

20기 정규세션

ToBig's 19기 강의자

최 경 주

Dimensionality Reduction

차원축소

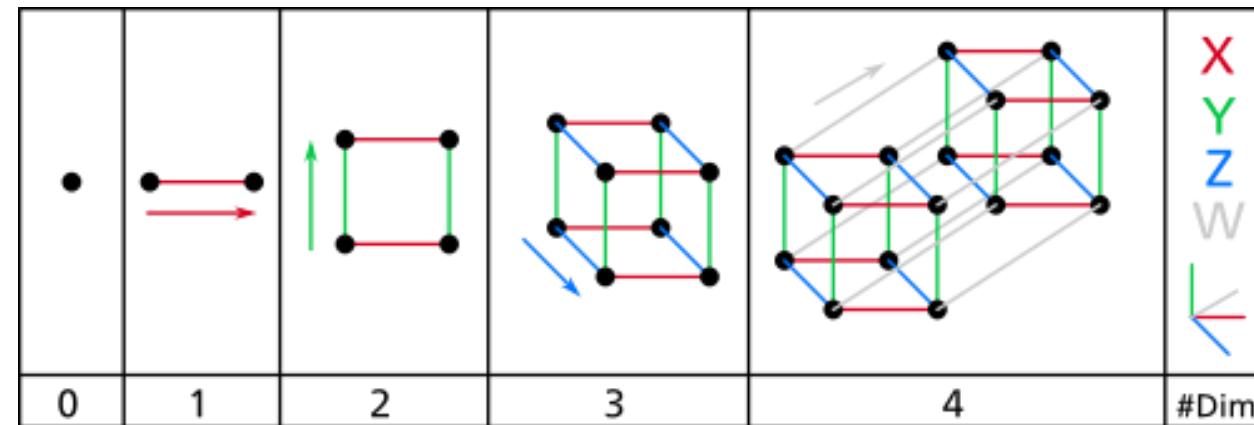
Contents

I. Unit 01	I. intro: Dimensionality Reduction
I. Unit 02	I. Eigen-Value Decomposition
I. Unit 03	I. PCA: Principal Component Analysis
I. Unit 04	I. LDA: Linear Discriminant Analysis
I. Unit 05	I. Manifold Learning

Contents

I. Unit 01	I. intro: Dimensionality Reduction
I. Unit 02	I. Eigen-Value Decomposition
I. Unit 03	I. PCA: Principal Component Analysis
I. Unit 04	I. LDA: Linear Discriminant Analysis
I. Unit 05	I. Manifold Learning

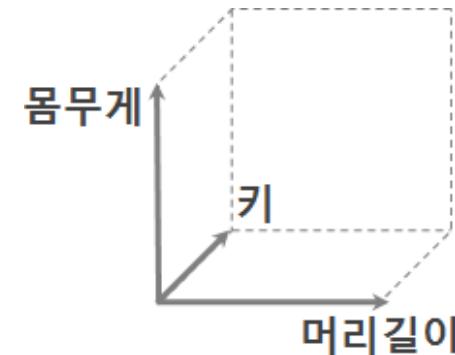
차원 Dimension



공간 내에 있는 점 등의 위치를 나타내기 위해 필요한 축의 개수

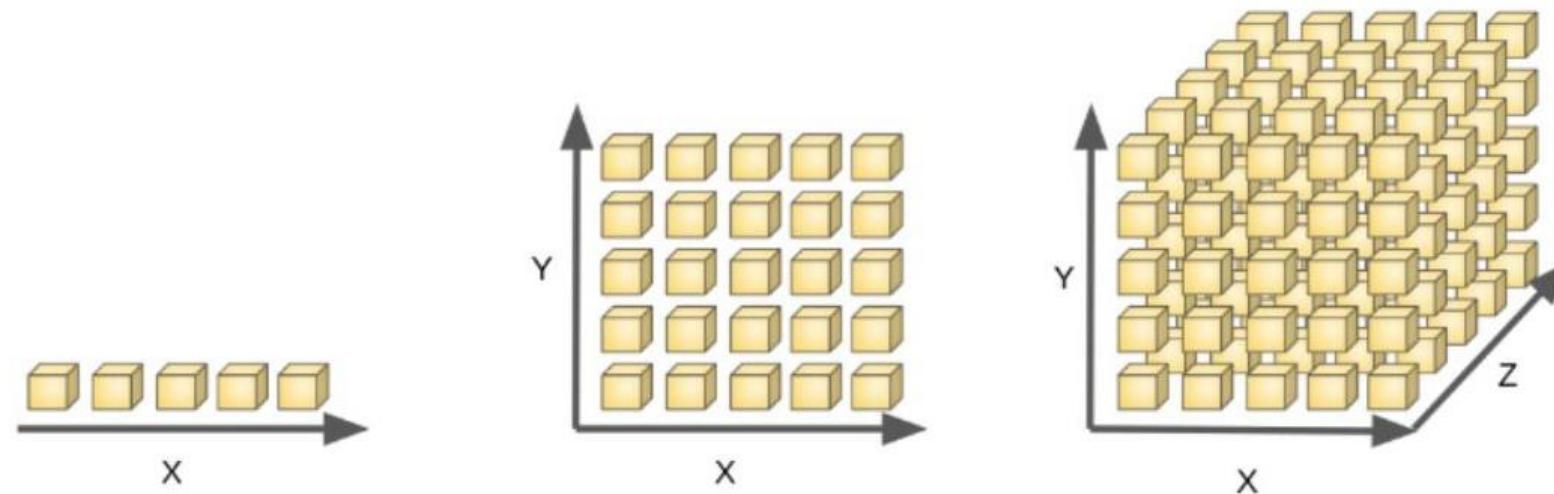
차원 Dimension

키	몸무게	머리길이
168	58	10
162	55	30
159	49	25
165	45	40



데이터에서의 차원이란?
데이터 분석에 사용되는 **변수의 개수**

차원의 저주 Curse of Dimensionality



변수의 수가 늘어나 차원이 커지면서 발생하는 문제
관측치 수보다 변수의 수가 많아질 경우

차원의 저주 Curse of Dimensionality

1) 필요한 데이터 수의 지수함수적 증식: **연산량 급증**

ex) 각 변수가 가질 수 있는 값이 **3개**일 때

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

변수A
1차원: 3개

1	2	3
4	5	6
7	8	9

변수A, B
2차원: 9개

19	20	21
22	23	24
25	26	27

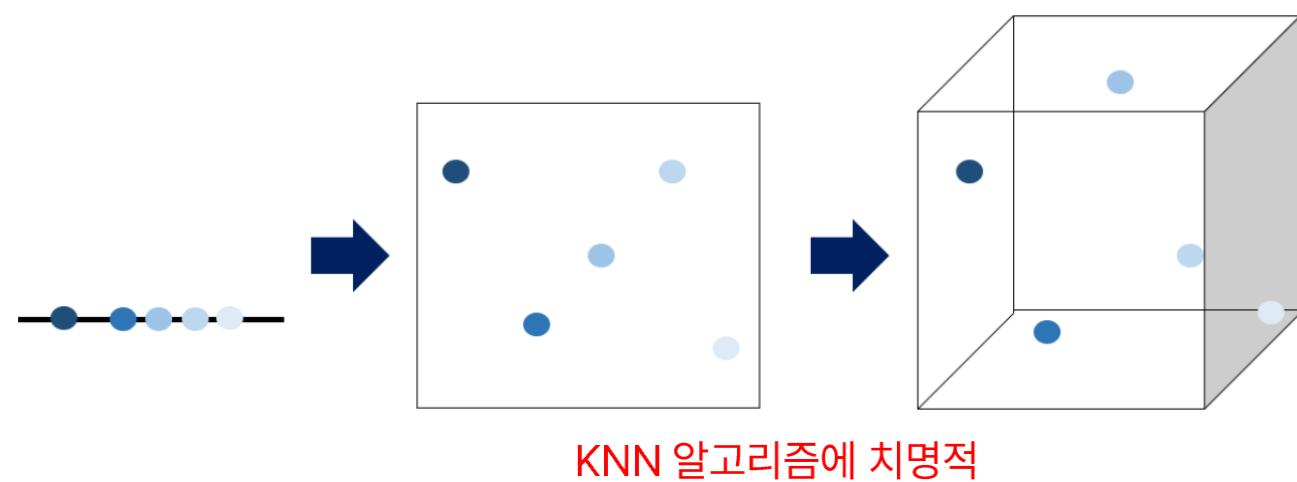
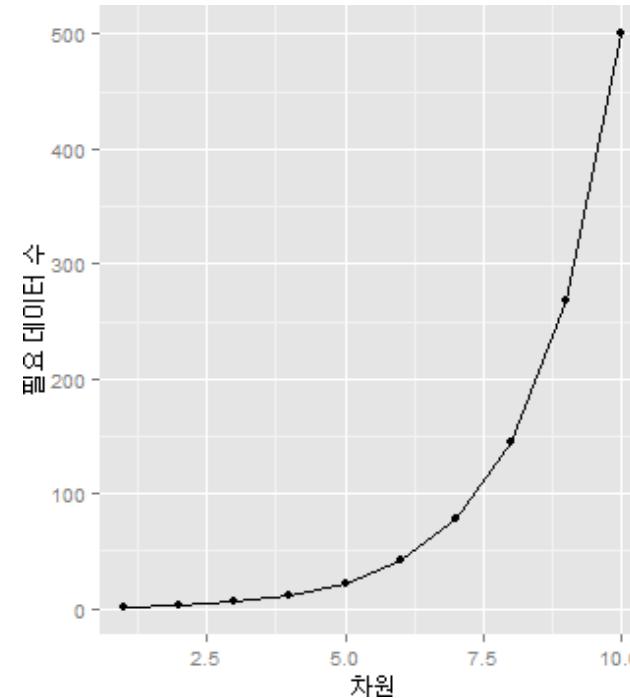
변수A, B, C
2차원: 27개



N차원: 3^{n} 개**

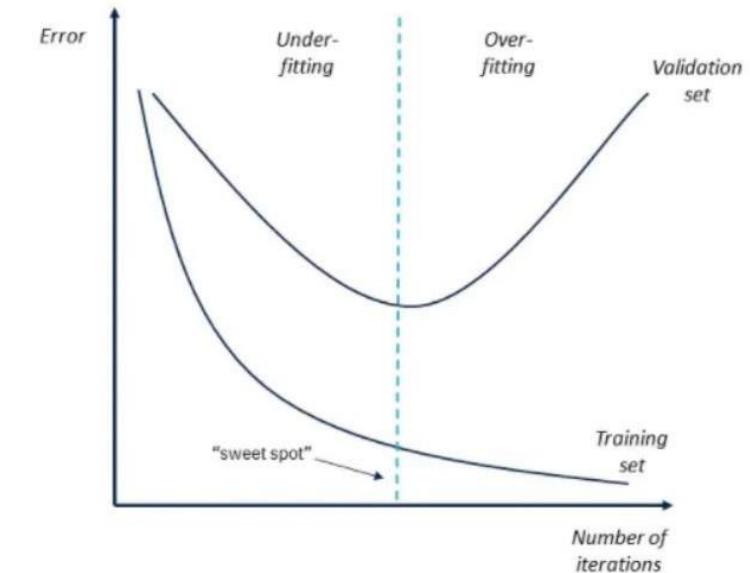
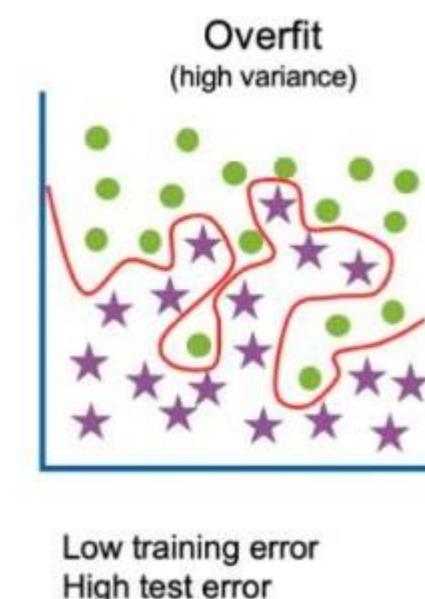
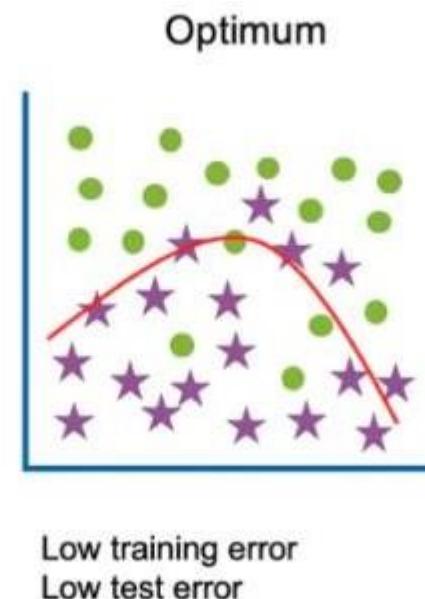
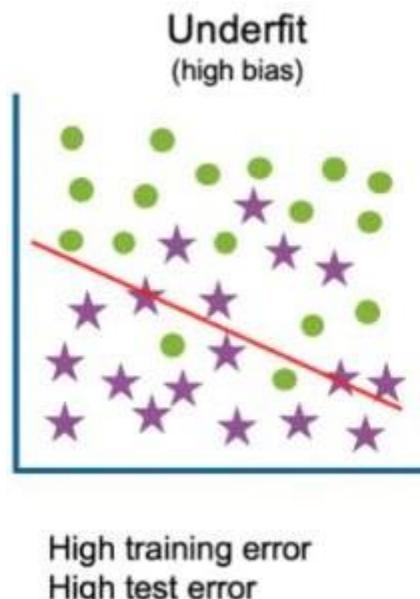
차원의 저주 Curse of Dimensionality

