

12기 정규세션

ToBig's 11기 임채빈

Dimensionality Reduction

Content

Unit 01 | intro : Dimensionality Reduction

Unit 02 | Eigen-value Decomposition

Unit 03 | PCA : Principal Component Analysis

Unit 04 | LDA : Linear Discriminant Analysis

Unit 05 | T-SNE : T – Stochastic Neighbor Embedding

Unit 01 | intro : Dimensionality Reduction

■ 차원 Dimension

: 공간 내의 있는 점 등의 위치를 나타내기 위해 필요한 축의 개수

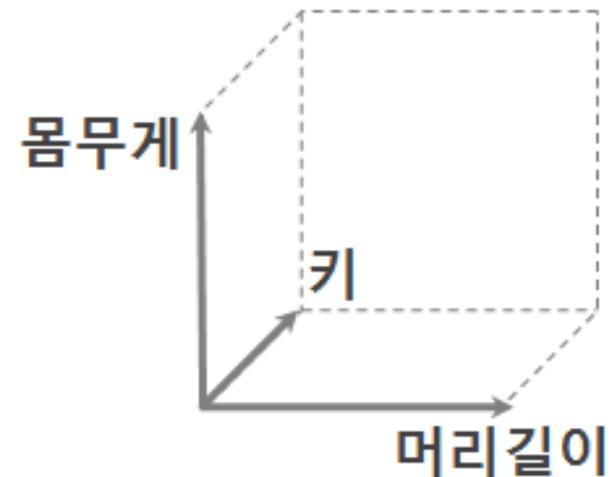
Unit 01 | intro : Dimensionality Reduction

■ 차원 Dimension

: 공간 내의 있는 점 등의 위치를 나타내기 위해 필요한 **축의 개수**

■ 한 데이터가 n개의 설명변수 x 를 가진다면 n차원의 좌표 상에 표현 할 수 있다.

ex) 사람을 키와 몸무게, 머리길이로 표현한 데이터를 가지고 있다면
3차원 좌표 상에 표현 할 수 있다.

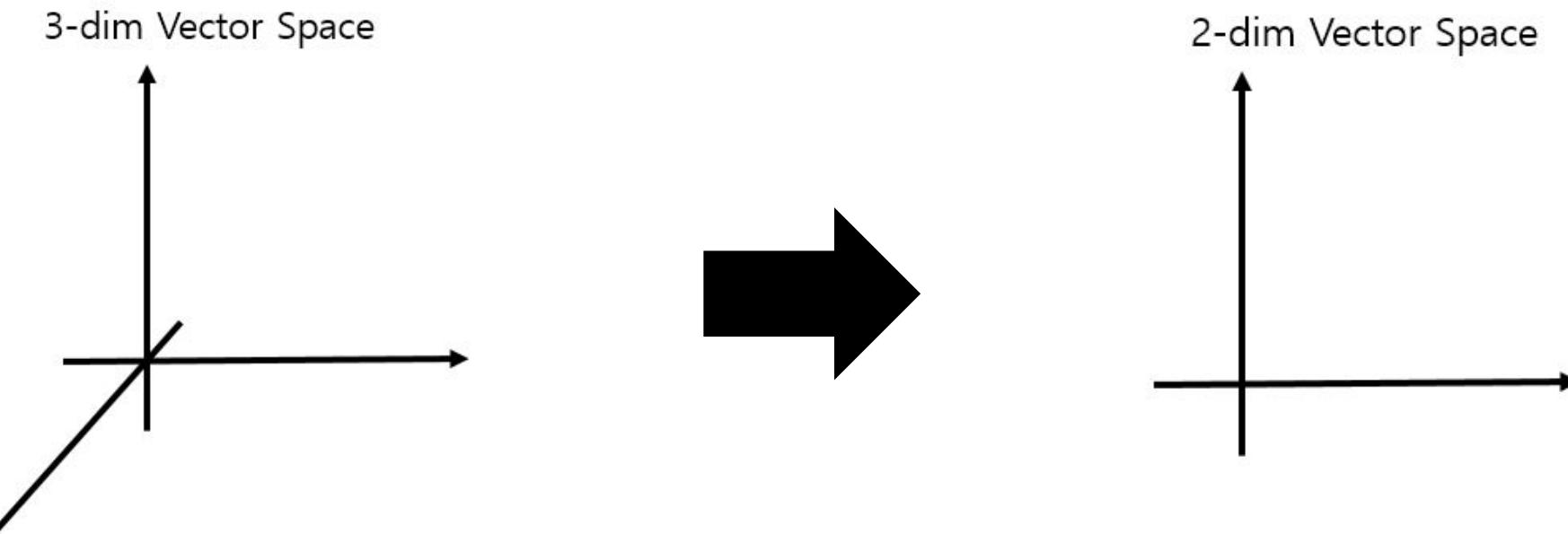


Unit 01 | intro : Dimensionality Reduction

■ 차원축소 Dimensionality Reduction

: 위치를 표현하기 위해 필요한 **축의 개수를 줄이는 것**.

ex) 3차원 \rightarrow 2차원



Unit 01 | intro : Dimensionality Reduction

- 차원의 저주 Curse of Dimensionality

- : 변수가 늘어나고 차원이 커지면서 발생하는 문제

Unit 01 | intro : Dimensionality Reduction

■ 차원의 저주 Curse of Dimensionality

: 변수가 늘어나고 차원이 커지면서 발생하는 문제

- 필요한 데이터 수의 지수함수적 증식
-> 연산량 급증

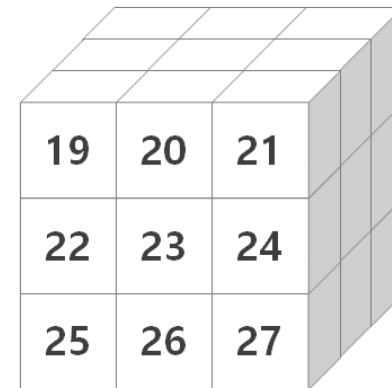
ex) 각 변수가 가질 수 있는 값이 3개일 때

1	2	3
---	---	---

1차원 : 3개

1	2	3
4	5	6
7	8	9

2차원 : 9개



3차원 : 27개



N차원 : 3의 n승 개

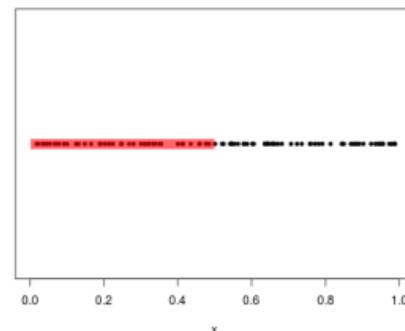
Unit 01 | intro : Dimensionality Reduction

■ 차원의 저주 Curse of Dimensionality

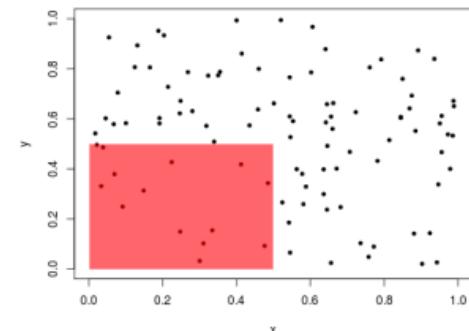
: 변수가 늘어나고 차원이 커지면서 발생하는 문제

- 필요한 데이터 수의 지수함수적 증식
-> 연산량 급증
- 정보의 밀도 감소

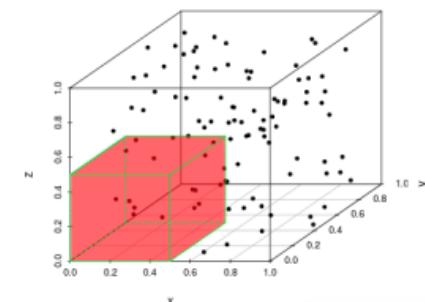
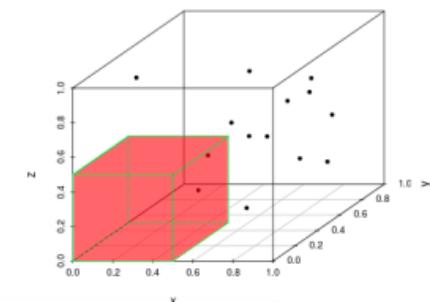
1-D: 42% of data captured.



2-D: 14% of data captured.



3-D: 7% of data captured.

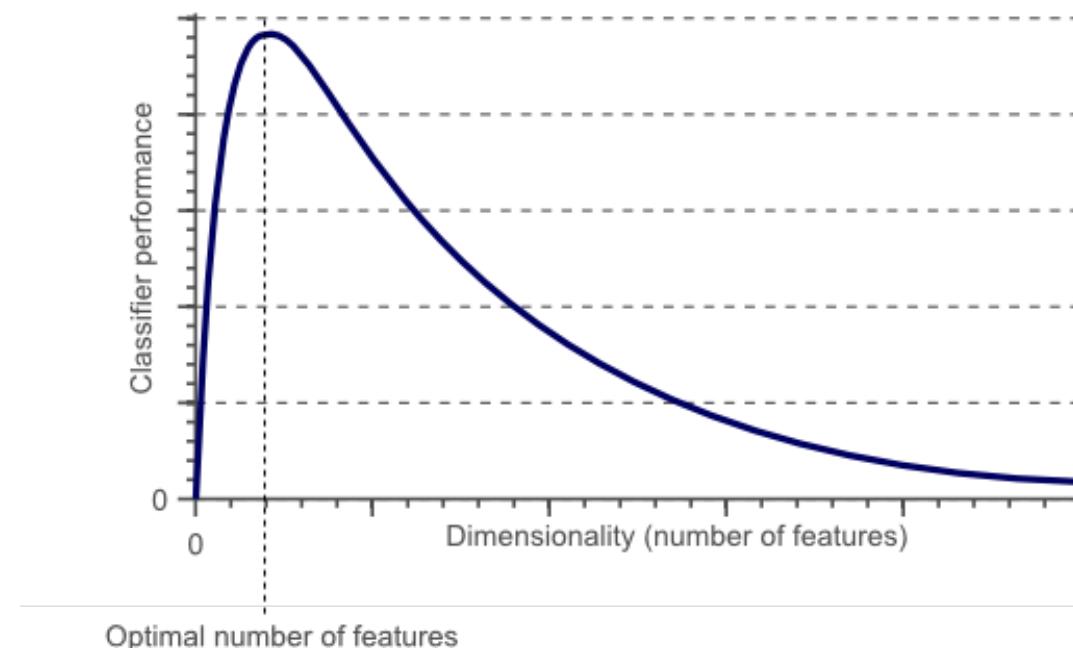
4-D: 3% of data captured.
t = 0

Unit 01 | intro : Dimensionality Reduction

■ 차원의 저주 Curse of Dimensionality

: 변수가 늘어나고 차원이 커지면서 발생하는 문제

- 필요한 데이터 수의 지수함수적 증식
-> 연산량 급증
- 정보의 밀도 감소
- 공간을 설명하기 위한 데이터의 부족
->**과적합**의 문제 & 성능 감소



Unit 01 | intro : Dimensionality Reduction

■ 차원축소 Dimensionality Reduction

: 관찰 대상을 잘 설명할 수 있는 **잠재 공간**(Latent space)은
실제 관찰되어지는 공간(observation space)보다 작을 수 있다.

■ 관찰 공간의 샘플들을 기반으로 잠재 공간을 파악하는 것

- 차원의 저주 해결
- 연산량 감소
- 시각화 용이

Unit 01 | intro : Dimensionality Reduction

■ 차원축소의 방법

1. 변수 선택 Feature selection

: 원본 데이터의 불필요한 특징 제거

ex)

몸무게 변수 불필요하다고 판단

나머지 변수만 선택

키, ~~몸무게~~, 머리 길이 -> 키, 머리 길이

2. 변수 추출 Feature selection

: 원본 데이터의 특징들의 조합으로
새로운 특징을 생성

ex)

키와 몸무게가 독립이 아니라고 판단

체구이라는 파생변수 생성

체구 = 0.2 * 키 + 0.8 * 몸무게

키, 몸무게, 머리 길이 -> 체구, 머리 길이

new

Unit 01 | intro : Dimensionality Reduction

■ 차원축소의 방법

1. 변수 선택 Feature selection

: 원본 데이터의 불필요한 특징 제거

Lasso

2. 변수 추출 Feature selection

: 원본 데이터의 특징들의 조합으로
새로운 특징을 생성

PCA, LDA, T-SNE

Unit 02 | Eigen-value Decomposition

- Eigen value and eigen vector 고유값과 고유벡터
 - Eigen vector 고유벡터
 - $n \times n$ 정방행렬 A 에 대해 $Av = \lambda v$ 를 만족하는 0 이 아닌 열 벡터
 - Eigen value 고유값
 - Eigen vector의 상수 λ 값

Unit 02 | Eigen-value Decomposition

- Eigen value and eigen vector 고유값과 고유벡터
 - Eigen vector 고유벡터
 - $n \times n$ 정방행렬 A 에 대해 $Av = \lambda v$ 를 만족하는 0이 아닌 열 벡터
: 선형 변환 A 에 의한 변환 결과가 자기 자신의 λ 배(상수배)가 되는 0이 아닌 벡터
 - Eigen value 고유값
 - Eigen vector의 상수 λ 값
: 선형 변환 A 에 의해 벡터의 크기가 변하는 정도

Unit 02 | Eigen-value Decomposition

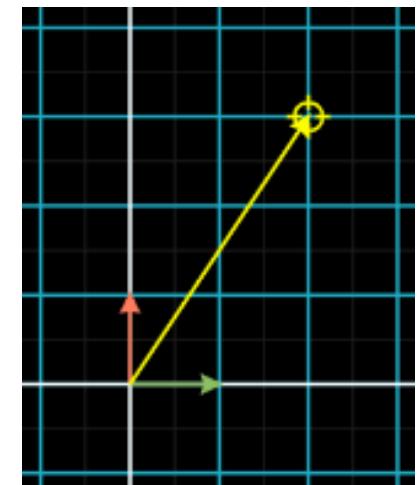
■ Linear transform 선형 변환

■ Basis 기저 (기준)

