17기 정규세션

ToBig's 16기 강의자

김주호

Dimensionality Reduction

차원 축소

Contents

- 차원이란?

  : 공간 내의 있는 점 등의 위치를 나타내기 위해 필요한 축의 개수 (변수의 개수)

  → 하나의 데이터 셋이 n개의 변수를 지닌다면 n차원의 좌표 상에 표현할 수 있다.

측정일시	측정소 코드	미세먼지	초미세먼지	풍속	습도	풍향
18-1-2 15:00	108	32	17	3.7	28	268
18-1-2 13:00	108	52	27	3.7	30	275
18-1-2 14:00	108	37	19	3.6	28	262
18-1-2 23:00	104	27	22	4	29	296
18-1-2 23:00	103	39	22	4	29	296
18-1-2 14:00	102	26	14	3.5	23	71
18-1-2 14:00	117	33	16	3.4	23	255
18-1-2 23:00	114	31	18	3.4	36	161
18-1-2 22:00	114	28	17	3.4	34	158
18-1-2 16:00	114	30	17	3.4	23	100
18-1-2 15:00	114	33	17	3.3	23	97

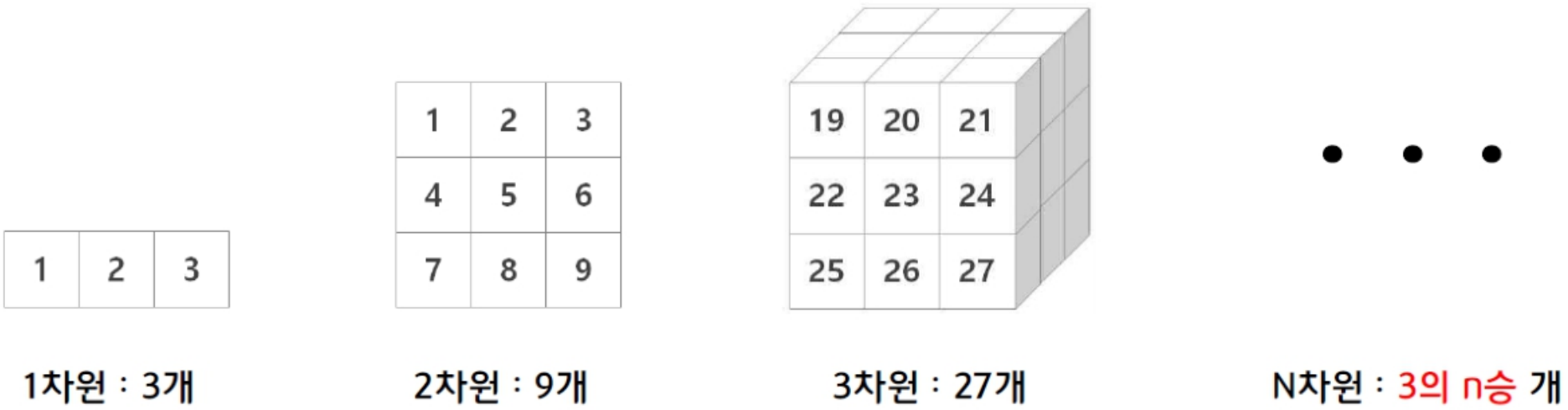
Unit 01 | Intro

- 차원을 축소해야하는 이유

  1) **차원의 저주**

  1-1) 변수의 수가 선형적으로 증가할 때, 동일한 설명력을 달성하기 위해 관측값의 수는 지수적으로 증가

  → 연산량 급증



- 차원을 축소해야하는 이유

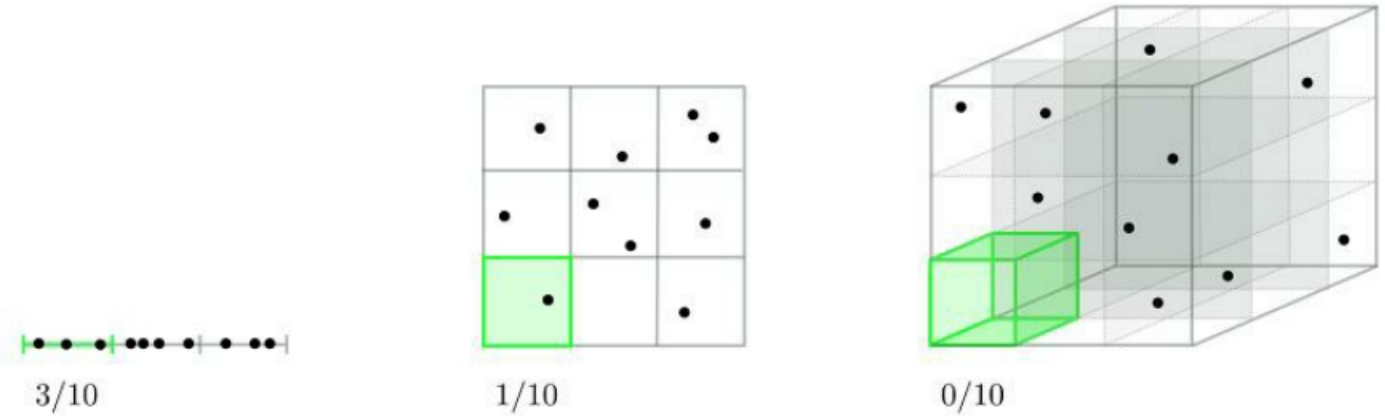
  1) **차원의 저주**

  1-1) 변수의 수가 선형적으로 증가할 때, 동일한 설명력을 달성하기 위한 관측값의 수는 지수적으로 증가

  → 연산량 급증

  1-2) 차원이 커질수록 해당 공간에서 데이터가 차지하는 밀도가 희소해짐

  → 과적합 및 성능 하락



Unit 01 | Intro

- 차원을 축소해야하는 이유

1) **차원의 저주**

: 연산량 급증 및 성능 하락

2) **Occam's Razor (Principle of parsimony)**

: Simple is the best, explainable model

3) **Intrinsic dimension < Original dimension**

: 객체의 본질적인 정보를 보존하는 내재적인 차원의 수는 실제 차원보다 훨씬 더 적은 경우가 많다.

Unit 01 | Intro

- 차원 축소 기법

1) 변수 선택 (Feature Selection)

: 원본 데이터의 불필요한 특징 제거

ex) 몸무게 변수 불필요하다고 판단

키, ~~몸무게~~, 머리 길이 → 키, 머리 길이

2) 변수 추출 (Feature Extraction)

: 원본 데이터의 특징을 조합으로 새로운 조합 생성

ex) 키와 몸무게를 결합한 새로운 파생변수 생성

키, 몸무게, 머리 길이 → 체구, 머리 길이

Unit 01 | Intro

- 차원 축소 기법

1) 변수 선택 (Feature Selection)

: 원본 데이터의 불필요한 특징 제거

ex) 키, ~~몸무게~~, 머리길이 → 키, 머리길이

Wrapper Method,
Filter Method,
Embedded Method

- **Wrapper Method** : 예측 정확도 측면에서 가장 좋은 성능을 보이는 변수의 집합을 구하는 방법
  ex) 전진 선택, 후방 제거, 단계별 선택, GA(Genetic Algorithm)

- **Filter Method** : 통계적 측정 방법을 사용하여 높은 상관관계를 가지는 피처를 사용하는 방법
  ex) Information gain, Correlation coefficient

- **Embedded method** : Filtering과 Wrapper의 장점을 결합한 방법으로, 모델의 중요도에 기여하는 피처를 선택하는 방법
  ex) Lasso, Ridge, Elastic, SelectFromModel

Unit 01 | Intro

- 차원 축소 기법

1) 변수 선택 (Feature Selection)

: 원본 데이터의 불필요한 특징 제거

ex) 키, ~~몸무게~~, 머리 길이 → 키, 머리 길이

Wrapper Method,
Filter Method,
Embedded Method

2) 변수 추출 (Feature Extraction)

