

Machine Learning

Dimension Reduction

Present by Sangmin Bae

Contents

1

Dimension
Reduction

3

SVD

5

t-SNE

2

PCA

4

LDA

5

UMAP

Part One

Dimension Reduction

Dimension Reduction

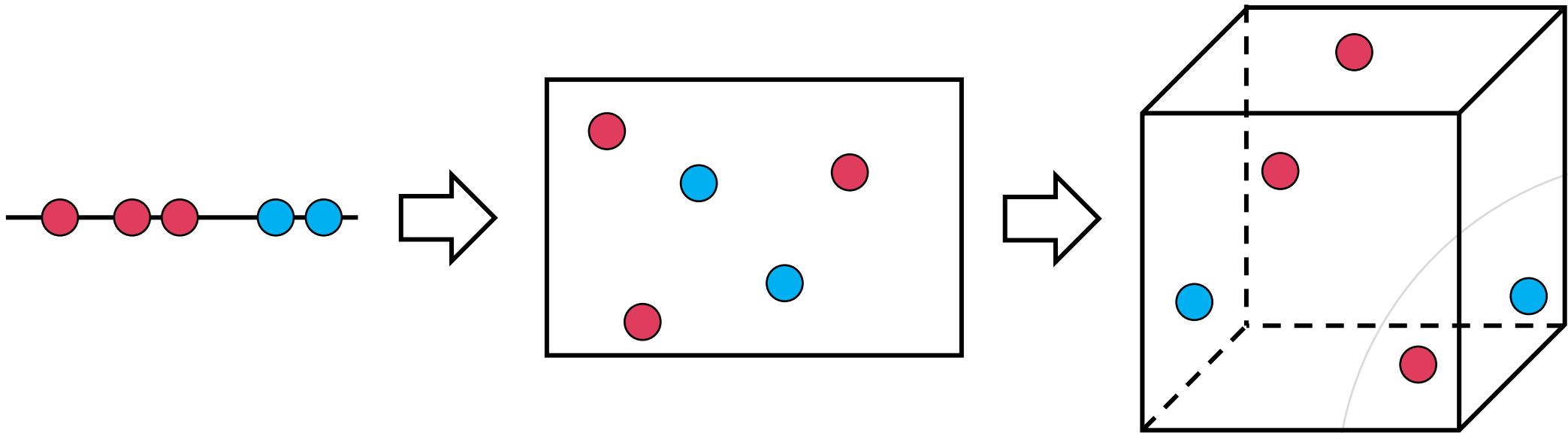
차원 축소 (dimension reduction)

- 데이터에 대한 차원 축소는 속도뿐만 아니라 성능면에서 필요함
- 모델 학습에 불필요한 피처(속도 향상)나 방해되는 피처(성능 향상)를 제거해야함
- 방해되는 피처란, over-fitting 문제를 발생시키는 피처로 이해 가능
- 이는 차원의 저주와 관련 있는 현상임

Curse of Dimensionality (1)

차원의 저주 (curse of dimensionality)

- 차원이 증가하면서, **학습 데이터 수(N)가 차원의 수(p)보다 적어져** 성능이 저하되는 현상
- 차원 내에 존재하는 데이터들이 희박해지는(sparse) 현상



Curse of Dimensionality (2)

차원의 저주 (curse of dimensionality)

- 빈 공간이 많아지며, 이는 정보가 없는 공간이 많아지는 것을 의미
- 해결할 수 있는 방법은 크게 두 가지가 있음
 1. 데이터를 수집
 2. 차원을 축소시킴
- 차원 축소 방법으로 크게 **형상 선택(feature selection)**과 **형상 추출(feature extraction)** 두 가지 사용

Feature Selection (1)

형상 선택 (feature selection)

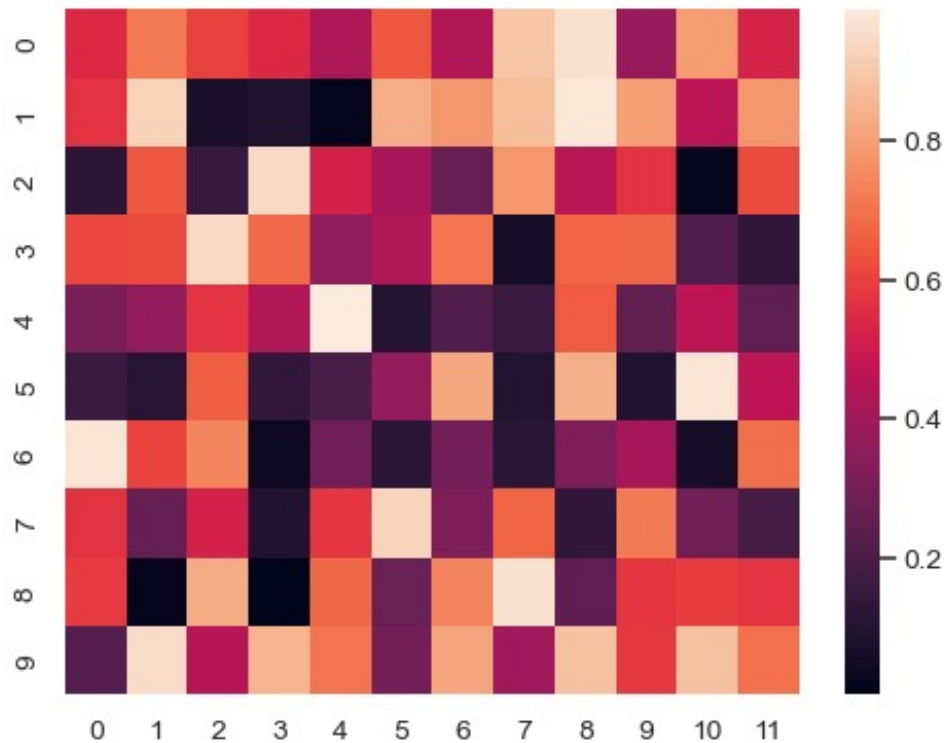
- 종속 변수와 가장 관련성이 높은 피처만을 선택해, 나머지를 제외시킴
(ex. 각각의 피처를 모델에 포함시킴으로써 손실값이 낮아지는 정도를 비교)
- (Remind) Bagged tree에서의 피처 중요도 계산