- 벡터 공간 V가 있을 때, 이 벡터 공간을 표현할 수 있는 **최소한의 단위 벡터**

ex) xy 평면의 기저 = $\begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, xyz 공간의 기저 = $\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$

xy 평면 상에 벡터 $\begin{bmatrix} 2 \\ 3 \end{bmatrix}$ 을 나타내 본다면, $\begin{bmatrix} 2 \\ 3 \end{bmatrix} = 2 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + 3 \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 3 \end{bmatrix}$



Unit 02 | Eigen-value Decomposition

■ Linear transform 선형 변환

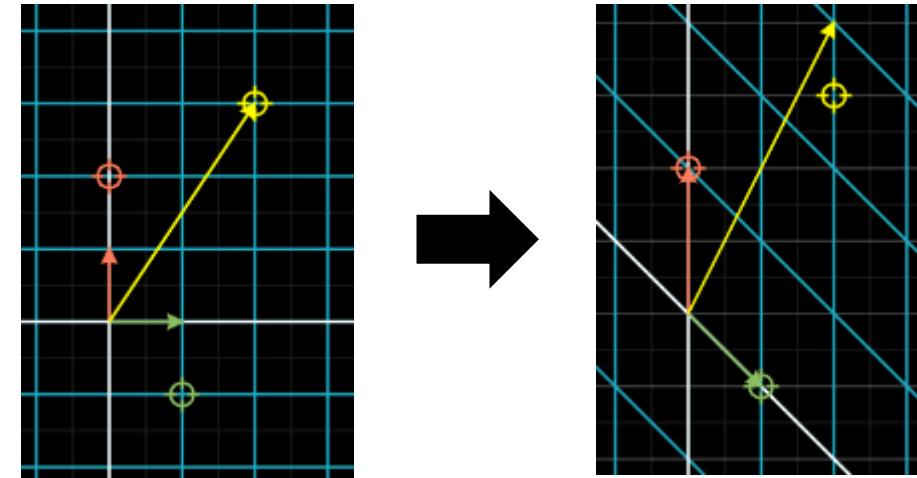
■ Basis transform 기저 변환

- 벡터 공간을 표현할 수 있는 최소한의 단위인 기저를 변경

ex) xy 평면의 기저 $\begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ 를 $\begin{bmatrix} 1 \\ -1 \end{bmatrix}, \begin{bmatrix} 0 \\ 2 \end{bmatrix}$ 로 변경

xy 평면 상에 벡터 $\begin{bmatrix} 2 \\ 3 \end{bmatrix}$ 을 나타내 본다면, $2\begin{bmatrix} 1 \\ -1 \end{bmatrix} + 3\begin{bmatrix} 0 \\ 2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -1 & 2 \end{bmatrix} \begin{bmatrix} 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \end{bmatrix}$

이 때, 벡터 $\begin{bmatrix} 2 \\ 4 \end{bmatrix}$ 는 정방행렬 $A \begin{bmatrix} 1 & 0 \\ -1 & 2 \end{bmatrix}$ 에 의해 벡터 $\begin{bmatrix} 2 \\ 3 \end{bmatrix}$ 가 선형 변환된 벡터이다.



Unit 02 | Eigen-value Decomposition

- Eigen value and eigen vector 고유값과 고유벡터
 - Eigen vector 고유벡터
 - $n \times n$ 정방행렬 A 에 대해 $Av = \lambda v$ 를 만족하는 0이 아닌 열 벡터
: 선형 변환 A 에 의한 변환 결과가 자기 자신의 λ 배(상수배)가 되는 0이 아닌 벡터
 - Eigen value 고유값
 - Eigen vector의 상수 λ 값
: 선형 변환 A 에 의해 벡터의 크기가 변하는 정도

Unit 02 | Eigen-value Decomposition

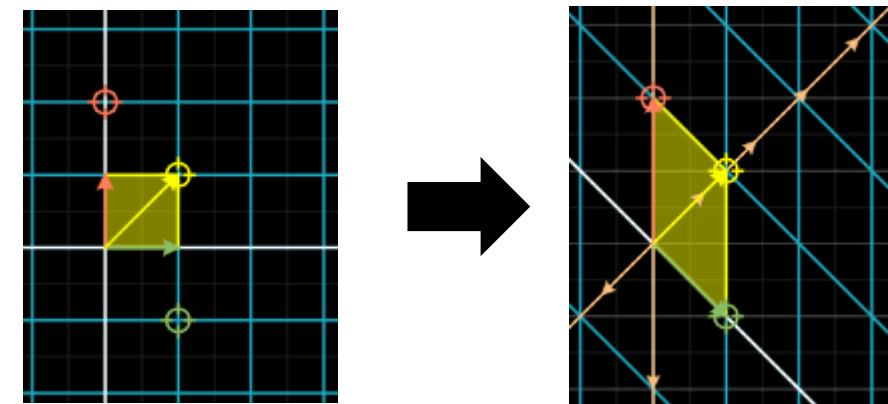
■ Linear transform 선형 변환

- 벡터 $\begin{bmatrix} 2 \\ 4 \end{bmatrix}$ 는 정방행렬 $A \begin{bmatrix} 1 & 0 \\ -1 & 2 \end{bmatrix}$ 에 의해 벡터 $\begin{bmatrix} 2 \\ 3 \end{bmatrix}$ 가 선형 변환된 벡터이다.

$\begin{bmatrix} 2 \\ 4 \end{bmatrix} \neq \lambda \begin{bmatrix} 2 \\ 3 \end{bmatrix}$, 상수배가 성립하지 않음.

- 벡터 $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$ 은?

$\begin{bmatrix} 1 & 0 \\ -1 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \lambda \begin{bmatrix} 1 \\ 1 \end{bmatrix}$, 상수($\lambda = 1$)배가 성립함.

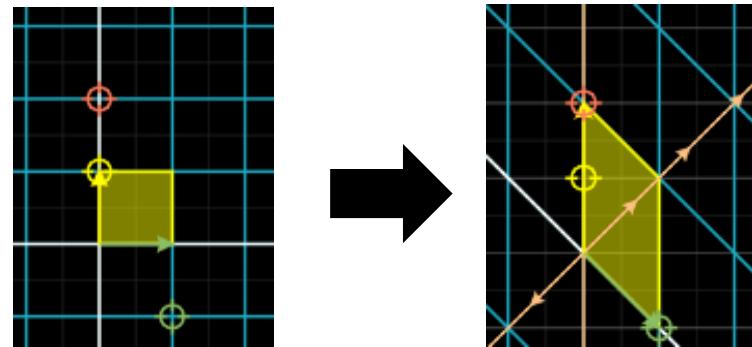


Unit 02 | Eigen-value Decomposition

■ Linear transform 선형 변환

- 벡터 $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$ 은?

$$\begin{bmatrix} 1 & 0 \\ -1 & 2 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 2 \end{bmatrix} = \lambda \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \text{ 상수 } (\lambda = 2) \text{ 배가 성립함.}$$



- 그러므로 정방행렬 $A = \begin{bmatrix} 1 & 0 \\ -1 & 2 \end{bmatrix}$ 는 고유값 1과 고유벡터 $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$, 고유값 2와 고유벡터 $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$ 을 가진다.

Unit 02 | Eigen-value Decomposition

■ Linear transform 선형 변환

■ 단위벡터의 정규화

고유 벡터에 실수를 곱한 벡터, 즉 방향이 같은 벡터는 모두 고유 벡터가 된다.

즉, $\begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 2 \\ 2 \end{bmatrix}, \begin{bmatrix} 3 \\ 3 \end{bmatrix} \dots$ 모두 $\begin{bmatrix} 1 & 0 \\ -1 & 2 \end{bmatrix}$ 의 고유 벡터가 된다.

그러므로 고유벡터를 표시할 때는 유클리드 거리가 1인 단위벡터가 되도록 정규화를 한다.

최종적으로 첫번째 고유값 1과 고유벡터 $\begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$, 두번째 고유값 2와 고유벡터 $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$ 를 가지게 된다.

Unit 02 | Eigen-value Decomposition

■ 수식적 접근

■ Eigen decomposition 고유값 분해

$$Av = \lambda v$$

$$(A - \lambda)v = (A - \lambda I)v = 0$$

A는 행렬, λ 는 상수 이기 때문에
 λ 에 단위행렬 $I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ 를 곱해서 행렬 꼴로 만든다.

■ Ex) $A = \begin{bmatrix} 1 & 0 \\ -1 & 2 \end{bmatrix}$

$$\begin{bmatrix} 1 - \lambda & 0 \\ -1 & 2 - \lambda \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

행렬식을 통하여 계산

Unit 02 | Eigen-value Decomposition

■ 수식적 접근

■ 행렬식 determinant

$$\begin{bmatrix} a - \lambda & b \\ c & d - \lambda \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

우변이 영벡터라면 좌변이 영행렬이 되기 위해선

$\begin{bmatrix} a - \lambda & b \\ c & d - \lambda \end{bmatrix}$ 가 full rank가 아니어야 함.

: 역행렬은 존재하지 않는다.

= 행렬식이 0이다.

Rank : 행렬 내의 선형독립인 행 혹은 열의 개수

Full rank : 행렬 내의 모든 행 혹은 열이 선형독립

역행렬 공식

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{ad-bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

행렬식

Unit 02 | Eigen-value Decomposition

■ 수식적 접근

■ Eigen decomposition 고유값 분해

$$Av = \lambda v$$

$$(A - \lambda)v = (A - \lambda I)v = 0$$

A 는 행렬, λ 는 상수 이기 때문에
 λ 에 단위행렬 $I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ 를 곱해서 행렬 꼴로 만든다.

■ Ex) $A = \begin{bmatrix} 1 & 0 \\ -1 & 2 \end{bmatrix}$

$$\begin{bmatrix} 1 - \lambda & 0 \\ -1 & 2 - \lambda \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \text{행렬식을 통하여 계산}$$

$$\begin{vmatrix} 1 - \lambda & 0 \\ -1 & 2 - \lambda \end{vmatrix} = (1 - \lambda)(2 - \lambda) - 0 * -1 = (1 - \lambda)(2 - \lambda) = 0$$

첫번째 고유값 1, 두번째 고유값 2

Unit 02 | Eigen-value Decomposition

■ 수식적 접근

i) $\lambda = 1$

$$\begin{bmatrix} 0 & 0 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\begin{cases} 0v_1 + 0v_2 = 0 \\ -v_1 + v_2 = 0 \end{cases} \rightarrow v_1 = v_2$$

$$v = \begin{bmatrix} v_1 \\ v_1 \end{bmatrix}$$

$$e = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

ii) $\lambda = 2$

$$\begin{bmatrix} -1 & 0 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\begin{cases} -v_1 + 0v_2 = 0 \\ -v_1 + 0v_2 = 0 \end{cases} \rightarrow v_1 = 0$$

$$v = \begin{bmatrix} 0 \\ v_2 \end{bmatrix}$$

$$e = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

정규화된 고유벡터 e로 표현

Unit 02 | Eigen-value Decomposition

Spectrum Decomposition 스펙트럼 분해

- n차 대칭행렬 A는 n개의 고유값과 n개의 정규화된 고유벡터로 표시할 수 있다.

$$\begin{aligned}
 A &= \lambda_1 * e_1 * e_1^{\top} + \lambda_2 * e_2 * e_2^{\top} + \cdots + \lambda_n * e_n * e_n^{\top} \\
 &= \sum \lambda_i * e_i * e_i^{\top} = [e_1 \ e_2 \ \cdots \ e_n] \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{bmatrix} \begin{bmatrix} e_1^{\top} \\ e_2^{\top} \\ \vdots \\ e_n^{\top} \end{bmatrix} \\
 &= P \Lambda P^{\top} \quad (\text{단, } PP^{\top} = P^{\top}P = I)
 \end{aligned}$$

P : 고유 벡터의 열로 구성된 행렬

Λ : 고유값을 대각원소로 하는 대각행렬

- 대칭행렬의 고유벡터들의 내적은 0이다. 즉, 서로 직교한다.

n차 양정치행렬 A의 성질

$$\begin{aligned}
 \text{tr}(A) &= \text{tr}(P \Lambda P^{\top}) = \text{tr}(P^{\top} P \Lambda) \\
 &= \text{tr}(\Lambda) = \sum \lambda_i
 \end{aligned}$$

