

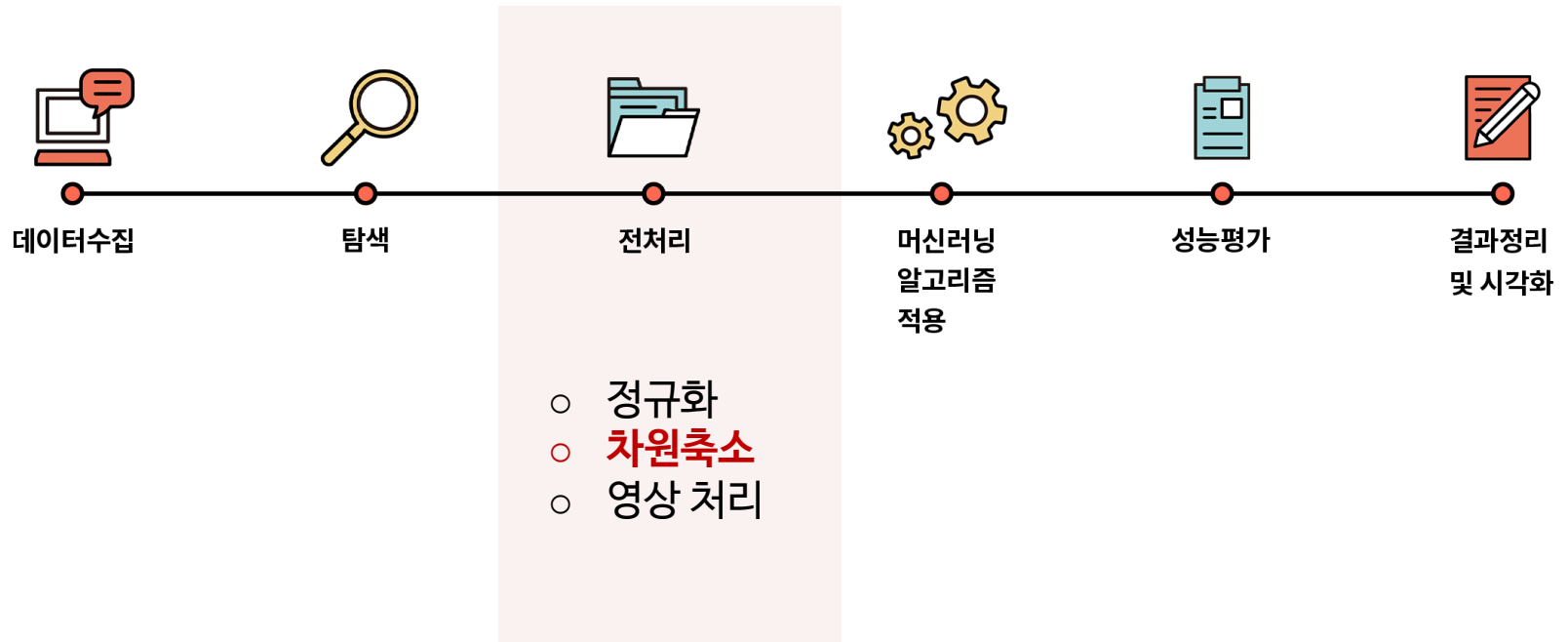
# 차원 축소

---

Dimension Reduction

# 데이터 분석 과정

## 머신러닝을 이용한 데이터 분석 과정



차원 축소는 데이터 분석 과정 중 전처리에서 주로 사용됨

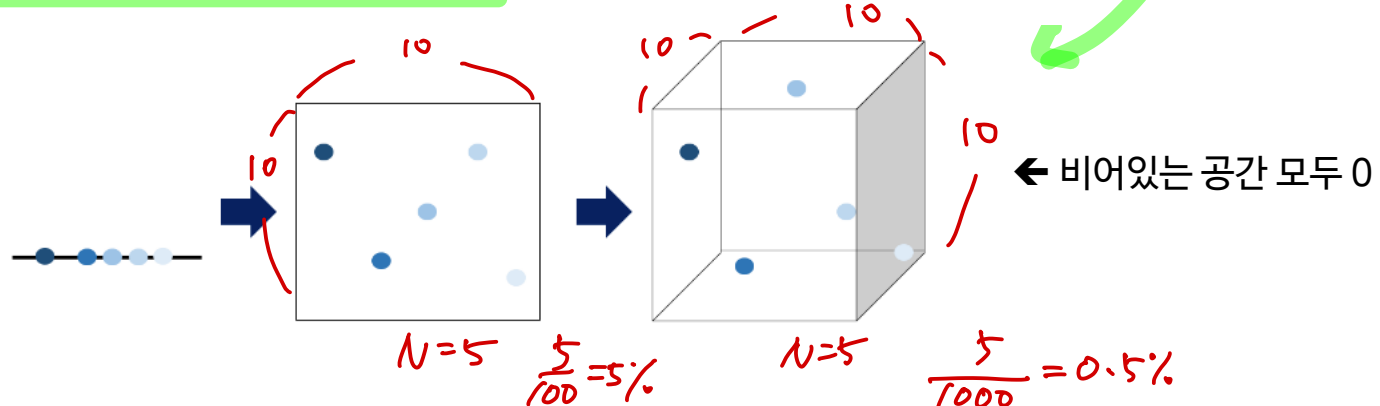
# 차원 축소의 필요성

고차원 → 저차원

- 고차원 데이터 (High dimensional data)의 예시
  - 문서 요약 (데이콘 예시)
    - 예) 한 문서의 크기 → 한 언어의 단어 수로 표현
    - Billions of documents \* bag of words
  - 추천 시스템
    - 예) 사용자 \* 총 영화 수 matrix 로 표현
    - 480,189 users \* 17,770 movie
  - 유전자 군집화
    - 예) 유전자 수 \* 유전자의 컨디션
    - 10,000 genes \* 1,000 conditions

# 차원 축소 개요

- 차원의 저주 (Curse of dimensionality)
  - 차원이 증가할 수록 동일 정보량을 표현하기 위해 필요한 데이터의 수는 지수적으로 증가한다는 의미
  - 데이터 학습을 위해 **차원이 증가**하면서 학습 데이터 수가 차원 수보다 적어져 모델의 **성능이 저하**되는 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)