2) 차원이 증가함에 따라서 빈공간이 발생: **정보의 밀도 감소**



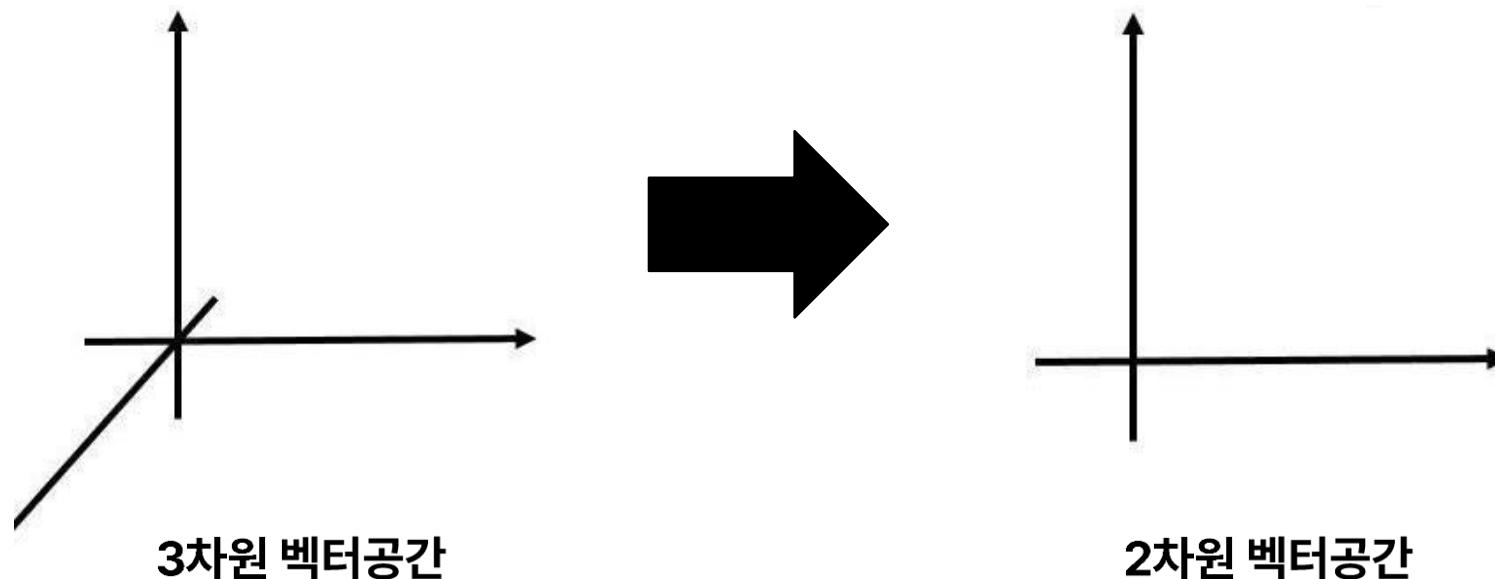
차원의 저주 Curse of Dimensionality

3) 공간을 설명하기 위한 데이터의 부족: 과적합 및 성능감소 우려



차원축소 Dimensionality Reduction

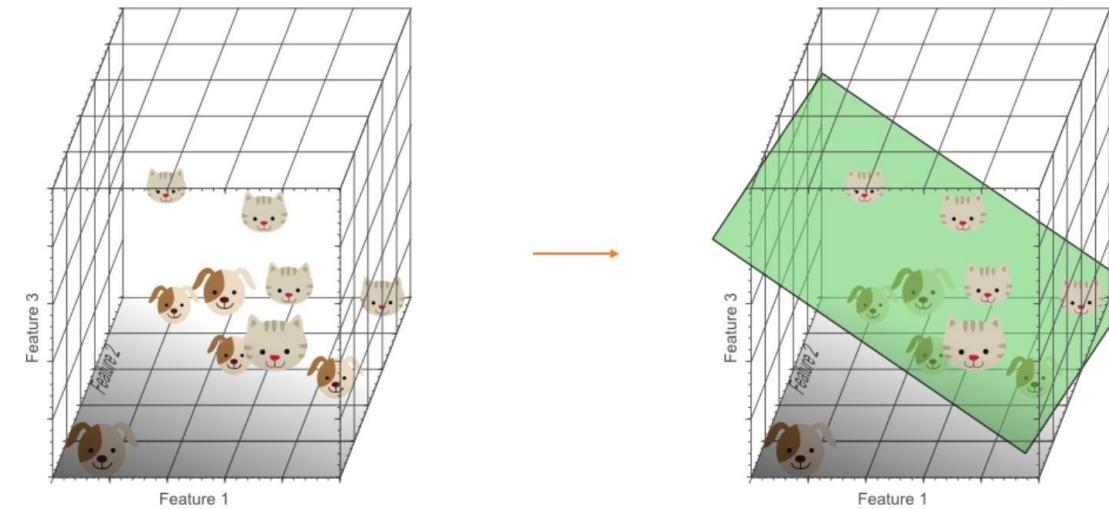
위치를 표현하기 위해 필요한 차원(변수의 수)을 줄이는 것



차원축소 Dimensionality Reduction

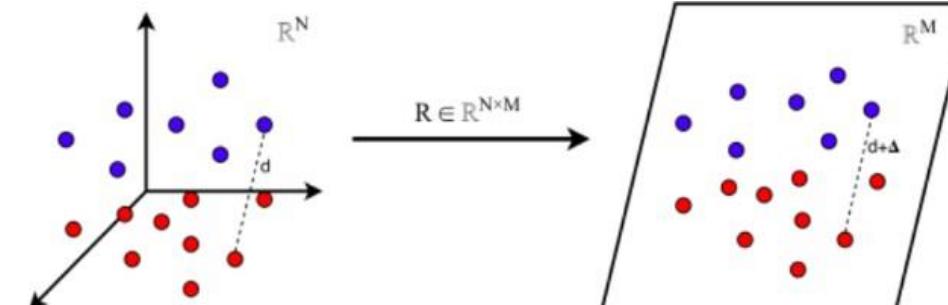
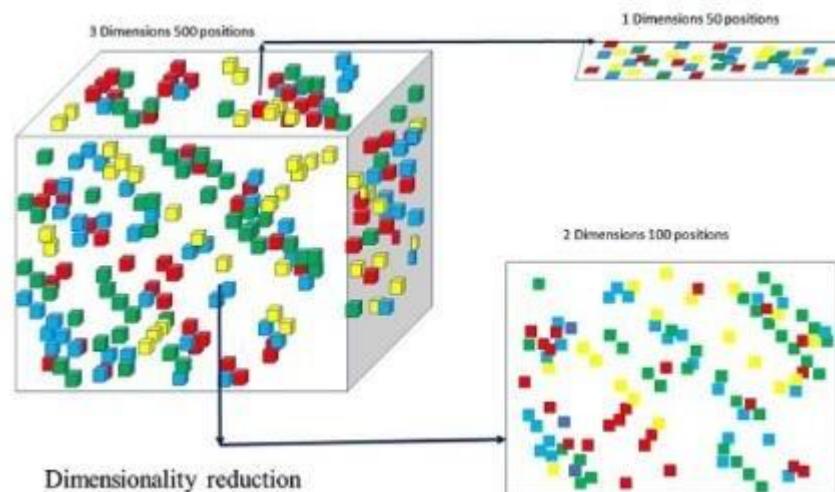
관찰 대상을 잘 설명할 수 있는 **잠재공간(Latent Space)**은 실제 관찰되어지는 공간보다 작을 수 있다.

단순히 데이터를 압축하는 것이 아니라 관측 데이터를 잘 설명할 수 있는 **잠재공간(Latent Space)**를 찾는 것



차원축소 Dimensionality Reduction

차원의 저주 해결
연산량 감소
시각화 용이



차원축소의 방법

1. 변수선택 Feature Selection

원본 데이터의 불필요한 특징 제거

키, ~~몸무게~~, 머리길이 → 키, 머리길이

몸무게를 제거한 나머지 변수만 선택

2. 변수추출 Feature Extraction

원본 데이터의 특징들 조합으로 **새로운 특징** 생성

키, **몸무게**, 머리길이 → ^{new}체구, 머리길이

키와 몸무게를 조합하여 '체구'라는 특징 생성

ex) 체구 = 몸무게 * 0.8 + 키 * 0.2

차원축소의 방법

1. 변수선택 Feature Selection

원본 데이터의 불필요한 특징 제거

키, ~~몸무게~~, 머리길이 → 키, 머리길이
Lasso

몸무게를 제거한 나머지 변수만 선택

2. 변수추출 Feature Extraction

원본 데이터의 특징들 조합으로 **새로운 특징** 생성

키, 몸무게, 머리길이 → **PCA, LDA^{new}...**

키와 몸무게를 조합하여 '체구'라는 특징 생성

ex) 체구 = 몸무게 * 0.8 + 키 * 0.2

Contents

I. Unit 01	I. intro: Dimensionality Reduction
I. Unit 02	I. Eigen-Value Decomposition
I. Unit 03	I. PCA: Principal Component Analysis
I. Unit 04	I. LDA: Linear Discriminant Analysis
I. Unit 05	I. Manifold Learning

고유값과 고유벡터 Eigen value & Eigen vector

Eigen vector : 고유벡터

$n \times n$ 정방행렬 A에 대해 $Ax = \lambda x$ 를 만족하는 0이 아닌 열 벡터

정방행렬(square matrix) : 행과 열의 개수가 같은 행렬

Eigen value : 고유값

Eigen vector의 상수 λ 값

$$Ax = \lambda x \quad (\text{상수 } \lambda)$$

$$\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \lambda \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$$

고유값과 고유벡터 Eigen value & Eigen vector <기하학적 의미>

Eigen vector : 고유벡터

정방행렬 A를 선형변환으로 봤을 때,
선형변환 A에 의한 변환 결과가 자기자신의 λ 배(상수배)가 되는 0이 아닌 벡터

$$Ax = \lambda x \quad (\text{상수 } \lambda)$$

Eigen value : 고유값

선형 변환 A에 의하여 벡터의 크기가 변하는 정도

$$\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \lambda \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$$

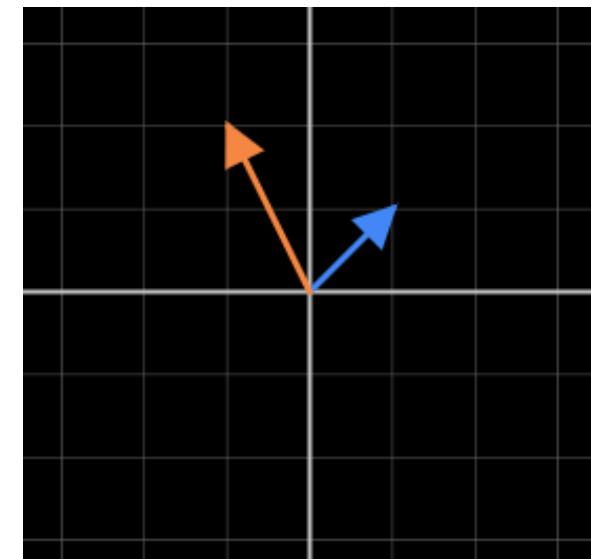
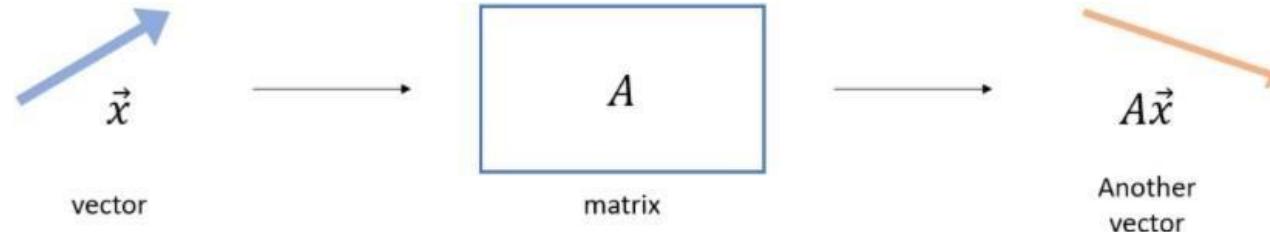
고유값과 고유벡터 Eigen value & Eigen vector

<기하학적 의미>

선형변환

좌표공간 내에서 일어날 수 있는 모든 변환

행렬이 벡터를 변환시켜 다른 벡터를 출력



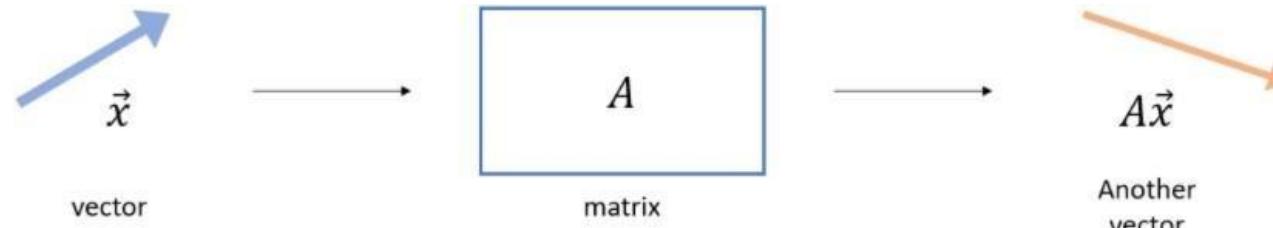
고유값과 고유벡터 Eigen value & Eigen vector

<기하학적 의미>

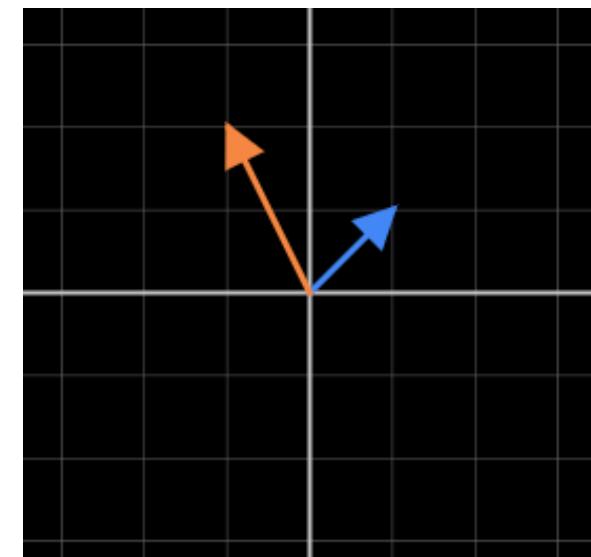
선형변환

특정한 벡터와 행렬은 선형변환 결과 크기만 바뀌고 방향은 그대로

입력벡터 x 를 A 로 선형변환 시킨 결과가 **상수배**라는 의미



$$A = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \quad \begin{bmatrix} 1 \\ -1 \end{bmatrix} \quad \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$



Unit 02 | Eigen-value Decomposition

차원축소

1) Eigenvalue & Eigenvector 고유값과 고유벡터 $\mathbf{Ax} = \lambda x$

- 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 곱해 나온 벡터이다.