: 원본 데이터의 특징을 조합으로 새로운 조합 생성

ex) 키와 몸무게를 결합한 새로운 파생변수 생성

키, 몸무게, 머리 길이 → 체구, 머리 길이

PCA,
LDA,
MDS,
T-SNE

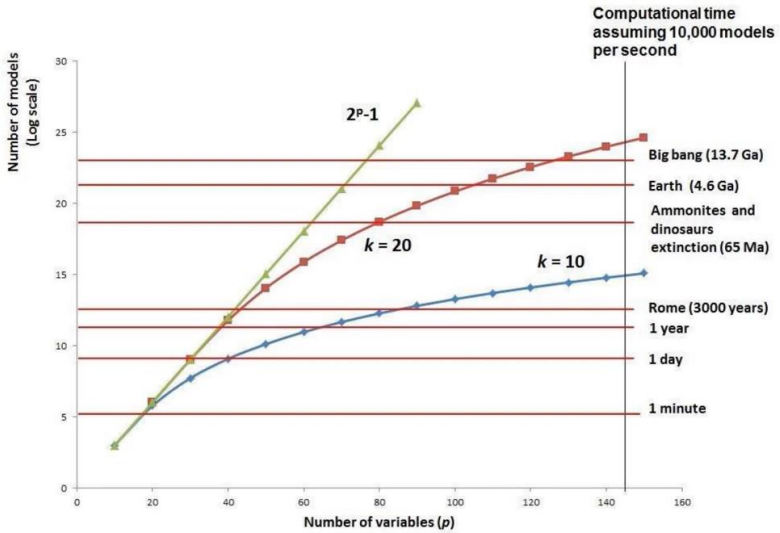
Contents

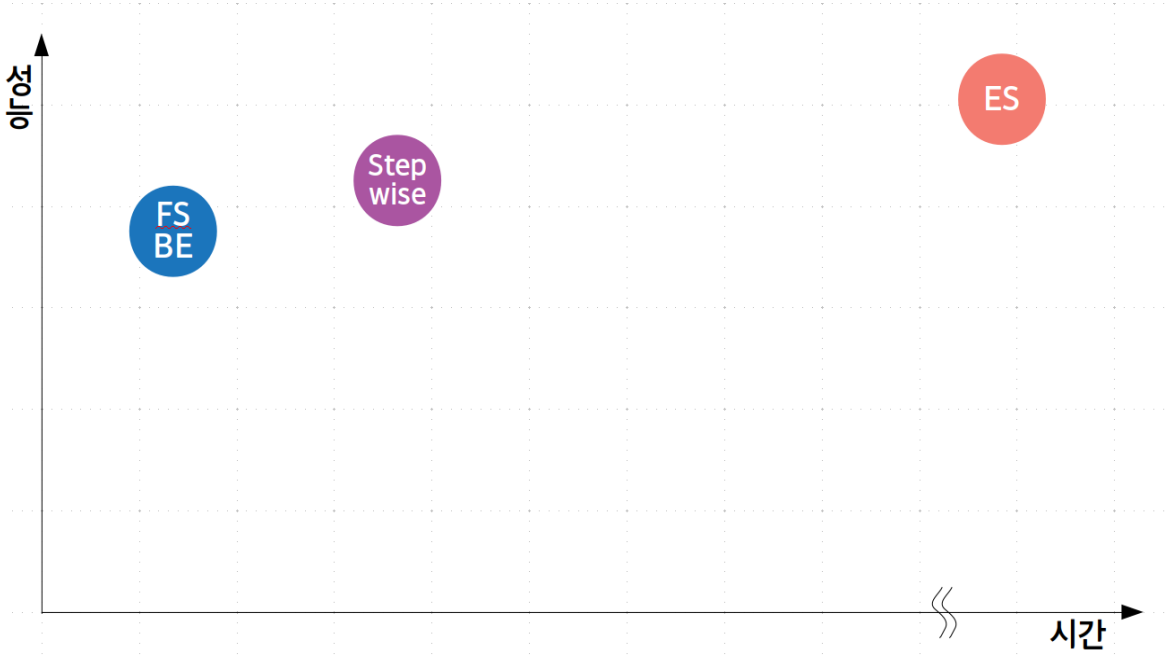
Unit 02 | Feature Selection techniques

▪ 변수 선택 기법들

Exhaustive Search

- Exhaustive search
  - ✓ Assume that we have a computer that can evaluate 10,000 models/second





Unit 02 | Feature Selection techniques

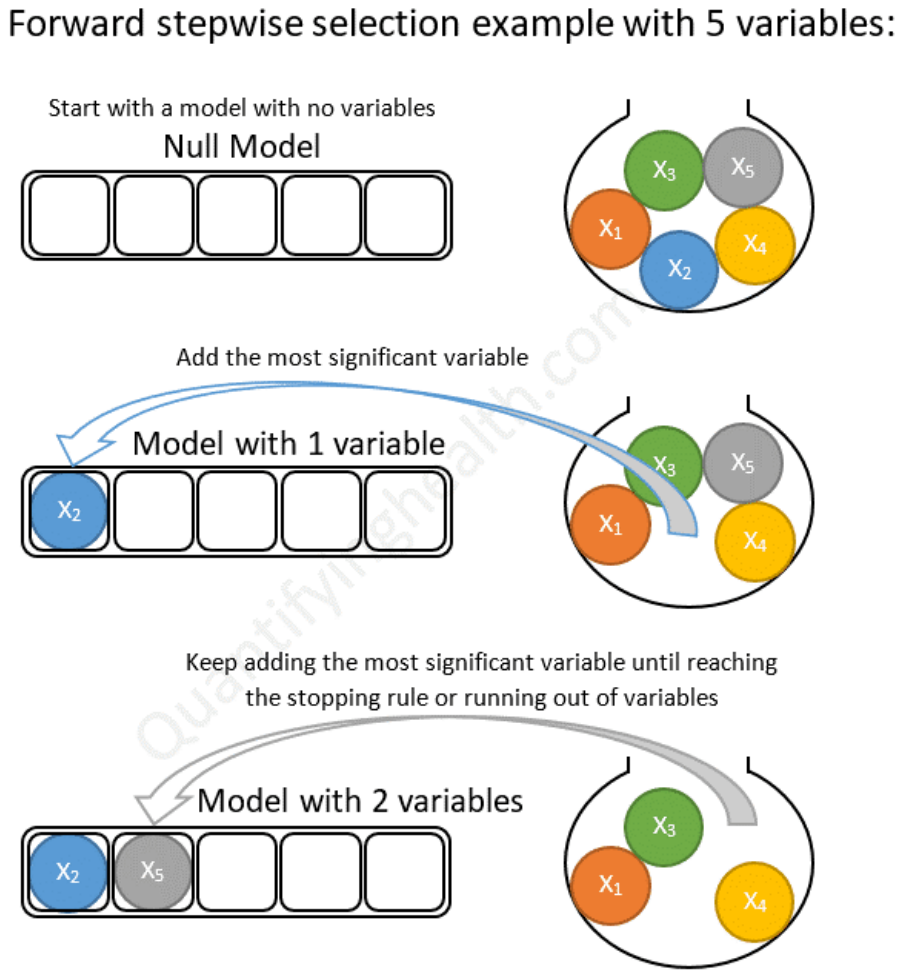
- 전진선택법

$\max(|t_j|) > t_F$: x_j변수 포함
$\max(|t_j|) < t_F$: x_j변수 제외하고 종료

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1$$
$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_2 x_2$$
$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_3 x_3$$
$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_4 x_4$$
$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_5 x_5$$

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_2 x_2 + \hat{\beta}_1 x_1$$
$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_2 x_2 + \hat{\beta}_3 x_3$$
$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_2 x_2 + \hat{\beta}_4 x_4$$
$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_2 x_2 + \hat{\beta}_5 x_5$$

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_2 x_2 + \hat{\beta}_5 x_5 + \hat{\beta}_1 x_1$$
$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_2 x_2 + \hat{\beta}_5 x_5 + \hat{\beta}_3 x_3$$
$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_2 x_2 + \hat{\beta}_5 x_5 + \hat{\beta}_4 x_4$$



Unit 02 | Feature Selection techniques

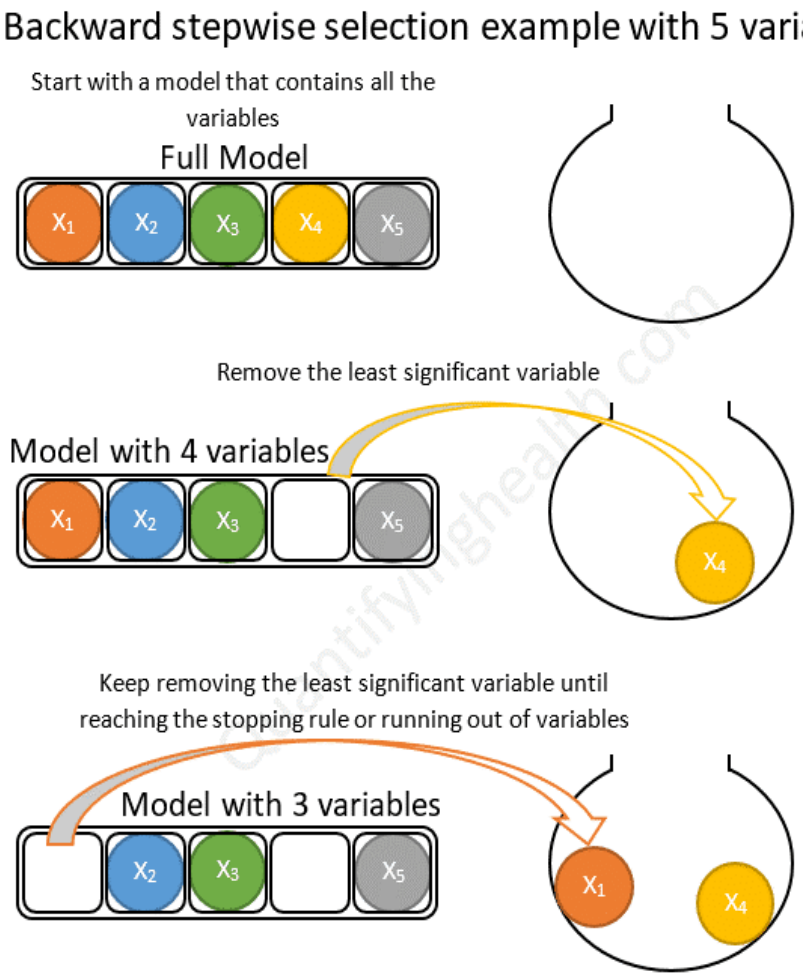
- 후방제거법

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \hat{\beta}_3 x_3 + \hat{\beta}_4 x_4 + \hat{\beta}_5 x_5$$

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \hat{\beta}_3 x_3 + \hat{\beta}_5 x_5$$

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_2 x_2 + \hat{\beta}_3 x_3 + \hat{\beta}_5 x_5$$