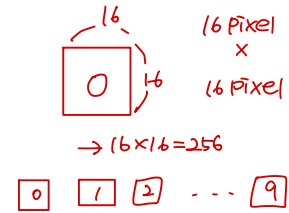


# 차원 축소 개요

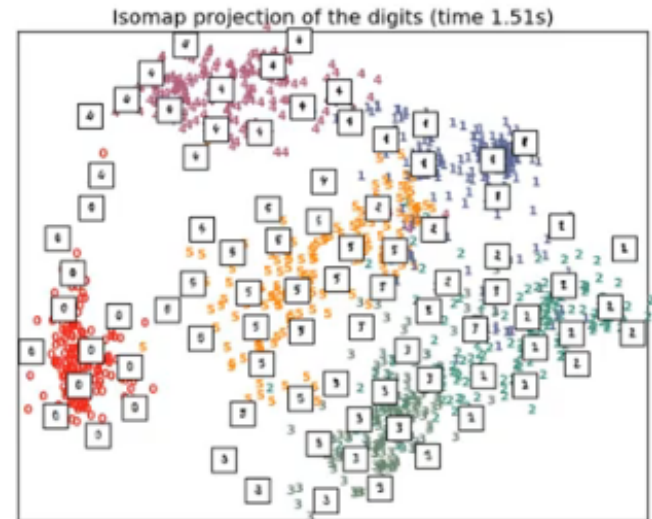
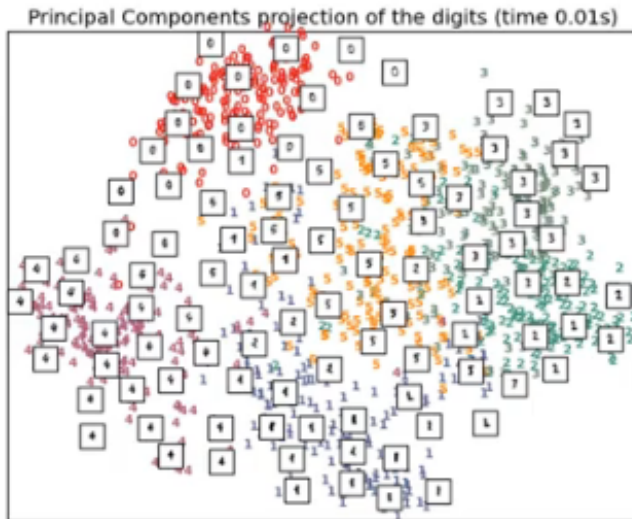
- 차원의 저주 (Curse of dimensionality)

- 일반적으로 intrinsic dimension은 original dimension 보다 상대적으로 작음

- 예시)
  - 고유차원
  - original dimension
  - MNIST 16x16 (256 dimensions) 데이터
  - PCA와 ISOMAP을 통한 2차원 데이터로 차원 축소



256차원 → 2차원



2차원이지만 대략적으로 MNIST 클래스의 형태가 유지됨

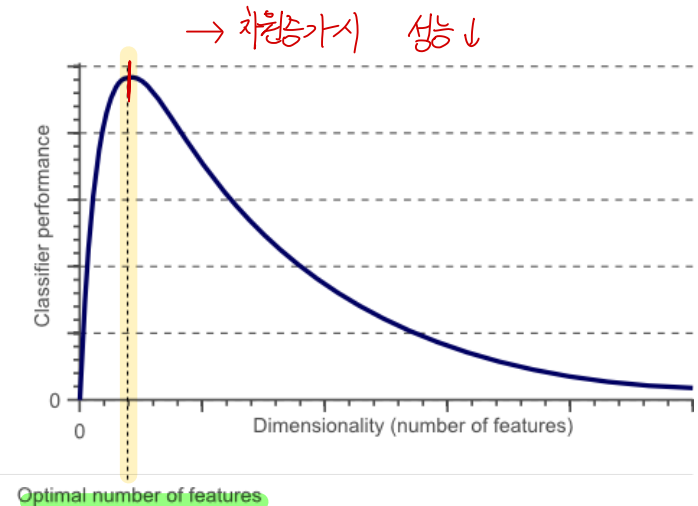
# 차원 축소 개요

- 차원의 저주 (Curse of dimensionality)

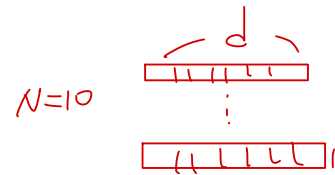
- 차원이 높을 수록 발생하는 문제
  - 데이터에 포함될 노이즈의 비율도 높아짐
    - 성능 감소를 야기함
  - 모델 학습과 추론의 계산 복잡도가 높아짐
  - 동일한 성능을 얻기 위해 더 많은 데이터의 수가 필요함

- 차원의 저주 해결 법

- 도메인 지식을 이용 → 중요한 특성만 사용
- 목적함수에 Regularization term 추가
- 차원 축소 기술을 전처리로 사용

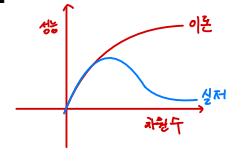


# 차원 축소 개요



## ■ 차원 축소 배경

- 이론적으로, 차원의 증가는 모델 성능을 향상 시킴
  - 가정: 모든 변수가 서로 독립일 경우 ✱
- 실제로, 차원의 증가는 모델 성능 저하를 가져옴
  - 모든 변수는 서로 상관관계가 있고, 노이즈가 존재함