양의 정부호 행렬 = 양정치행렬

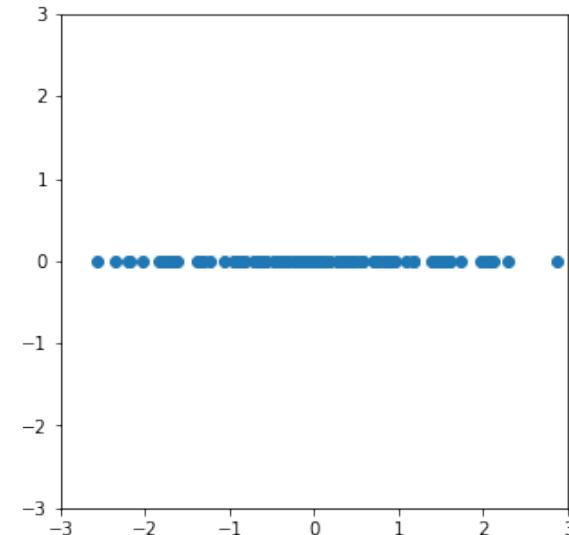
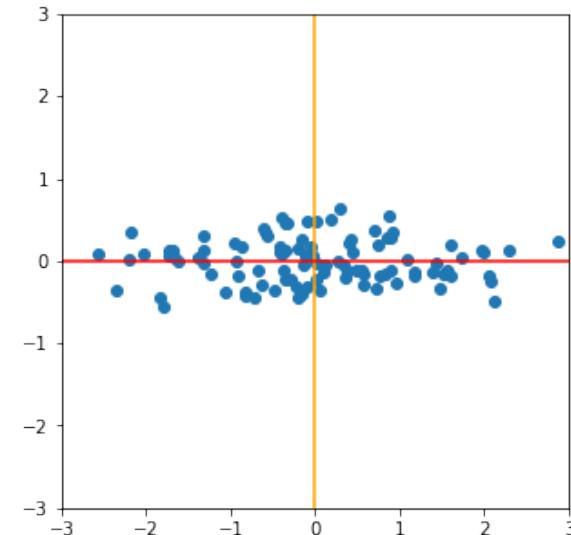
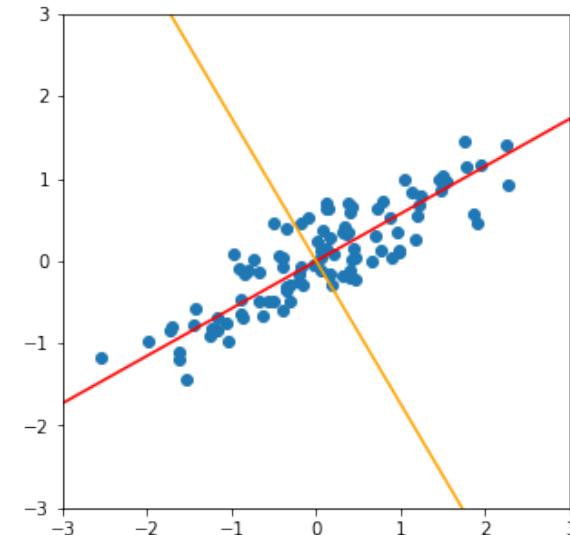
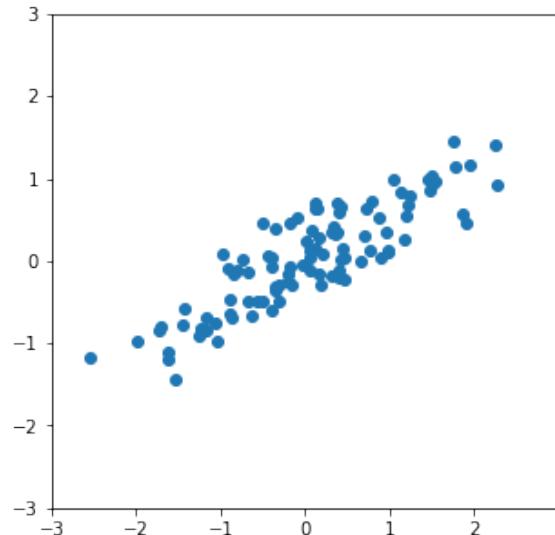
Tr : trace, 대각원소의 합

공분산행렬은 언제나 양정치행렬

Unit 03 | PCA : Principal Component Analysis

■ Principal Component Analysis 주성분 분석

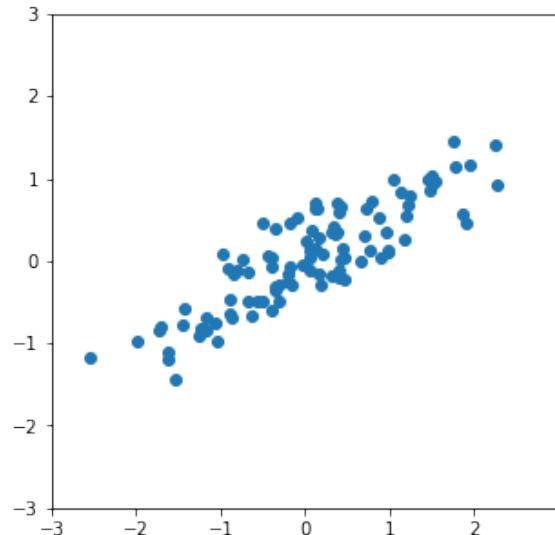
- 변수들의 전체분산 대부분을 소수의 주성분을 통하여 설명하는 것.
 - 고차원 데이터의 최대 분산 방향을 찾아 새로운 공간에 저차원으로 투영하는 것
 - 기하학적 측면에서 좌표축을 회전시켜 얻어진 새로운 좌표축 선택



Unit 03 | PCA : Principal Component Analysis

■ Principal Component Analysis 주성분 분석

- Covariance Matrix 공분산 행렬
- $\Sigma = \begin{bmatrix} 1.026 & 0.548 \\ 0.548 & 0.389 \end{bmatrix}$
- 대각원소의 k번째 원소는 변수 k의 분산
- k행 j열의 원소는 변수 k와 변수 j의 공분산으로 이루어진 행렬

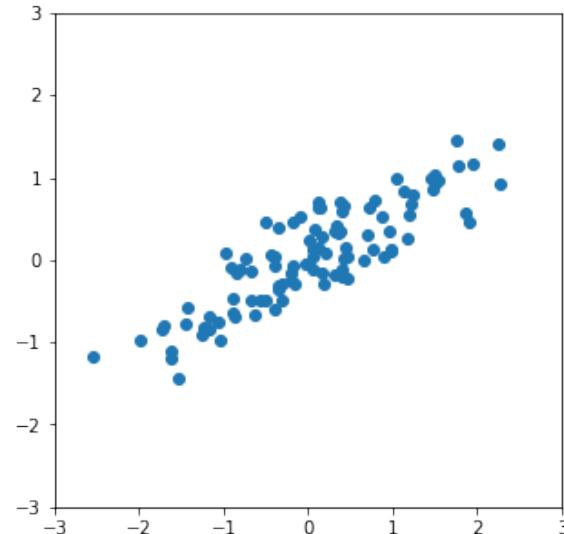


Unit 03 | PCA : Principal Component Analysis

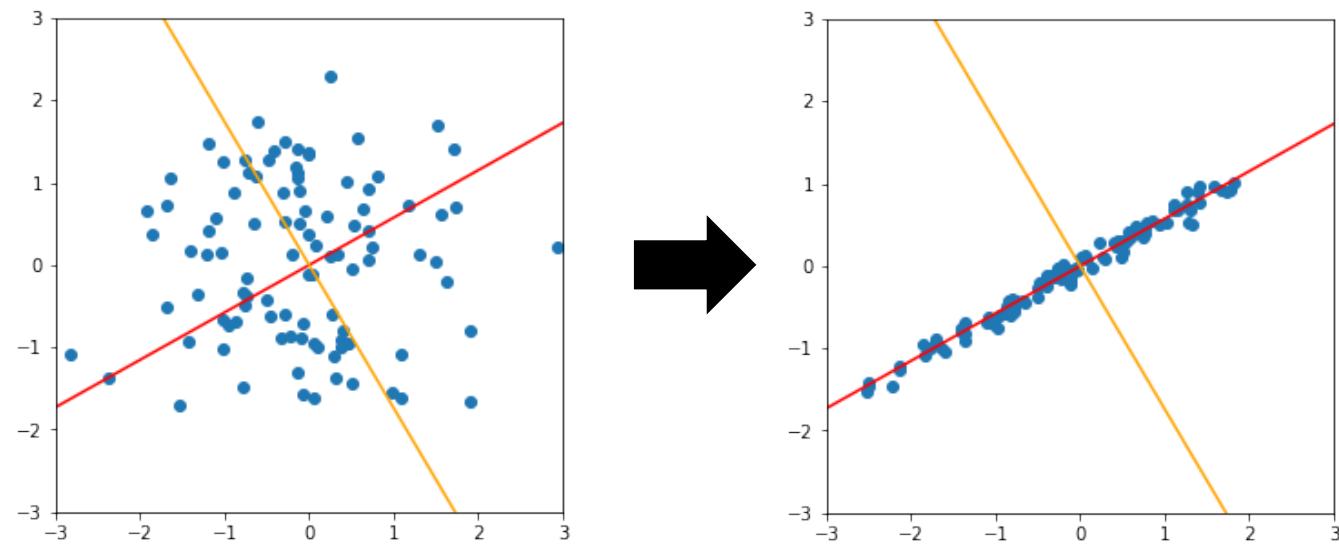
■ Principal Component Analysis 주성분 분석

- Covariance Matrix 공분산 행렬

- $\Sigma = \begin{bmatrix} 1.026 & 0.548 \\ 0.548 & 0.389 \end{bmatrix}$



- 공분산 행렬로 선형변환 <랜덤 벡터 100개>



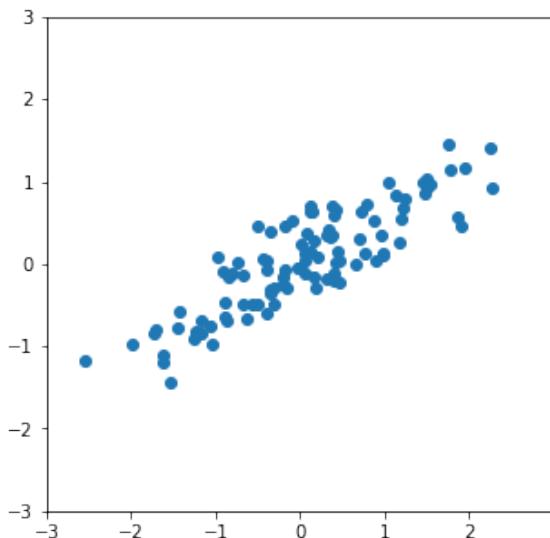
- 고유벡터는 방향이 변하지 않음
- 대칭행렬의 고유벡터는 직교

Unit 03 | PCA : Principal Component Analysis

■ Principal Component Analysis 주성분 분석

■ Covariance Matrix 공분산 행렬

$$\Sigma = \begin{bmatrix} 1.026 & 0.548 \\ 0.548 & 0.389 \end{bmatrix}$$



■ 공분산 행렬의 스펙트럼 분해

$$\Sigma = \begin{bmatrix} 1.026 & 0.548 \\ 0.548 & 0.389 \end{bmatrix}$$

$$= P \Lambda P'$$

P : 고유 벡터의 열로 구성된 행렬

Λ : 고유값을 대각원소로 하는 대각행렬

$$= \begin{bmatrix} 0.867 & 0.499 \\ -0.499 & 0.867 \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} 0.867 & -0.499 \\ 0.499 & 0.867 \end{bmatrix}$$

e_1

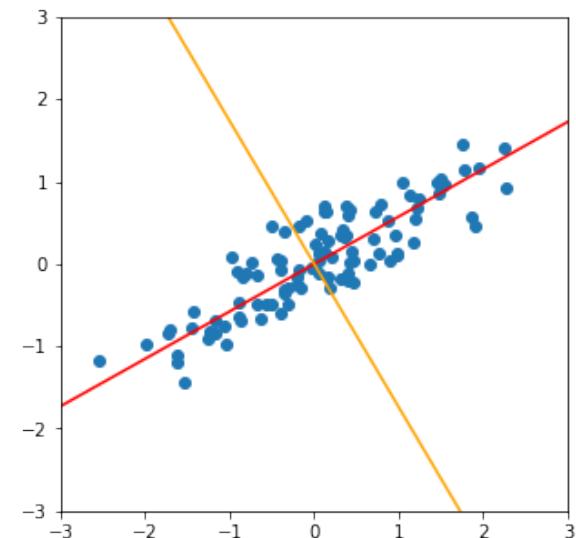
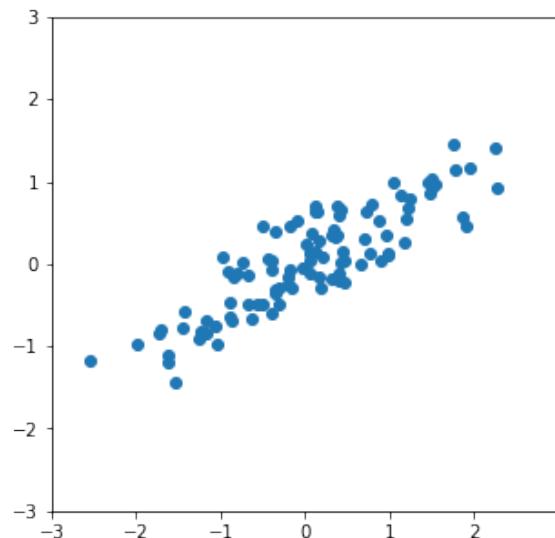
e_2

Unit 03 | PCA : Principal Component Analysis

■ Principal Component Analysis 주성분 분석

- $\Sigma = \begin{bmatrix} 1.026 & 0.548 \\ 0.548 & 0.389 \end{bmatrix}$

- $\lambda_1 = 1.342, e_1 = \begin{bmatrix} 0.867 \\ -0.499 \end{bmatrix}, \lambda_2 = 0.073, e_2 = \begin{bmatrix} 0.499 \\ 0.867 \end{bmatrix}$



- 고유벡터 : 데이터가 분산된 방향
- 고유값 : 전체 분산에 대한 주성분 PC_i 의 설명 비율

전체 분산 = $tr(\Sigma) = tr(\Lambda) = \lambda_1 + \lambda_2 + \cdots + \lambda_n$

PC_i 의 분산 = λ_i

→ 분산(고유값)이 큰 주성분 부터 사용

→ k개 주성분의 설명 비율

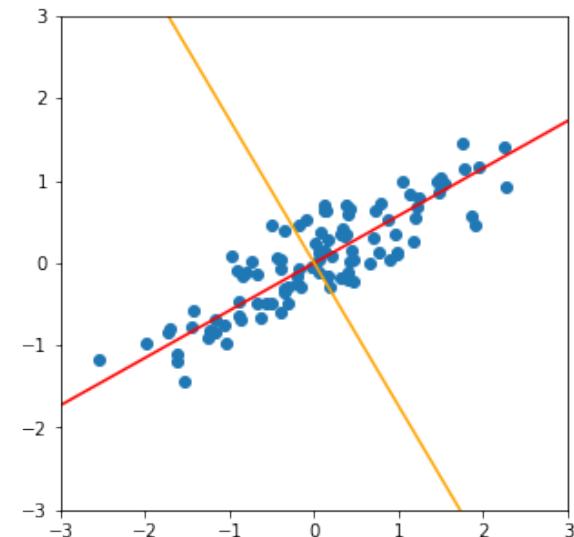
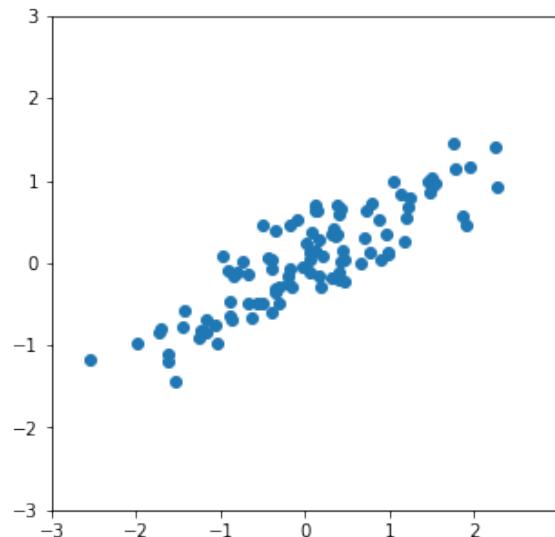
$$= (\lambda_1 + \lambda_2 + \cdots + \lambda_k) / (\lambda_1 + \lambda_2 + \cdots + \lambda_n)$$

Unit 03 | PCA : Principal Component Analysis

■ Principal Component Analysis 주성분 분석

- $\Sigma = \begin{bmatrix} 1.026 & 0.548 \\ 0.548 & 0.389 \end{bmatrix}$

- $\lambda_1 = 1.342, e_1 = \begin{bmatrix} 0.867 \\ -0.499 \end{bmatrix}, \lambda_2 = 0.073, e_2 = \begin{bmatrix} 0.499 \\ 0.867 \end{bmatrix}$



$$PC_1 = 0.867 * x_1 - 0.499 * x_2$$

$$PC_2 = 0.499 * x_1 + 0.867 * x_2$$

$$\text{전체 분산} = 1.342 + 0.073 = 1.415$$

$$PC_1 \text{의 분산} = \frac{1.342}{1.415} = 0.945$$

$$PC_2 \text{의 분산} = \frac{0.073}{1.415} = 0.052$$

첫번째 주성분이 전체 분산의 95%를 설명 ?