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

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

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

Unit 02 | Eigen-value Decomposition

차원축소

1) Eigenvalue & Eigenvector 고유값과 고유벡터 $Ax = \lambda x$

- 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.

기저 Basis

Basis 기저 (기준)

벡터 공간 V 가 있을 때, 이 벡터 공간을 표현할 수 있는 최소한의 단위 벡터 (크기가 1인 벡터)

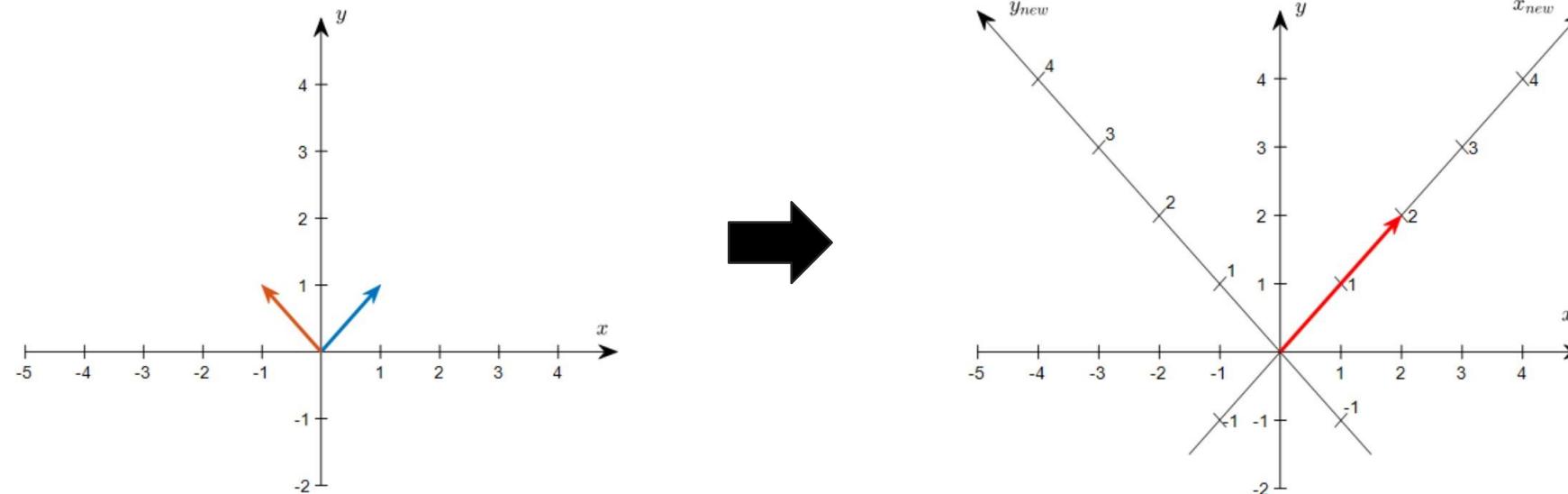
2차원 공간: xy평면, 3차원 공간: xyz공간 차원: 기저벡터의 개수!

$$\text{One basis for } R^2: \begin{bmatrix} v_1 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} v_2 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

$$\text{One basis for } R^3: \begin{bmatrix} v_1 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} v_2 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} v_3 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

기저 변환 Basis transform

벡터 공간을 표현할 수 있는 최소한의 단위인 기저를 변경



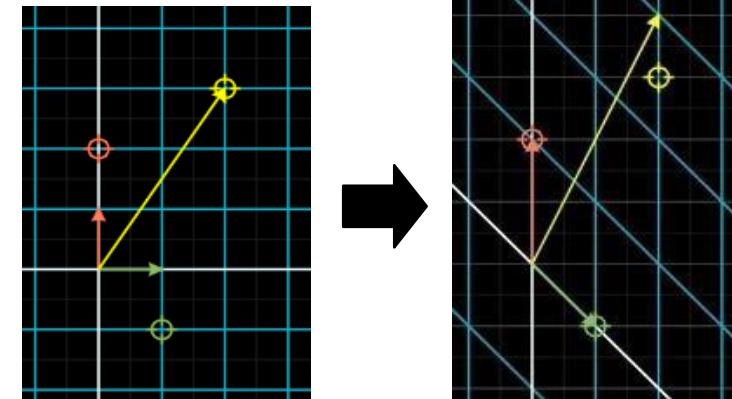
기저 변환 Basis transform

벡터 공간을 표현할 수 있는 최소한의 단위인 기저를 변경

ex) xy평면의 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, $2\begin{bmatrix} 1 \\ -1 \end{bmatrix} + 3\begin{bmatrix} 0 \\ 2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -1 & 2 \end{bmatrix} \begin{bmatrix} 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \end{bmatrix}$

이때, 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 변환**된 벡터이다.



고유벡터 정규화 (크기를 1로!)

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

수식적 접근

- Eigen decomposition 고유값 분해

$$Av = \lambda v$$

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

A 는 행렬, λ 는 상수이기 때문에

λ 에 단위행렬 $I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ 를 곱해서 행렬 꼴로 만든다.

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

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

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

특정방정식: 행렬식을 통하여 계산

수식적 접근

행렬식 determinant

- 역행렬(Inverse Matrix) : 원래 행렬과 곱했을 때, 단위행렬이 되는 행렬

$$A^{-1}A = AA^{-1} = I$$

- 계산법:

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

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

행렬식

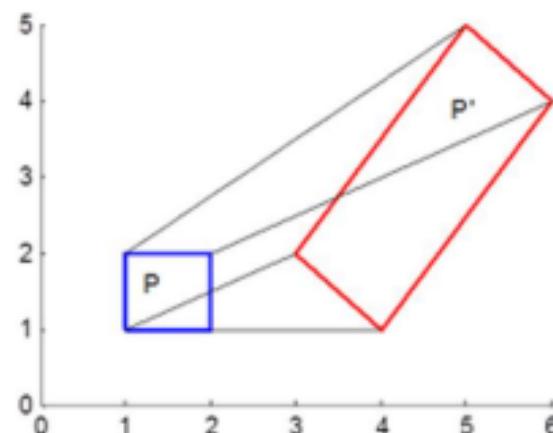
- 행렬식 계산식: $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \rightarrow \det(A) = ad - bc$

$$A = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix} \rightarrow \det(A) = a(ei-fh) - b(di-fg) + c(dh-eg)$$

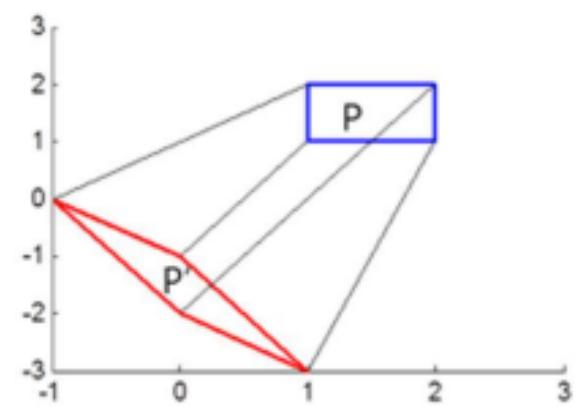
Unit 02 | Eigen-value Decomposition

차원축소

- 행렬식(determinant) <기하학적 의미>

면적(P') = $|det(A)| \times$ 면적(P) (2D의 경우)

$$A = \begin{pmatrix} 1 & 2 \\ -1 & 3 \end{pmatrix}, \det(A) = 5$$



$$A = \begin{pmatrix} 1 & -1 \\ -2 & 1 \end{pmatrix}, \det(A) = -1$$

도형 P (1,1), (1,2), (2,2), (2,1) 이루어짐
행렬 A 로 선형변환 하면

왼쪽 그림)
 P 의 면적 $1 *$ 행렬식 $5 =$ 변형된 P' 의 면적 5
 P 는 $(3,2), (5,5), (6,4), (4,1)$ 로 변환

오른쪽 그림)
 P 의 면적 $1 *$ 행렬식 $-1 =$ 변형된 P' 의 면적 1
 P 는 $(0,-1), (-1,0), (0,-2), (1,-3)$ 로 변환

Unit 02 | Eigen-value Decomposition

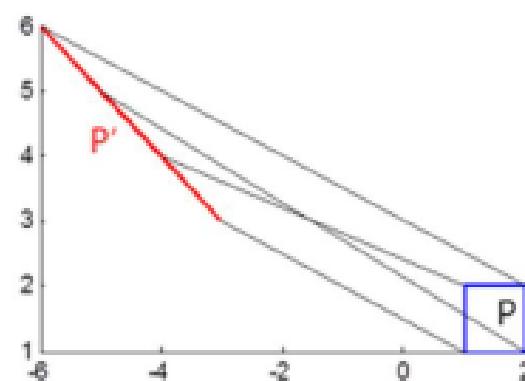
차원축소

- 행렬식(determinant)

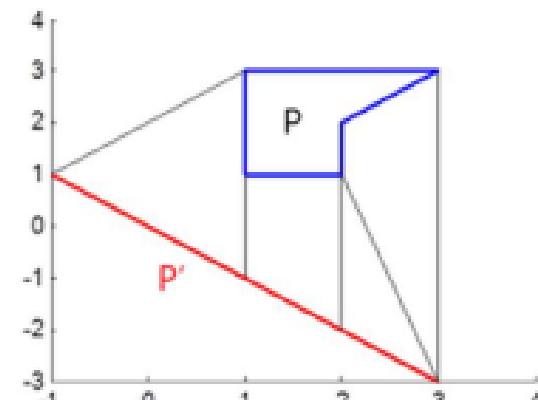
<기하학적 의미>

면적(P') = $|det(A)| \times$ 면적(P) (2D의 경우)

If) 행렬식이 0이라면???



$$A = \begin{pmatrix} -2 & -1 \\ 2 & 1 \end{pmatrix}, \det(A)=0$$



$$A = \begin{pmatrix} 2 & -1 \\ -2 & 1 \end{pmatrix}, \det(A)=0$$

직선(선분)으로 변환된다!

면적(P') = $|det(A)| \times$ 면적(P) (2D의 경우)

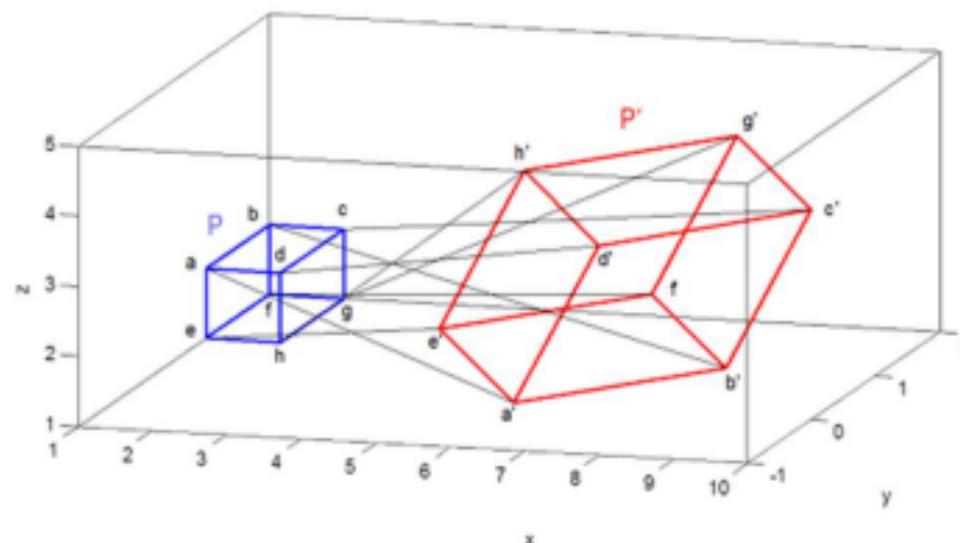
위의 식도 0으로 성립

Unit 02 | Eigen-value Decomposition

차원축소

- 행렬식(determinant) <기하학적 의미>

부피(P') = $|det(A)| \times$ 부피(P) (3D의 경우)



길이 1인 P를 A를 통해 변환 시키면

P의 부피 $1 * \text{행렬식 } -7 = \text{변형된 } P'\text{의 부피 } 7$

$$A = \begin{pmatrix} 2 & 2 & 1 \\ -1 & 1 & 0 \\ 3 & 0 & -1 \end{pmatrix}, \det(A) = -7$$

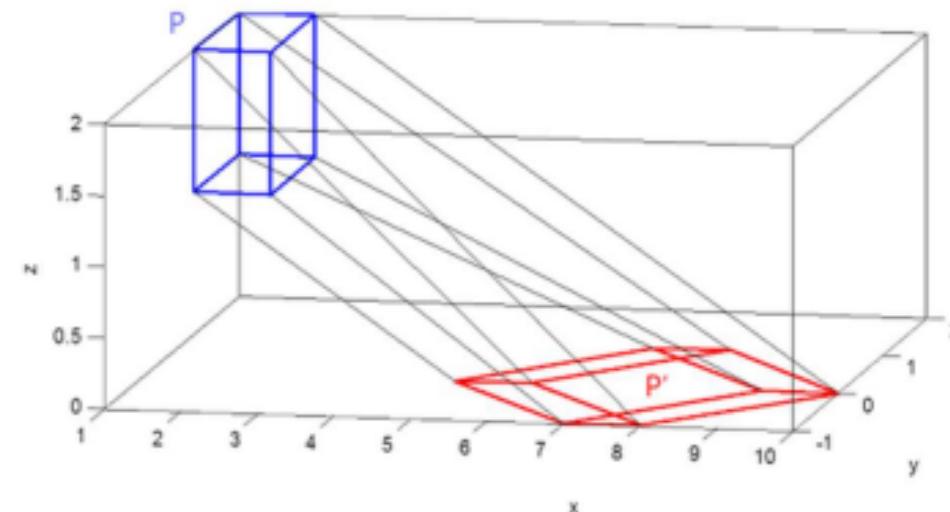
Unit 02 | Eigen-value Decomposition

차원축소

- 행렬식(determinant) <기하학적 의미>

$$\text{부피}(P') = |\det(A)| \times \text{부피}(P) \quad (3D \text{의 경우})$$

If) 행렬식이 0이라면???