## ■ 차원 축소 목적

- 모델의 성능을 최대로 해주는 변수의 일부 셋을 찾는 것

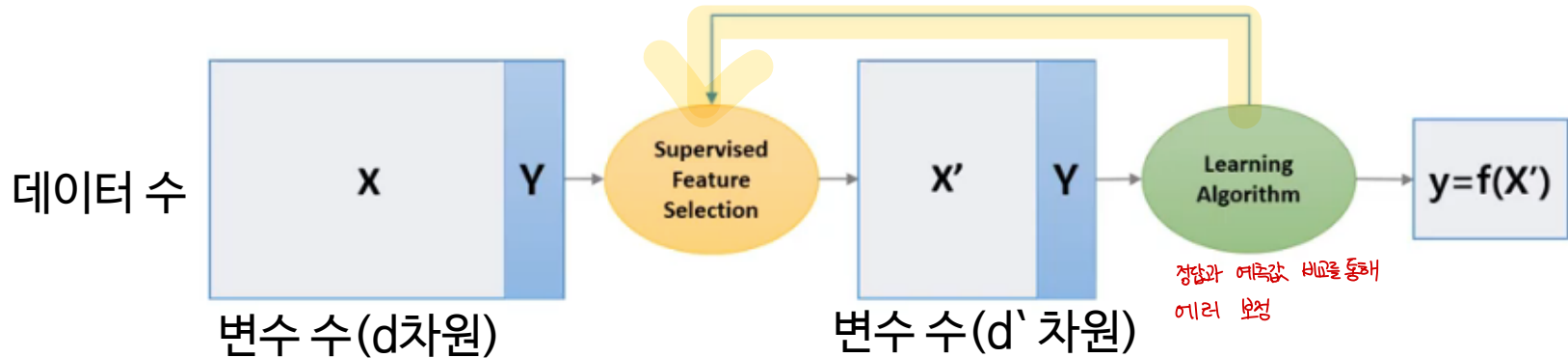
## ■ 차원 축소 효과

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

# 차원 축소 개요

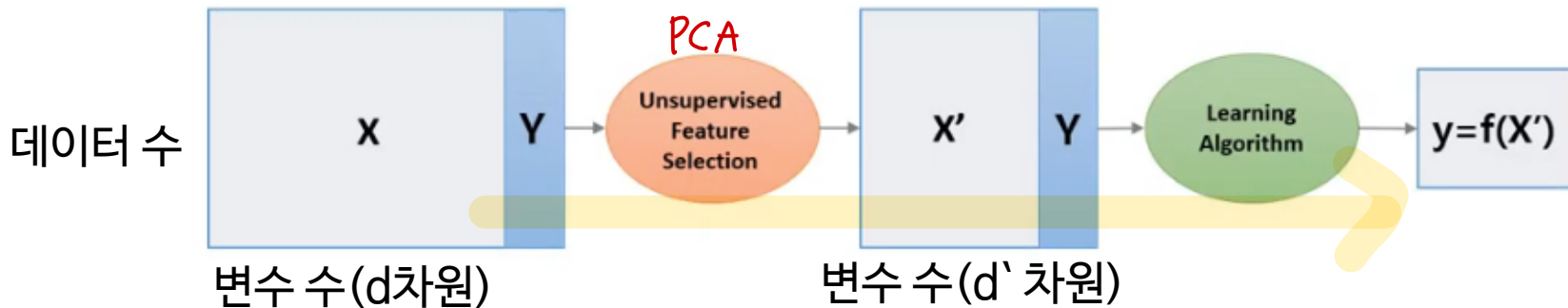
## 지도학습 기반 차원 축소

- 학습결과가 피드백 되어 Feature Selection 을 반복함



## 비지도학습 기반 차원 축소

- 지도학습 처럼 피드백을 통한 Feature Selection 반복 없음





# 차원 축소 개요

지도 / 비지도 ) 2x2 4가지 방법  
선택 / 추출

## ■ 차원 축소 방법

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

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