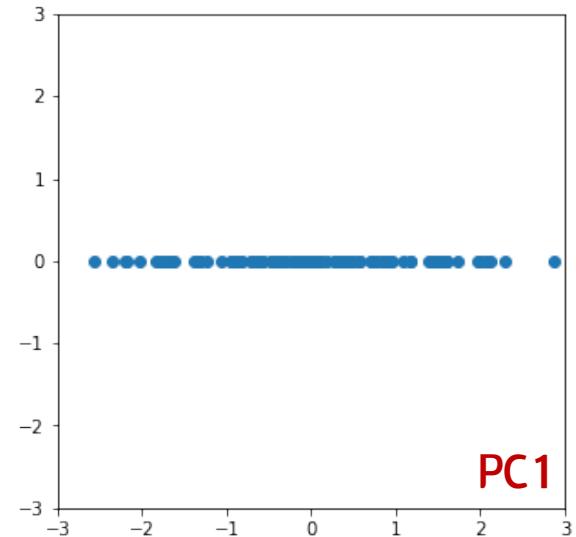
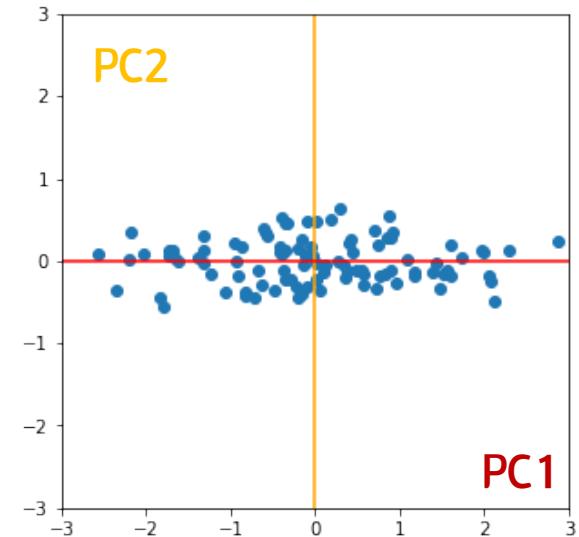
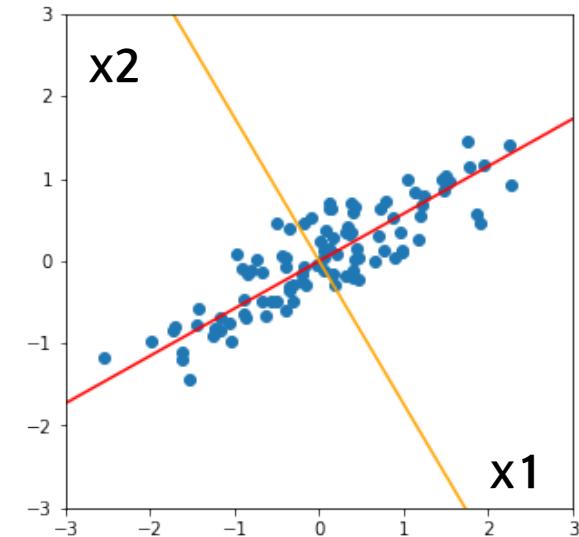
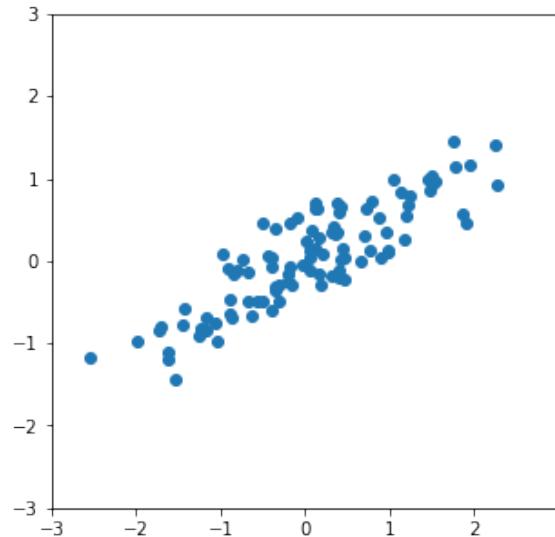
→ 실데이터에선 측정단위의 영향을 받음

: 표준화 사용!

Unit 03 | PCA : Principal Component Analysis

■ Principal Component Analysis 주성분 분석

- 변환된 축이 최대 분산 방향과 정렬되도록 좌표 회전
 - : xy 평면의 기저를 정규화된 공분산 행렬의 고유벡터 행렬로 기저변환
- 전체분산 대부분을 설명하는 주성분 1로 차원축소 \rightarrow PC1 축으로 데이터 투영

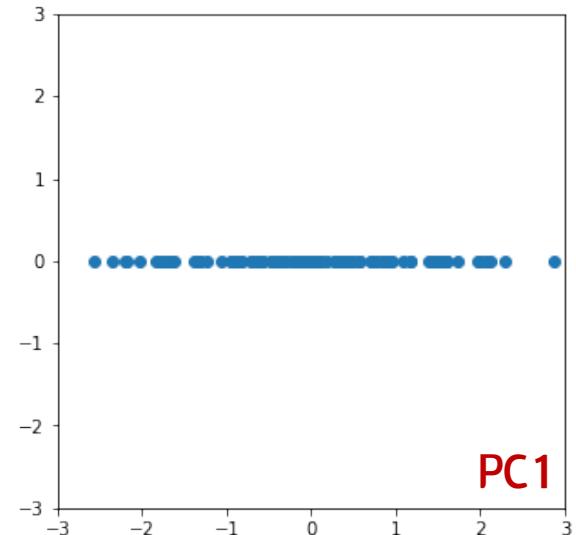
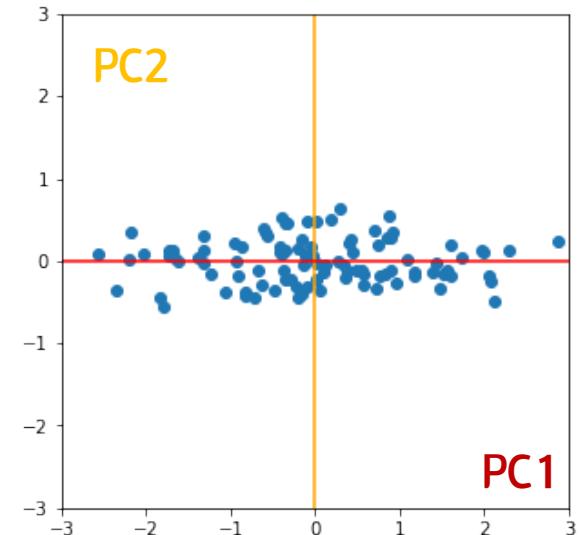
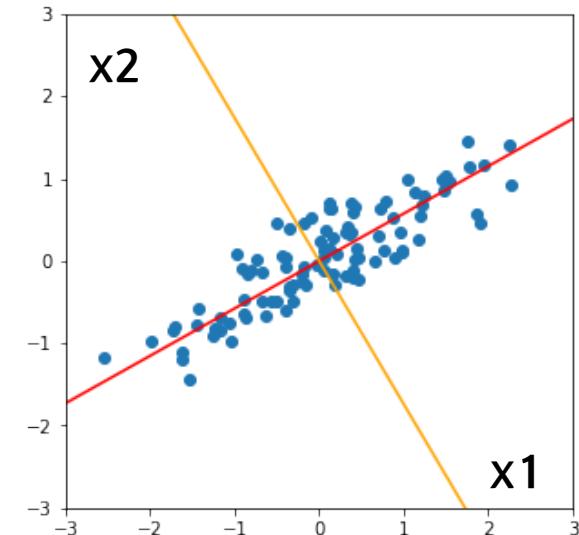
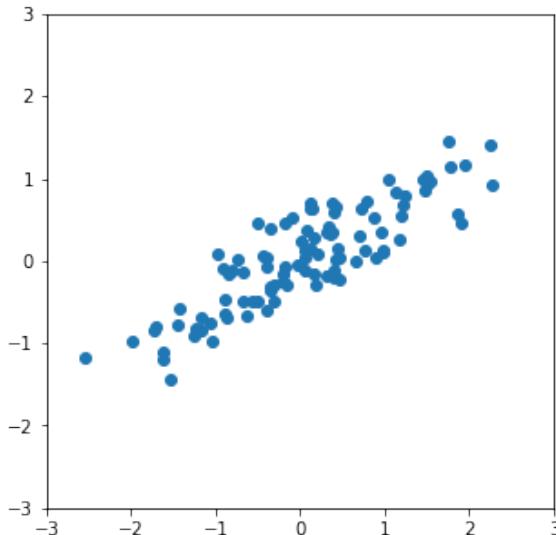


Unit 03 | PCA : Principal Component Analysis

■ Principal Component Analysis 주성분 분석

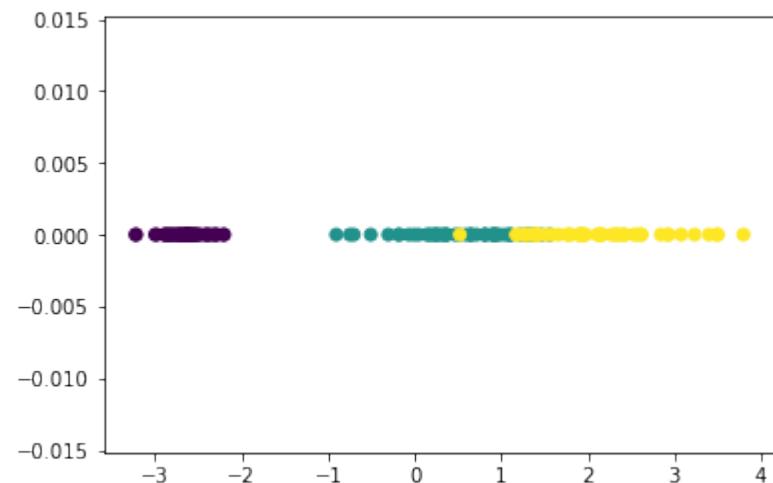
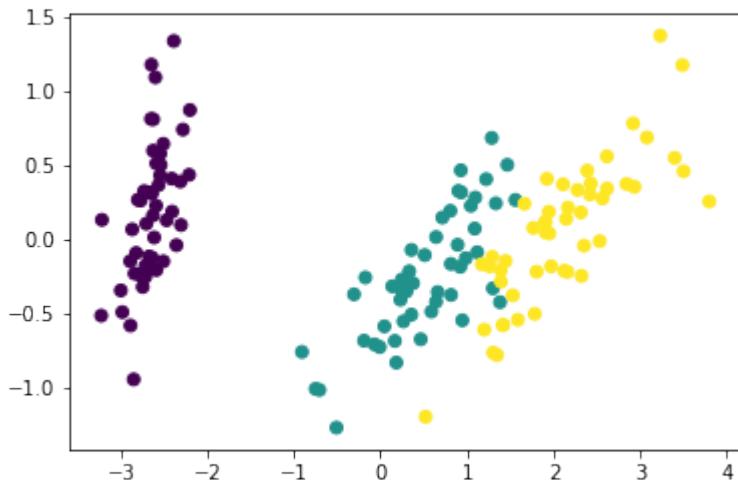
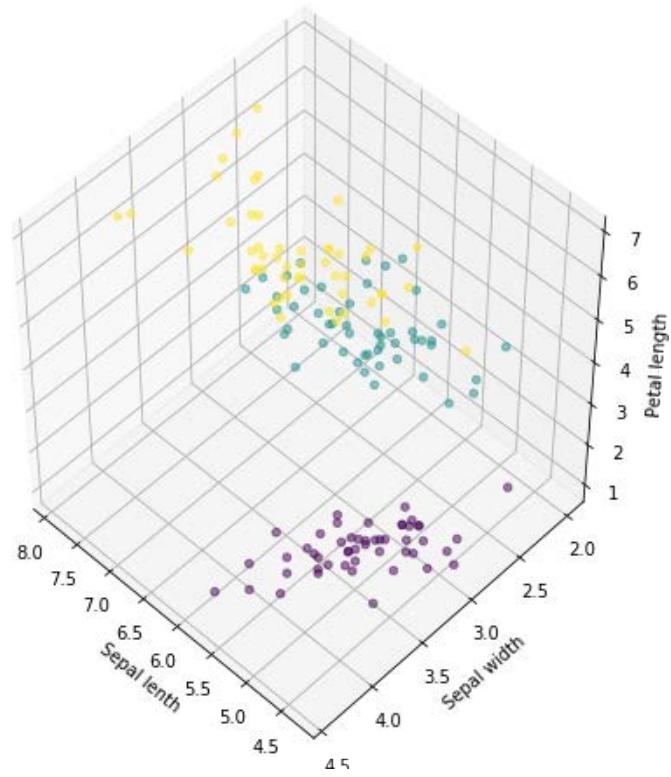
- 공분산 행렬의 모든 고유벡터는 **직교**한다. 즉, 모든 주성분들은 서로 **독립**이다.
- 데이터 분산이 가장 큰 첫번째 축을 찾고