Feature Selection (2)

형상 선택 (feature selection)


- 피처 사이의 상관 관계가 매우 높아서, 한 쪽이 의미 없는 경우 제외시킴
- 일반적으로 히트맵(heatmap)을 통해 여러 피처의 공분산(covariance) 분석



Feature Extraction

형상 추출 (feature extraction)

- 개별 피처를 제거하는 대신, **저차원 공간으로 투영**시켜 데이터와 모델을 단순화시킴
- 다음과 같은 알고리즘들이 있음
 1. 주성분 분석 (PCA)
 2. 특이값 분해(SVD)
 3. LDA
 4. t-SNE
 5. UMAP



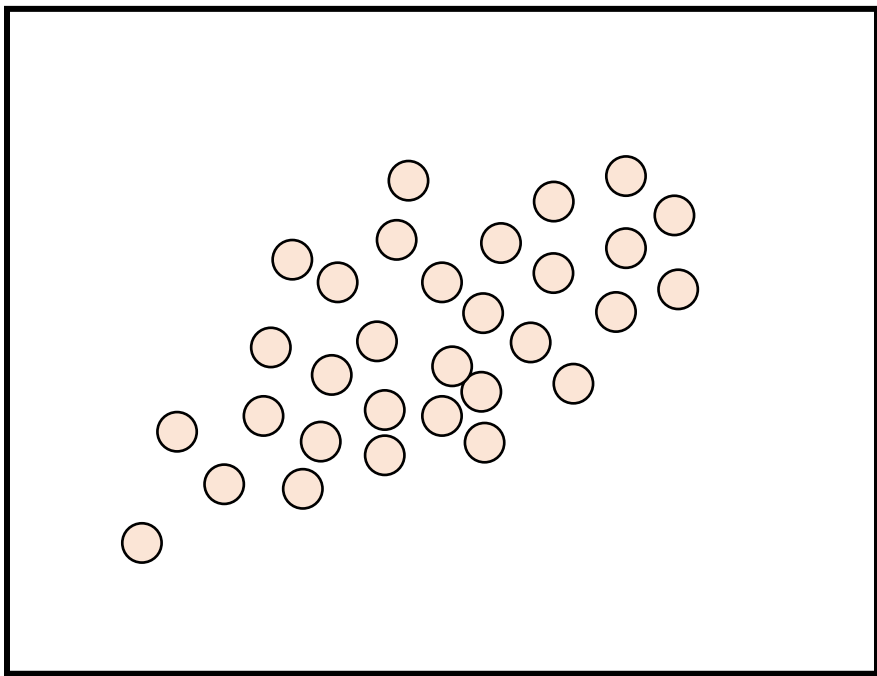
Part Two

PCA

Principal Component Analysis (1)

주성분 분석 (PCA)

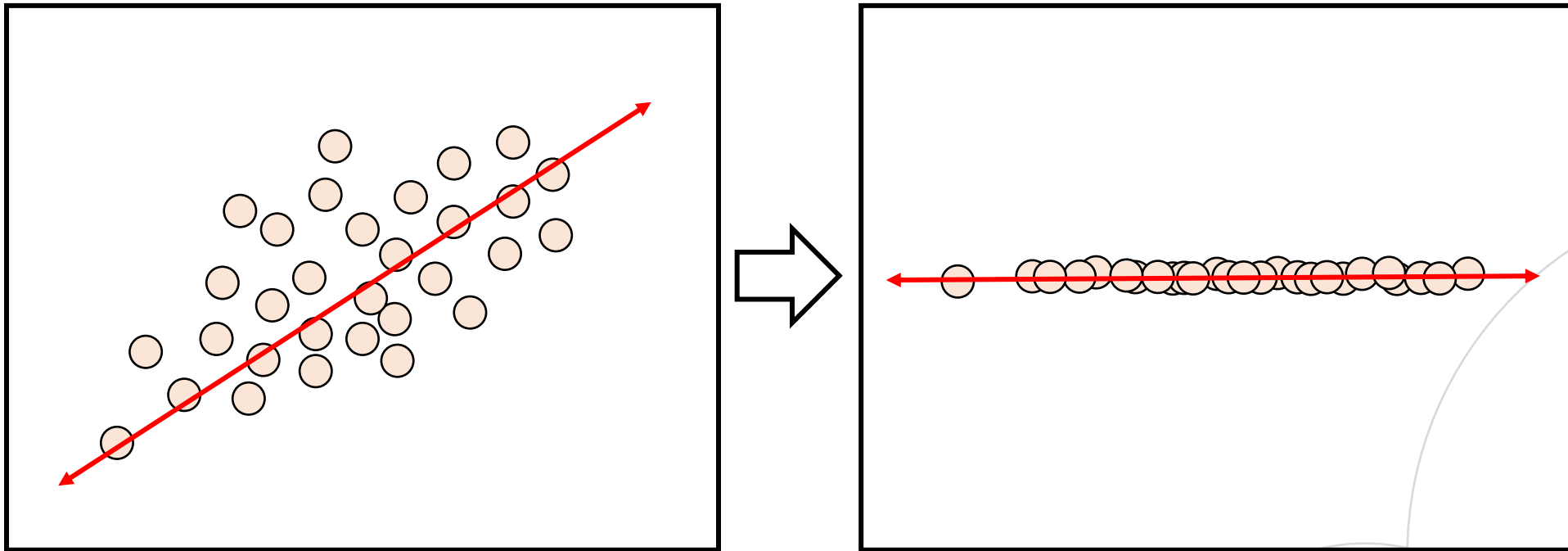
- 전체 데이터의 분포를 가장 잘 설명할 수 있는 주성분을 찾아주는 방법론
- 주성분이란 **그 방향으로 데이터들의 분산이 가장 큰 방향벡터**를 의미함