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

### ■ 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

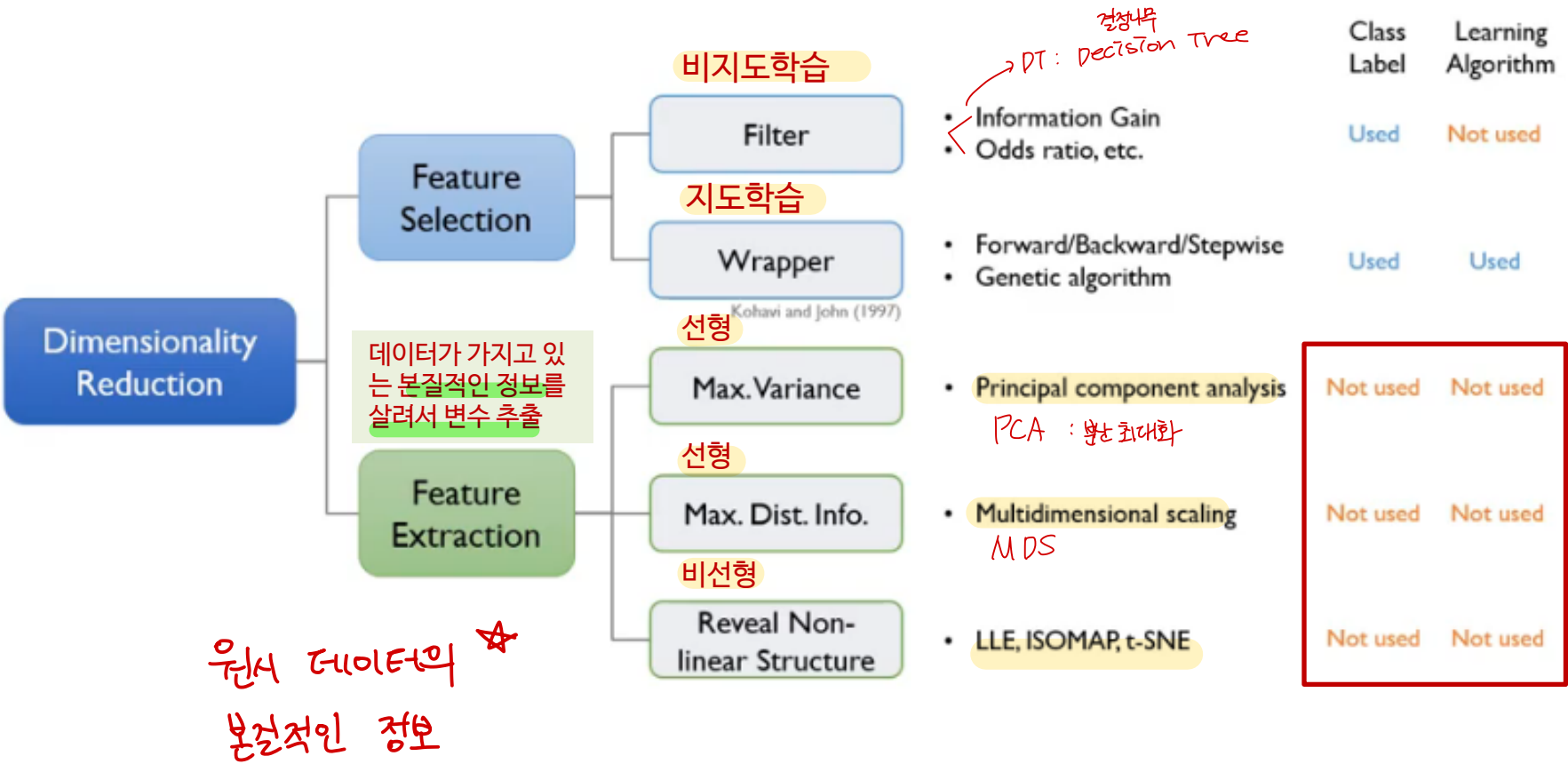
선형 결합

$$z_1 = a_{11}x_1 + a_{12}x_2 + a_{13}x_3 \dots a_{1n}x_{100}$$

$$z_2 = a_{21}x_1 + a_{22}x_2 + a_{23}x_3 \dots a_{2n}x_{100}$$

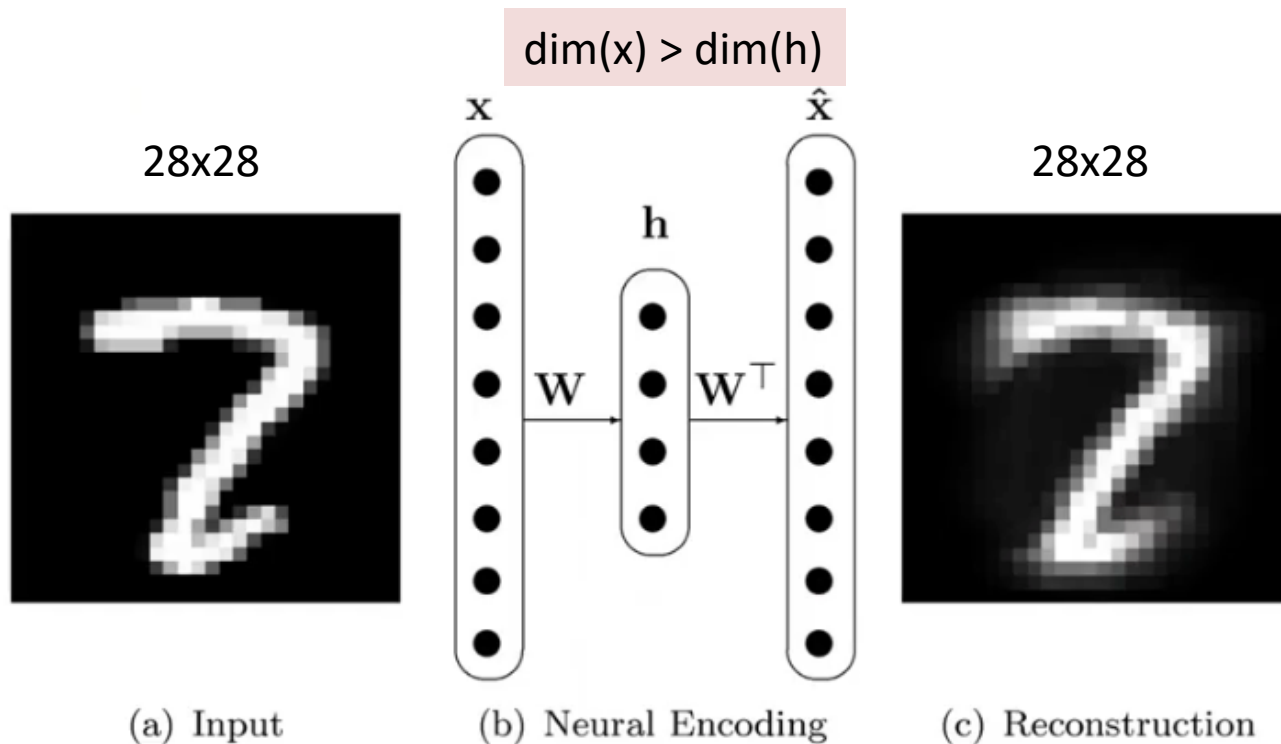
# 차원 축소 (Dimension Reduction)

## 차원 축소 방법 정리



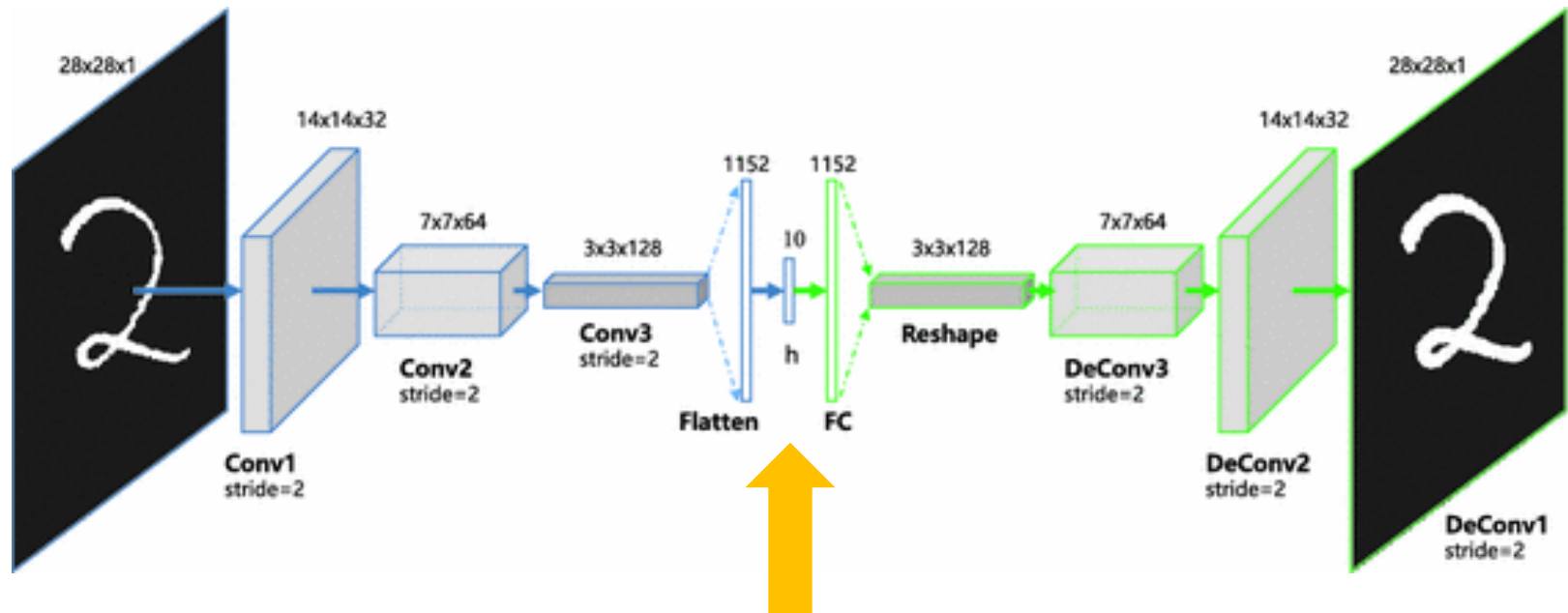
# 차원 축소 (Dimension Reduction)