그 축과 직교하면서 분산이 다음으로 큰 두번째 축을 찾고 … 반복



Unit 03 | PCA : Principal Component Analysis

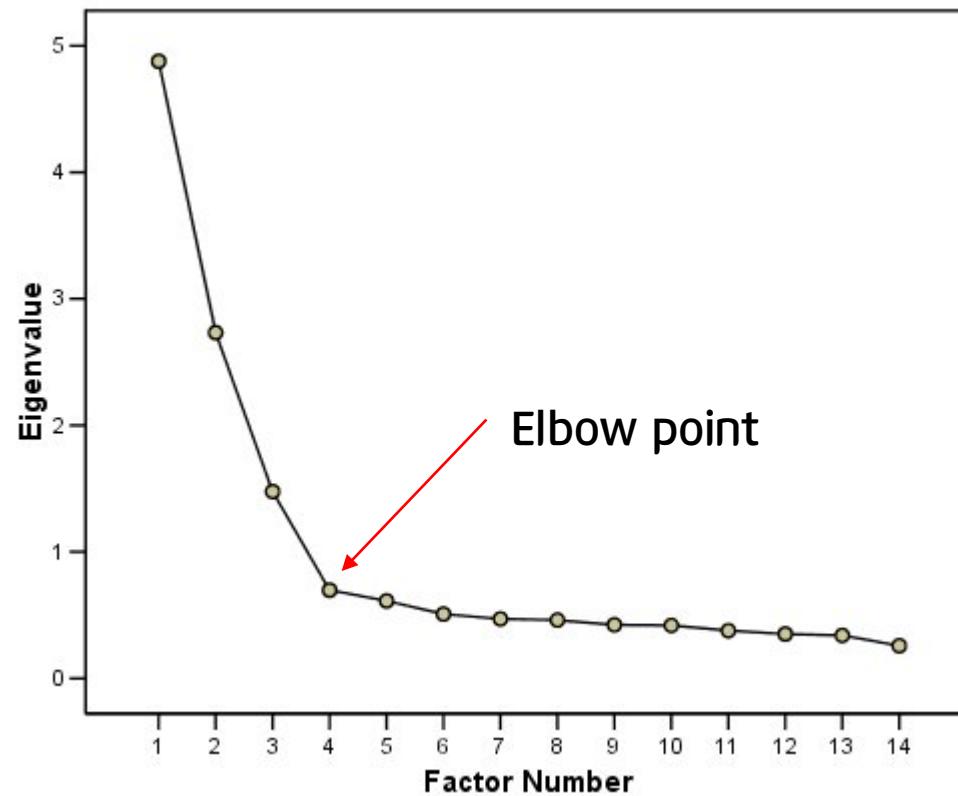
■ Principal Component Analysis 주성분 분석



Unit 03 | PCA : Principal Component Analysis

■ 주성분 개수의 결정

< Scree plot >



1. Elbow point
: 곡선의 기울기가 급격히 감소하는 지점
2. Kaiser's Rule
: 고유값 1 이상의 주성분들
3. 누적설명률이 70%~80% 이상인 지점

Unit 03 | PCA : Principal Component Analysis

■ 응용 - PCA Regression 주성분 회귀분석

- 주성분을 설명변수로 사용하여 종속변수를 설명

$$PC_1 = a_{11} * x_1 + a_{12} * x_2 + \dots + a_{1n} * x_n$$

$$PC_2 = a_{21} * x_1 + a_{22} * x_2 + \dots + a_{2n} * x_n$$

$$y^* = \beta_0 + \beta_1 * PC_1 + \beta_2 * PC_2$$

$$= \beta_0 + \beta_1 * (a_{11} * x_1 + a_{12} * x_2 + \dots + a_{1n} * x_n) + \beta_2 * (a_{21} * x_1 + a_{22} * x_2 + \dots + a_{2n} * x_n)$$

$$= \beta_0 + (\beta_1 * a_{11} + \beta_2 * a_{21}) * x_1 + (\beta_1 * a_{12} + \beta_2 * a_{22}) * x_2 + \dots + (\beta_1 * a_{n1} + \beta_2 * a_{n2}) * x_n$$

주의사항

- Test dataset을 정규화 할 때 Train dataset의 mean & std 사용
- 범주형 자료에는 사용 X

Unit 03 | PCA : Principal Component Analysis

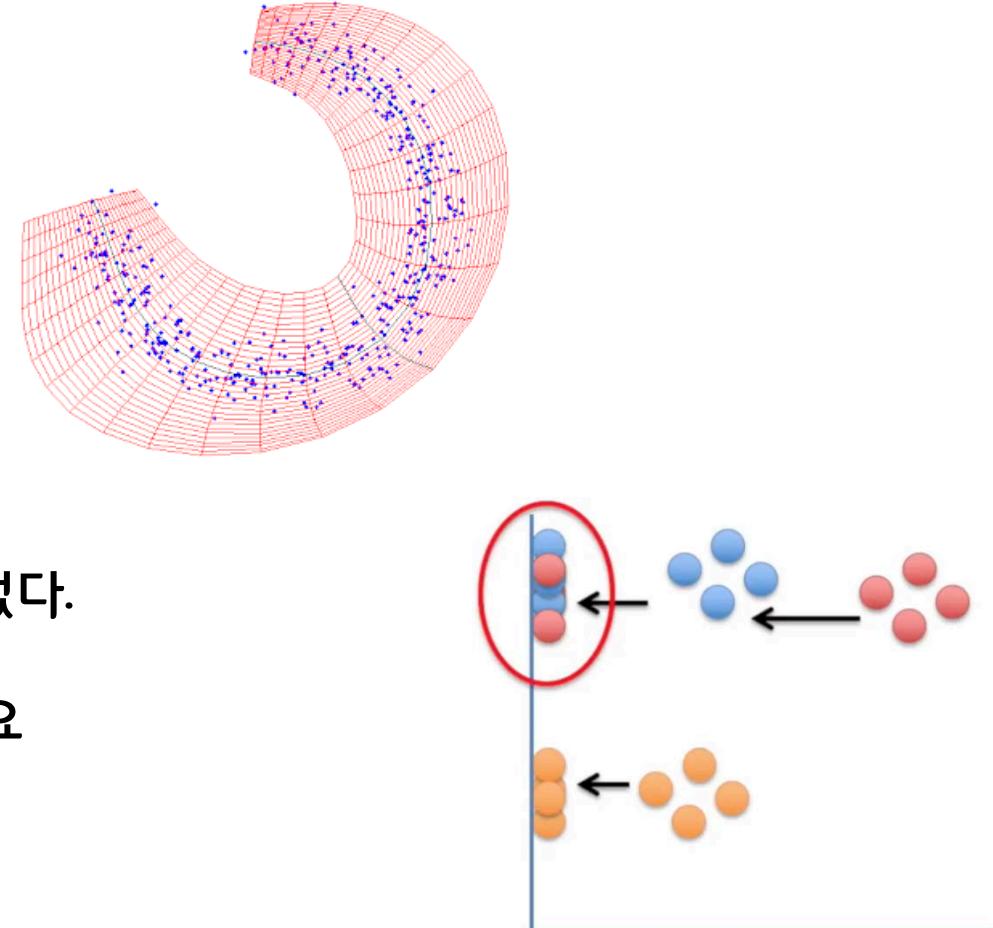
- PCA의 가정
 - Linearity : 데이터가 선형성을 띤다
 - Orthogonality : 찾은 주축들은 서로 직교한다.
 - 큰 분산을 갖는 방향이 중요한 정보를 담고 있다.
- PCA의 장점
 - 변수 간 상관관계 및 연관성을 이용해 변수 생성
 - 차원 축소로 인한 차원의 저주 해결 (속도 상승 & 과적합 방지)
 - 다중공선성 문제 해결

Unit 03 | PCA : Principal Component Analysis

■ PCA의 단점

- 데이터가 선형성을 띠지 않으면 적용 불가
→ 해결 : Kernel PCA를 통해 비선형 데이터에 적용 가능
- 특징 벡터의 클래스를 고려하지 않기 때문에
최대 분산 방향이 특징 구분을 좋게 한다는 보장이 없다.
- 새로 형성된 주성분의 해석을 위한 도메인 지식이 필요
(주성분은 모든 변수의 선형결합으로 이루어짐)

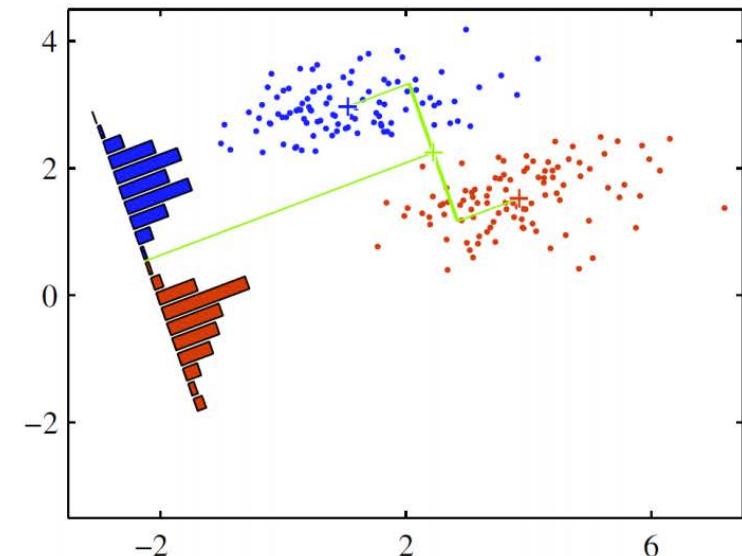
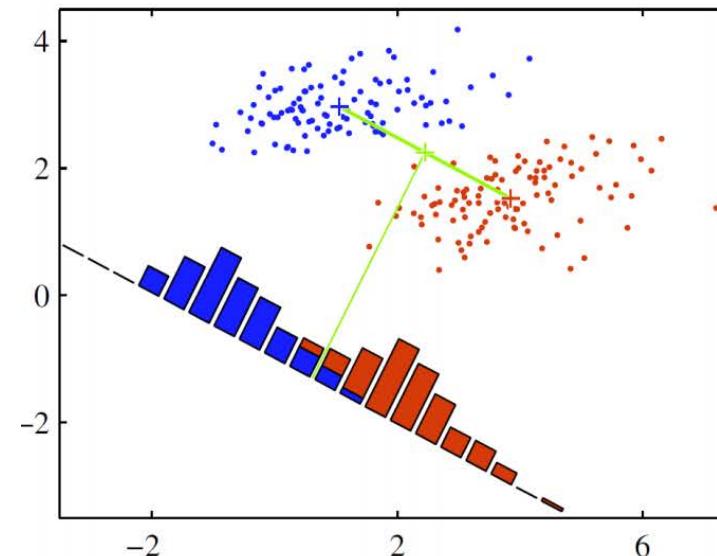
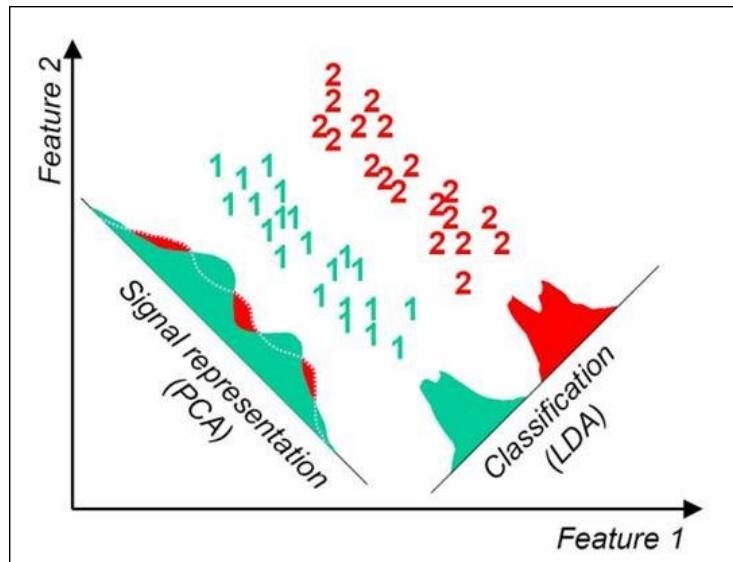
$$PC_1 = a_{11} * x_1 + a_{12} * x_2 + \dots + a_{1n} * x_n$$



Unit 04 | LDA : Linear Discriminant Analysis

■ Linear Discriminant Analysis 선형 판별 분석

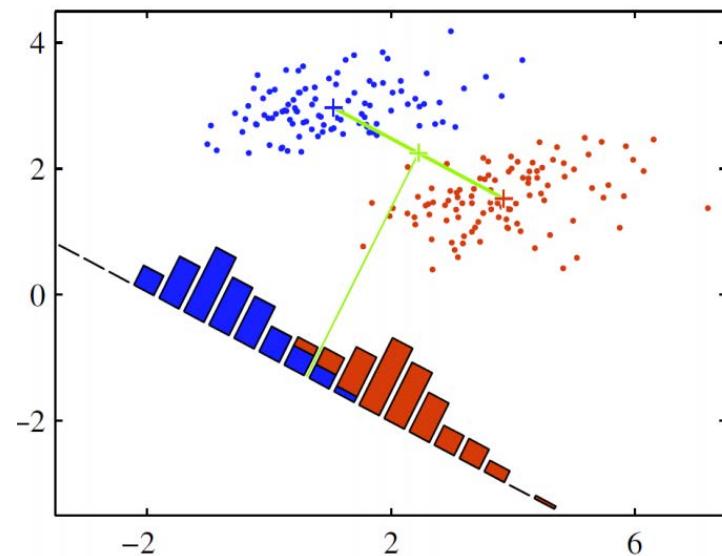
- 데이터의 분포를 학습하여 분리를 최적화하는 결정경계를 만들어 데이터를 분류하는 모델
 - PCA가 최적 표현을 위해 최대분산을 찾아 차원을 축소한다면
- LDA는 **최적 분류**를 위해 분별정보를 최대한 유지시키면서 차원을 축소한다.