$\min(|t_j|) > t_B$: x_j변수 포함하고 종료
$\min(|t_j|) < t_B$: x_j변수 제거



Unit 02 | Feature Selection techniques

- 단계별 선택법

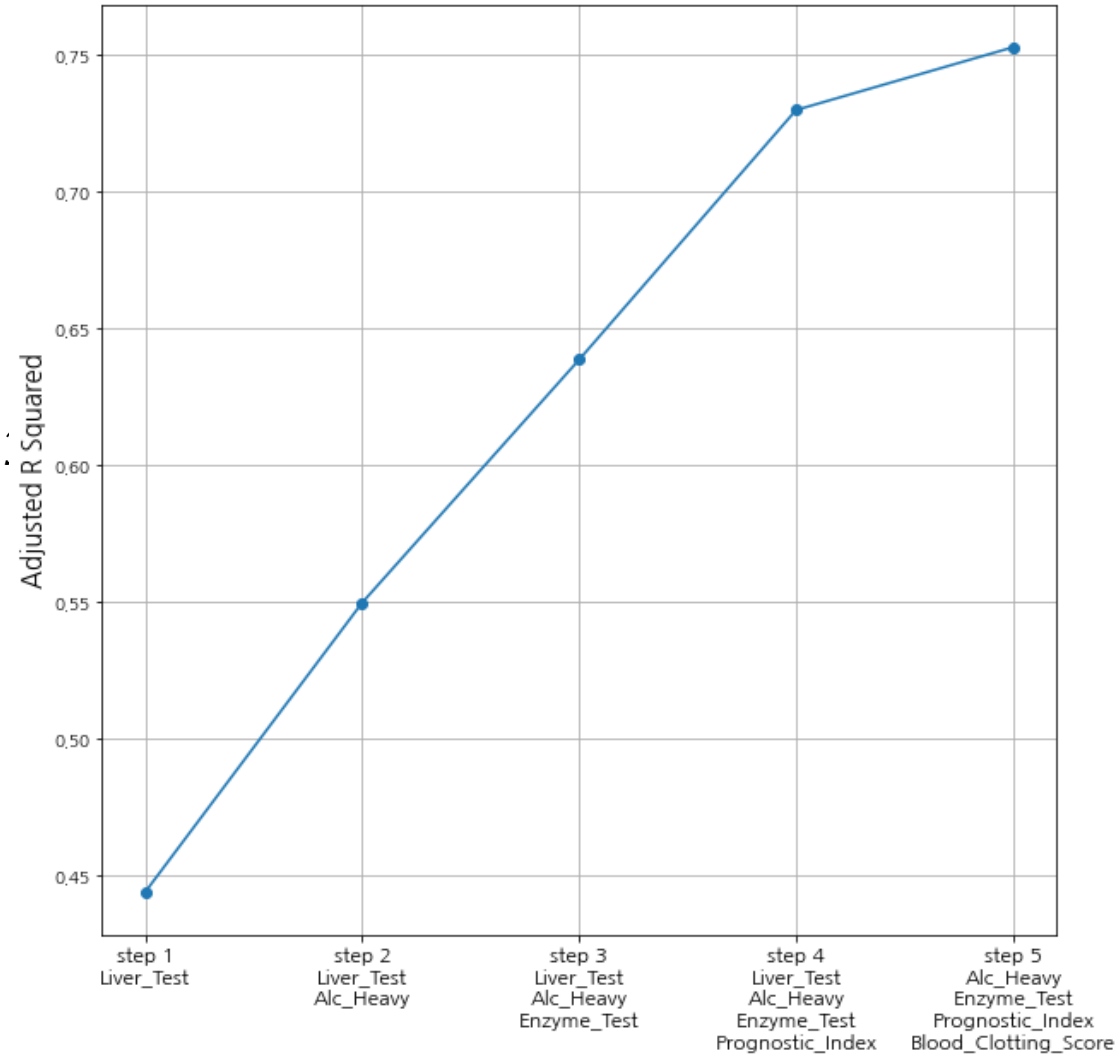
$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \hat{\beta}_3 x_3$$

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \hat{\beta}_3 x_3 + \hat{\beta}_4 x_4$$

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_2 x_2 + \hat{\beta}_3 x_3 + \hat{\beta}_4 x_4$$

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_2 x_2 + \hat{\beta}_3 x_3 + \hat{\beta}_4 x_4 + \hat{\beta}_5 x_5$$

한 번 선택된 변수를 제거하지 않고, 한 번 제거된 변수를 선택하지 않는 문제를 해결



Unit 02 | Feature Selection techniques

- Genetic Algorithm

1) 염색체(chromosome) 만들기

	x_1	x_2	x_3	x_4	x_5	x_6
①	0.2	0.9	0.1	0.3	0.6	0.2
②	0.8	0.1	0.1	0.9	0.6	0.7
③	0.1	0.9	0.1	0.5	0.4	0.7

➡

			점수	가중치
①	0 1 0 0 1 0	→	0.60	0.177
②	1 0 0 1 1 1	→	0.65	0.185
③	0 1 0 1 0 1	→	0.45	0.118

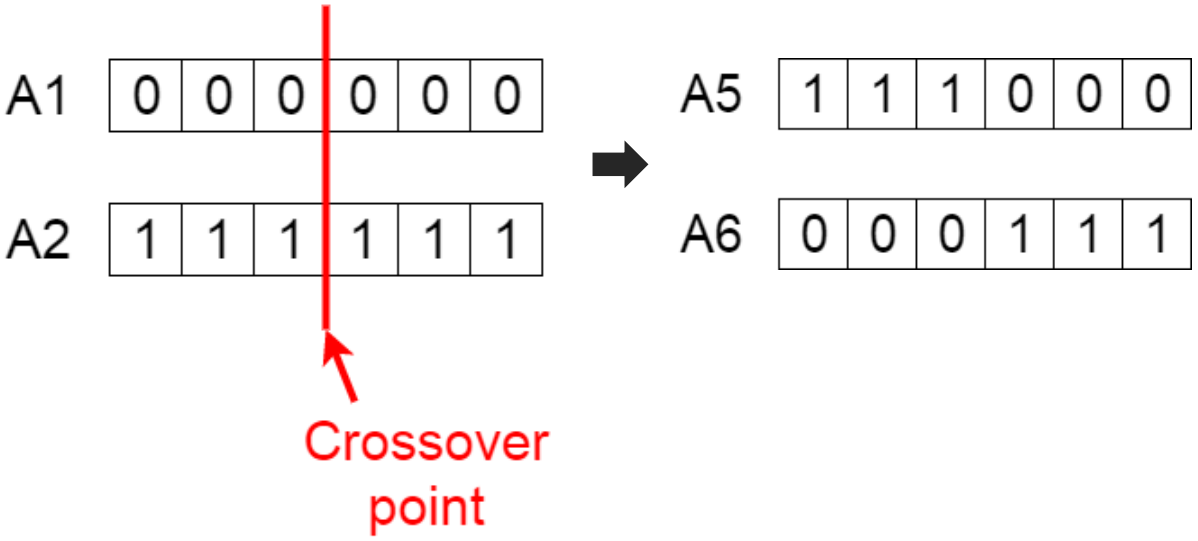
2) Selection

① Deterministic selection
: fitness 점수가 높은 순으로 정렬 후, 설정한 cut-off 값을 기준으로 부모세대 chromosome 선택
ex) cut-off = 0.5의 경우, ①, ②번이 부모 세대가 됨

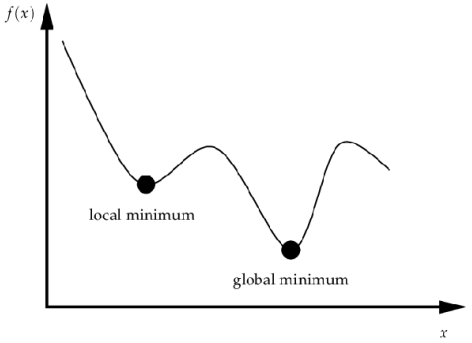
② Probabilistic selection
: 난수 생성 후, 해당되는 염색체(부모세대)를 선택

- Genetic Algorithm

3) Cross-over



4) Mutation



Before Mutation

A5 | 1 | 1 | 1 | 0 | 0 | 0 |

After Mutation

A5 | 1 | 1 | 0 | 1 | 1 | 0 |

x_1 x_2 x_3 x_4 x_5 x_6

Empirical Study

- Rankings in terms of
  - ✓ (1) Error rate improvement
  - ✓ (2) Variable reduction rate
  - ✓ (3) Computational efficiency

Variable selection technique	Error rate improvement	Variable reduction rate	Computational efficiency
Forward	5	4	1
Backward	4	3	2
Stepwise	3	2	6
GA	1	6	7
Ridge	2	7	5
LASSO	7	1	3
Enet	6	5	4

Contents

Unit 03 | Eigen-value Decomposition

▪ 고유값과 고유벡터 (Eigenvalue & Eigenvector)

$n \times n$인 정방행렬 A는 n개의 고유값 $\lambda_1 \lambda_2, \dots, \lambda_n$과 각각의 고유값에 해당하는 고유벡터 $X_1, X_2, \dots, X_n$을 갖는다.

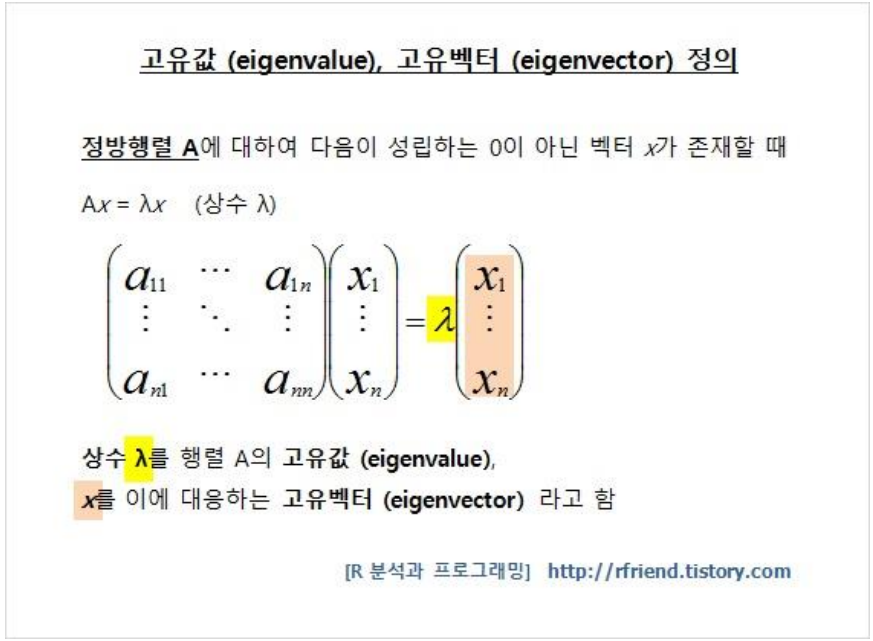
1) 고유벡터 (Eigenvector)

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

cf. 벡터란? 크기와 방향을 갖춘 물리량!