- 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



# 차원 축소: PCA

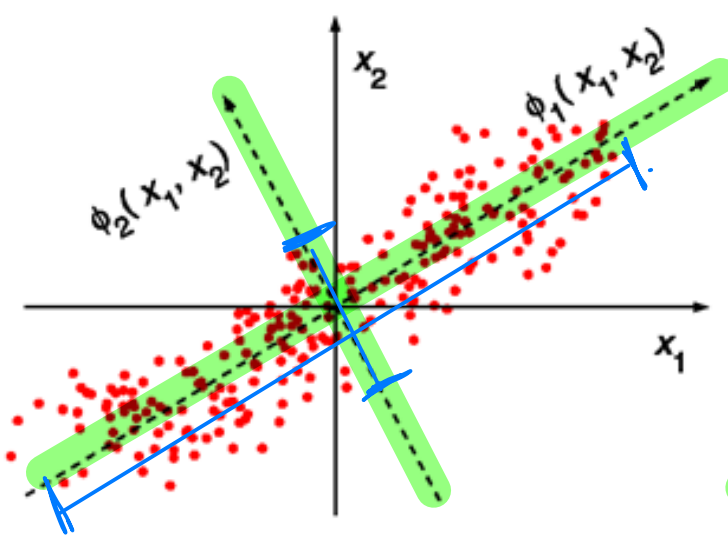
---

Dimension Reduction

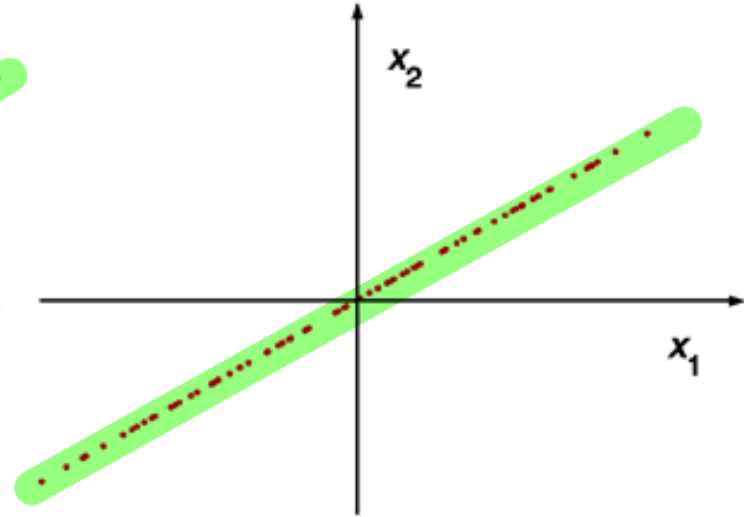
# 주성분 분석(PCA)

- 주성분 분석의 목적

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



(a) PCA basis



(b) PCA reduction to 1D

# 주성분 분석(PCA)

- MNIST의 예시

Feature  
→ 1/D vector

⇒ 2D, 3D

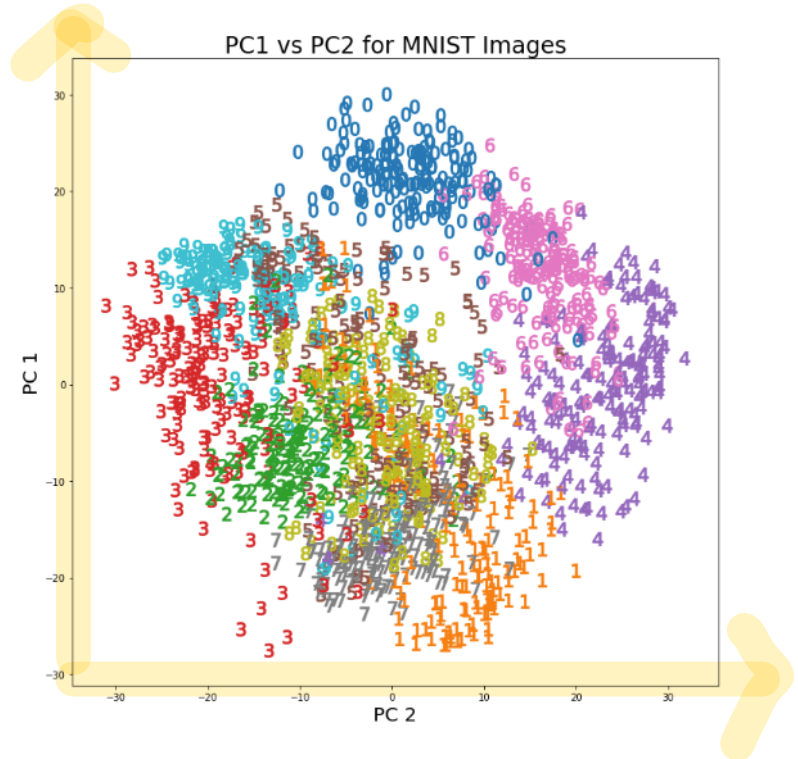
방법: t-SNE

$16 \times 16 = 256$

2D



MNIST

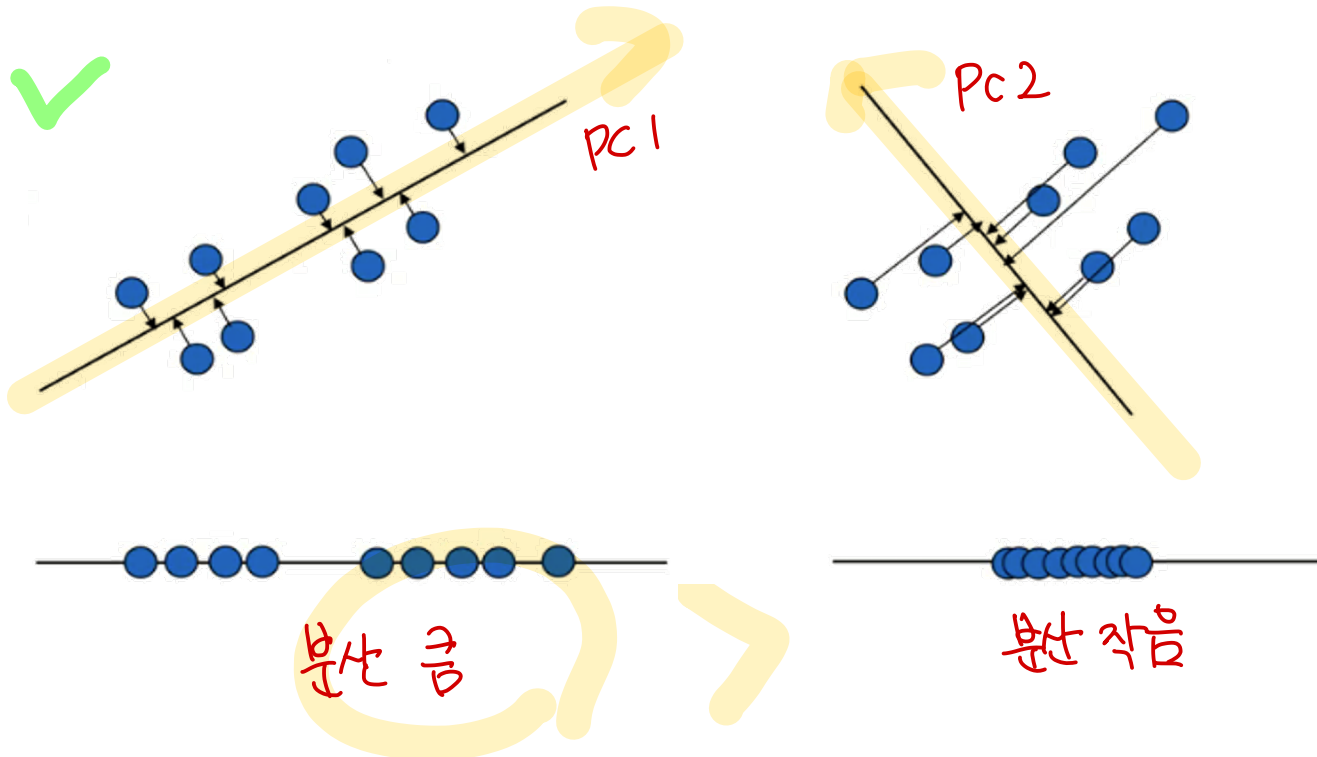


PCA

# 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 최대한 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

주성분 기저

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

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

⋮

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

$X_1, X_2, \dots, X_p$ : 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)