길이 1인 P를 A를 통해 변환 시키면

$$P \text{의 부피 } 1 * \text{행렬식 } 0 = \text{변형된 } P' \text{의 부피 } 0$$

평면으로 바뀐다!!

행렬식이 의미하는 것 =>

$$A = \begin{pmatrix} 2 & 2 & 1 \\ -1 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \det(A) = 0$$

선형변환 될 때 단위 면적이 얼마만큼 늘어나는가?

수식적 접근

Eigen decomposition 고유값 분해

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

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

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

$$\begin{vmatrix} 1 - \lambda & 0 \\ -1 & 2 - \lambda \end{vmatrix} = (1 - \lambda)(2 - \lambda) - 0 * -1 = (1 - \lambda)(2 - \lambda) = 0 \quad \text{첫번째 고유값 1, 두번째 고유값 2}$$

⇒ 선형 변환을 했을 때, 그 크기는 변하고 방향이 변하지 않는 벡터가 있다고 할 때,

⇒ 그 벡터의 크기는 각각 1배, 2배가 된다!

수식적 접근

i) $\lambda = 1$

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

$$\begin{aligned} 0v_1 + 0v_2 &= 0 \\ v_1 + v_2 &= 0 \end{aligned} \rightarrow v_1 = v_2$$

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

정규화된 고유벡터 e로 표현

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

ii) $\lambda = 2$

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

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

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

정규화된 고유벡터 e로 표현

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

\Rightarrow 이 벡터들은 각각 선형변환 A를 취해주면 그 방향은 변하지 않고, 크기가 각각 1배, 2배가 된다!

스펙트럼 분해 Spectrum Decomposition

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

$$A\mathbf{v}_1 = \lambda_1 \mathbf{v}_1$$

$$A\mathbf{v}_2 = \lambda_2 \mathbf{v}_2$$

 \vdots

$$A\mathbf{v}_n = \lambda_n \mathbf{v}_n$$



$$A[\mathbf{v}_1 \mathbf{v}_2 \cdots \mathbf{v}_n] = [\lambda_1 \mathbf{v}_1 \lambda_2 \mathbf{v}_2 \cdots \lambda_n \mathbf{v}_n]$$

$$= [\mathbf{v}_1 \mathbf{v}_2 \cdots \mathbf{v}_n] \begin{bmatrix} \lambda_1 & & 0 \\ & \lambda_2 & \\ 0 & & \ddots & \lambda_n \end{bmatrix}$$

$$AP = P\Lambda$$

$$A = \lambda_1 * e_1 * e_1^T + \lambda_2 * e_2 * e_2^T + \cdots + \lambda_n * e_n * e_n^T$$

P : 고유 벡터의 열로 구성된 행렬

Λ : 고유값을 대각원소로 하는 대각행렬

$A = P\Lambda P^{-1}$

$$= \sum \lambda_i * e_i * e_i^T = [e_1 \ e_2 \ \cdots \ e_n] \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{bmatrix} [e_1^T \ e_2^T \ \vdots \ e_n^T]$$

$$= P\Lambda P^T (\text{단, } PP^T = P^TP = I)$$

스펙트럼 분해 Spectrum Decomposition

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

$$\begin{aligned}
 A &= P \Lambda P^{-1} \\
 &= \sum \lambda_i * e_i * e_i^T = [e_1 \ e_2 \ \dots \ e_n] \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_n \end{bmatrix} [e_1^T \ e_2^T \ \vdots \ e_n^T]
 \end{aligned}$$

- 대칭행렬은 모두 고유값분해 가능, 직교행렬로 분해 할 수 있다
- 대칭행렬의 고유벡터들의 내적은 0! 즉, 서로 직교한다.

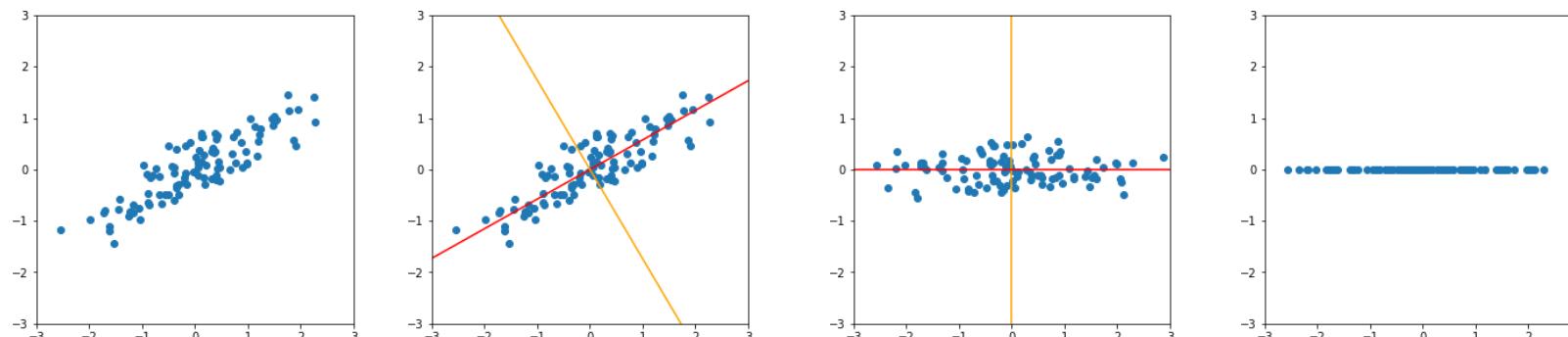
Contents

I. Unit 01	I. intro: Dimensionality Reduction
I. Unit 02	I. Eigen-Value Decomposition
I. Unit 03	I. PCA: Principal Component Analysis
I. Unit 04	I. LDA: Linear Discriminant Analysis
I. Unit 05	I. Manifold Learning

주성분 분석 Principal Component

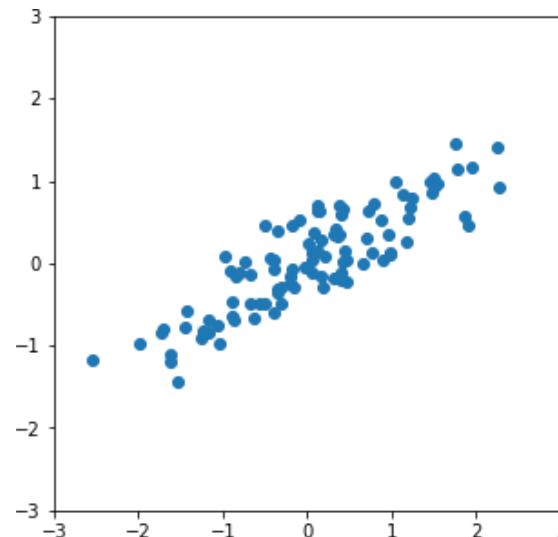
변수들의 전체분산 대부분을 소수의 주성분을 통하여 설명하는 것.

- 고차원 데이터의 최대 분산 방향을 찾아 새로운 공간에 저차원으로 투영하는 것
- 기하학적 측면에서 좌표축을 회전시켜 얻어진 새로운 좌표축 선택
⇒ 기존 변수들을 선형결합을 이용해 새로운 변수들(새로운 좌표)로 변환하는 것!
- ⇒ 이 때 좌표의 변환에 이용되는 것 = 고유벡터
- ⇒ 전체 고유값이 아닌 가장 큰 고유값 몇 개만 선택하여, 관련 고유벡터만을 적용하여 차원 축소



주성분 분석 Principal Component

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



cf) 공분산 행렬 Matrix의 각 원소들이 의미하는 것



주성분 분석 Principal Component

- Covariance Matrix 공분산 행렬
- 공분산 행렬의 스펙트럼 분해

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

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

$$= P \Lambda P'$$

P : 고유 벡터의 열로 구성된 행렬

Λ : 고유값을 대각원소로 하는 대각행렬

$$= \begin{bmatrix} 0.867 & 0.499 \\ -0.499 & 0.867 \end{bmatrix} \begin{bmatrix} 1.342 & 0 \\ 0 & 0.073 \end{bmatrix} \begin{bmatrix} 0.867 & -0.499 \\ 0.499 & 0.867 \end{bmatrix}$$

e_1

e_2

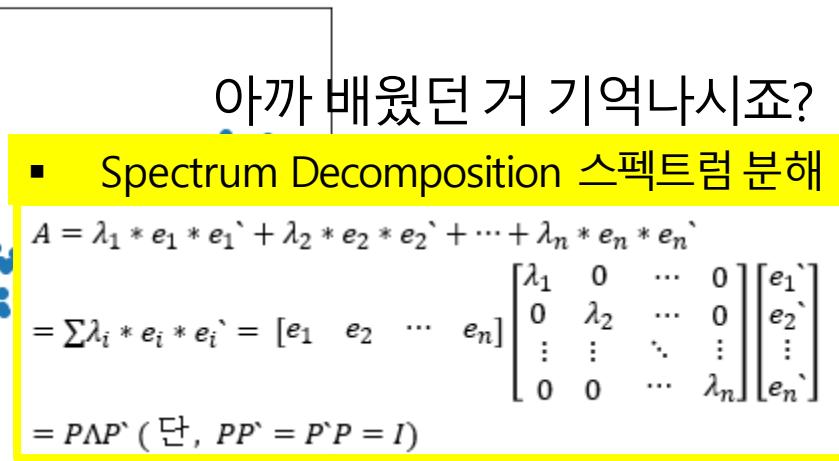
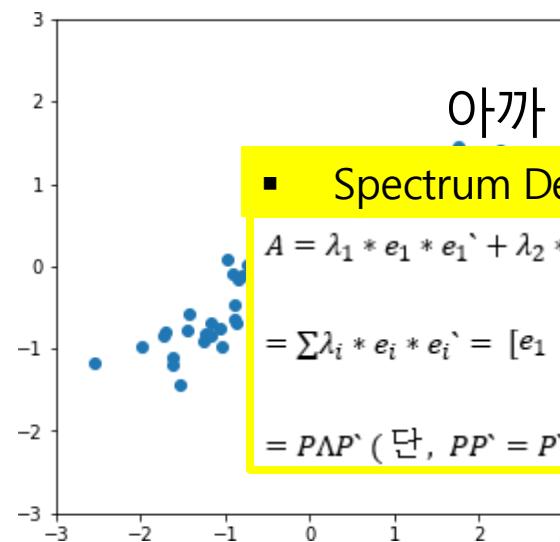
아까 배웠던 거 기억나시죠?

- Spectrum Decomposition 스펙트럼 분해

$$A = \lambda_1 * e_1 * e_1' + \lambda_2 * e_2 * e_2' + \dots + \lambda_n * e_n * e_n'$$

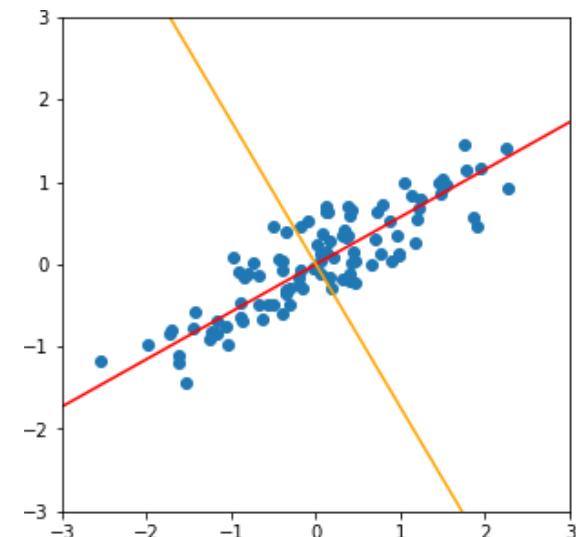
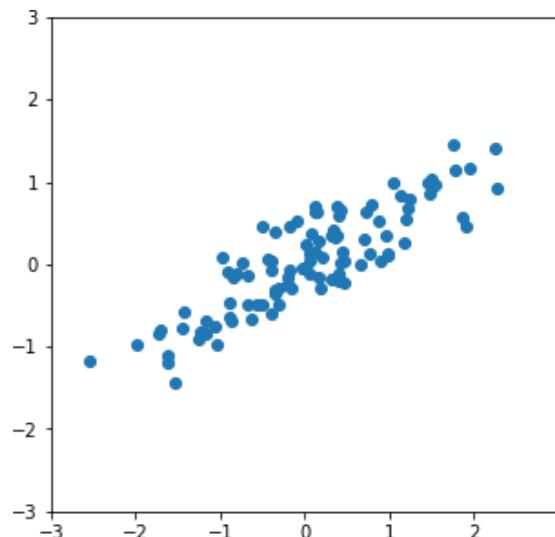
$$= \sum \lambda_i * e_i * e_i' = [e_1 \ e_2 \ \dots \ e_n] \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_n \end{bmatrix} \begin{bmatrix} e_1' \\ e_2' \\ \vdots \\ e_n' \end{bmatrix}$$

$$= P \Lambda P' \quad (\text{단, } PP' = P'P = I)$$



주성분 분석 Principal Component

- $\Sigma = \begin{bmatrix} 1.026 & 0.548 \\ 0.548 & 0.389 \end{bmatrix}$
- $\lambda_1 = 1.342, e_1 = \begin{bmatrix} 0.867 \\ -0.499 \end{bmatrix}, \lambda_2 = 0.073, e_2 = \begin{bmatrix} 0.499 \\ 0.867 \end{bmatrix}$