2) 고유값 (Eigenvalue)

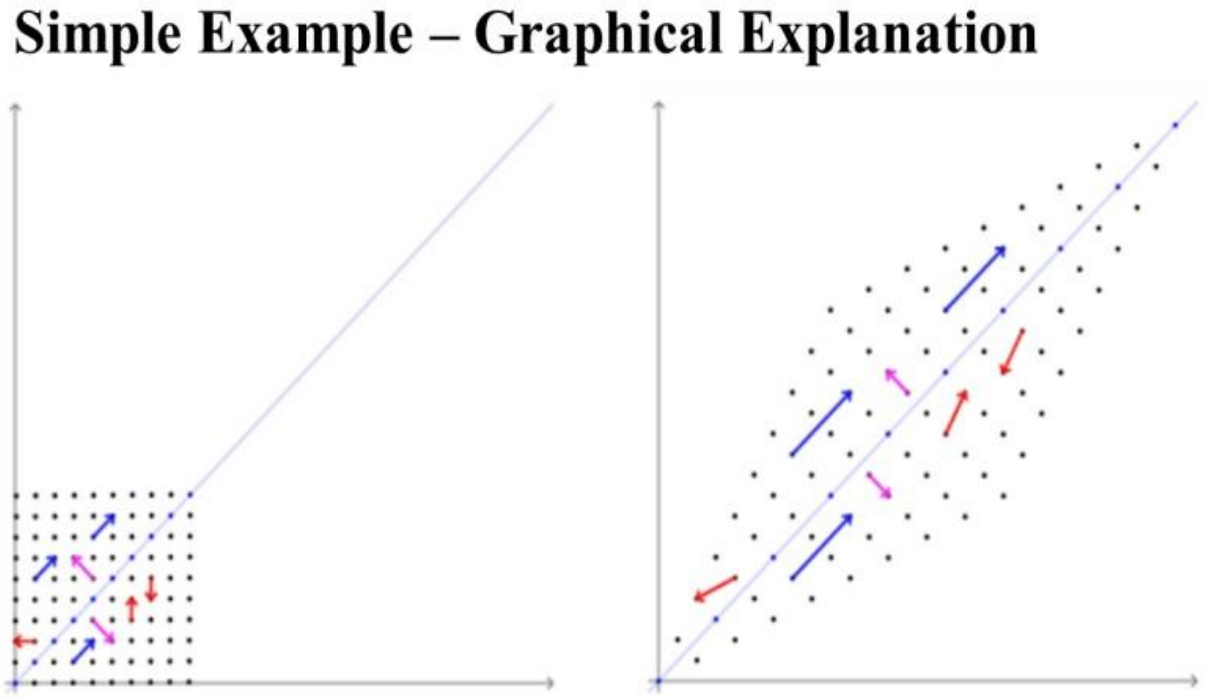
: 고유벡터의 상수 λ값



Unit 03 | Eigen-value Decomposition

- 고유값과 고유벡터 <기하학적 의미>

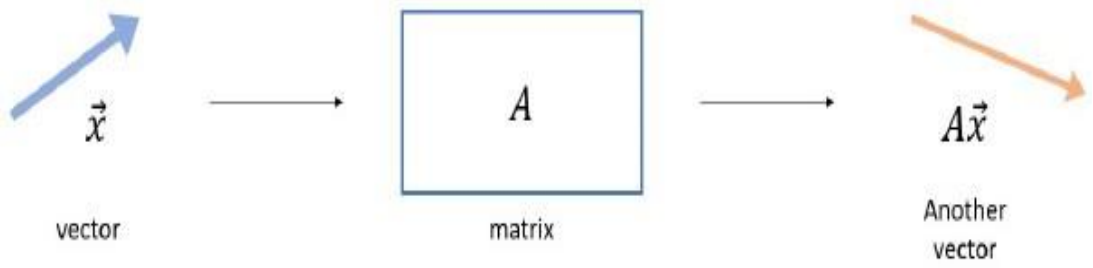
- 고유벡터의 기하학적 의미
  : 선형변환 이후 방향이 바뀌지 않는 벡터!



Unit 03 | Eigen-value Decomposition

- 선형변환

  : 좌표 공간 내에서 일어날 수 있는 선형적 변환



Unit 03 | Eigen-value Decomposition

- 고유값 분해 (정방행렬의 대각화)

Simple Example

$$\begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix} x = \lambda x \qquad eigenvalue = \lambda = 1, \quad eigenvector = (1, -1)$$

$$\begin{pmatrix} 2-\lambda & 1 \\ 1 & 2-\lambda \end{pmatrix} x = 0 \qquad eigenvalue = \lambda = 3, \quad eigenvector = (1, 1)$$

$$(2-\lambda)^2 - 1 = 0$$

$$\lambda = 1, 3$$

SW

Eigenvalue(Spectral) Decomposition

$$A = \begin{pmatrix} v_1 & v_2 & \dots & v_n \end{pmatrix} \begin{bmatrix} \lambda_1 & 0 & 0 & 0 \\ 0 & \lambda_2 & 0 & 0 \\ 0 & 0 & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{bmatrix} \begin{pmatrix} v_1 & v_2 & \dots & v_n \end{pmatrix}^{-1}$$

$n \times n$ $n \times n$ $n \times n$ $n \times n$

rotation streching Inverse rotation

A scalar λ is an eigenvalue of an $n \times n$ matrix A if and only if λ satisfies the characteristic equation

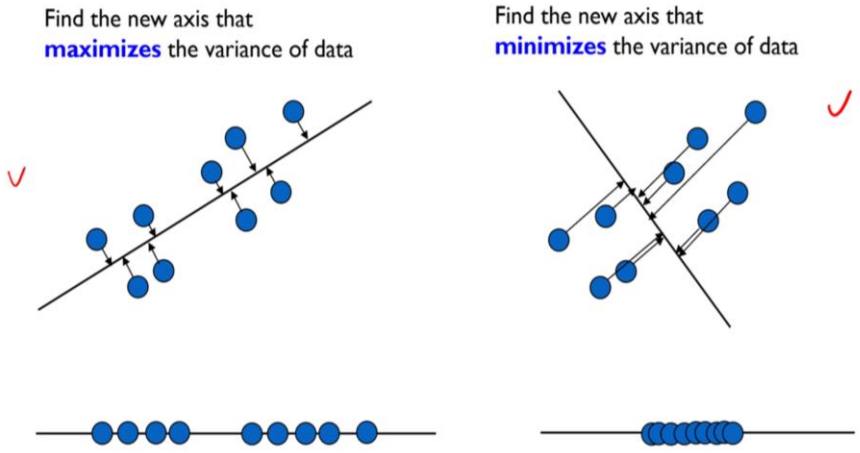
$$\det(A - \lambda I) = 0$$

Contents

Unit 04 | Linear : PCA, LDA, MDS

- 주성분 분석(Principal Component Analysis)
  - 원래 데이터의 분산을 최대한 보존하는 새로운 축을 찾고, 해당 축에 데이터를 사영(projection)시키는 대표적인 차원축소 기법

    ⇒ 기존의 변수들의 선형결합을 이용해 새로운 변수들로 변환하는 것!



우리의 목표는 주성분 점수 $a_{i,j}$ 를 찾는것!
HOW? Covariance Matrix

Z is a linear combination (선형결합) of the original p variables in X

$$Z_1 = \alpha_1^T X = \alpha_{11}X_1 + \alpha_{12}X_2 + \cdots + \alpha_{1p}X_p$$
$$Z_2 = \alpha_2^T X = \alpha_{21}X_1 + \alpha_{22}X_2 + \cdots + \alpha_{2p}X_p$$
$$\vdots \qquad\qquad \vdots$$
$$Z_p = \alpha_p^T X = \alpha_{p1}X_1 + \alpha_{p2}X_2 + \cdots + \alpha_{pp}X_p$$

- $X_1, X_2, \ldots, X_p$: 원래 변수 (original variable)
- $\alpha_i = [\alpha_{i1}, \alpha_{i2}, \ldots, \alpha_{ip}]$: i 번째 기저(basis) 또는 계수 (Loading)
- $Z_1, Z_2, \ldots, Z_p$: 각 기저로 사영된 변환 후 변수 (주성분, Score)

▪ 주성분 분석(Principal Component Analysis)

- 공분산 행렬 (Covariance Matrix) : 데이터의 구조를 설명, 각 feature의 변동 알 수 있음

- covariance
  - $\mathrm{cov}(x,y) = E[(x-m_x)(y-m_y)]$
- covariance matrix
  - $x=[x_1,\ldots,x_n]^T$: sample data, n차원 열벡터
  - $C = E[(x-m_X)(x-m_X)^T]$: n×n 행렬
  - $\langle C\rangle_{ij} = E[(x_i-m_{xi})(x_j-m_{xj})^T]$: i번째 성분과 j번째 성분의 공분산
  - C is real and symmetric

$$C=\begin{pmatrix} C_{11} & \cdots & C_{1n} \\ \vdots & \ddots & \vdots \\ C_{n1} & \cdots & C_{nn} \end{pmatrix}$$

cf) 공분산 행렬의 각 원소들이 의미하는 것

- 고유벡터는 행렬이 벡터에 작용하는 주축(principal axis)의 방향을 나타냄
- 공분산 행렬의 고유벡터는 데이터가 어떤 방향으로 분산되어 있는지 보여줌
- 고유값은 고유벡터 방향으로 얼만큼의 크기로 벡터 공간이 늘려져 있는지 보여줌

⇒ 고유값이 큰 순서대로 고유벡터를 정렬하면 중요한 순서의 주성분을 구할 수 있게 된다.

▪ 주성분 분석 (Principal Component Analysis)의 과정

① 실제 데이터에선 측정단위의 영향을 받음 ! : 표준화 사용

→ 각 변수가 평균 0, 표준편차 1이 되도록 표준화 $(z = \frac{x_i - \bar{x}_i}{\sqrt{\sigma_{ii}}})$

$$X = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_p \end{bmatrix} \longrightarrow Z = \begin{bmatrix} Z_1 \\ Z_2 \\ \vdots \\ Z_P \end{bmatrix}$$

$$\Sigma(X) = Cov(X) = E(X-\mu)(X-\mu)^T$$
$$= E\left(\begin{bmatrix} X_1 - \mu_1 \\ X_2 - \mu_2 \\ \vdots \\ X_p - \mu_p \end{bmatrix} \begin{bmatrix} X_1 - \mu_1 & X_2 - \mu_2 & \cdots & X_p - \mu_p \end{bmatrix} \right)$$
$$= \begin{bmatrix} \sigma_{11} & \sigma_{12} & \cdots & \sigma_{1p} \\ \sigma_{21} & \sigma_{22} & \cdots & \sigma_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{p1} & \sigma_{p2} & \cdots & \sigma_{pp} \end{bmatrix}$$

$$\longrightarrow Cov(Z) = \begin{pmatrix} 1 & \cdots & \rho_{1p} \\ \vdots & \ddots & \vdots \\ \rho_{p1} & \cdots & 1 \end{pmatrix} = R : Correlation\, matrix$$

$$Cov(z_1, z_2) = \frac{Cov(X_1, X_2)}{\sqrt{\sigma_{11}}\sqrt{\sigma_{12}}} = corr(X_1, X_2) = \rho_{12}$$

- 상관계수 행렬의 고유값/고유벡터를 사용한 주성분분석과 동일
- 전체 분산 = $\lambda_1 + \lambda_2 + \cdots + \lambda_p = p$
- 전체 변동 중 k 번째 주성분이 설명하는 변동 비율 = λ_k / p

▪ 주성분 분석(Principal Component Analysis)

② 공분산 행렬 (Covariance Matrix) 구하기

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

③ 공분산 행렬 스펙트럼 분해

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

$$= \begin{vmatrix} \underbrace{0.867}_{e_1} & \underbrace{0.499}_{e_2} \\ -0.499 & 0.867 \end{vmatrix} \begin{vmatrix} \overset{\lambda_1}{1.342} & 0 \\ 0 & \underbrace{0.073}_{\lambda_2} \end{vmatrix} \begin{vmatrix} 0.867 & -0.499 \\ 0.499 & 0.867 \end{vmatrix}$$

$$\Sigma = \begin{bmatrix} 1.026 & 0.548 \\ 0.548 & 0.389 \end{bmatrix} = 1.342 \begin{bmatrix} 0.867 \\ -0.499 \end{bmatrix} [0.867 \quad -0.499] + 0.073 \begin{bmatrix} 0.499 \\ 0.867 \end{bmatrix} [0.499 \quad 0.867]$$

- $\lambda_2 \approx 0$
- 앞의 λ_1, e_1만으로도 공분산 행렬 Σ 표현 가능!