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

$$\underset{[d \times d]}{\text{Cov}(X)} = \frac{1}{n} \underset{[d \times n]}{(X - \bar{X})} \underset{[n \times d]}{(X - \bar{X})^T}$$

- 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)

$$A = \begin{bmatrix} 1 & 3 & 5 \\ 5 & 4 & 1 \\ 3 & 8 & 6 \end{bmatrix},$$

$$\text{Conv}(A) = \begin{bmatrix} 2.67 & 0.67 & -2.67 \\ 0.67 & 4.67 & 2.33 \\ -2.67 & 2.33 & 4.67 \end{bmatrix}$$

다 더하면 전체분산

# 주성분 분석(PCA): 수리적 배경

- 사영 (Projection)



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

$$\vec{x} = p\vec{a} = \frac{\vec{b}^T \vec{a}}{\vec{a}^T \vec{a}} \vec{a} = 1$$

If  $\vec{a}$  is unit vector


$$p = \vec{b}^T \vec{a} \Rightarrow \vec{x} = p\vec{a} = (\vec{b}^T \vec{a}) \vec{a}$$

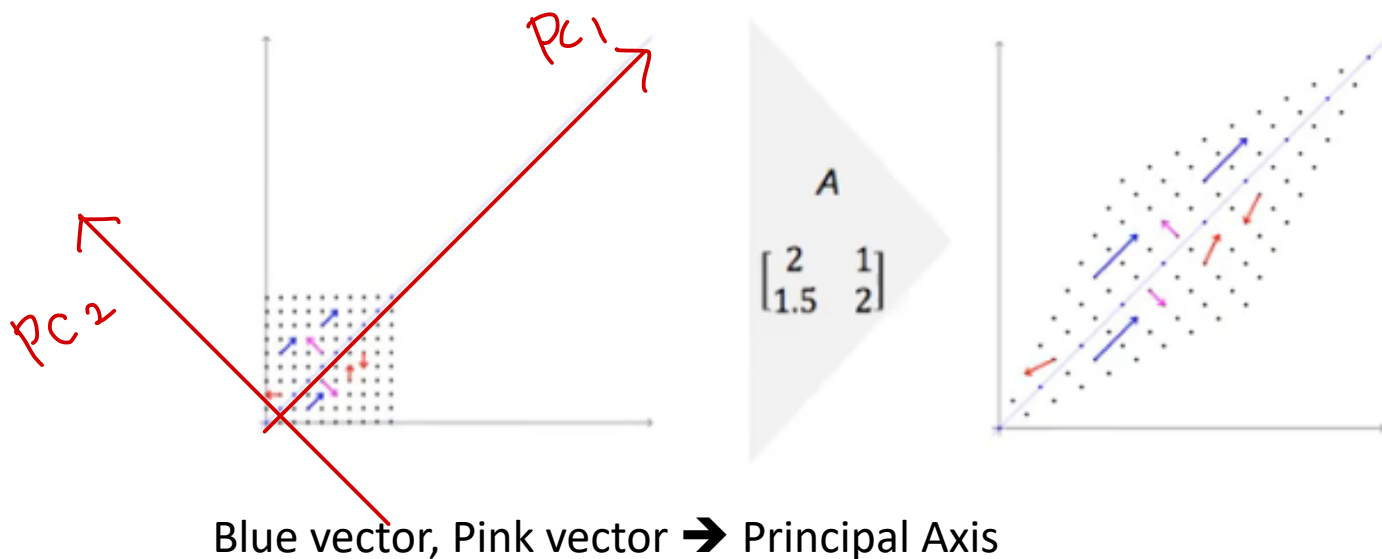
$\vec{x}$ : 사영후 벡터,  $p$ : 직교 사영을 위한 스케일러  
 $\vec{b}$  = 데이터,  $\vec{a}$  = 기저축 (PC)

# 주성분 분석(PCA): 수리적 배경

- 고유값(eigenvalue)과 고유벡터(eigenvector)

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

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



# 주성분 분석(PCA): 수리적 배경

- 고유값(eigenvalue)과 고유벡터(eigenvector)

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

- 행렬  $A$  가 Non-singular 하다면,  $d$ 개의 고유값과 고유벡터가 존재함
- 고유벡터는 서로 직교함(orthogonal)
- $\text{tr}(A) = \lambda_1 + \lambda_2 + \dots + \lambda_d$

PCA 주성분 서로 직교

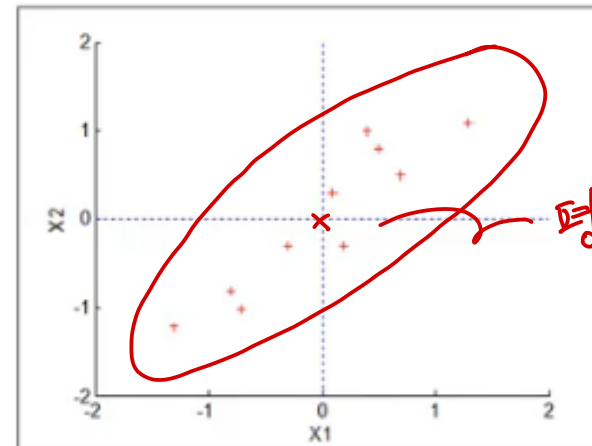
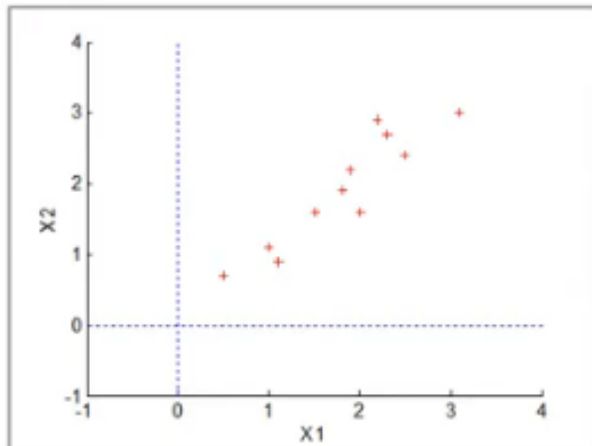
# 주성분 분석 알고리즘 PCA