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

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

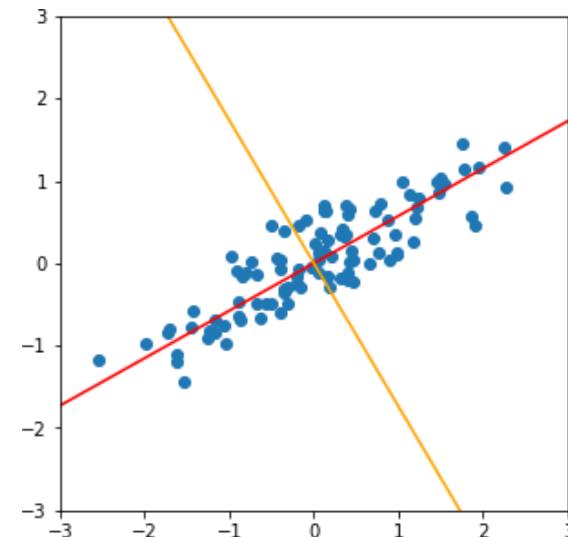
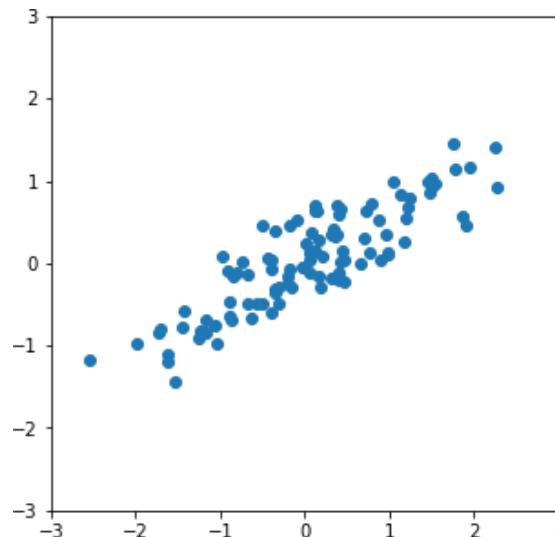
PC_i 의 분산 = λ_i

- 분산(고유값)이 큰 주성분부터 사용
- k개 주성분의 설명 비율
 $= (\lambda_1 + \lambda_2 + \dots + \lambda_k) / (\lambda_1 + \lambda_2 + \dots + \lambda_n)$

주성분 분석 Principal Component

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

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



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

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

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

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

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

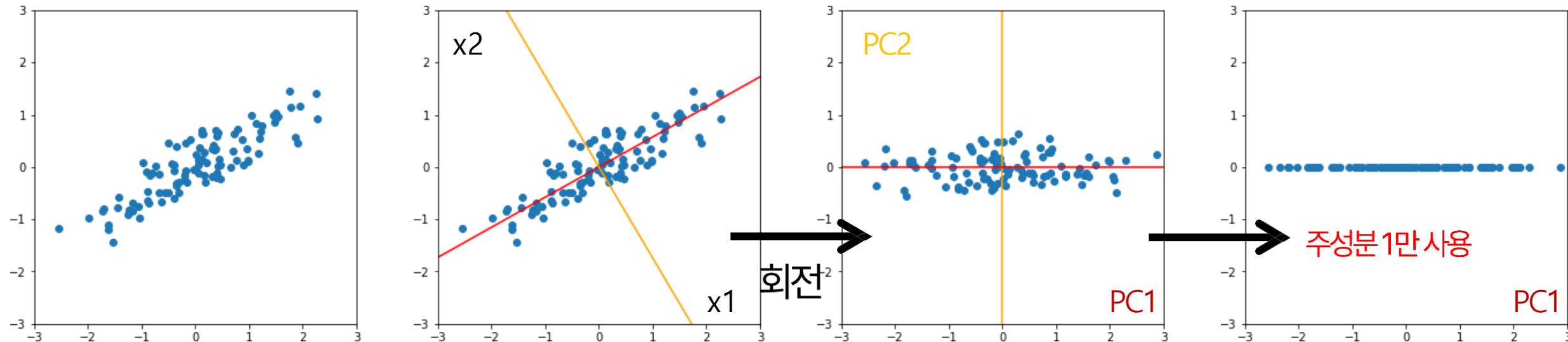
첫번째 주성분이 전체 분산의 95%를 설명?

- 실데이터에선 측정단위의 영향을 받음

: 표준화 사용!

주성분 분석 Principal Component

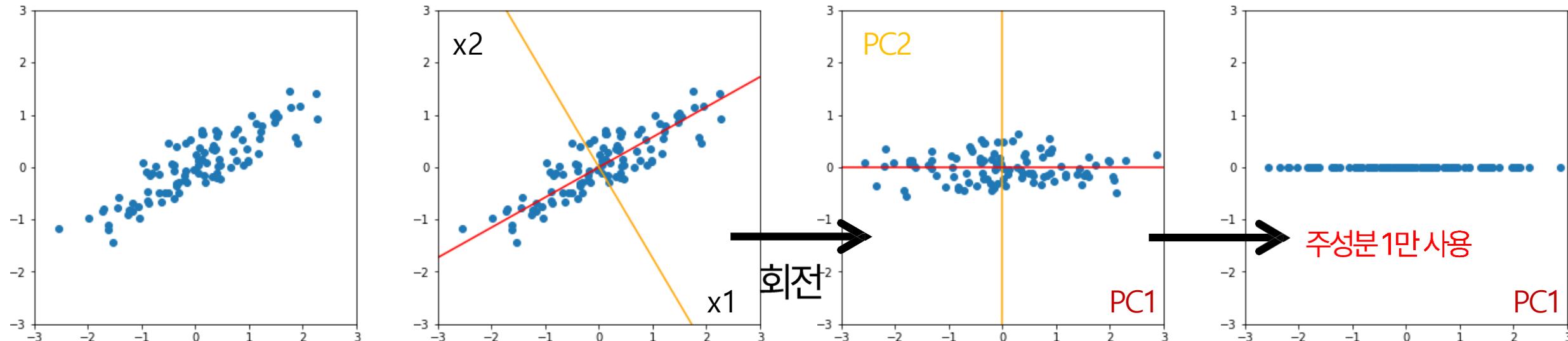
- 변환된 축이 최대 분산 방향과 정렬되도록 **좌표 회전**
: xy 평면의 기저를 정규화된 공분산 행렬의 고유벡터 행렬로 **기저변환**
- 전체분산 대부분을 설명하는 주성분 1로 차원축소 -> PC1 축으로 데이터 투영



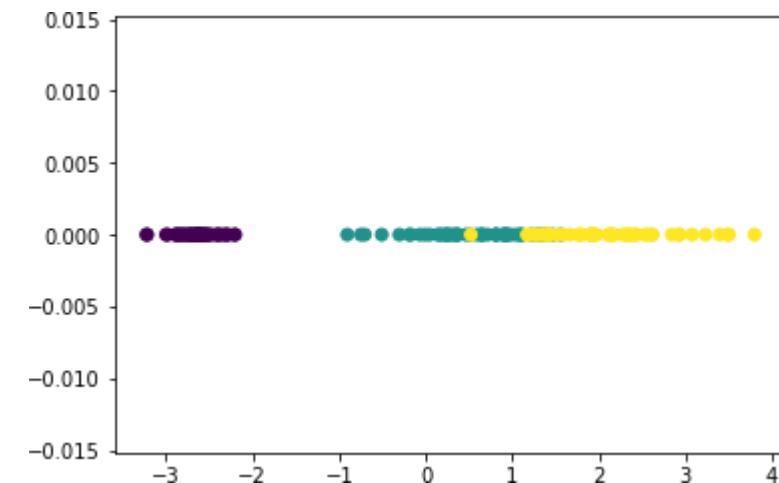
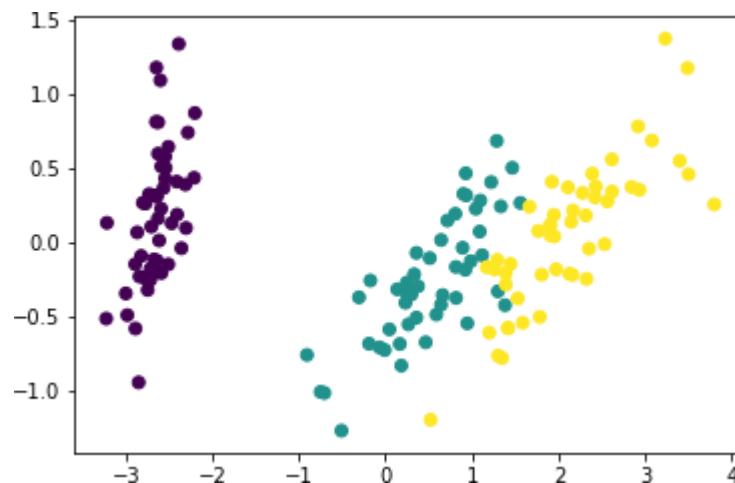
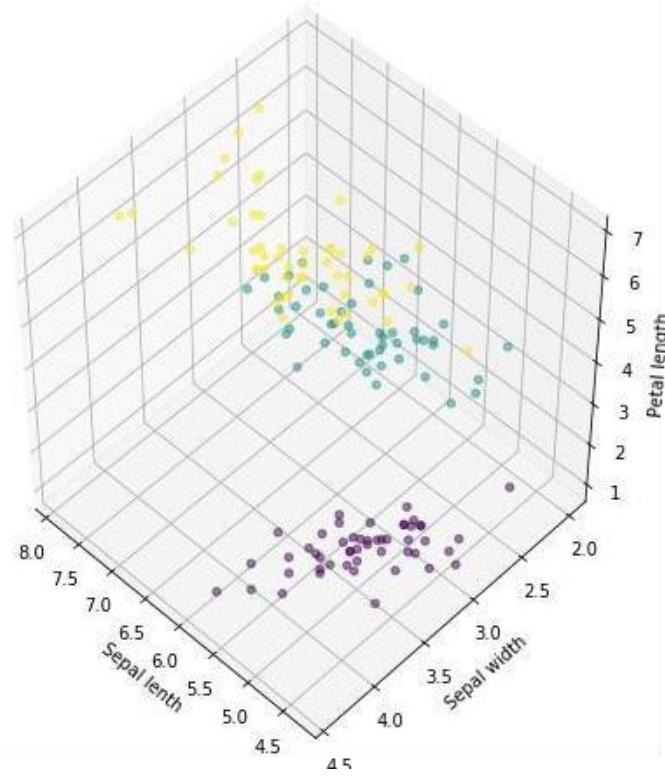
주성분 분석 Principal Component

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

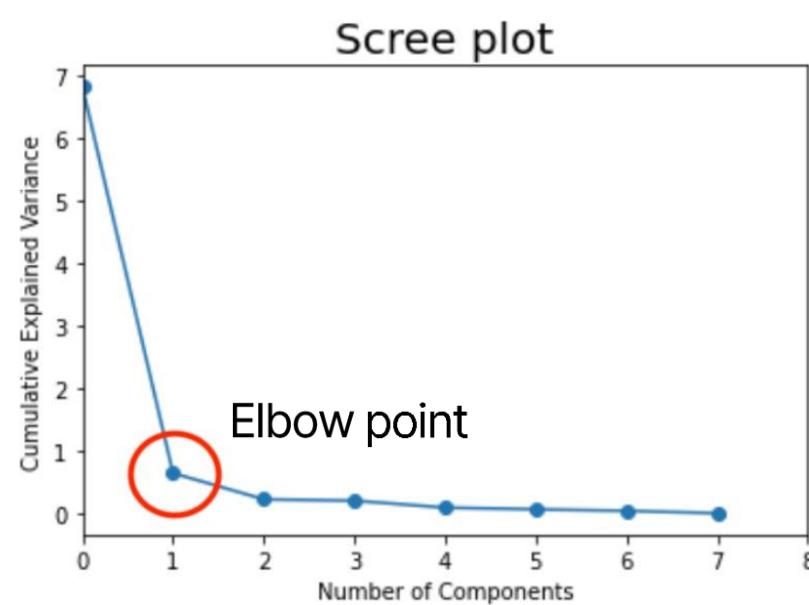
그 축과 직교하면서 분산이 다음으로 큰 두번째 축을 찾고 ... 반복



주성분 분석 Principal Component



주성분 개수의 결정



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

주성분 분석 Principal Component

- 과정을 정리해보면!
- 1) 데이터 표준화 -> 변수의 스케일의 차이로 인한 분산의 왜곡 방지
 - 2) 데이터에 대한 공분산 행렬 계산
 - 3) 공분산 행렬에 대한 Eigenvalue-decomposition을 통해 고유값과 고유벡터
(공분산 행렬을 가장 잘 설명하는 벡터)를 구한다
 - 4) 찾은 벡터를 몇 개 정하여(방금 전 그 기준들로)
해당 space로 투영시켜 데이터를 다루는 차원축소를 수행

Unit 03 | PCA : Principal Component Analysis

차원축소

■ 응용 - PCA Regression 주성분 회귀분석 -> 다중공선성 해결

- 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

주성분 분석 Principal Component

■ PCA의 가정

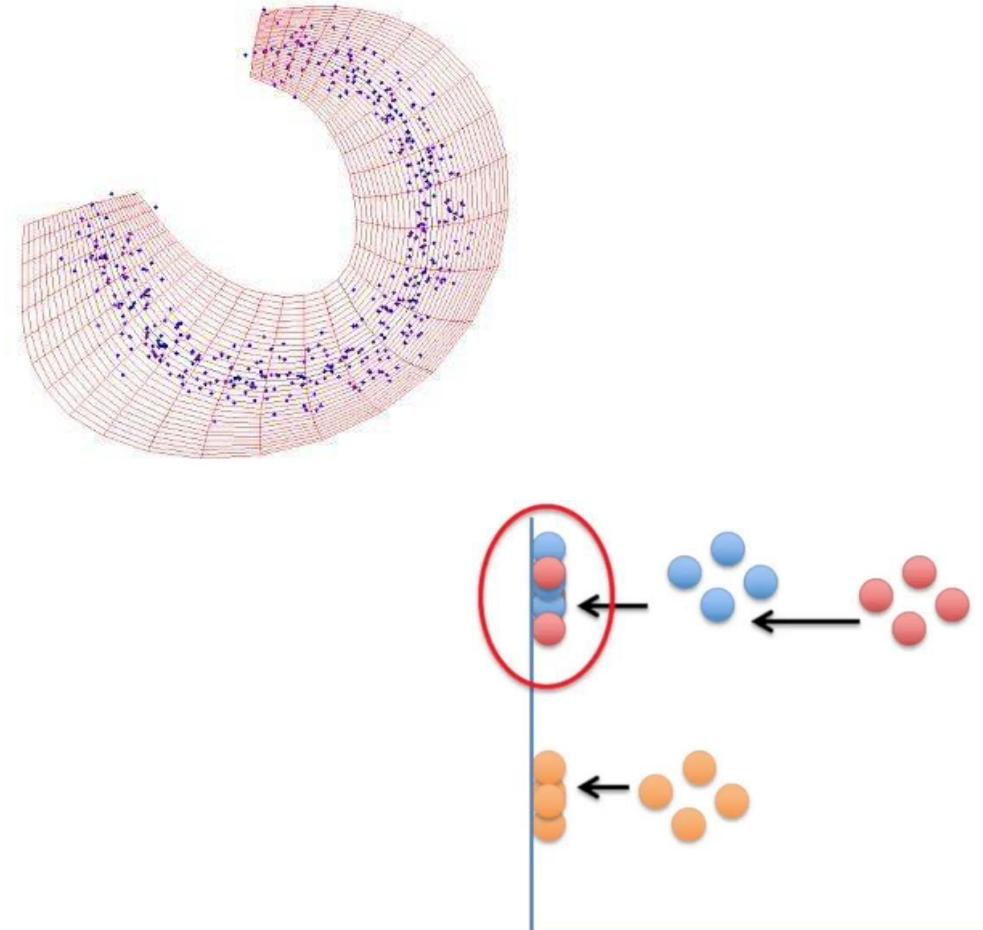
- Linearity : 데이터가 선형성을 띤다
- Orthogonality : 찾은 주축들은 서로 직교한다.
- 큰 분산을 갖는 방향이 중요한 정보를 담고 있다.