Spectrum Decomposition

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

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

$= P\Lambda P'$ (단, $PP' = P'P = I$) $e_i^T e_i = 1, e_i^T e_j = 0$ (직교, 정규)

- 주성분 분석(Principal Component Analysis)의 과정

④ 고유값 크기 순으로 고유벡터를 정렬하여 원래 데이터와 선형결합

→ $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$: 분산(고유값)이 큰 주성분부터 사용

→ $\frac{(\lambda 1+\lambda 2+\cdots+\lambda k)}{(\lambda_1+\lambda_2+\cdots+\lambda_n)}$: 처음 k개의 주성분에 의해 설명되는 변동의 비율

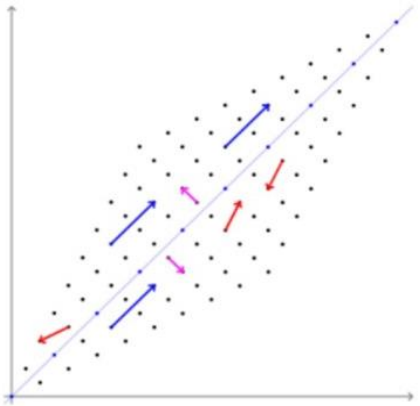
→ 주성분 점수 a_{ij} : 고유값 λ_i 에 대응되는 고유벡터 e_i

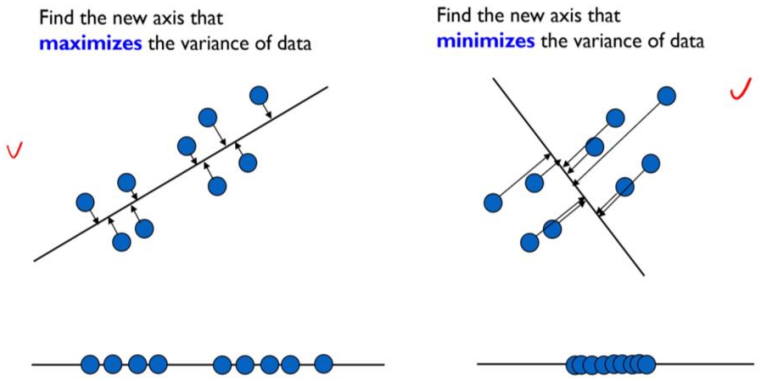
우리의 목표는 주성분 점수 a_{ij} 를 찾는것!
HOW? Covariance Matrix

Z is a linear combination (선형결합) of the original p variables in X

$$Z_1 = \alpha_1^T X = \alpha_{11}X_1 + \alpha_{12}X_2 + \cdots + \alpha_{1p}X_p$$
$$Z_2 = \alpha_2^T X = \alpha_{21}X_1 + \alpha_{22}X_2 + \cdots + \alpha_{2p}X_p$$
$$\vdots \qquad\qquad \vdots$$
$$Z_p = \alpha_p^T X = \alpha_{p1}X_1 + \alpha_{p2}X_2 + \cdots + \alpha_{pp}X_p$$

- $X_1, X_2, \ldots, X_p$: 원래 변수 (original variable)
- $\alpha_i = [\alpha_{i1}, \alpha_{i2}, \ldots, \alpha_{ip}]$: i 번째 기저(basis) 또는 계수 (Loading)
- $Z_1, Z_2, \ldots, Z_p$: 각 기저로 사영된 변환 후 변수 (주성분, Score)

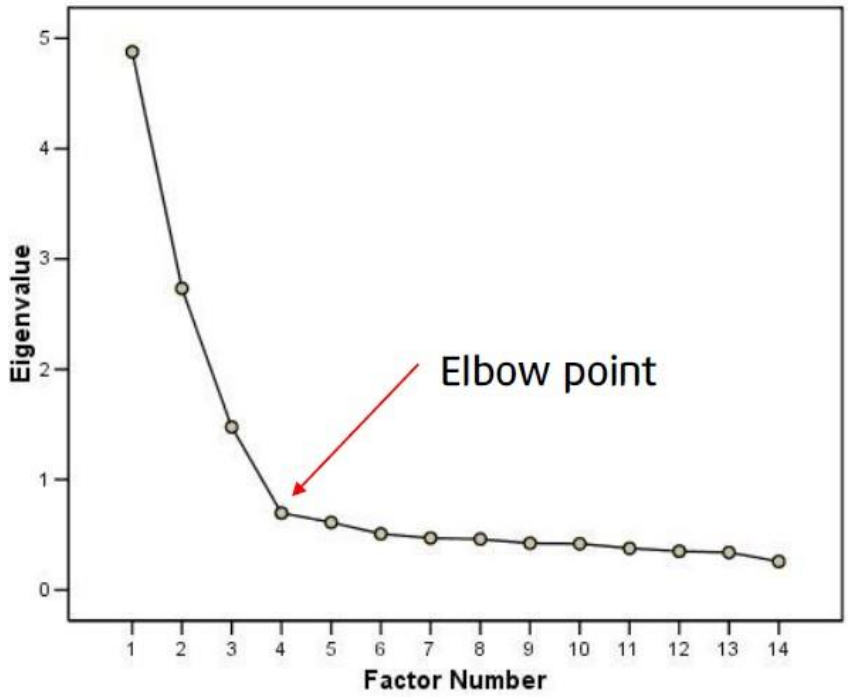




Unit 04 | Linear : PCA, LDA, MDS

- 주성분 개수의 결정

< Scree plot >



1. Rule of Thumb

   - 총분산 설명하는 비중이 70%~90% 사이에서 선택

   - 평균 고유값 $(\sum_i \lambda_i) / p$ 보다 작은 고유값을 갖는 주성분 제거
     : 평균고유값 = 평균분산
     : 표준화된 변수를 사용한다면 평균분산=1 이므로 1 보다 작은 고유값 제거
     : 0.7 보다 작은 고유값을 제거하는 것 제안하기도 함

2. Scree plot 활용

   : 곡선의 기울기가 급격히 감소하는 시점(Elbow point)

Unit 04 | Linear : PCA, LDA, MDS

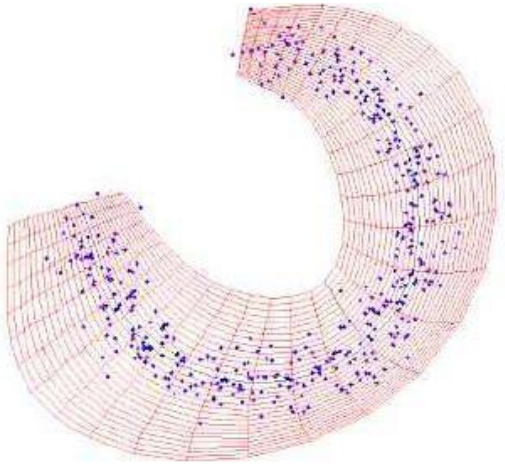
- PCA의 가정
  - Linearity : 데이터가 선형성을 띈다
  - Orthogonality : 찾은 주축들은 서로 직교한다.
  - 큰 분산을 갖는 방향이 중요한 정보를 담고 있다.

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

Unit 04 | Linear : PCA, LDA, MDS

- PCA의 단점
  - 데이터가 선형성을 띄지 않으면 적용 불가

    -> 해결 : Kernel PCA를 통해 비선형 데이터에 적용 가능



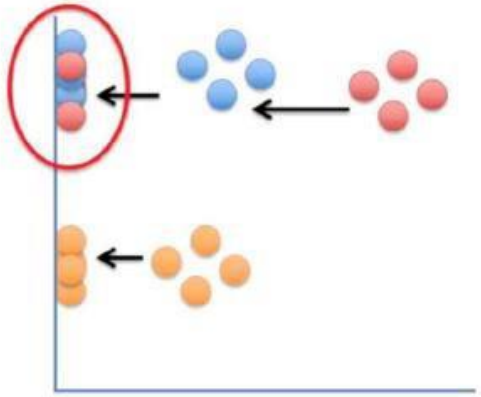
  - 특징 벡터의 클래스를 고려하지 않기 때문에

    최대 분산 방향이 특징 구분을 좋게 한다는 보장이 없다.(label과 관련이 없다)

  - 새로 형성된 주성분의 해석을 위한 도메인 지식이 필요

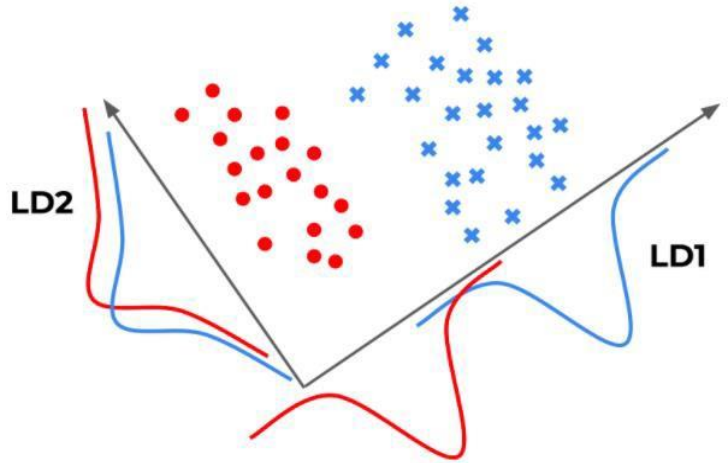
    (주성분은 모든변수의 선형결합으로 이루어짐)