Principal Component Analysis (2)

주성분 분석 (PCA)

- 분산이 가장 크다는 말은 데이터가 가장 넓게 퍼져있어 구분짓기 쉽다는 뜻임
- 주성분 방향벡터를 찾아 데이터를 **그 위로 투영(projection)**시켜 차원을 한 단계 낮춤



Covariance Matrix (1)

공분산 (covariance)

- 두 변수 x, y 의 공분산은 다음과 같이 정의됨

$$\text{cov}(x, y) = E[(x - \bar{x})(y - \bar{y})] = E[xy] - \bar{x}\bar{y}$$

- 한 쪽 변수의 값이 커짐에 따라 다른 변수 또한 커지면 양의 상관관계를, 작아지면 음의 상관관계를 가짐
- 서로 상관관계가 없는 독립적인 관계에서는 공분산이 0이 됨

Covariance Matrix (2)

공분산 행렬 (covariance matrix)

- 각각의 데이터 피쳐들 사이의 공분산 값을 원소로 하는 행렬
- $x = [x_1, x_2, \dots, x_N]$
- $C = E[(x - \bar{x})(x - \bar{x})^T] : N \times N$ 행렬

$$C_{ij} = E[(x_i - \bar{x})(x_j - \bar{x})^T]$$

- 열에 대한 부분이 피쳐가 되며, 피쳐 사이의 공분산이 원소가 됨
- 공분산 행렬은 symmetric 성질을 만족함

Eigenvalue & Eigenvector (1)

고유값과 고유벡터 (eigenvalue & eigenvector)

- 행렬 A 를 선형변환으로 봤을 때, 선형변환 A 이후에도 자기 자신의 상수배가 되는 벡터를 고유벡터(eigenvector)라 부르고 이 때의 상수배 값을 고유값(eigenvalue)라 함
- $Av = \lambda v$
- 즉, 선형변환 A 에 의해 방향은 보존되고 스케일만 변환되는 방향 벡터와 스케일 값을 의미

Eigenvalue & Eigenvector (2)

고유값과 고유벡터 (eigenvalue & eigenvector)

- Ex. A가 회전 변환이라 가정했을 때, 회전축 벡터가 eigenvector, eigenvalue는 1이 됨



Eigen Decomposition (1)

고유값 분해 (eigen decomposition)

- 행렬 A 를 고유벡터를 열벡터로 하는 행렬과 고유값을 대각원소로 하는 행렬의 곱으로 분해
- 고유값과 고유벡터는 **정방행렬($n \times n$)**에 대해서만 정의됨
- 단, 행렬 A 가 n 개의 일차독립인 고유벡터를 가져야 함
(일차독립: 어느 한 벡터도 다른 벡터들의 일차결합으로 표현 불가능)
- **대칭 행렬(symmetric matrix)은 항상 고유값 분해 가능**

Eigen Decomposition (2)

고유값 분해 (eigen decomposition)

- 고유값과 고유벡터들을 λ_i, v_i ($i = 1, 2, \dots, n$ and $\mathbf{v} \neq \mathbf{0}$, $v_i: n \times 1$ matrix)라 할때,

$$Av_1 = \lambda_1 v_1$$

$$\vdots$$

$$Av_n = \lambda_n v_n$$

$$\bullet \quad A[v_1 \ v_2 \ \dots \ v_n] = [\lambda_1 v_1 \ \lambda_2 v_2 \ \dots \ \lambda_n v_n] = [v_1 \ v_2 \ \dots \ v_n] \begin{bmatrix} \lambda_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \lambda_n \end{bmatrix}$$

$$\mathbf{A} = [v_1 \ v_2 \ \dots \ v_n] \begin{bmatrix} \lambda_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \lambda_n \end{bmatrix} [v_1 \ v_2 \ \dots \ v_n]^{-1} = \mathbf{PDP}^{-1}$$

Eigen Decomposition (3)

고유값 분해 (eigen decomposition)

- 실제 계산은 다음과 같이 진행함
- $Av = \lambda v$

$$Av - \lambda v = 0$$

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

- 만약 $(A - \lambda I)$ 의 역행렬이 존재한다면 $v = 0$ 의 해를 항상 갖게 되지만, 이는 정의에 부합하지 않음

Eigen Decomposition (4)

고유값 분해 (eigen decomposition)

- $\det(A - \lambda I) = 0$
- 행렬식(determinant): 어떤 행렬의 역행렬 존재 여부에 대한 판별값 역할

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$$

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

- 특성 방정식(characteristic equation): 위 형태의 행렬식에 대한 방정식을 의미

Eigen Decomposition (5)

고유값 분해 (eigen decomposition)

- $A = \begin{bmatrix} 2 & 0 & -2 \\ 1 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix}$
- $\det(A - \lambda I) = \begin{vmatrix} 2 - \lambda & 0 & -2 \\ 1 & 1 - \lambda & -2 \\ 0 & 0 & 1 - \lambda \end{vmatrix} = (2 - \lambda)(1 - \lambda)^2 = 0$
- $\lambda = 1, 2$ 의 **고유값에 대한** 두 가지 해가 존재
- 참고로 이중근은 2개, 삼중근은 3개의 고유벡터를 얻을 수 있음

Eigen Decomposition (6)