■ PCA의 장점

- 변수 간 상관관계 및 연관성을 이용해 변수 생성
- 차원 축소로 인한 차원의 저주 해결 (속도 상승 & 과적합 방지)
- 다중공선성 문제 해결

주성분 분석 Principal Component

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



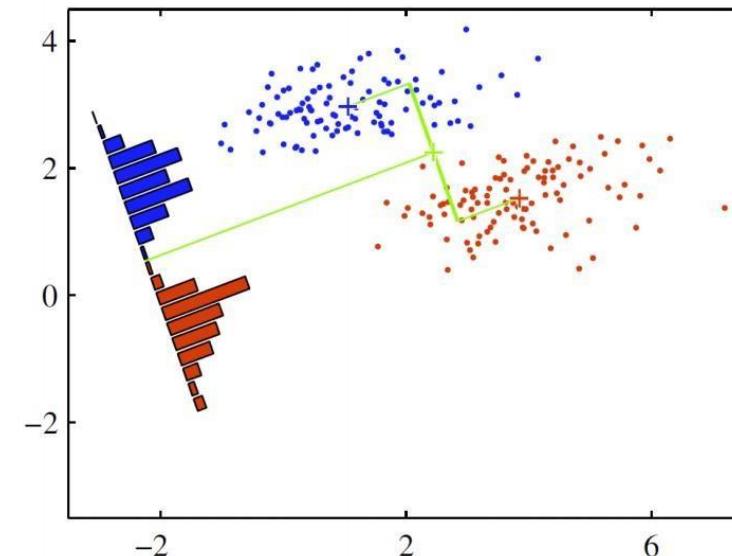
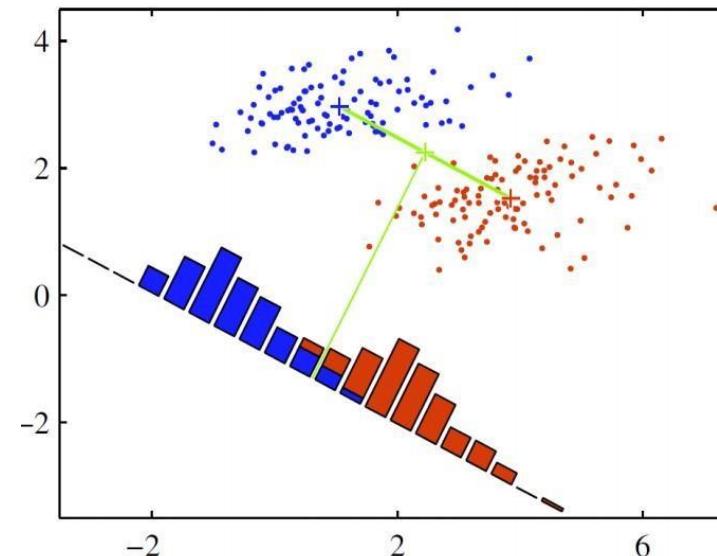
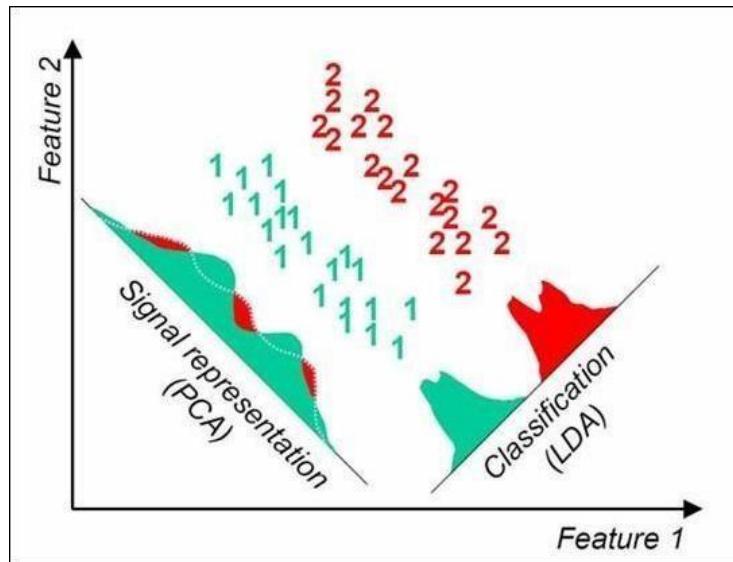
$$PC_1 = a_{11} * x_1 + a_{12} * x_2 + \dots + a_{1n} * x_n$$

Contents

I. Unit 01	I. intro: Dimensionality Reduction
I. Unit 02	I. Eigen-Value Decomposition
I. Unit 03	I. PCA: Principal Component Analysis
I. Unit 04	I. LDA: Linear Discriminant Analysis
I. Unit 05	I. Manifold Learning

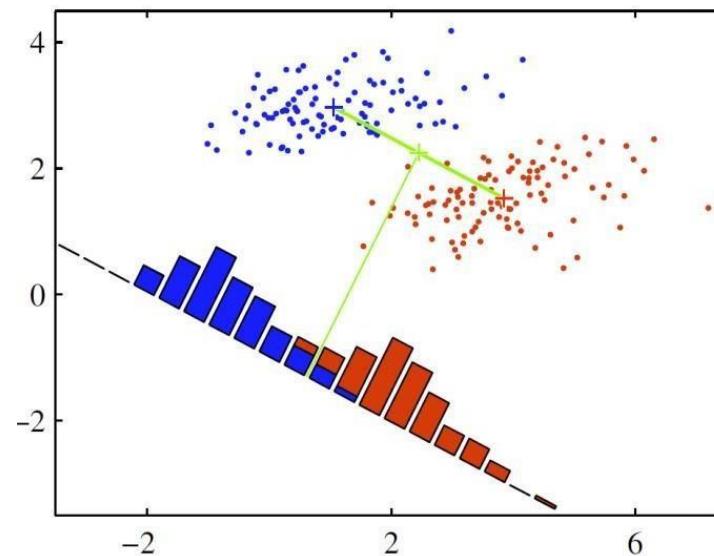
선형 판별 분석 Linear Discriminant Analysis

- 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.