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



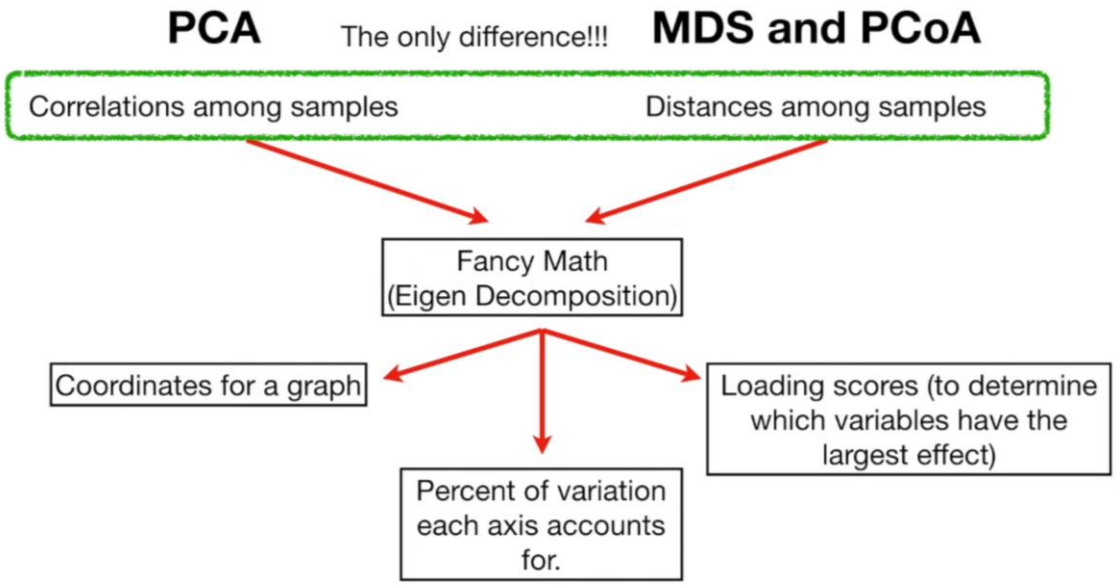
Unit 04 | Linear : PCA, LDA, MDS

- LDA (Linear Discriminant Analysis)
  - 데이터의 분포를 학습하여 분리를 최적화하는 결정경계(Decision boundary) 데이터를 분류하는 모델
  - PCA가 최적 표현을 위한 최대 분산을 찾아 차원을 축소한다면
    LDA는 최적 분류를 위해 분별 정보를 최대한 유지하면서 차원을 축소한다
  - LDA의 핵심은 PCA와 달리, 공분산 행렬이 아닌 클래스 간 분산 행렬과 클래스 내부 분산 행렬을
    내적하여 분해하는 것에 있다



Unit 04 | Linear : PCA, LDA, MDS

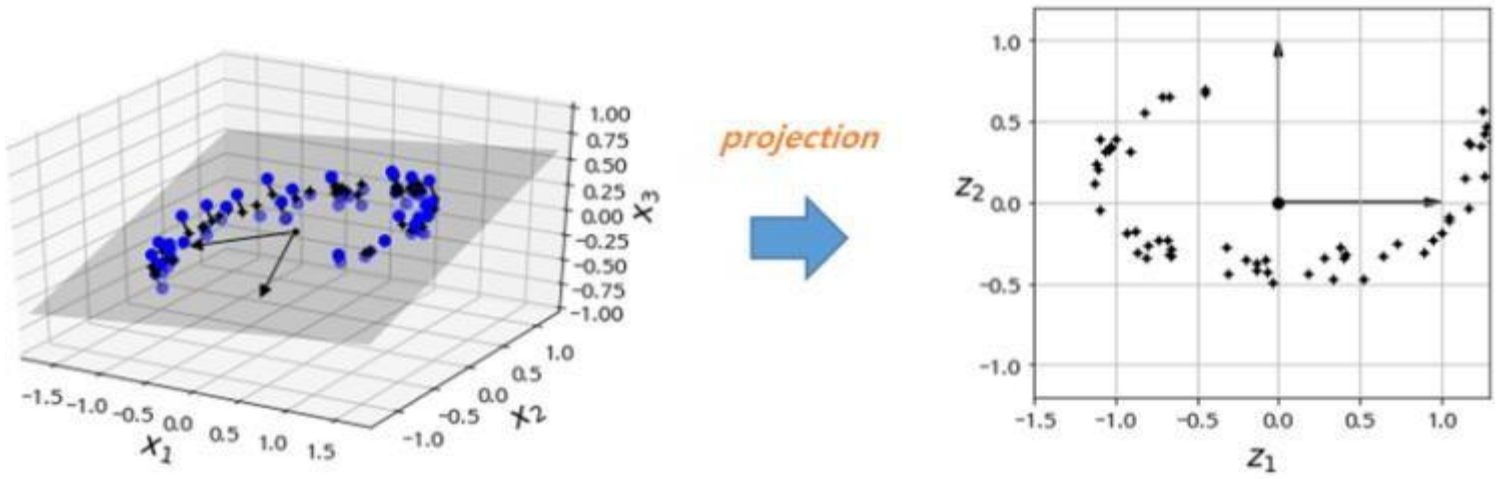
- MDS (Multi-Dimensional Scaling)
  - 데이터 포인트 사이의 유사성/비유사성을 측정하여 저차원의 공간 상에 투영하는 방식
  - MDS 또한, 공분산행렬이 아닌 거리 행렬을 이용하는 것 외에는 PCA와 동일하다



Contents

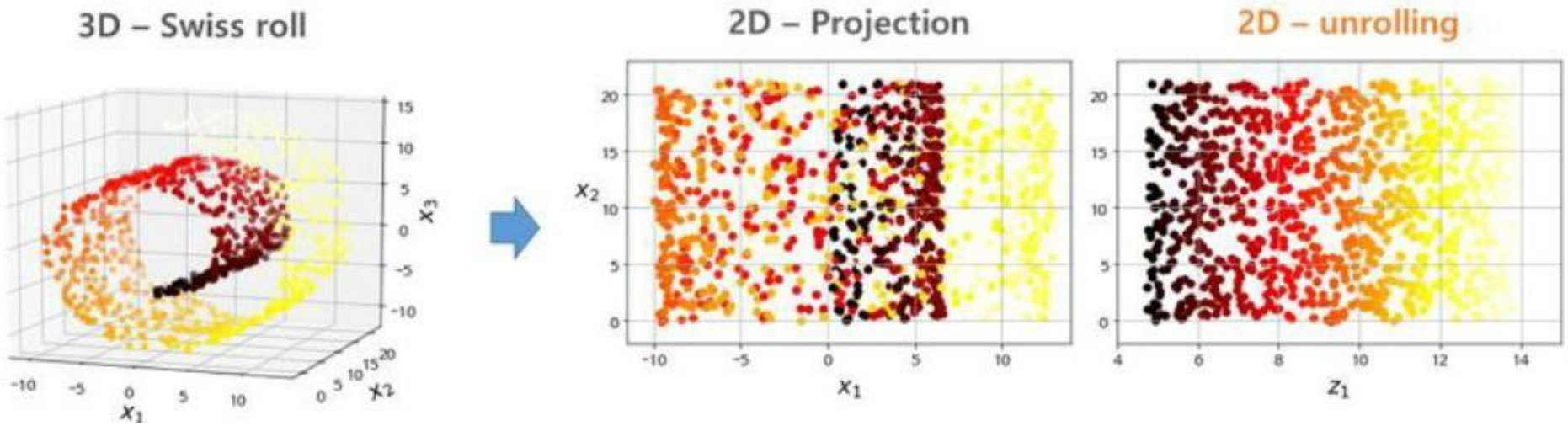
Unit 05 | Non-Linear : Isomap, LLE, SNE

- Non-linear transformation
  - 앞선 PCA와 LDA는 데이터를 새로운 차원에 선형으로 projection(투영)시키는 과정



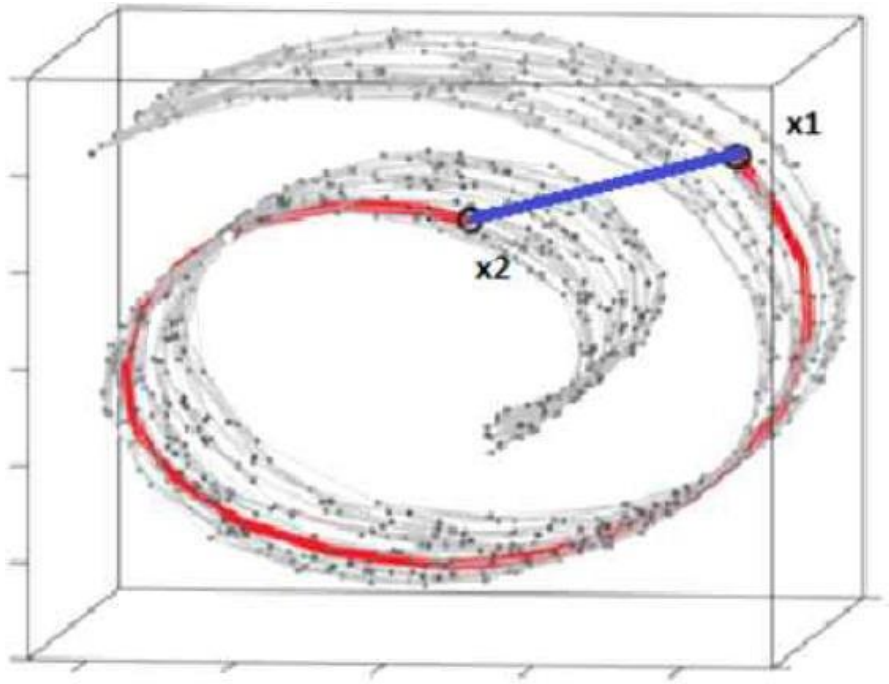
Unit 05 | Non-Linear : Isomap, LLE, SNE

- Non-linear transformation
  - 앞선 방법론들은 부분공간이 비선형적인 경우, 좋은 성능을 보이지 못함
  - 따라서 부분공간이 비선형적인 경우 (ex. 스위스 롤),
    데이터를 잘 아우르는 저차원으로 축소할 필요성 → Manifold Learning