Unit 04 | LDA : Linear Discriminant Analysis

■ Fisher's LDA

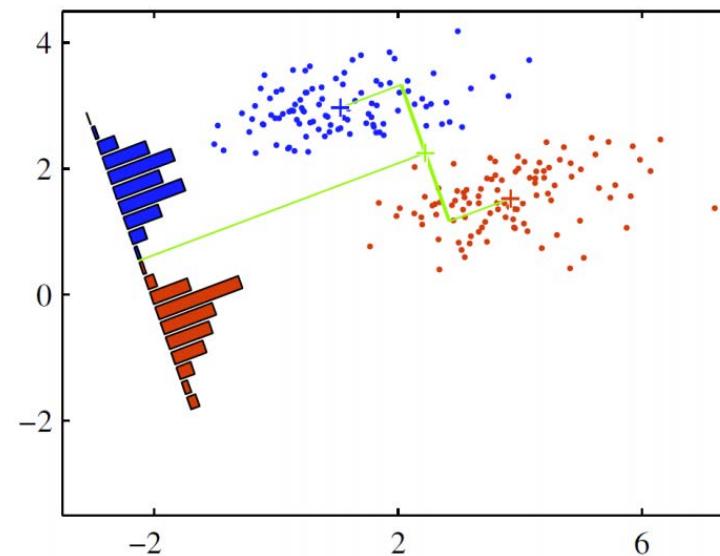
- 두 클래스의 중심점을 멀게 \rightarrow 클래스 간의 분산은 크게
- 클래스 내의 분산은 작게



중심점 간의 거리가 크다

클래스 내의 분산, 클래스 간의 분산
동시에 고려

VS



클래스 내의 분산이 작아서 더 좋은 분류기

Unit 04 | LDA : Linear Discriminant Analysis

■ LDA 수식적 접근

- P차원의 입력벡터 x 를 w 라는 벡터(축)으로 투영 한 후 생성되는 1차원 좌표값을 y 로 정의
- 각각 N_1, N_2 개의 관측치를 갖는 두 범주에 대해 입력공간에서 각 범주의 중심 벡터를 M_1, M_2 로 정의

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

투영 후 두 중심과 W 의 관계

$$M_1 = \frac{1}{N_1} \sum_{n \in C_1} x_n$$

$$m_2 - m_1 = w^T (M_2 - M_1)$$

투영 후 분산

$$M_2 = \frac{1}{N_2} \sum_{n \in C_2} x_n$$

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

 M, w, x : 벡터 m, s, y : 상수

Unit 04 | LDA : Linear Discriminant Analysis

■ LDA 의 목적함수

- 클래스 간의 분산은 크게
- 클래스 내의 분산은 작게

$$J(\vec{w}) = \frac{(m_1 - m_0)^2}{s_0^2 + s_1^2} = \frac{\text{두 클래스 평균 차의 제곱}}{\text{두 클래스의 분산의 합}}$$

■ 분자

$$(m_1 - m_0)^2 = (w^T M_1 - w^T M_2)^2$$

$$= (w^T(M_1 - M_2))^2$$

$$= w^T(M_1 - M_2)(M_1 - M_2)^T w$$

$$= w^T S_B w \quad \therefore S_B = (M_1 - M_2)(M_1 - M_2)^T$$

Unit 04 | LDA : Linear Discriminant Analysis

- LDA 의 목적함수 $J(\vec{w}) = \frac{(m_1 - m_0)^2}{s_0^2 + s_1^2}$

- 분모

$$s_0^2 = \sum_{n \in C_0} (w^T \underline{X_i} - \underline{w^T M_0})^2 = \sum_{n \in C_0} (w^T X_i - w^T M_0)^2$$

$$= \sum_{n \in C_0} [w^T (X_i - M_0)(X_i - M_0)^T w]$$

$$= w^T [\sum_{n \in C_0} (X_i - M_0)(X_i - M_0)^T] w$$

s_1^2 도 동일

$$s_0^2 + s_1^2 = w^T [\sum_{n \in C_0} (X_i - M_0)(X_i - M_0)^T + \sum_{n \in C_0} (X_i - M_0)(X_i - M_0)^T] w$$

$$= w^T S_W w$$

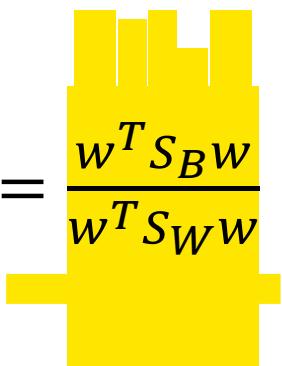
$$\therefore S_w = \sum_{n \in C_0} (X_i - M_0)(X_i - M_0)^T + \sum_{n \in C_0} (X_i - M_0)(X_i - M_0)^T$$

Unit 04 | LDA : Linear Discriminant Analysis

■ LDA 의 목적함수

■ W로 정리한 목적함수

$$J(\vec{w}) = \frac{(m_1 - m_0)^2}{s_0^2 + s_1^2} = \frac{\vec{w}^T S_B \vec{w}}{\vec{w}^T S_W \vec{w}}$$



S_B : between – class scatter
 S_W : within – class scatter

■ 목적함수의 최대화를 위해 w로 미분 (분수의 미분)

$$J(\vec{w}) = \frac{2S_B \vec{w} (\vec{w}^T S_W \vec{w})}{(\vec{w}^T S_W \vec{w})^2} - \frac{2\vec{w}^T S_B \vec{w} S_W \vec{w}}{(\vec{w}^T S_W \vec{w})^2} = \frac{2S_B \vec{w}}{\vec{w}^T S_W \vec{w}} - \frac{2\vec{w}^T S_B \vec{w} S_W \vec{w}}{(\vec{w}^T S_W \vec{w})^2} = 0$$

Unit 04 | LDA : Linear Discriminant Analysis

- LDA 의 목적함수

- 목적함수의 최대화를 위해 w 로 미분 (분수의 미분)

$$\frac{d}{d\vec{w}} J(\vec{w}) = \frac{2S_B w (w^T S_W w)}{(w^T S_W w)^2} - \frac{2w^T S_B w S_W w}{(w^T S_W w)^2} = \frac{2S_B w}{w^T S_W w} - \frac{2w^T S_B w S_W w}{(w^T S_W w)^2} = 0$$

여기서 $\frac{2}{w^T S_W w}$, $\frac{2w^T S_B w}{(W^T S_W w)^2}$ 는 scalar 상수값 \rightarrow 하나의 값으로 표현 : λ

$S_B \vec{w} = \lambda S_W \vec{w}$ 표현될 수 있다.

$\rightarrow S_W$ 는 역행렬을 가짐 LDA에서 모든 변수들은 통계적으로 독립이라고 가정

최종식 : $S_W^{-1} S_B \vec{w} = \lambda \vec{w}$

Unit 04 | LDA : Linear Discriminant Analysis

■ LDA 의 분류

- 최종식 : $S_W^{-1}S_B\vec{w} = \lambda\vec{w}$ 어디서 많이 본 식 : 고유값 분해 ($Av = \lambda v$)

이 방적식을 만족시키는 \vec{w} 들은 $S_W^{-1}S_B$ 행렬의 고유벡터가 됨.

고유치의 크기 순서대로 고유 벡터들을 정렬 후 q개 (축소 후의 차원 수)를 선택하여 이를 열로 하는 변환행렬 W를 구성하여 Train set을 변환한다.

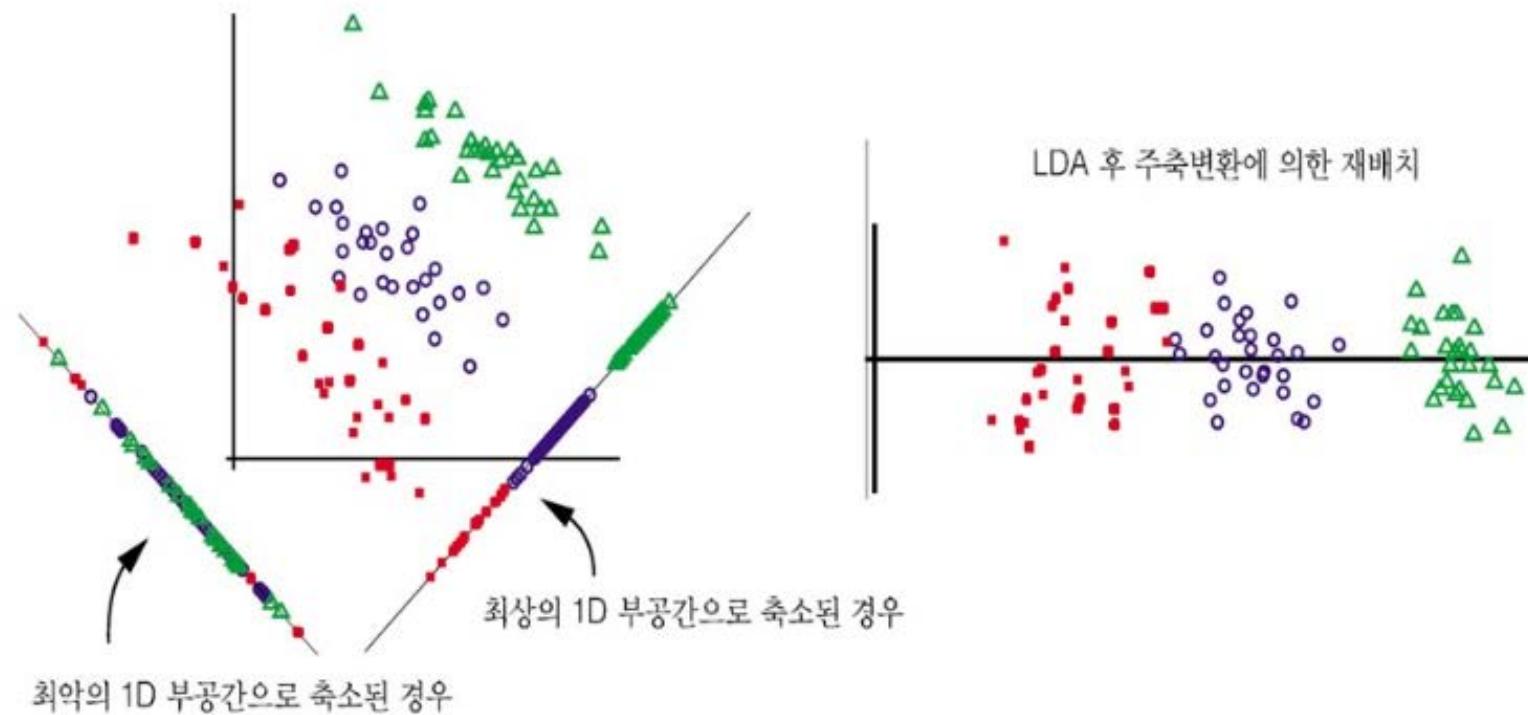
$$y = W^T x$$

변환된 Train set의 class 평균을 이용하여 test set의 class 결정

Unit 04 | LDA : Linear Discriminant Analysis

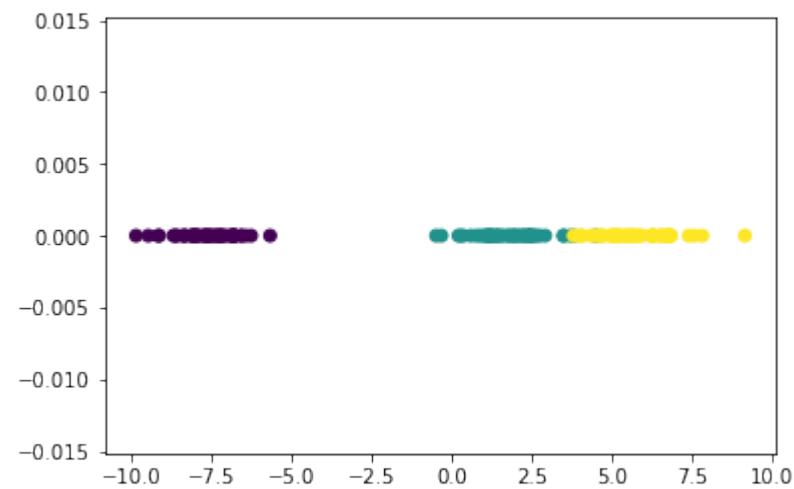
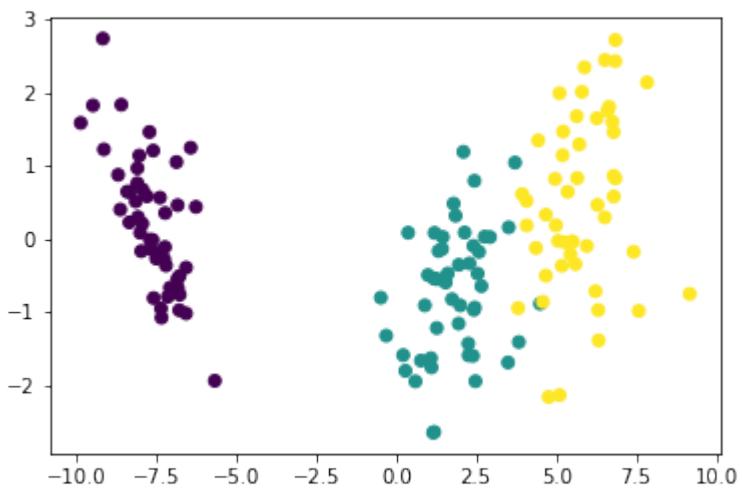
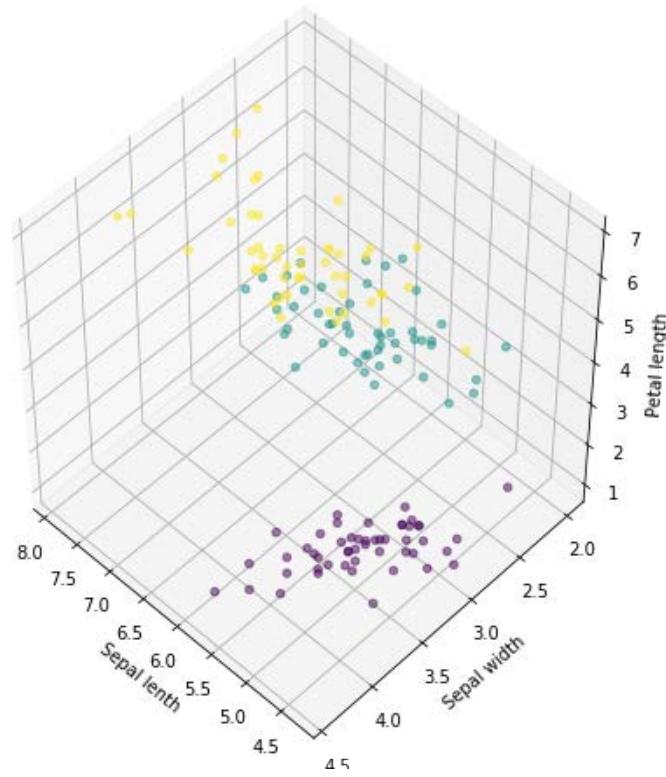
■ LDA 의 분류

