283	-0.26046798
sugars	0.22806853	0.35143444	-0.35540518	-0.02270711	-0.31487244	-0.15208226	0.22798519
potass	-0.40196434	0.30054429	0.06762024	0.09087842	-0.14836049	0.02515389	0.14880823
vitamins	0.11598022	0.1729092	0.38785872	-0.6041106	-0.04928682	0.12948574	0.29427618
shelf	-0.17126338	0.26505029	-0.00153102	-0.63887852	0.32910112	-0.05204415	-0.17483434
weight	0.05029929	0.45030847	0.24713831	0.15342878	-0.22128329	-0.39877367	0.01392053
cups	0.29463556	-0.21224795	0.13999969	0.04748911	0.12081645	0.09946091	0.74856687
rating	-0.43837839	-0.25153893	0.1818424	0.0383162	0.05758421	-0.18614525	0.06344455

분산	3.63360572	3.1480546	1.90934956	1.01947618	0.98935974	0.72206175	0.67151642
구산비(%)	27.95081329	24.21580505	14.6873045	7.84212446	7.61045933	5.55432129	5.16551113
누적분산비(%)	27.95081329	52.16661835	66.85391998	74.69604492	82.3065033	87.86082458	93.02633667

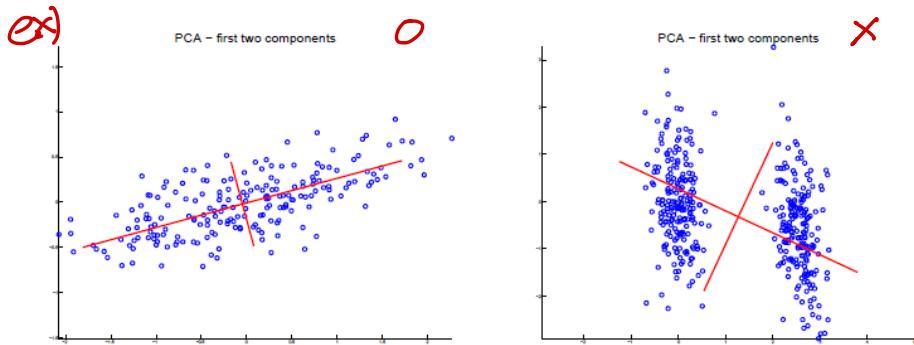
# 주성분 분석 한계

PCA

# [개의 차원은 높도록 찾는다]  
가정 !! unimodel.

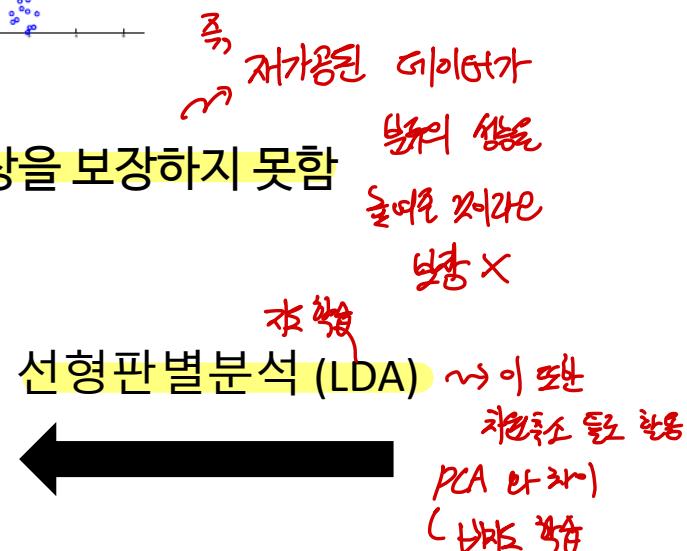
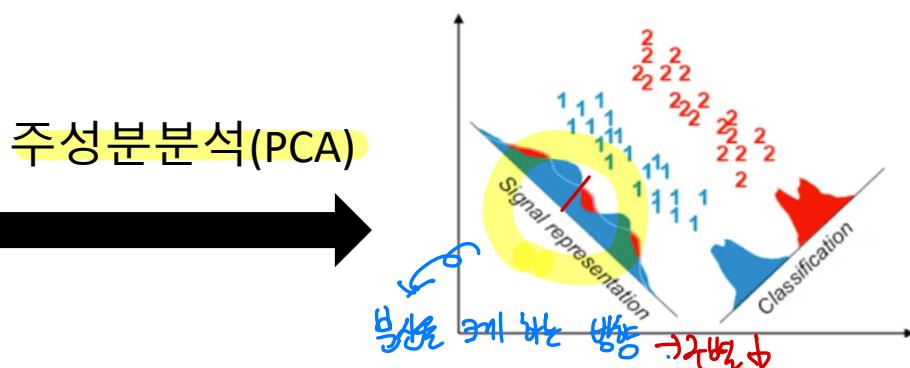
## 한계점 1

- 데이터 분포가 가우시안(non-gaussian)이 아니거나 다중 가우시안(multimodal gaussian) 자료들에 대해서는 적용하기 어려움



## 한계점 2

- 분류 문제를 위해 디자인되지 않음, 즉 분류 성능 향상을 보장하지 못함



# 차원 축소: 기타

---

Dimension Reduction

# Randomized PCA, Kernelized PCA

일반적인 PCA의 단점

$N \ll d$

$N \gg k$   
 $d \gg k$

~ 연산량 ↑

## 랜덤 PCA의 개념

- 자료의 크기 또는 특성변수의 크기가 매우 크면 주성분  $W$ 를 구하기 위한 SVD 계산이 불가능하거나 시간이 많이 소요됨

④ 이런 경우 Randomized PCA 가 유용

- Randomized PCA는 QR 분해를 이용하여 행렬의 SVD를 수행함

↳ 고유값과 고유벡터를 구하기 위한 일련의 ~

## 커널 PCA 개념

- PCA는 선형 변환이고 Kernelized PCA는 비선형 변환임
- SVM의 커널트릭을 PCA에서도 사용
- 특성 변수  $x$ 를 비선형  $h(x)$ 로 변환한 후 이에 대해 PCA를 하여 차원 축소를 하는 방법임