- 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



# 주성분 분석 알고리즘

PCA 방향성 : 분산의 최대화  
↑  
사용된 데이터

## ▪ Step2: 최적화 문제 정의

- 데이터 X를 기저 벡터 W에 사영(projection)하면, 사용 후 분산은 다음과 같음

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

분산

- S는 X의 covariace matrix
- PCA의 목적은 사영 이후 분산 V를 최대화 하는 것

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

$$\text{s. t. } \mathbf{w}^T \mathbf{w} = 1 \quad \Leftarrow \quad \mathbf{w}^T \mathbf{w} = 1 \quad (\mathbf{w}^T \cdot \mathbf{w} = 1)$$

➡

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

# 주성분 분석 알고리즘

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

- Step3: 최적화 문제 솔루션
  - 라그랑지 멀티플라이어(Lagrangian multiplier) 적용

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

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

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

W는 S의 eigen vector  
λ는 S의 eigen value.

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

S의 W

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

S의 λ



# 주성분 분석 알고리즘

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

분산이 크나 = λ가 크나 = eigenvalue가 크나

## Step4: 주축 정렬

- Eigenvalue에 해당되는 eigenvectors를 순서대로 정렬

$$\text{Eigenvectors} = \begin{bmatrix} 0.6779 & -0.7352 \\ 0.7352 & 0.6779 \end{bmatrix} \quad \text{Eigenvalues} = (1.2840 \quad 0.0491)$$

제일큰거

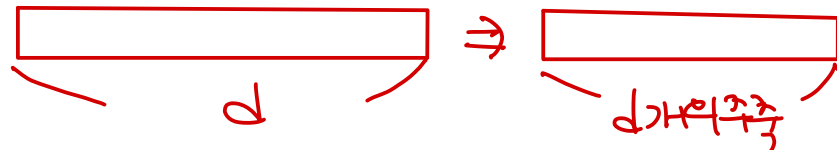
- W<sub>1</sub>은 Eigen vector이고 λ<sub>1</sub>은 대응되는 eigen value이다.
- (아래 식의 유도에 의하면) W<sub>1</sub>에 사용된 데이터의 분산은 λ<sub>1</sub>이다.

basis 중 하나 (Pc1, Pc2..)

$$\mathbf{v} = (\mathbf{w}_1^T \mathbf{X})(\mathbf{w}_1^T \mathbf{X})^T = \mathbf{w}_1^T \mathbf{X} \mathbf{X}^T \mathbf{w}_1 = \mathbf{w}_1^T \mathbf{S} \mathbf{w}_1$$

$$\text{Since } \mathbf{S} \mathbf{w}_1 = \lambda_1 \mathbf{w}_1, \quad \mathbf{w}_1^T \mathbf{S} \mathbf{w}_1 = \mathbf{w}_1^T \lambda_1 \mathbf{w}_1 = \lambda_1 \mathbf{w}_1^T \mathbf{w}_1 = \lambda_1$$

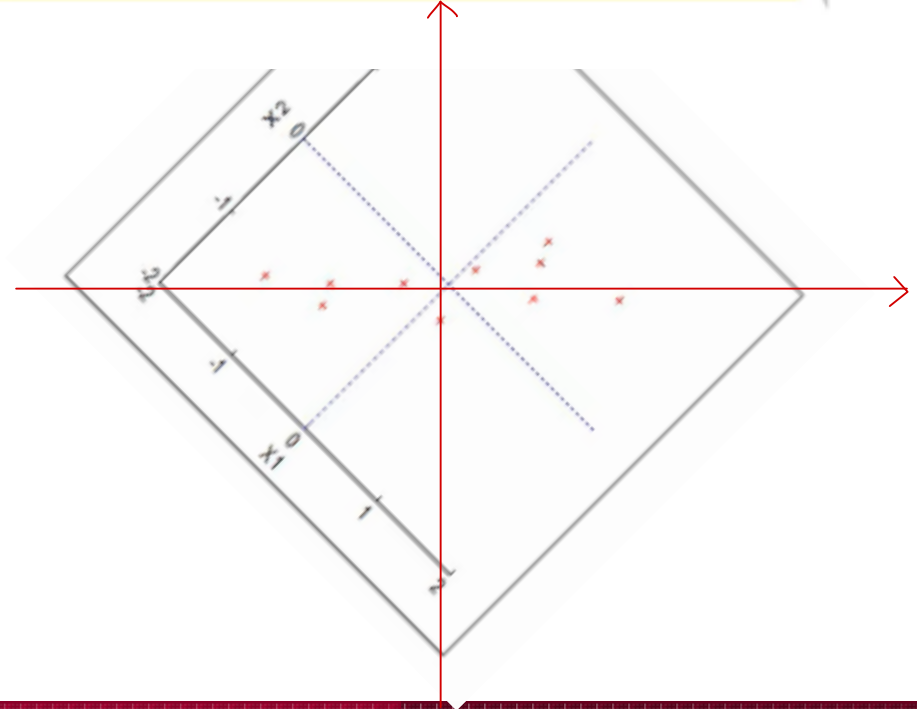
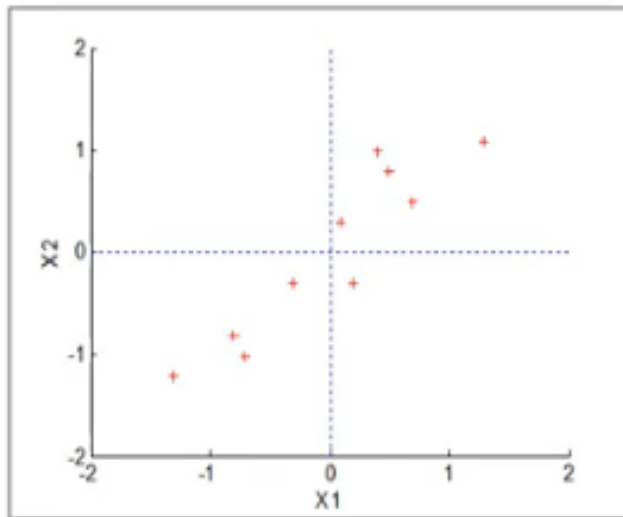
- 첫번째 주성분으로 설명가능한 데이터 비율  $= \frac{\lambda_1}{\lambda_1 + \lambda_2} = \frac{1.2840}{1.2840 + 0.0491} = 0.96$



# 주성분 분석 알고리즘

- Step5: PCA로 변환된 데이터
  - Principle Component 1 =  $Z_1 = W_1^T X$   
주축, 변환된 데이터