ex) 고유벡터 하나만 이용해 $2d \rightarrow 1d$ 변환



Unit 03 | PCA : Principal Component Analysis

■ Linear Discriminant Analysis 선형 판별 분석



Unit 04 | LDA : Linear Discriminant Analysis

- 성능

- 보통 데이터의 패턴, 내재적 속성을 밝혀내는 것이 중요할 때는 PCA의 성능이 더 좋음
- LDA가 더 좋은 성능을 낼 때
 1. 데이터가 정규분포(다변량 정규분포)를 따름
 2. 각각의 class들이 **동일한** 공분산 행렬을 가짐

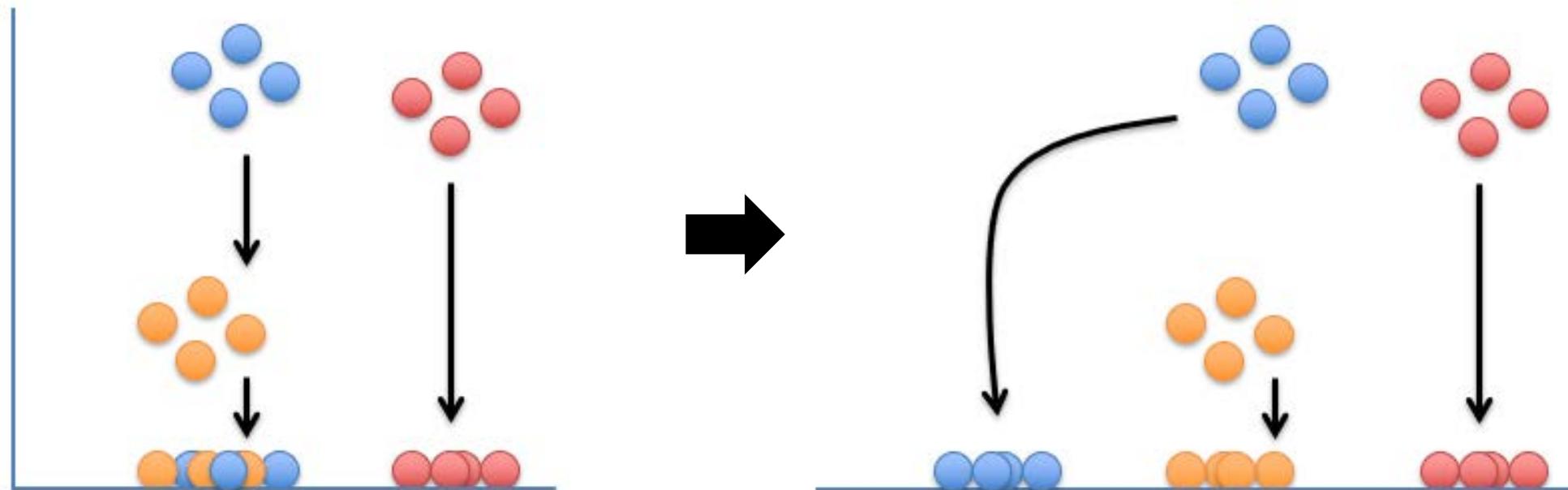
Unit 05 | T-SNE : T – Stochastic Neighbor Embedding

- T – Stochastic Neighbor Embedding T-분포 확률적 임베딩
 - 고 차원의 데이터를 저 차원의 데이터로 거리 관계를 유지하며 **임베딩** 시키는 기법
 - 직관적으로 데이터의 구조를 시각화하여 확인할 수 있다. Ex) 50차원의 데이터 -> 2차원 평면 시각화

Unit 05 | T-SNE : T – Stochastic Neighbor Embedding

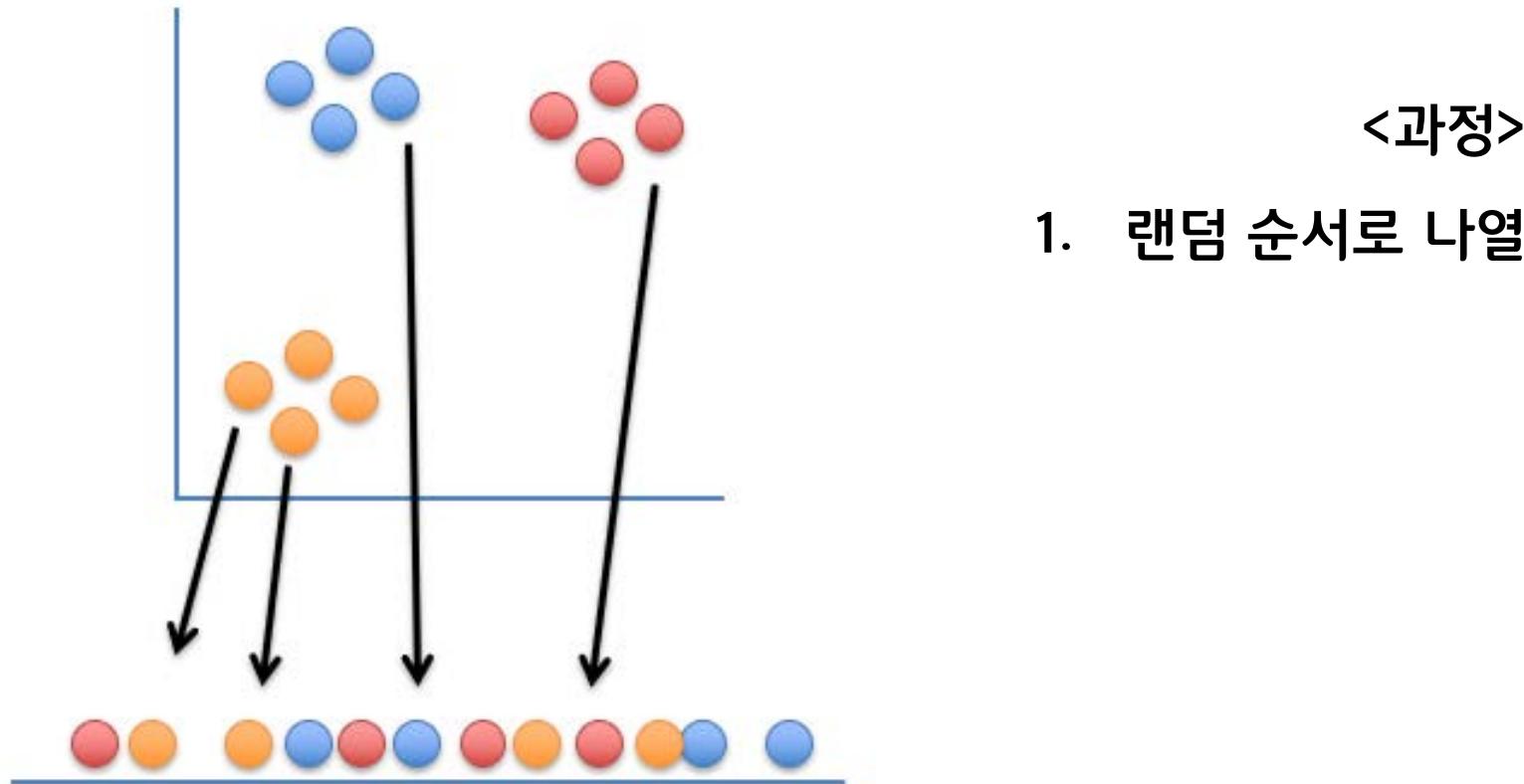
■ T – Stochastic Neighbor Embedding T-분포 확률적 임베딩

- 고 차원의 데이터를 저 차원의 데이터로 거리 관계를 유지하여 임베딩 시키는 기법
- 직관적으로 데이터의 구조를 시각화하여 확인할 수 있다. Ex) 50차원의 데이터 → 2차원 평면 시각화



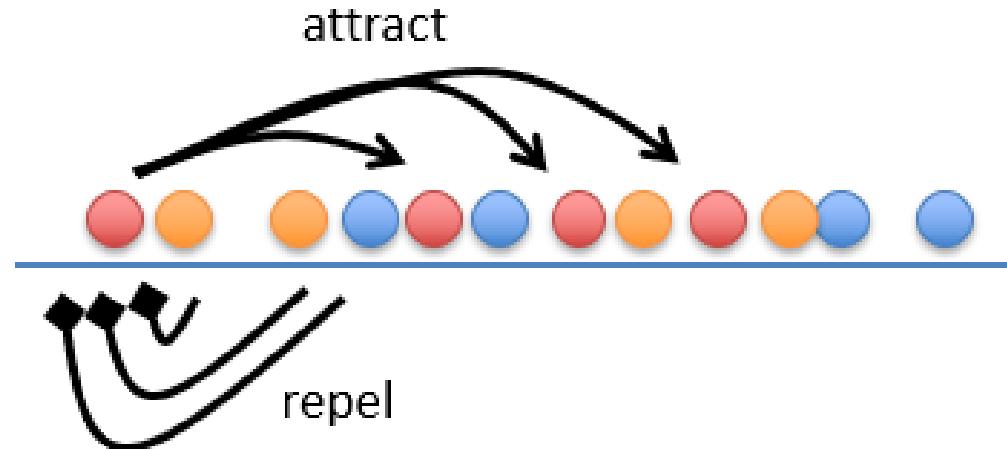
Unit 05 | T-SNE : T – Stochastic Neighbor Embedding

- T – Stochastic Neighbor Embedding T-분포 확률적 임베딩



Unit 05 | T-SNE : T – Stochastic Neighbor Embedding

- T – Stochastic Neighbor Embedding T-분포 확률적 임베딩



<과정>

1. 랜덤 순서로 나열
2. 동일 군집은 당기는 힘
다른 군집은 미는 힘 작용

Unit 05 | T-SNE : T – Stochastic Neighbor Embedding

- T – Stochastic Neighbor Embedding T-분포 확률적 임베딩



<과정>

1. 랜덤 순서로 나열
2. 동일 군집은 당기는 힘
다른 군집은 미는 힘 작용
3. 힘의 균형에 맞는 위치로 이동

Unit 05 | T-SNE : T – Stochastic Neighbor Embedding

■ T – Stochastic Neighbor Embedding T-분포 확률적 임베딩



<https://youtu.be/NEaUSP4YerM?t=226>

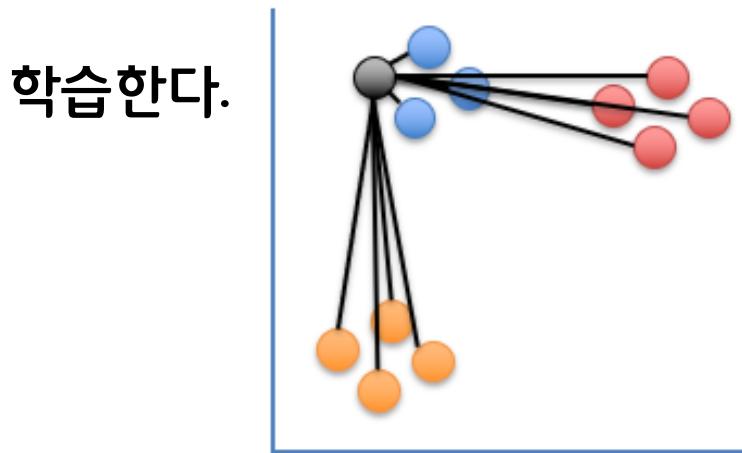
<과정>

1. 랜덤 순서로 나열
2. 동일 군집은 당기는 힘
다른 군집은 미는 힘 작용
3. 힘의 균형에 맞는 위치로 이동
4. 모든 점들이 2~3 과정 반복

Unit 05 | T-SNE : T – Stochastic Neighbor Embedding

■ T – Stochastic Neighbor Embedding T-분포 확률적 임베딩

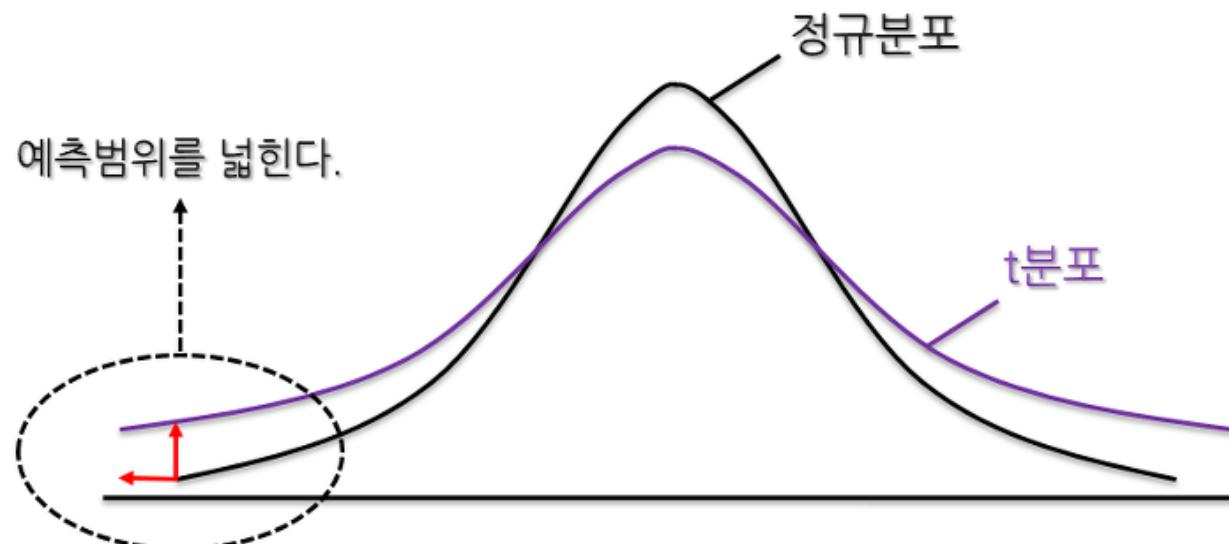
- 고차원 공간의 유클리디안 거리를 이용하여 저차원의 거리를 학습한다.
- 기존의 방법 : SNE – 정규 분포 사용
 - 문제점
 - 고차원의 거리가 멀어지면 gradient 값이 0이 되서 저차원의 거리가 멀어지게 학습을 못 시킴



저차원의 거리가 멀어지게 학습을 못 시킴
= 적당히 먼 거리와 매우 먼 거리를 구별 못함!!