고유값 분해 (eigen decomposition)

- $(A - 2I)v = \begin{bmatrix} 0 & 0 & -2 \\ 1 & -1 & -2 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$
- $v_z = 0, v_x = v_y$ 을 만족해야 되고, 이를 만족하는 벡터는 무수히 많음
- 행렬에 대한 고유값은 유일하지만, 고유벡터는 유일하지 않음
 $Av = \lambda v, A(cv) = \lambda(cv), A(v_1 + v_2) = \lambda(v_1 + v_2), \dots$
- 일반적으로 벡터의 크기가 1인 단위벡터를 사용함
- $v_1 = \left[\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0 \right]^T$

Eigen Decomposition (7)

고유값 분해 (eigen decomposition)

- $(A - I)v = \begin{bmatrix} 1 & 0 & -2 \\ 1 & 0 & -2 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$
- $v_x = 2v_z$ 를 만족하는 단위벡터를 찾아야 함
- $v = [2t, s, t]^T = t[2, 0, 1]^T + s[0, 1, 0]^T$ 의 선형결합으로 표현 가능
- $v_2 = \left[\frac{2}{\sqrt{5}}, 0, \frac{1}{\sqrt{5}}\right]^T, v_3 = [0, 1, 0]^T$

Principal Component Analysis (1)

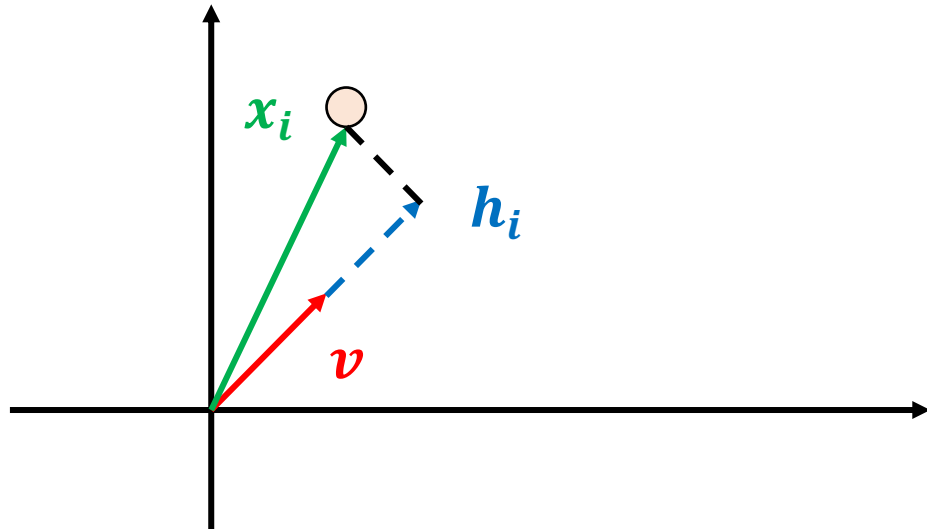
주성분 분석 (PCA)

- 전체 데이터의 분포를 가장 잘 설명할 수 있는 주성분을 찾아주는 방법론
- 이를 **데이터의 공분산 행렬에 대한 고유값 분해**를 통해 얻음
- 공분산 행렬은 대칭 행렬이기 때문에 고유값 분해가 항상 가능함
- 가장 큰 분산을 가진 방향의 고유벡터로 데이터를 투영시켜 차원을 축소시킴

Principal Component Analysis (2)

주성분 분석 (PCA)

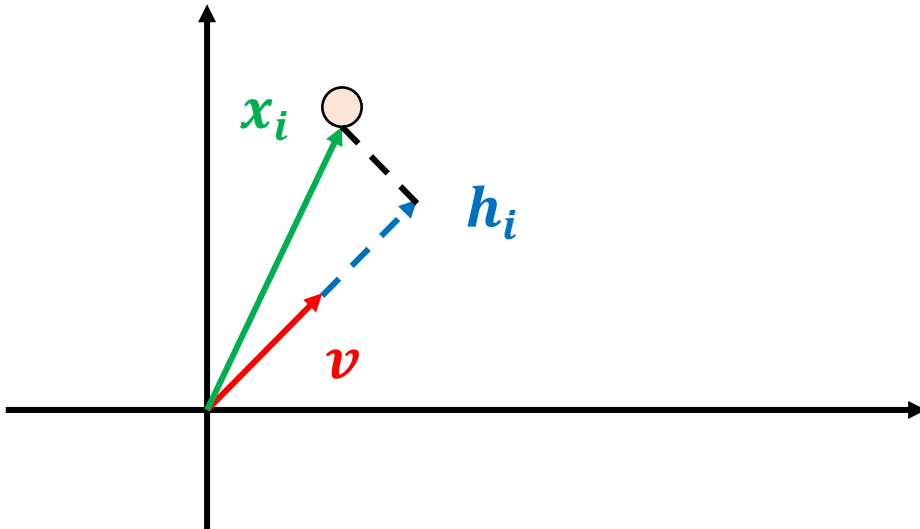
- Q. 공분산 행렬의 고유값, 고유벡터가 왜 분산과 분산에 관한 방향벡터인가?
- 데이터를 $X = [x_1, x_2, \dots, x_n]^T$ 로 하고 평균을 원점이라 가정
- 임의의 단위벡터를 v 로 가정
- 단위벡터 v 로 데이터를 투영시켰을 때의 벡터를 h_i 라 가정



Principal Component Analysis (3)

주성분 분석 (PCA)

- $h_i = (\|x_i\| \cos \theta) v = \left(\|x_i\| * \frac{x_i \cdot v}{\|x_i\| * \|v\|} \right) v = (x_i \cdot v) v$
- 데이터들의 분산이 최대화되는 방향이란, h_i 의 크기에 대한 분산이 최대화된 것임
- $\sigma_{\|h_i\|}^2$ 를 최대화시키는 v 를 찾을 것임