Unit 05 | Non-Linear : Isomap, LLE, SNE

- Manifold 학습의 종류



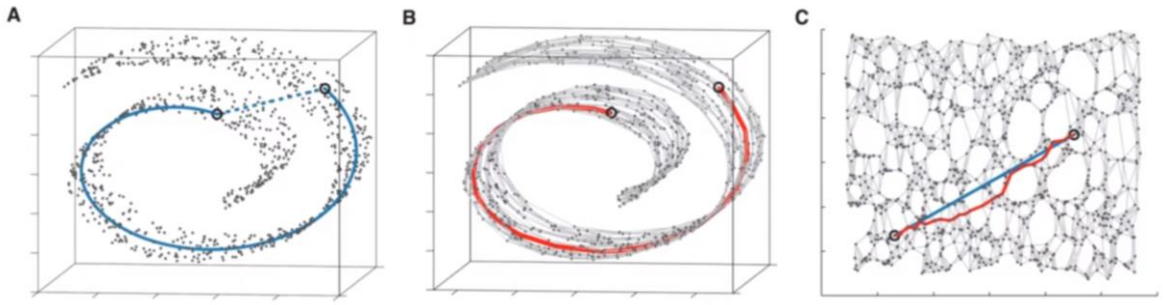
- **ISOMAP**
  : 각 데이터 포인트를 가장 가까운 이웃과 연결하는 방식으로 거리 행렬을 계산하며 이외에는 MDS와 동일함

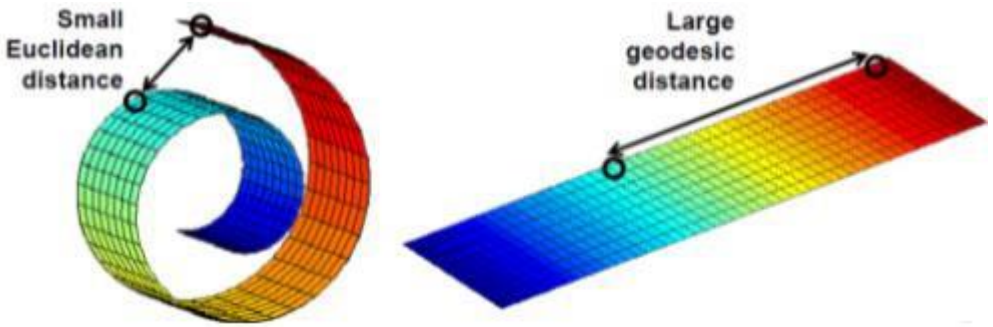
- **LLE(Locally Linear Embedding)**
  : 큰 틀에서 ISOMAP과 동일하지만 Locality를 반영하는 방식이 ISOMAP과 상이함

- **SNE**
  : cut-off 값으로 이웃을 정하는 LLE와 달리, 가까운 이웃과 먼 이웃을 확률적으로 계산함

Unit 05 | Non-Linear : Isomap, LLE, SNE

ISOMAP





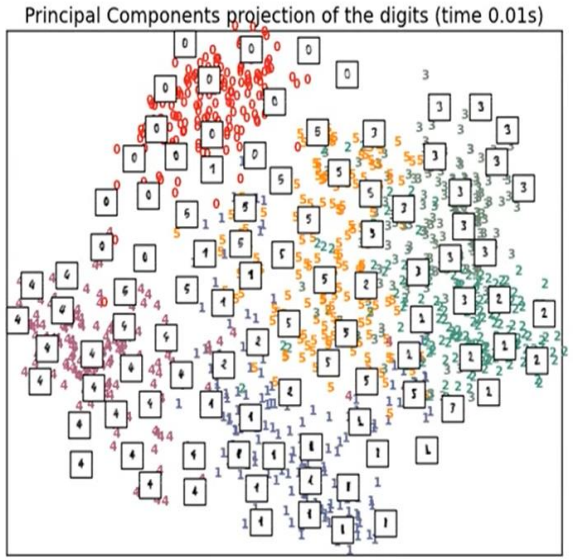
- Isomap procedure
  - ✓ Step 1: Construct neighborhood graph
    - ε-Isomap: connect two points if they are closer than ε
    - k-Isomap: connect the point i to the point j if the i is one of the k-nearest neighbor of j
  - ✓ Step 2: Compute the shortest paths
    - Initialize $d_G(i, j) = d_X(i, j)$ if i and j are linked by an edge, $d_G(i, j) = \inf$ otherwise
    - For each value of $k = 1,2, \ldots, N$ in turn, replace all entries $d_G(i, j)$ by

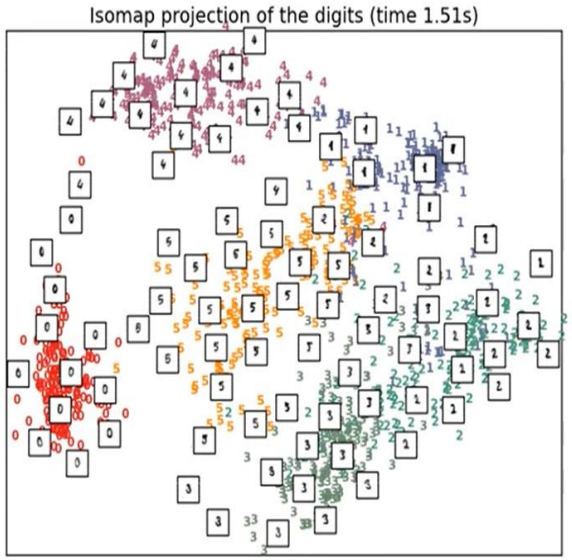
    $\min \{d_G(i, j), d_G(i, k)+d_G(k, i)$
  - ✓ Step 3: Construct d-dimensional embedding by traditional MDS

Unit 05 | Non-Linear : Isomap, LLE, SNE

- ISOMAP

  - Isomap example: Hand digit recognition





Unit 05 | Non-Linear : Isomap, LLE, SNE

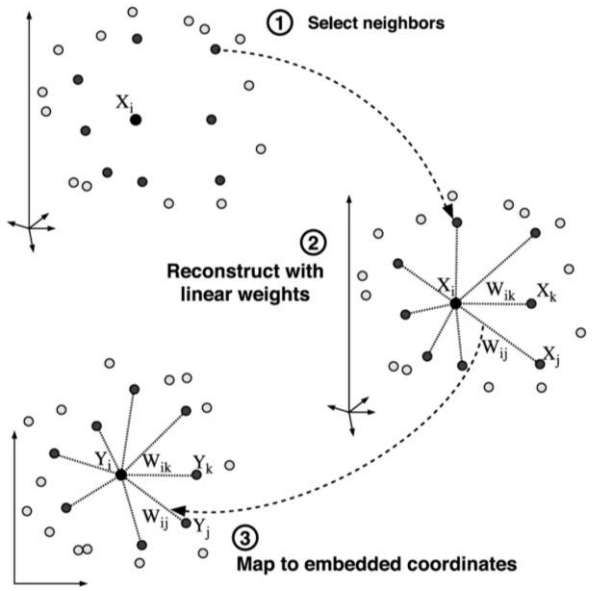
- Locally Linear Embedding (LLE)
  - 국소적인 평면의 데이터들이 축소된 차원에서도 인접하도록 반영
  - 최인접 이웃의 정보, locality에 집중
  - 데이터 사이의 선형적인 구조를 보존하며 저차원으로 임베딩

① $$E(\mathbf{W}) = \sum_i \left| \mathbf{x}_i - \sum_j \mathbf{W}_{ij}\mathbf{x}_j \right|^2$$

$$s.t.\ \mathbf{W}_{ij} = 0 \text{ if } \mathbf{x}_j \text{ does not belong to the neighbor of } \mathbf{x}_i$$

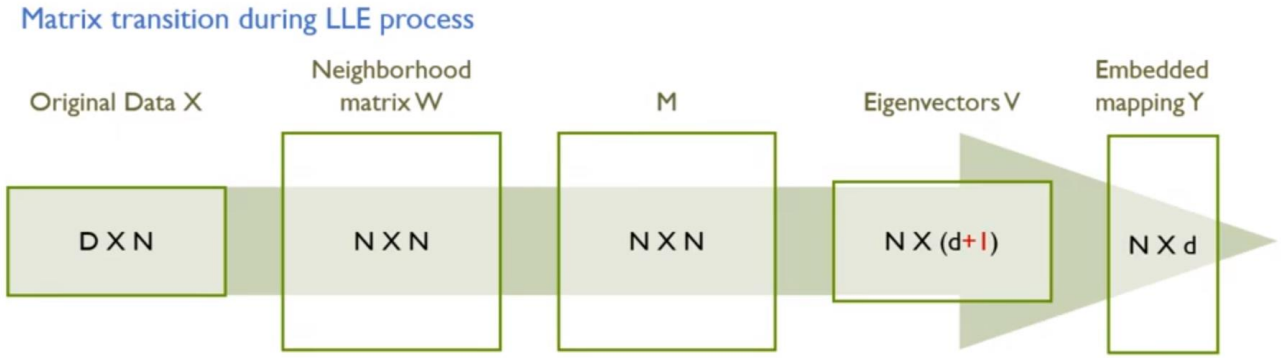
$$\sum_j \mathbf{W}_{ij} = 1 \text{ for all } i$$

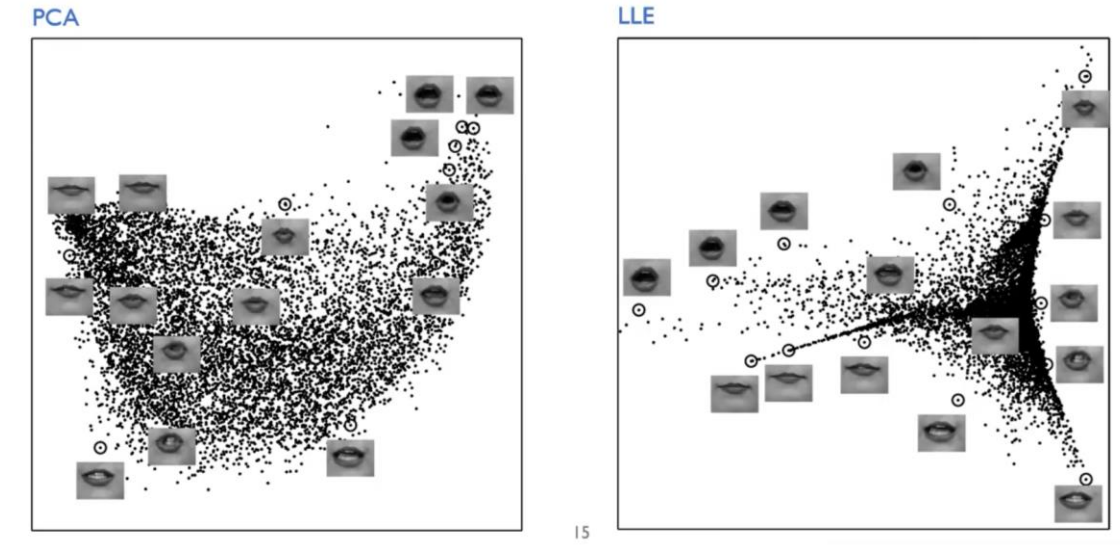
② $$\Phi(\mathbf{W}) = \sum_i \left| \mathbf{y}_i - \sum_j \mathbf{W}_{ij}\mathbf{y}_j \right|^2$$