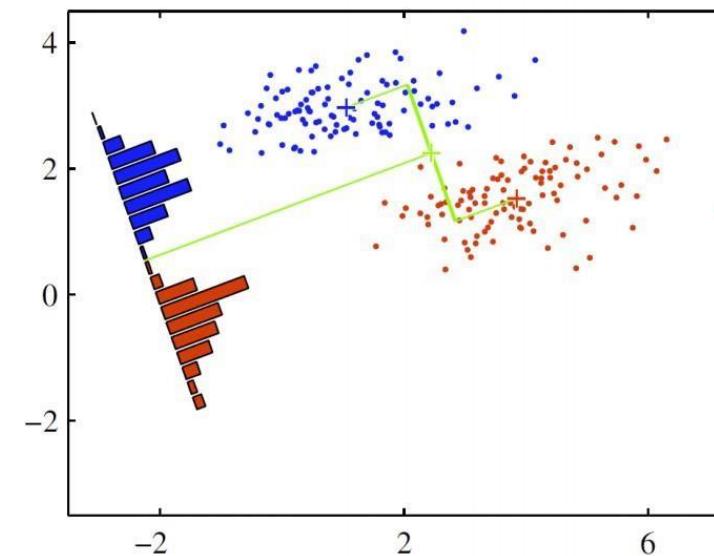


선형 판별 분석 Linear Discriminant Analysis

- 두 클래스의 중심점을 멀게 -> 클래스 간의 분산은 크게
- 클래스 내의 분산은 작게 => 클래스 내의 분산, 클래스 간의 분산 동시에 고려



VS



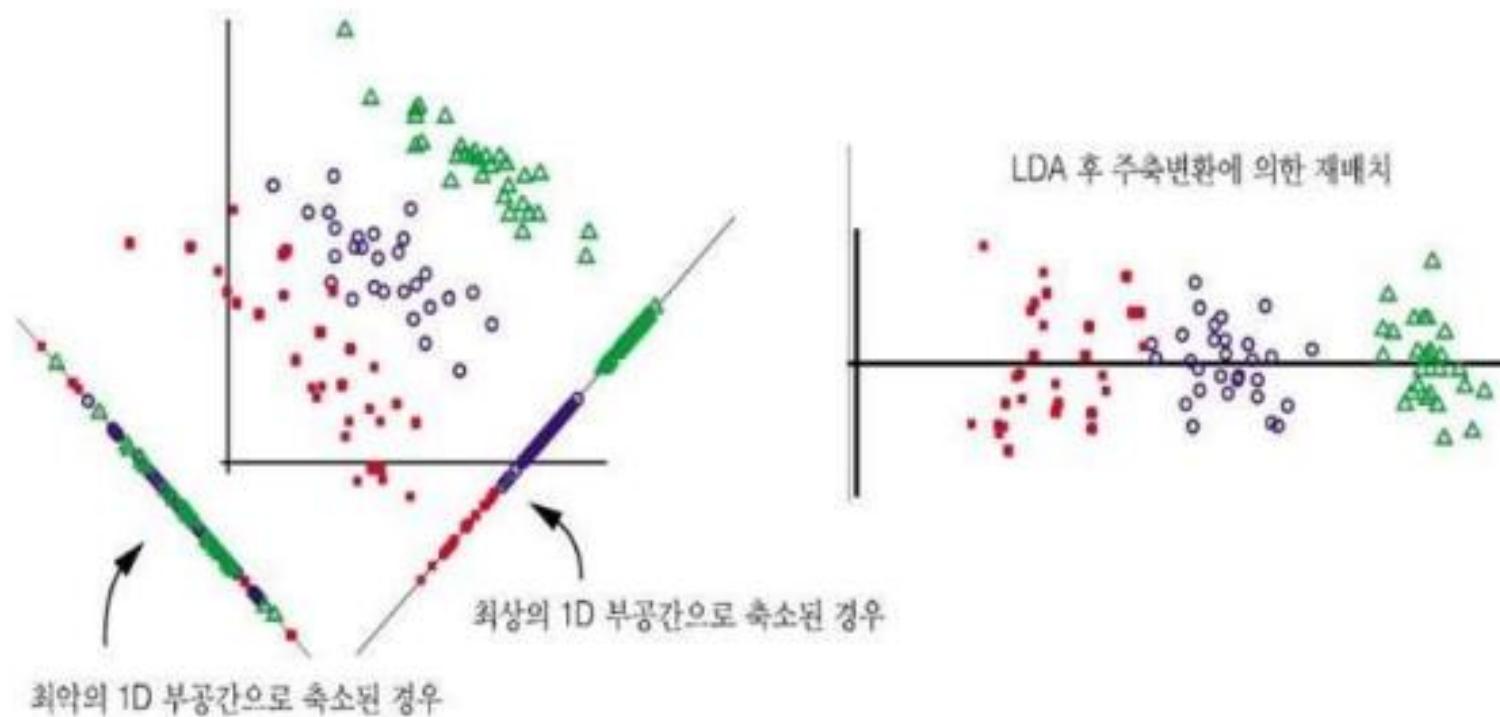
중심점 간의 거리가 크고 클래스 내의 분산이 작아서 더 좋은 분류기

Unit 03 | PCA : Principal Component Analysis

차원축소

■ LDA의 분류

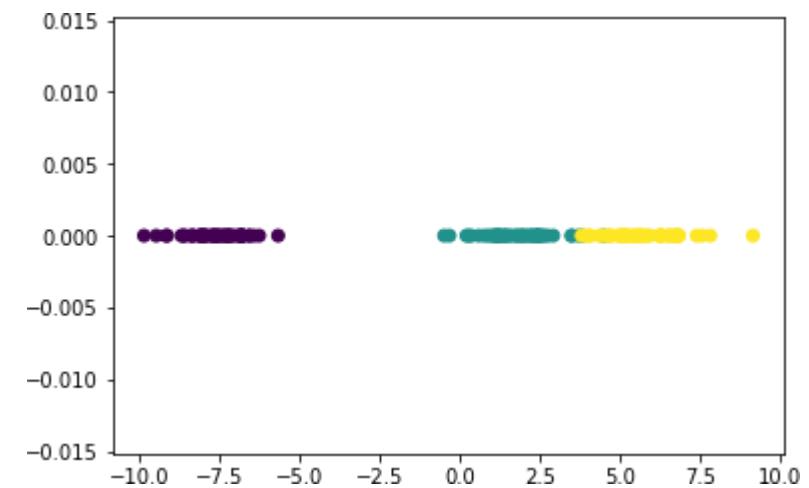
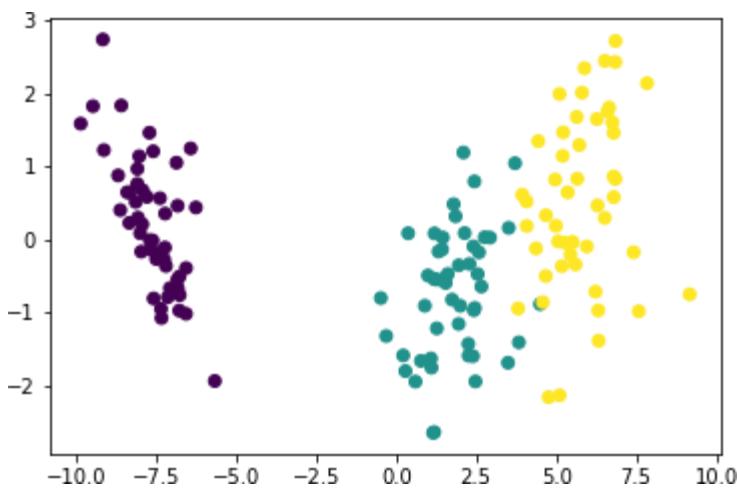
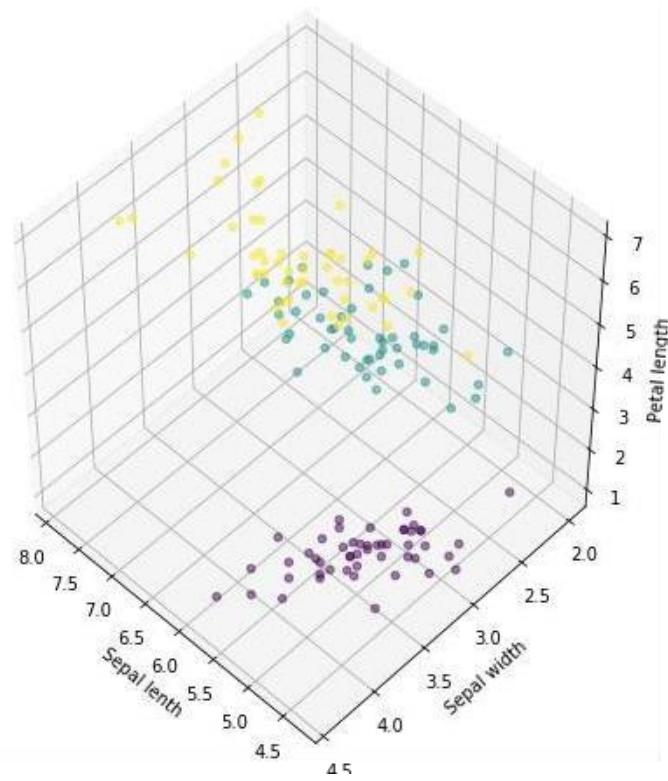
ex) 고유벡터 하나만 이용해 2d → 1d 변환



Unit 03 | PCA : Principal Component Analysis

차원축소

선형 판별 분석 Linear Discriminant Analysis



선형 판별 분석 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

Contents

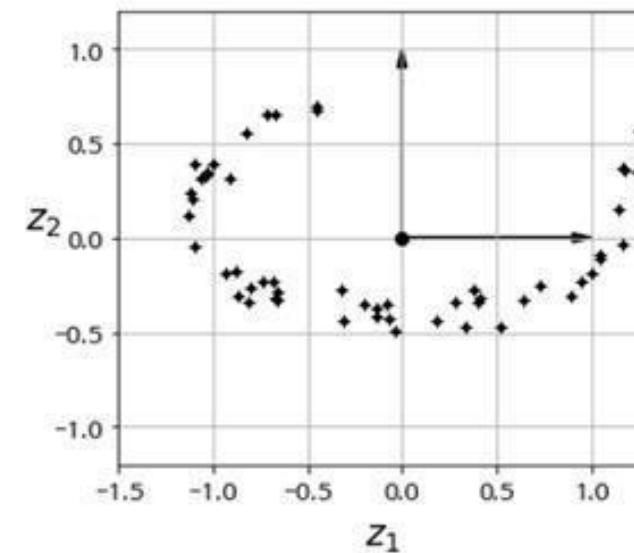
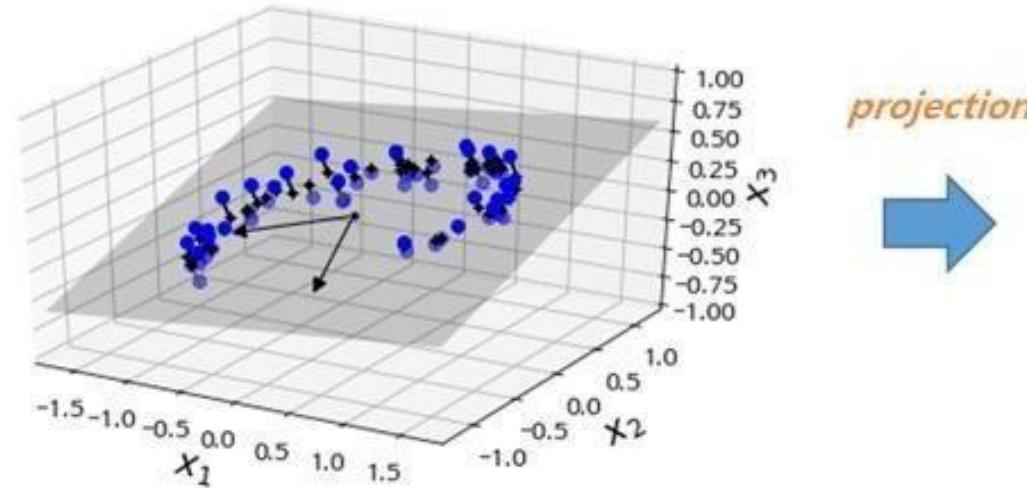
I. Unit 01	I. intro: Dimensionality Reduction
I. Unit 02	I. Eigen-Value Decomposition
I. Unit 03	I. PCA: Principal Component Analysis
I. Unit 04	I. LDA: Linear Discriminant Analysis
I. Unit 05	I. Manifold Learning

Unit 05 | Manifold Learning

매니폴드 학습

Manifold Learning

PCA와 LDA는 Eigen decomposition 을 기반으로
데이터를 새로운 축에 '선형으로' projection(투영)시키는 과정이 핵심이었다!



Unit 05 | Manifold Learning

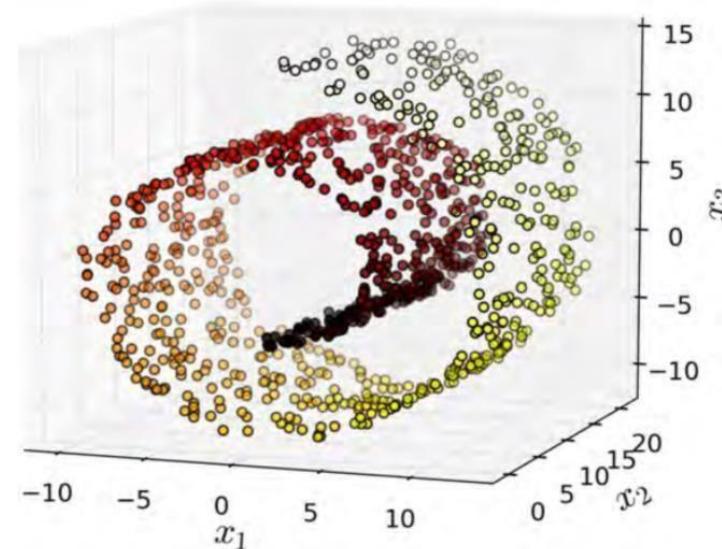
매니폴드 학습 Manifold Learning

- 방금까지 알아본 PCA와 LDA는 Eigen decomposition 을 기반으로 데이터를 새로운 축에 '선형으로' projection(투영)시키는 과정이 핵심이었다!

⇒ 하지만, 이것은 부분공간이 선형적인 상황에서만 가능하다...

⇒ 이런 생긴 친구는 어쩌지...?

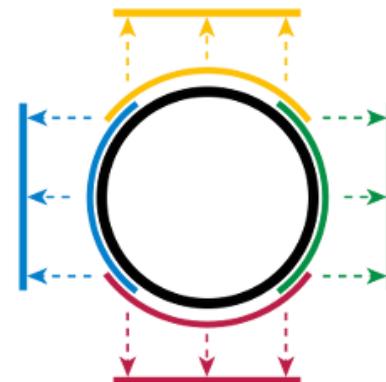
⇒ Manifold learning!



매니폴드 학습 Mainfold Learning

- 매니폴드 (manifold)

- 다양체라고도 하며, 국소적으로 유클리드 공간과 닮은 위상공간
- 즉, 국소적으로는 유클리드 공간과 구별할 수 없으나, 대역적으로 독특한 위상수학적 구조를 가질 수 있다
- Ex)



=> 모든 점에 대해서 국소적으로 적선과 같은 구조를 가지는 1차원 매니폴드

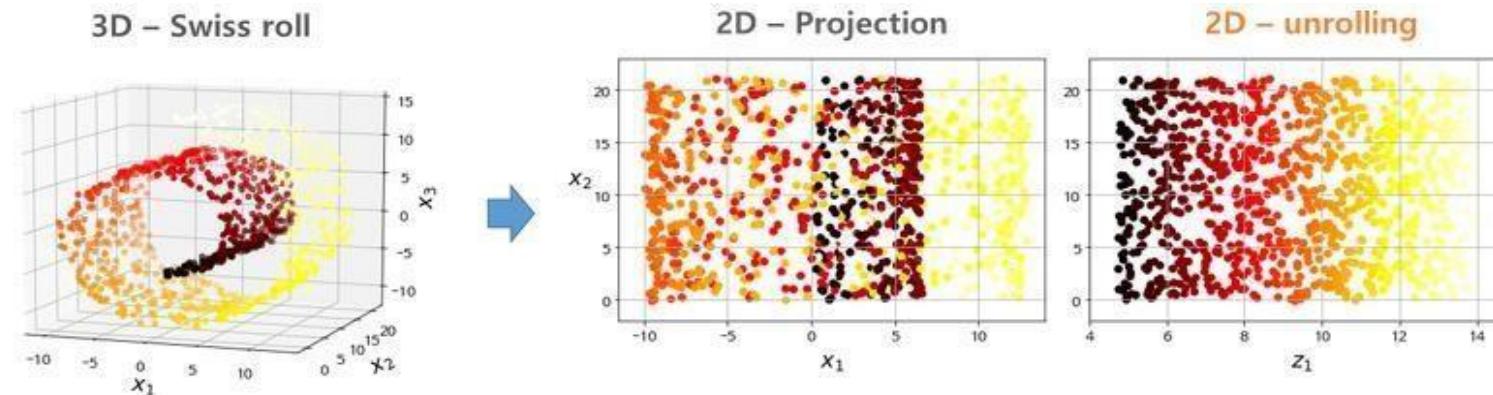
Unit 05 | Manifold Learning

매니폴드 학습

Manifold Learning

■ 매니폴드 (manifold)

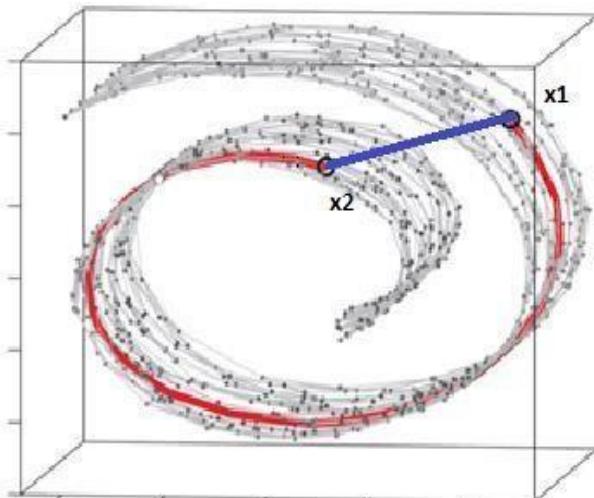
- 일반적으로, d -차원 매니폴드는 국소적으로 d -차원 초평면으로 볼 수 있는 n -차원 공간의 일부($d < n$)
- Ex) 스위스 룰(swiss roll, 룰케이크 모양의 데이터셋) 데이터셋 (2D-매니폴드의 한 예)
2D-매니폴드: 고차원(3차원) 공간에서 휘거나 말린 2D 모양
 \Rightarrow 스위스 룰은 $d = 20$ 이고 $n = 3$ 인, 국소적으로는 2D 평면이지만, 3차원으로 말려있는 데이터



매니폴드 학습 Manifold Learning

- Manifold Learning 종류

- Manifold learning :
데이터 분포의 비선형(non-linear) 구조를 직접적으로 고려!
- Isomax : 국소적으로 eigen-decompose 하는 방법
- LLE(Locally Linear Embedding) :
국소적인 평면의 데이터들이 축소된 차원에서도 인접하도록 반영
- SNE(Stochastic Neighbor Embedding) :
LLE에서 했던 국소적 평면의 데이터들이 축소된 차원에서도 인접하게 되도록 확률적으로 계산