Principal Component Analysis (4)

주성분 분석 (PCA)

- $$\sigma_{\|h_i\|}^2 = \frac{1}{n} (\sum_i (x_i \cdot v)^2) - \left(\frac{1}{n} \sum_i (x_i \cdot v) \right)^2 = \frac{1}{n} \sum_i (x_i \cdot v)^2$$

$$= \frac{1}{n} (Xv)^T (Xv) = \frac{1}{n} v^T X^T X v = \mathbf{v}^T \frac{X^T X}{n} \mathbf{v} = \mathbf{v}^T \mathbf{C} \mathbf{v}$$
- (Remind) 공분산 행렬: $C = E[(x - \bar{x})(x - \bar{x})^T] = \frac{X^T X}{n}$
- $$\max_v \mathbf{v}^T \mathbf{C} \mathbf{v}$$

subject to $\mathbf{v}^T \mathbf{v} = 1$

Principal Component Analysis (5)

주성분 분석 (PCA)

- $\max_v v^T C v$
subject to $v^T v = 1$
- 위의 문제는 라그랑지 승수 벡터를 통해 다음과 같이 표현 가능:

$$\max_v v^T C v - \lambda(v^T v - 1) = \max_v L$$

- $\frac{\partial L}{\partial v} = 2Cv - 2\lambda v = 0$
 $\Rightarrow Cv = \lambda v$

Principal Component Analysis (6)

주성분 분석 (PCA)

- $Cv = \lambda v$
- 공분산 행렬 C에 대한 고유벡터가 분산을 최대화시키는 방향벡터임
- 또한, 그 때의 분산의 크기가 고유값이 됨
- 공분산 행렬에 대한 고유값 λ_i 를 정렬해, 데이터를 투영할 방향벡터의 순서를 결정함


$$C = [v_1 \ v_2 \ \cdots \ v_n] \begin{bmatrix} \lambda_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_n \end{bmatrix} [v_1 \ v_2 \ \cdots \ v_n]^{-1}$$

- 이 후, 원하는 차원의 크기까지 순서대로 투영시키면 됨

Principal Component Analysis (7)

주성분 분석 (PCA)

- Q. PCA를 통해 얻은 주성분 벡터들은 왜 수직인 관계인가?
- 고유값 분해가 가능한 정방행렬은 n 개의 일차독립인 고유벡터를 가져야 함
- 서로 독립의 관계를 가지기 때문에, 주성분 벡터들은 서로 수직임



Part Three

SVD

Singular Value Decomposition (1)

특이값 분해 (singular value decomposition)

- 특이값 분해는 정확히 말하면 “차원 축소”보단 “데이터 압축”을 시킬 수 있는 방법임
- 고유값 분해처럼 행렬을 대각화하는 방법으로, 모든 크기의 $m \times n$ 행렬 A 에 대해 적용 가능
- $A = U\Sigma V^T$
- U 와 V 는 각각 $m \times m$, $n \times n$ 크기의 직교행렬이고, Σ 는 $m \times n$ 행렬임
- 직교행렬(orthogonal matrix)이란, 전치행렬이 역행렬이 되는 행렬

$$UU^T = I, U^{-1} = U^T$$

Singular Value Decomposition (2)

특이값 분해 (singular value decomposition)

- 특이값 분해의 기하학적 의미는 다음과 같음
- 2x2 크기의 직교행렬은 다음의 꼴과 동치임

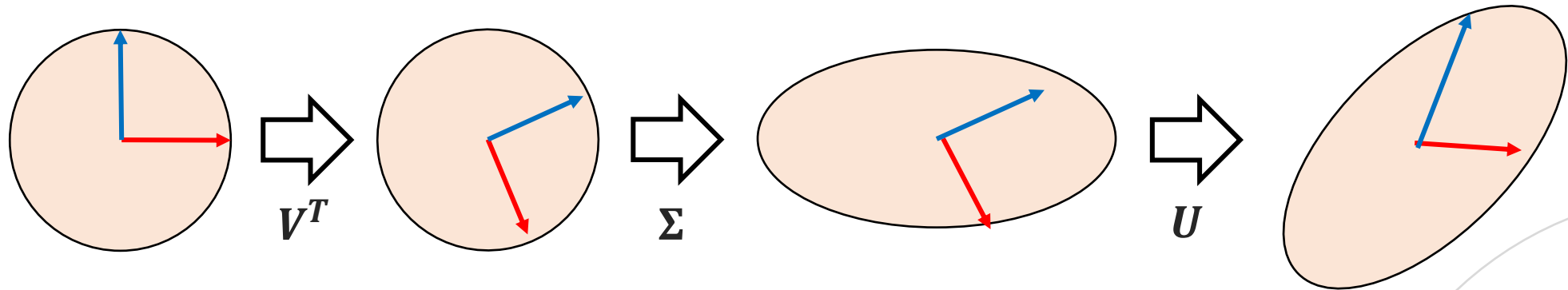
$$A = \begin{pmatrix} \cos\theta & -\sin\theta \\ \epsilon \sin\theta & \epsilon \cos\theta \end{pmatrix} \quad \theta \in \mathbb{R}, \epsilon = \pm 1$$

- $\epsilon = 1$ 일 때는 회전변환행렬(rotation matrix)와 같은 꼴이며, $\epsilon = -1$ 일 때는 뒤집혀진 회전변환(reflection rotation matrix)을 의미함
- 대각행렬(diagonal matrix)은 각 축으로의 스케일변환을 의미함

Singular Value Decomposition (3)

특이값 분해 (singular value decomposition)