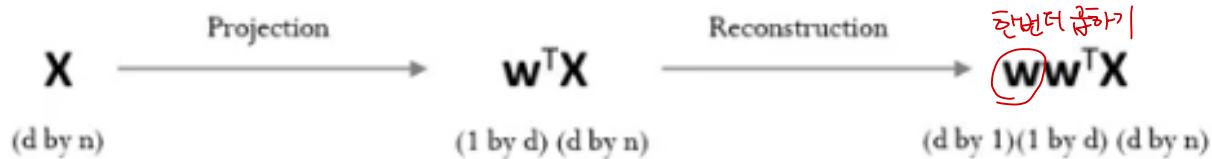
$x_1$	0.69	-1.31	0.39	0.09	1.29	0.49	0.19	-0.81	-0.31	-0.71
$x_2$	0.49	-1.21	0.99	0.29	1.09	0.79	-0.31	-0.81	-0.31	-1.01
$z_1$	0.83	-1.78	0.99	0.27	1.68	0.91	-0.10	-1.14	-0.44	-1.22



# 주성분 분석 알고리즘

PCA 압축 → 복원

- Step6: 원데이터로 복원 *Loss 발생*
  - 2D 데이터 → 1D PCA → 2D 데이터 복원(reconstruction)



$x_1$	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

= 몇차원으로 줄일 것인가?

## ■ 주성분 개수 선정 법 → 몇개의 주성분을 사용해야 할까?

### ■ 선택 방법 #1

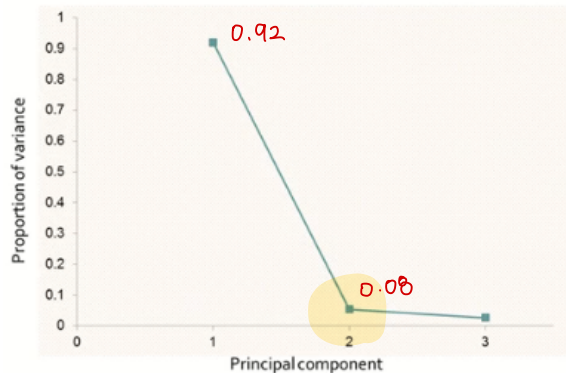
- 고유 값 감소율이 유의미하게 낮아지는 Elbow Point에 해당하는 주성분을 선택

### ■ 선택 방법 #2

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

$PC_1$	$PC_2$	$PC_3$	$PC_4$	...
10	10	30	20	...

→ 4개 선택



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

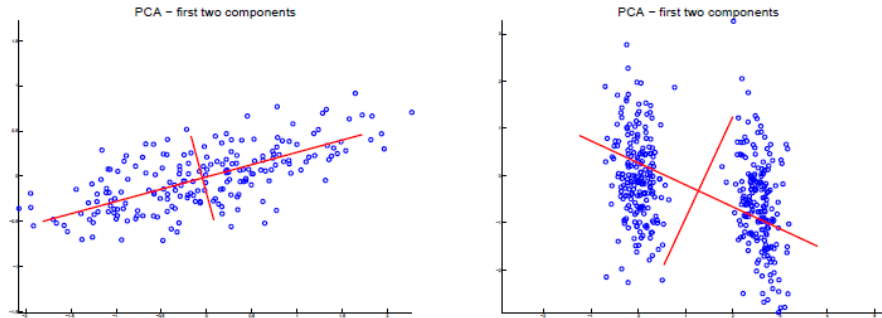
- 몇개의 주성분을 사용해야 할까? 5개
  - 조건: 전체 분산의 80%를 포함해야 함
- 솔루션
  - (위) 원 데이터, (아래) PCA 결과
  - PCA결과의 누적분산비를 살펴보면  
PC1~PC5까지 선택해야 전체 데이터  
분산의 82%에 도달할 수 있음

시리얼 이름	제조업체명	유형	칼로리	단백질	지방	나트륨	식이섬유	복합탄수화물	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		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

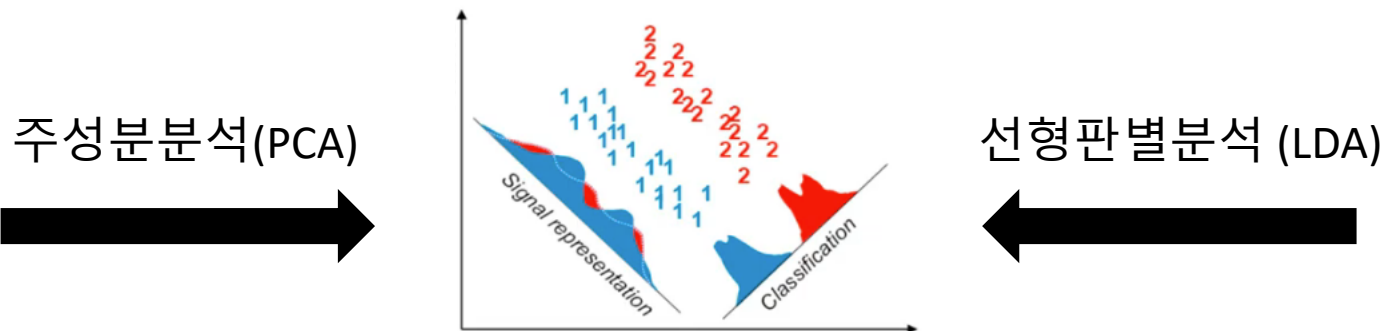
변수이름	PC1 1	PC2 2	PC3 3	4	5	6	7
calories	0.2995424	0.39314792	0.11485746	0.20435865	0.20389892	-0.25590625	-0.02559552
protein	-0.30735639	0.16532333	0.27728197	0.30074316	0.319749	0.120752	0.28270504
fat	0.03991544	0.34572428	-0.20489009	0.18683317	0.58699332	0.34796733	-0.05115468
sodium	0.18339655	0.13722059	0.38943109	0.12033724	-0.33836424	0.66437215	-0.28370309
fiber	-0.45349041	0.17981192	0.06976604	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	0.06752024	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
분산비(%)	27.95081329	24.21580505	14.6873045	7.84212446	7.61045933	5.55432129	5.16551113
누적분산비(%)	27.95081329	52.16661835	66.85391988	74.69604492	82.3065033	87.86082458	93.02633667

# 주성분 분석 한계

- 한계점 1 *only 단일 가우시안 unit model*
  - 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

## ■ 랜덤 PCA의 개념

- 자료의 크기 또는 특성변수의 크기가 매우 크면 주성분  $W$  를 구하기 위한 SVD 계산이 불가능하거나 시간이 많이 소요됨
- 이런 경우 Randomized PCA 가 유용
- Randomized PCA는 QR 분해를 이용하여 행렬의 SVD를 수행함

## ■ 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