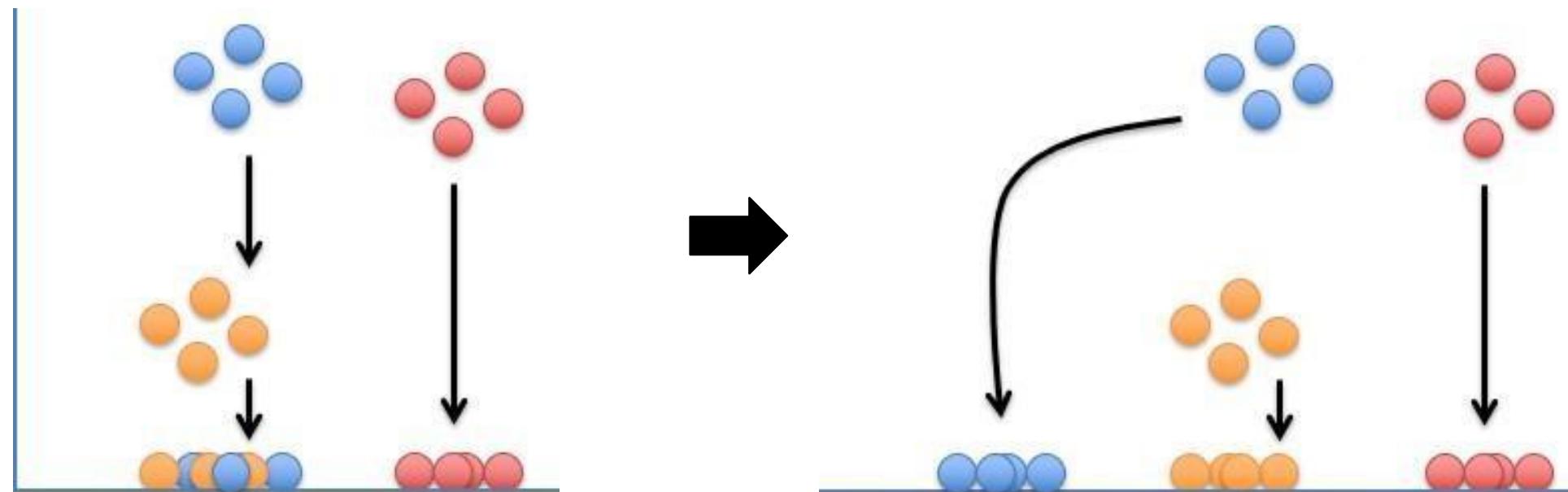
<https://wordbe.tistory.com/entry/Manifold-Learning-IsoMap-LLE-t-SNE-%EC%84%A4%EB%AA%85>

Unit 05 | Manifold Learning

■ T – Stochastic Neighbor Embedding T-분포 확률적 임베딩

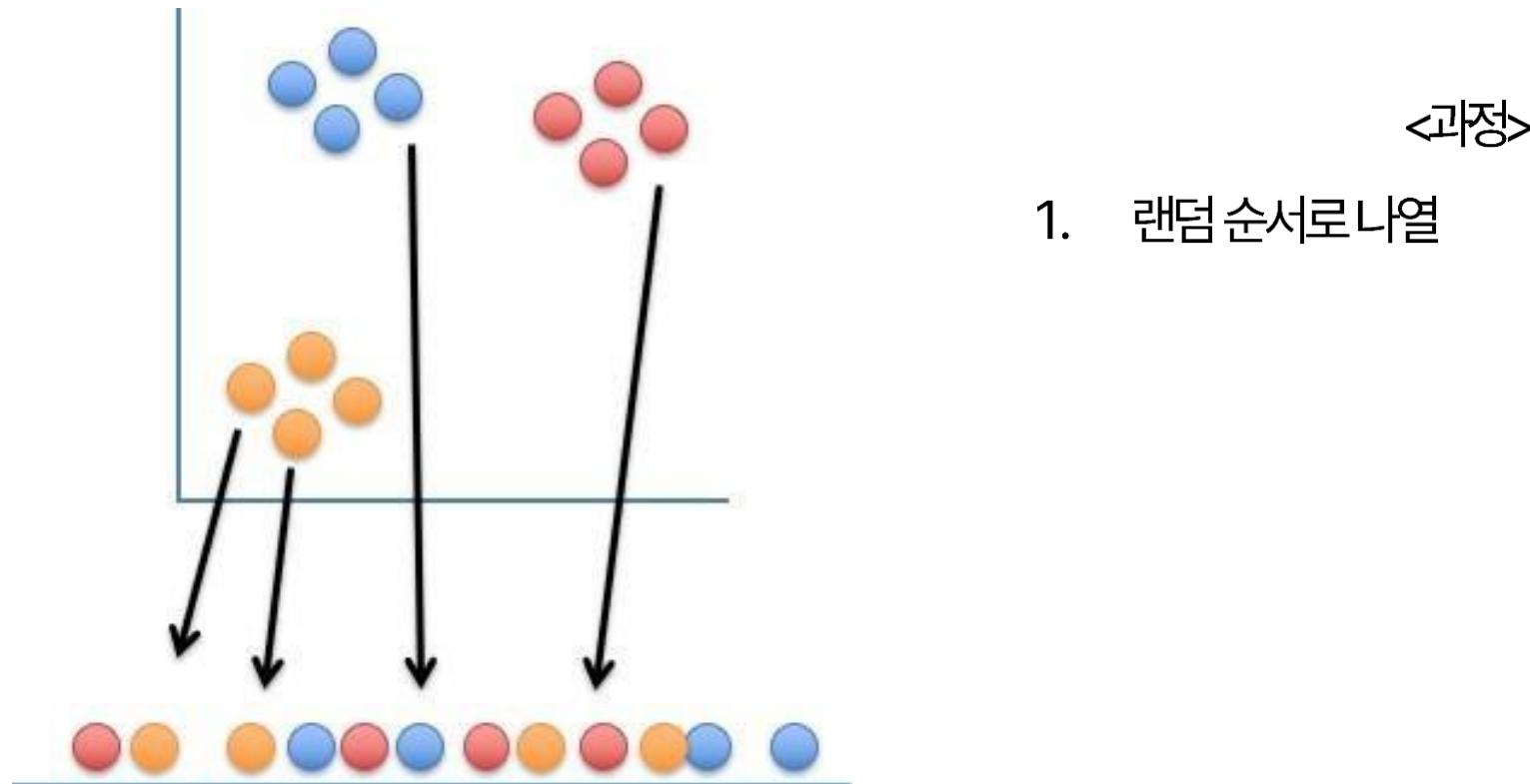
- 고차원의 데이터를 저 차원의 데이터로 거리 관계를 유지하며 **임베딩** 시키는 기법
- 직관적으로 데이터의 구조를 시각화하여 확인할 수 있다.

Ex) 50차원의 데이터 -> 2차원 평면 시각화



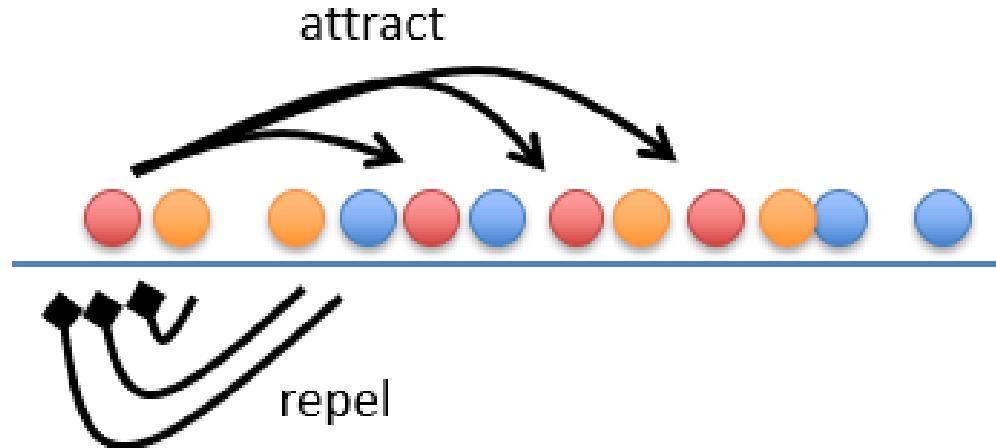
Unit 05 | Manifold Learning

■ T – Stochastic Neighbor Embedding T-분포 확률적 임베딩



Unit 05 | Manifold Learning

■ T – Stochastic Neighbor Embedding T-분포 확률적 임베딩



<과정>

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

- 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

■ T – Stochastic Neighbor Embedding T-분포 확률적 임베딩



<과정>

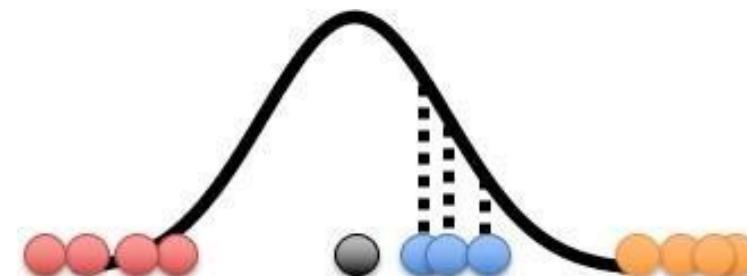
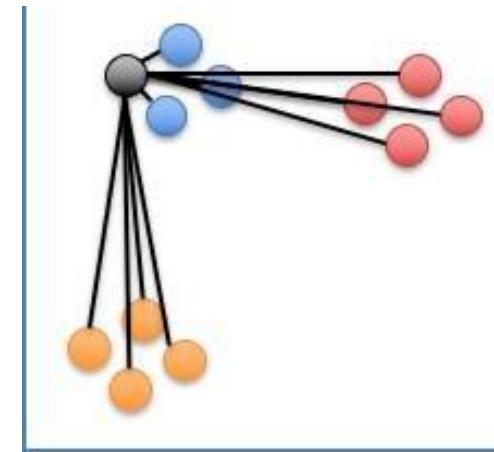
1. 랜덤 순서로 나열
2. 동일 군집은 당기는 힘 다른
군집은 미는 힘 작용
3. 힘의 균형에 맞는 위치로 이동
4. 모든 점들이 2~3 과정 반복

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

Unit 05 | Manifold Learning

■ T – Stochastic Neighbor Embedding T-분포 확률적 임베딩

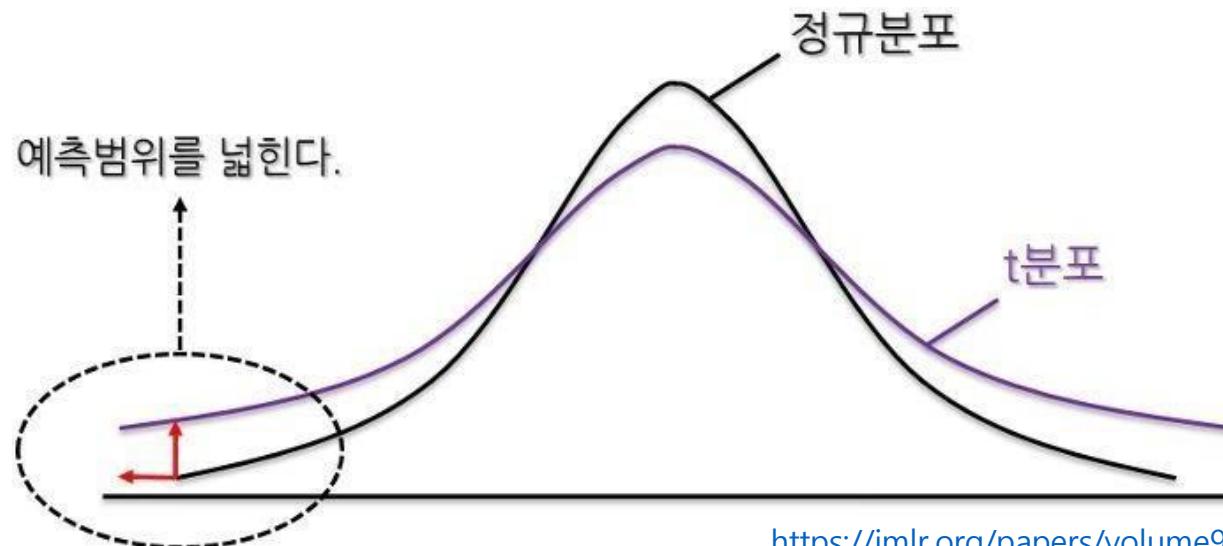
- 고차원 공간의 유클리디안 거리를 이용하여 저차원의 거리를 학습한다.
- 기존의 방법: SNE – 정규 분포 사용
 - 문제점
 - 고차원의 거리가 멀어지면 gradient 값이 0이 되어 저차원의 거리가 멀어지게 학습을 못 시킴
 - = 적당히 먼 거리와 매우 먼 거리를 구별 못함!



Unit 05 | Manifold Learning

■ T – Stochastic Neighbor Embedding T-분포 확률적 임베딩

- T-SNE: 자유도가 1인 T-분포 사용
 - 정규분포보다 꼬리가 두꺼워 예측 범위 상승
 - 가까운 거리를 더 깊게 먼 거리를 더 멀게 배치할 수 있게 됨



■ 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

Summary

차원축소

Summary

- 차원 축소

- **PCA:** 변수들의 **전체분산** 대부분을 소수의 **주성분**을 통하여 설명하는 것
고차원 데이터의 최대 분산 방향을 찾아 새로운 공간에 저차원으로 투영!
- **LDA:** 데이터의 분포를 학습하여 분리를 최적화하는 결정경계를 만들어 데이터를 분류하는 모델
- **T-SNE:** manifold learning
고차원의 데이터를 저차원의 데이터로 거리 관계를 유지하며 임베딩 시키는 기법

Homework

- Homework1: PCA 과정 차근차근 봅아보기
 - Week4_Dimensionality_Reduction_Assignment1.ipynb 파일을 실행해주세요.
- Homework2: 차원축소 Mnist data에 적용해보기
 - Week4_Dimensionality_Reduction_Assignment2.ipynb 파일을 실행해주세요.

Homework

- Homework3: 다른 차원축소기법 조사하기
 - UMAP과 PaCMAP 기법을 A4 한페이지로 정리해주세요.

UMAP : https://github.com/lmcinnes/umap_paper_notebooks/tree/master

PaCMAP : <https://github.com/YingfanWang/PaCMAP>

Appendix

- 참고자료

11기 임채빈님 강의자료
14기 박준영님 강의자료
16기 김주호님 강의자료
18기 지윤혁님 강의자료

feature selection : <https://www.analyticsvidhya.com/blog/2016/12/introduction-to-feature-selection-methods-with-an-example-or-how-to-select-the-right-variables/>

공돌이의 수학정리노트

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

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>