- 행렬 A 의 선형변환을 회전변환과 스케일변환으로 나누는 것이 특이값 분해임 ($A = U\Sigma V^T$)



Singular Value Decomposition (4)

특이값 분해 (singular value decomposition)

- $A = U\Sigma V^T$
- U 는 AA^T 의 고유벡터들을 열벡터로 가지고 있는 행렬로, 이 때의 열벡터들을 A 의 left singular vector라 부름

$$U = [u_1 \ u_2 \ \cdots \ u_m], \quad (AA^T)u_i = \lambda_i u_i$$

- V 는 $A^T A$ 의 고유벡터들을 열벡터로 가지고 있는 행렬로, 이 때의 열벡터들을 A 의 right singular vector라 부름

$$V = [v_1 \ v_2 \ \cdots \ v_n], \quad (A^T A)v_i = \lambda_i v_i$$

Singular Value Decomposition (5)

특이값 분해 (singular value decomposition)

- $A = U\Sigma V^T$
- Σ 는 AA^T 혹은 $A^T A$ 의 고유값들의 root 값(σ)을 대각원소로 하는 $m \times n$ 직사각 대각행렬임
- 이 때의 원소들을 특이값(singular value)이라 부름

$$\Sigma = \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_n \\ & & & 0 \end{bmatrix} (m > n) \text{ or } \Sigma = \begin{bmatrix} \sigma_1 & & & \\ & \ddots & & \\ & & \sigma_m & \\ & & & 0 \end{bmatrix} (m < n)$$

Property in SVD (1)

특이값 분해 (singular value decomposition)

- AA^T 와 $A^T A$ 는 모두 정방행렬이므로 고유값 분해가 가능함
- 이 때의 고유값들은 모두 0 이상임

$$A^T A v = \lambda v \implies v^T A^T A v = \lambda v^T v \implies (Av)^T A v = \lambda v^T v \implies \|Av\|^2 = \lambda \|v\|^2$$

- 0이 아닌 고유값들은 서로 동일하여, 하나의 행렬 Σ 로 표현

$$A^T A v = \lambda v \implies A(A^T A)v = \lambda A v \implies AA^T(Av) = \lambda(Av) \text{ if } Av \neq 0$$

(if $Av = 0$, $A^T A v = \lambda v$ 때문에 고유값은 0이므로 가정에 위반)

Property in SVD (2)

특이값 분해 (singular value decomposition)

- 참고로 고유값이 0일 때도 존재함
- $Av = \lambda v = 0$
- 고유벡터는 영벡터가 될 수 없으므로, **A는 역행렬이 없는 특이 행렬(singular matrix)임**

Property in SVD (3)

특이값 분해 (singular value decomposition)

- $\Sigma = \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_n \\ & & & 0 \end{bmatrix} (m > n)$
- $m > n$ 인 경우, $\mathbf{u}_{n+1}, \dots, \mathbf{u}_m$ 고유벡터들의 고유값은 0인 상황임
 $\Rightarrow A A^T$ 행렬이 특이 행렬일 것임
- $\sigma_1, \dots, \sigma_n \geq 0$ 이므로, 0의 값인 singular value가 존재할 수도 있음
 $\Rightarrow A^T A$ 행렬이 특이 행렬일 것임

Property in SVD (4)

특이값 분해 (singular value decomposition)

- $\Sigma = \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_n \\ & & & 0 \end{bmatrix} (m > n)$
- $\sigma_n = 0$ 이라면, $u_n, u_{n+1} \dots, u_m$ 고유벡터들의 순서는 무작위로 바뀌어도 상관 없음
- 고유벡터의 수는 무수히 많은데 어떤 방향 벡터를 사용해야 되는가?
⇒ 직교행렬의 성질을 만족하기 위해 u_i, v_i 모두 단위벡터로 사용

Data Compression using SVD (1)

데이터 압축

- $m \times n$ 크기의 행렬 A 를 SVD를 통해 분해하는 상황
- Full SVD

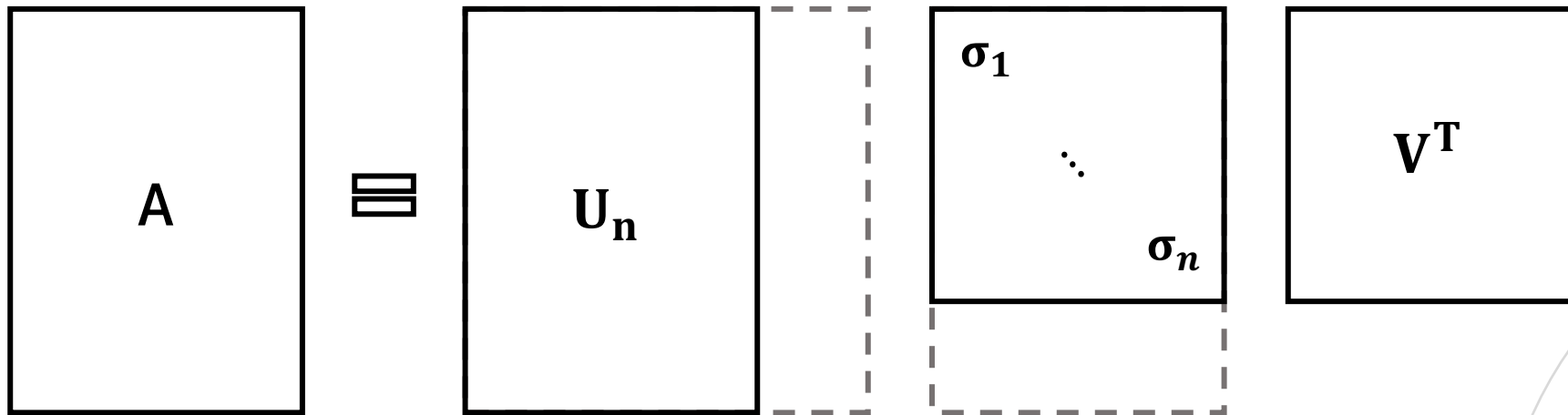
The diagram shows the Full SVD decomposition of matrix A . Matrix A is represented by a square box on the left. To its right is an equals sign, followed by three square boxes representing the components of the decomposition: matrix U , matrix Σ , and matrix V^T . Matrix Σ is depicted with its diagonal elements $\sigma_1, \dots, \sigma_n$ and a zero at the bottom, indicating the presence of zero singular values in the full SVD.