Unit 05 | Non-Linear : Isomap, LLE, SNE

- Locally Linear Embedding (LLE)





- SNE (Stochastic Neighbor Embedding)
  - SNE란? : LLE와 큰 틀에서 동일하나 데이터 포인트들 간의 인접성을 확률적으로 계산하는 차원축소방식
  - 고차원 공간에서 유클리드 거리를 가우시안 정규분포에 대입하여 조건부확률로 변환
  - 고차원의 데이터 포인트(x_i 와 x_j)의 유사성 : $p_{j|i}$, 저차원 데이터 포인트(y_i 와 y_j)의 유사성: $q_{j|i}$

→ 만약, 고차원의 데이터 간의 거리 정보가 저차원에서도 잘 보존 되었다면, $p_{j|i}$ 와 $q_{j|i}$ 는 유사할 것이다

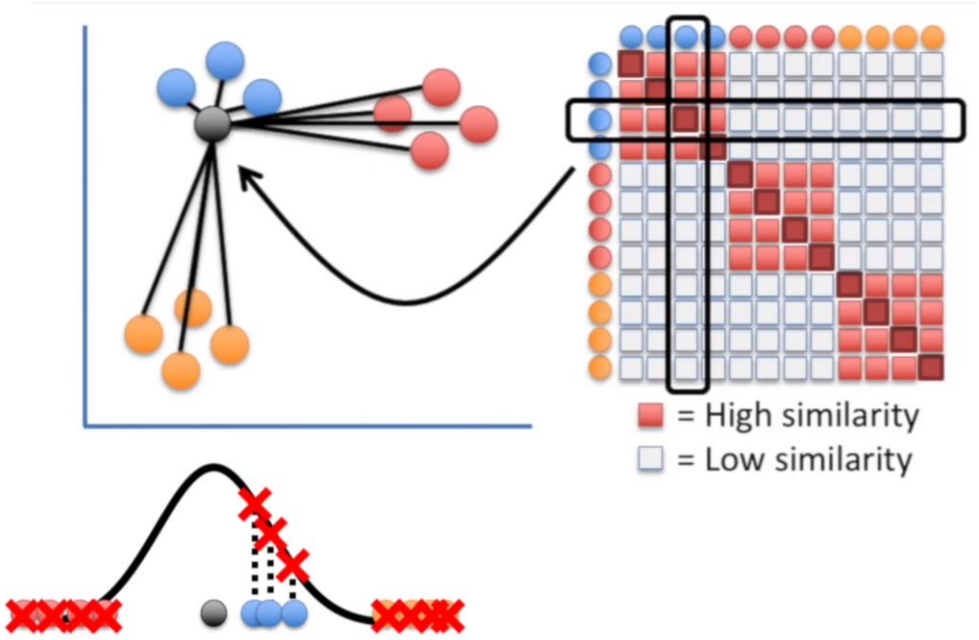
→ 확률 분포의 유사도(거리) 측정 : KL-divergence(Kullback-Leibler divergence)

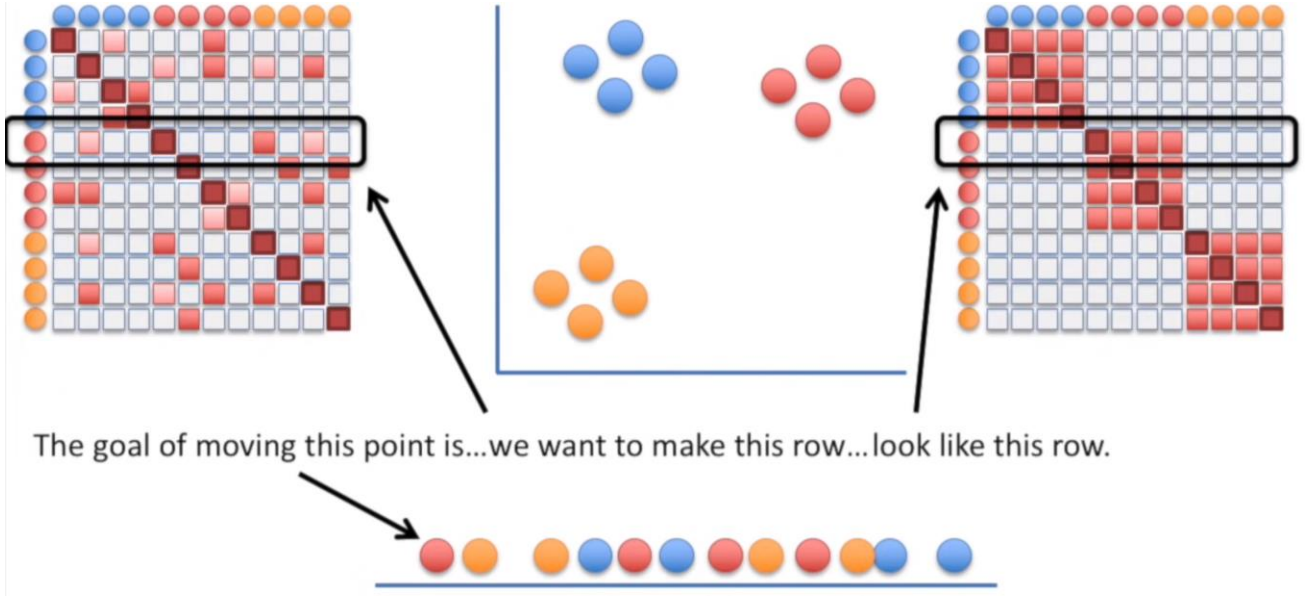
$$Cost = \sum_i KL(P_i||Q_i) = \sum_i \sum_j p_{j|i} log \frac{p_{j|i}}{q_{j|i}}$$

→ KL-divergence가 최소화되는 방향으로 학습이 진행
→ p_{ij}에 가장 가깝도록 q_{ij} 를 학습

Unit 05 | Non-Linear : Isomap, LLE, SNE

- SNE (Stochastic Neighbor Embedding)





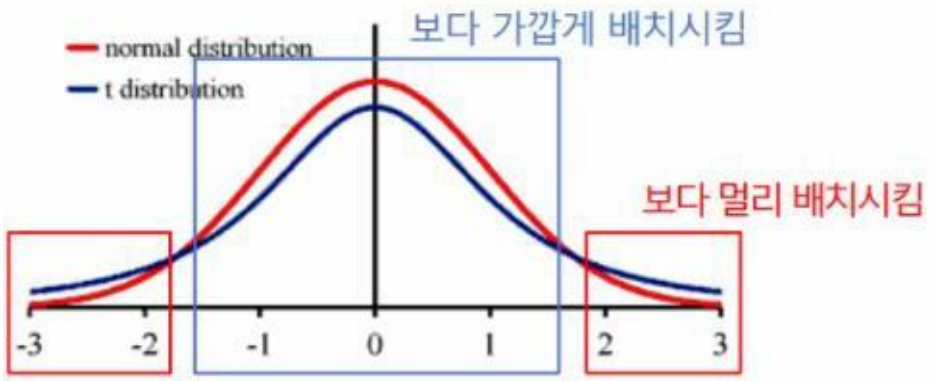
Unit 05 | Non-Linear : Isomap, LLE, SNE

- t-SNE (Stochastic Neighbor Embedding)
  - SNE의 문제점 : 가우시안 정규분포

    고차원의 거리가 멀어지면 gradient값이 0이 되어 i와 적당히 먼 데이터 포인트와

    매우 먼 데이터 포인트 간의 선택될 확률 차이가 발생하지 않는다

    → 따라서 꼬리가 두터운 t분포를 사용하는 것이 t-SNE !



가우스 분포와 자유도 1의 t분포의 비교

Unit 05 | Non-Linear : Isomap, LLE, SNE

- t-SNE (Stochastic Neighbor Embedding)
  - 장점
    - PCA와 달리 군집이 중복되지 않는다.
    - 군집성이 유지되기 때문에 시각화를 통한 분석에 유용
  - 단점
    - 거리를 학습하며 계속 업데이트하기 때문에 값이 매번 바뀜
    - 데이터 수가 많아지면 시간이 오래 걸림

Unit 06 | 과제

- 실습

  1) Dimensionality_reduction_practice.ipynb

- 과제

  1) 과제1 : PCA 과정 밟아보기 + 다른 차원축소 알고리즘 구현해보기 https://www.youtube.com/watch?v=9IDXYHhAfGA

     Week5_dimensionality_reduction_assignment1.ipynb

  2) 과제2 : 차원축소 Mnist data에 적용해보기

     Week5_dimensionality_reduction_assignment2.ipynb

Unit 06 | 참고문헌

- ✓ 투빅스 15기 권오현님 차원축소 강의자료
- ✓ 고려대학교 산업경영공학부 DSBA 연구실 강필성 교수님 차원축소 강의

  : https://www.youtube.com/watch?v=INHwh8k4XhM&list=PLetSlH8YjIfWMdw9AuLR5ybkVvGcoG2EW&index=1
- ✓ 고려대학교 산업경영공학부 김성범 교수님 강의 : https://www.youtube.com/watch?v=FhQm2Tc8Kic
- ✓ 고려대학교 김홍중 교수님 선형대수 강의자료, 고려대학교 박관영 교수님 회귀분석 강의자료
- ✓ 파이썬 머신러닝 완벽가이드
- ✓ GA : https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3
- ✓ MDS : https://yngie-c.github.io/machine%20learning/2020/10/02/mds/
- ✓ LLE : https://t-lab.tistory.com/26
- ✓ t-SNE : https://www.youtube.com/watch?v=NEaUSP4YerM&t=388s

들어주셔서 감사합니다.