Unit 05 | T-SNE : T – Stochastic Neighbor Embedding

- T – Stochastic Neighbor Embedding T-분포 확률적 임베딩
 - T-SNE : 자유도가 1인 T-분포 사용
 - 정규분포보다 꼬리가 두꺼워 예측 범위 상승
 - 가까운 거리를 더 가깝게 먼 거리를 더 멀게 배치할 수 있게 됨



Unit 05 | T-SNE : T – Stochastic Neighbor Embedding

- T – Stochastic Neighbor Embedding T-분포 확률적 임베딩

- 장점

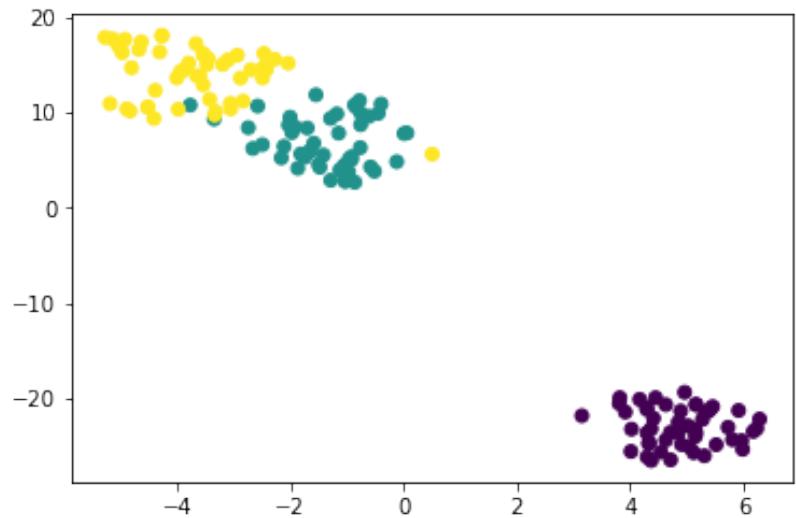
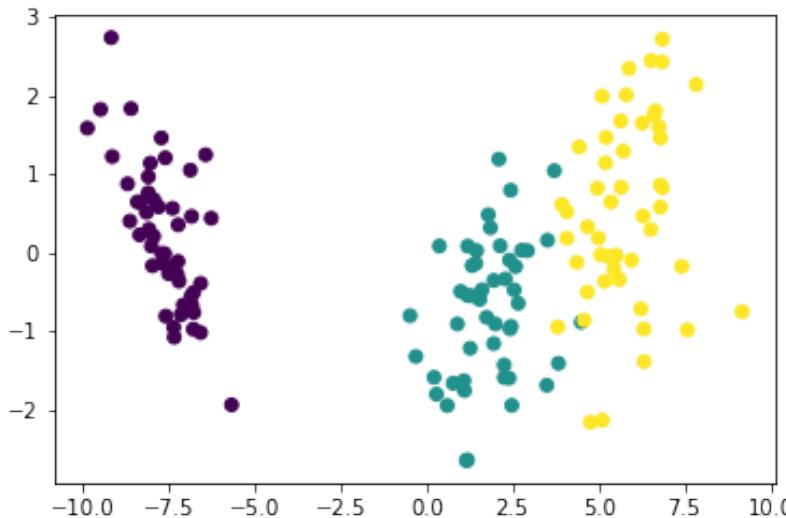
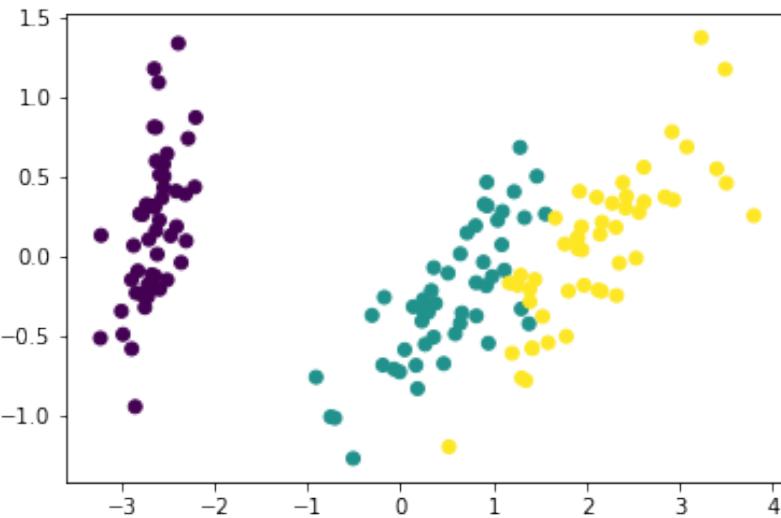
- PCA와 달리 군집이 중복되지 않는다.
 - 군집성이 유지되기 때문에 시각화를 통한 분석 유용

- 단점

- 거리를 학습하며 계속 업데이트 하기 때문에 값이 매번 바뀜
 - : 결과를 feature로 사용할 수 없다.
 - 데이터 수가 많아지면 시간이 오래 걸림

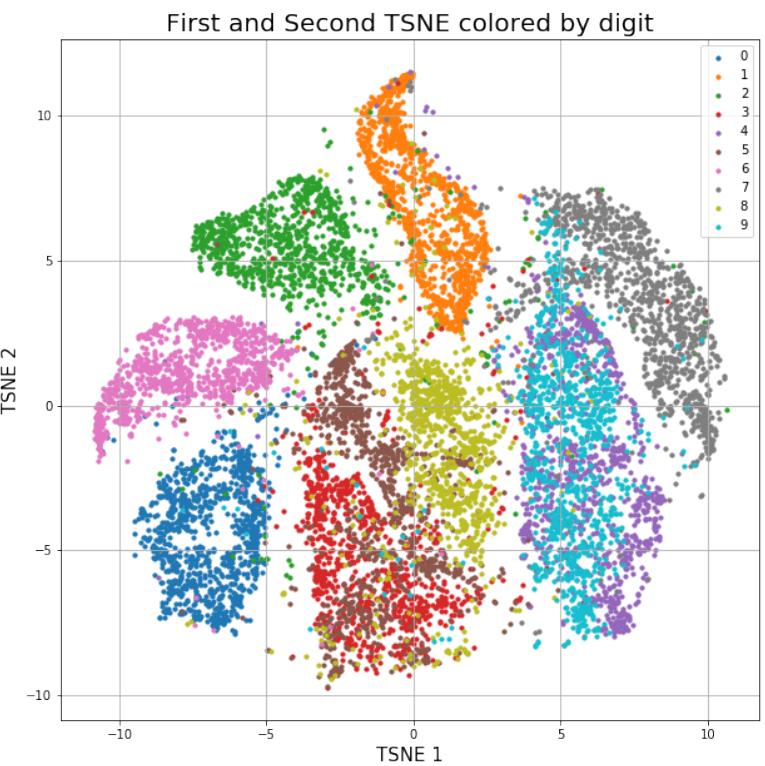
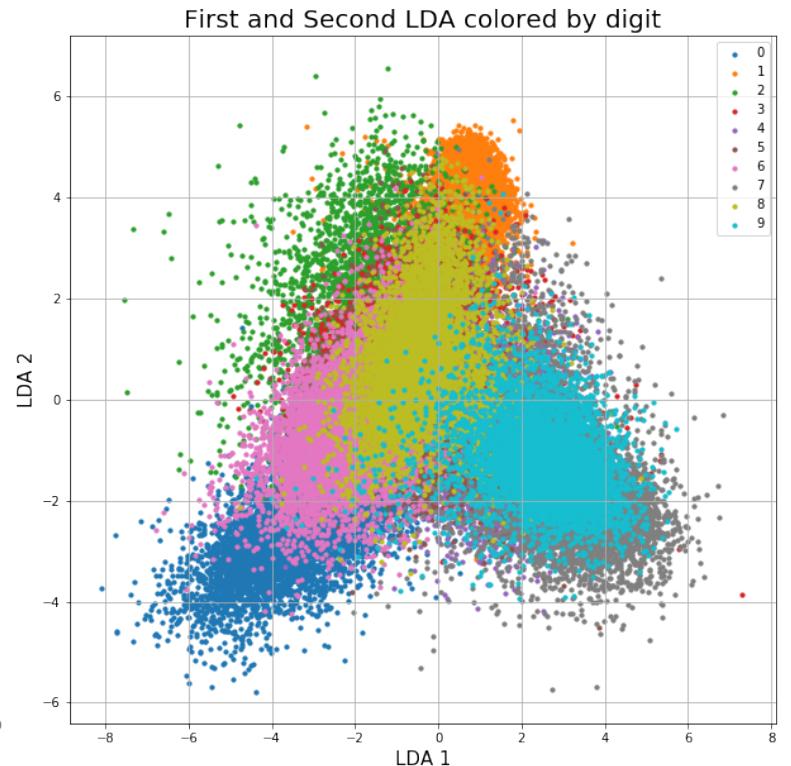
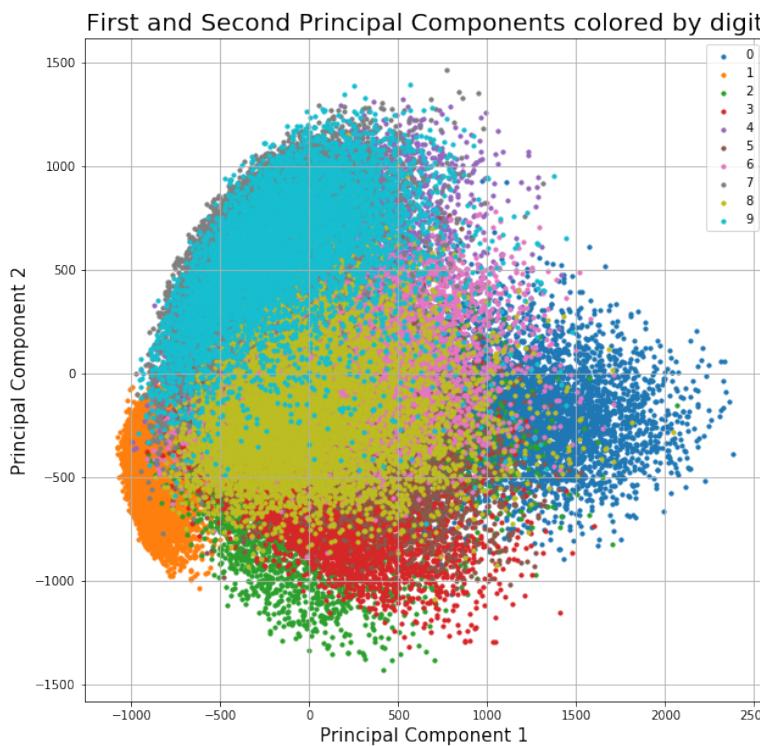
Unit 05 | T-SNE : T – Stochastic Neighbor Embedding

- PCA vs LDA vs T-SNE
 - IRIS



Unit 05 | T-SNE : T – Stochastic Neighbor Embedding

- PCA vs LDA vs T-SNE
 - MNIST



Assignment

과제 1

- Mnist 데이터를 사용 (8:2 비율로 train set, test set split)
- 원본 데이터 & PCA 축소 데이터 & LDA 축소 데이터 비교
- 지금까지 배웠던 다항 분류기 2개 이상 사용 (KNN, random forest, NB 등등)
- time stamp 찍어서 training 시간과 test accuracy 비교하기

+ 해석

Q & A

들어주셔서 감사합니다.

Appendix

- 참고자료

10기 이민주님 강의 -> 강의자료 및 실습코드

http://www.datamarket.kr/xe/index.php?mid=board_jPWY12&page=2&document_srl=52335

eigen vector & eigen value 관련 youtube

<https://www.youtube.com/watch?v=PFDu9oVAE-g&t=185s>

공돌이의 수학정리노트

PCA : <https://wikidocs.net/7646>

LDA : <https://wikidocs.net/5957>

ratsgo's blog

PCA : <https://ratsgo.github.io/machine%20learning/2017/04/24/PCA/>

LDA : <https://ratsgo.github.io/machine%20learning/2017/03/21/LDA/>

T-SNE 관련 youtube

<https://www.youtube.com/watch?v=NEaUSP4YerM>