$$A = U \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_n \\ & & & 0 \end{bmatrix} V^T$$

Data Compression using SVD (2)

데이터 압축

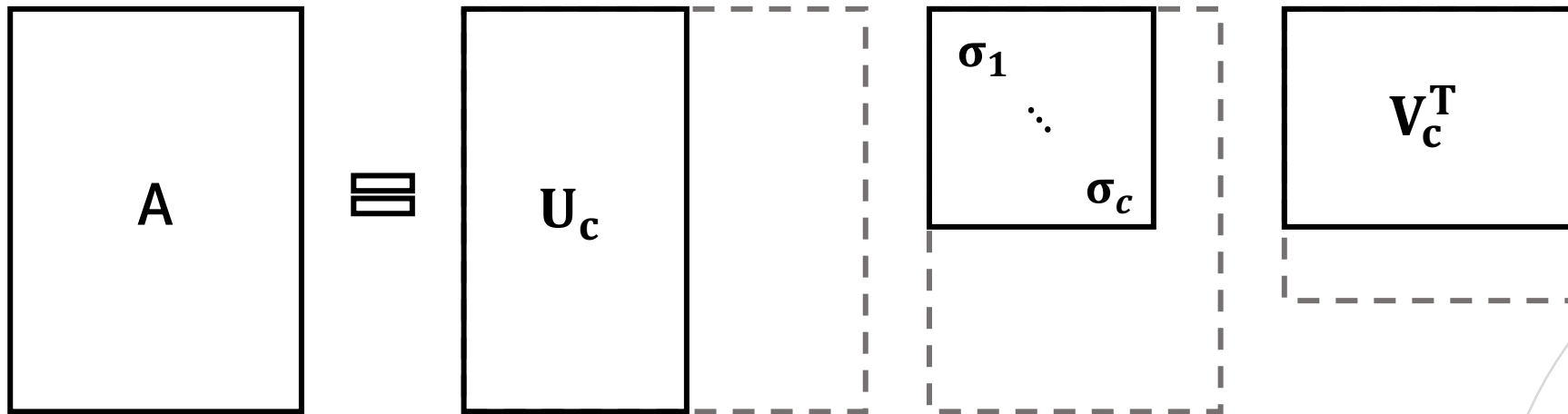
- $m \times n$ 크기의 행렬 A 를 SVD를 통해 분해하는 상황
- **Thin SVD**: 동일한 행렬 A 출력



Data Compression using SVD (3)

데이터 압축

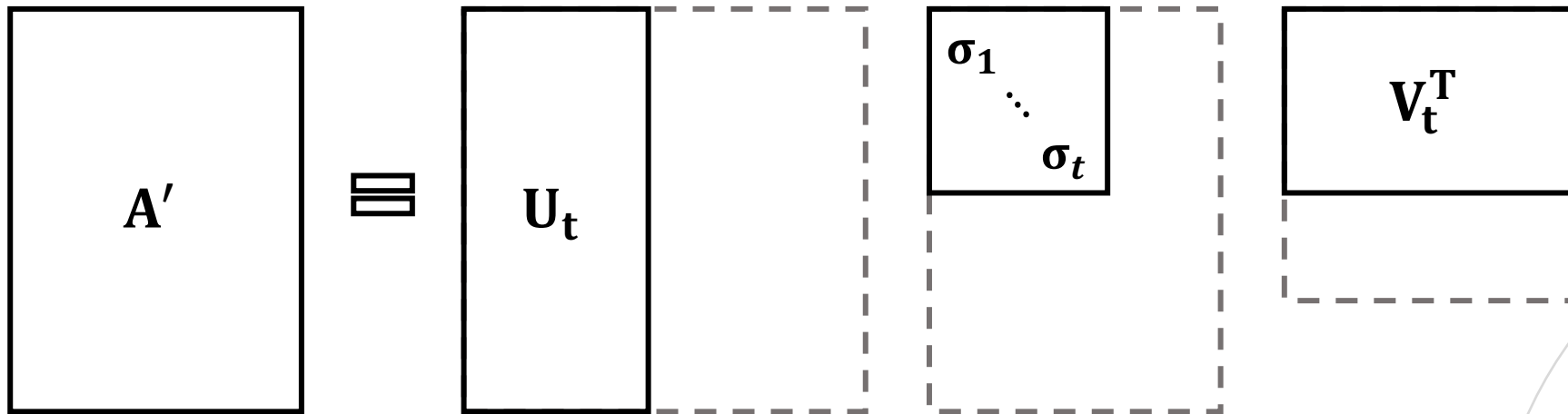
- $m \times n$ 크기의 행렬 A 를 SVD를 통해 분해하는 상황
- **Compact SVD**: 0의 특이값을 제외하는 방법으로, 동일한 행렬 A 출력



Data Compression using SVD (4)

데이터 압축

- $m \times n$ 크기의 행렬 A 를 SVD를 통해 분해하는 상황
- **Truncated SVD**: 작은 특이값까지 제외하는 방법으로, 근사 행렬 A' 출력





Part Four

LDA

Linear Discriminant Analysis (1)

(Remind) LDA

- $$P(Y = k|X = x) = \frac{P(X = x|Y = k)P(Y=k)}{\sum_{l=1}^K P(X = x|Y = l)P(Y=l)}$$
$$= \frac{\pi_k f_k(x)}{\sum_{l=1}^K \pi_l f_l(x)}$$
- 다음의 두 가지 가정을 사용함
 1. Density function이 Normal 혹은 Gaussian density를 따른다
 2. $\sigma_k = \sigma$ for all k

Linear Discriminant Analysis (2)

(Remind) LDA

- $$p_k(\mathbf{x}) = P(Y = k | X = \mathbf{x}) = \frac{\pi_k \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{\mathbf{x}-\mu_k}{\sigma}\right)^2}}{\sum_{l=1}^K \pi_l \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{\mathbf{x}-\mu_l}{\sigma}\right)^2}}$$
- 판별 함수는 $p_k(\mathbf{x})$ 에 대한 값을 뺀 함수
- $$\arg \max_{1 \leq k \leq K} P(Y = k | X = \mathbf{x}) = \arg \max_{1 \leq k \leq K} \pi_k \exp\left(-\frac{1}{2\sigma^2} (\mathbf{x}^2 - 2\mu_k \mathbf{x} + \mu_k^2)\right)$$

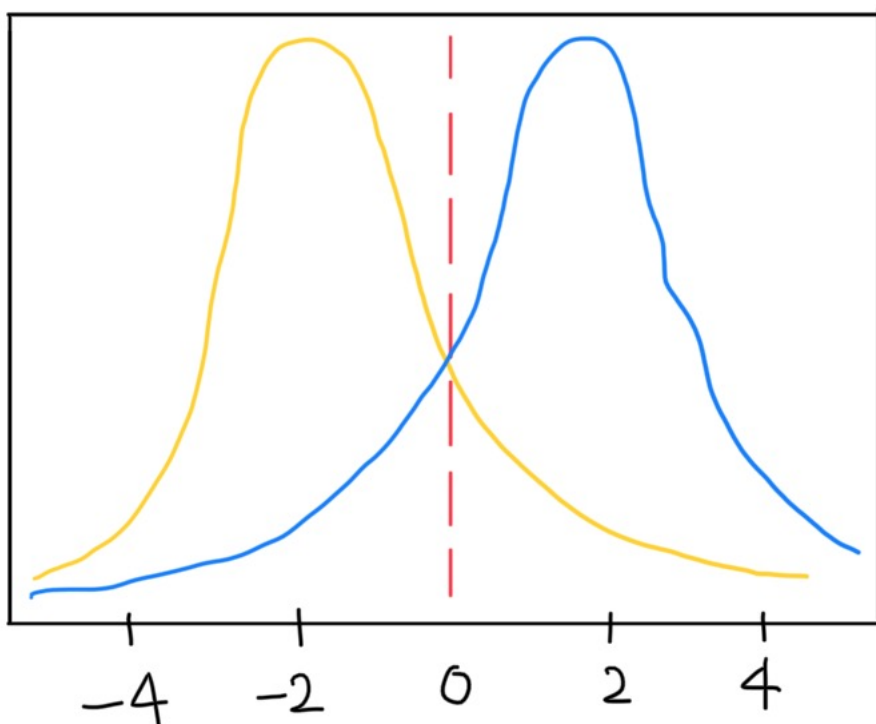
$$= \arg \max_{1 \leq k \leq K} \mathbf{x} \cdot \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k)$$

Linear Discriminant Analysis (3)

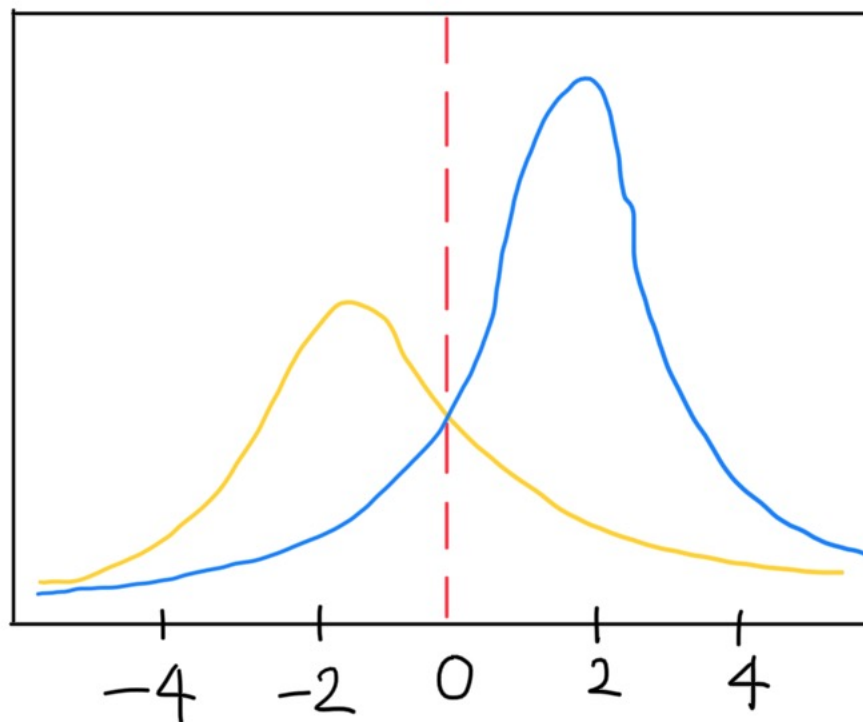
(Remind) LDA

- 결정 경계는 $\delta_1(x) = \delta_2(x)$ 를 통해 얻을 수 있음

$$\pi_1 = 0.5, \pi_2 = 0.5$$



$$\pi_1 = 0.3, \pi_2 = 0.7$$



Dimension Reduction with LDA (1)

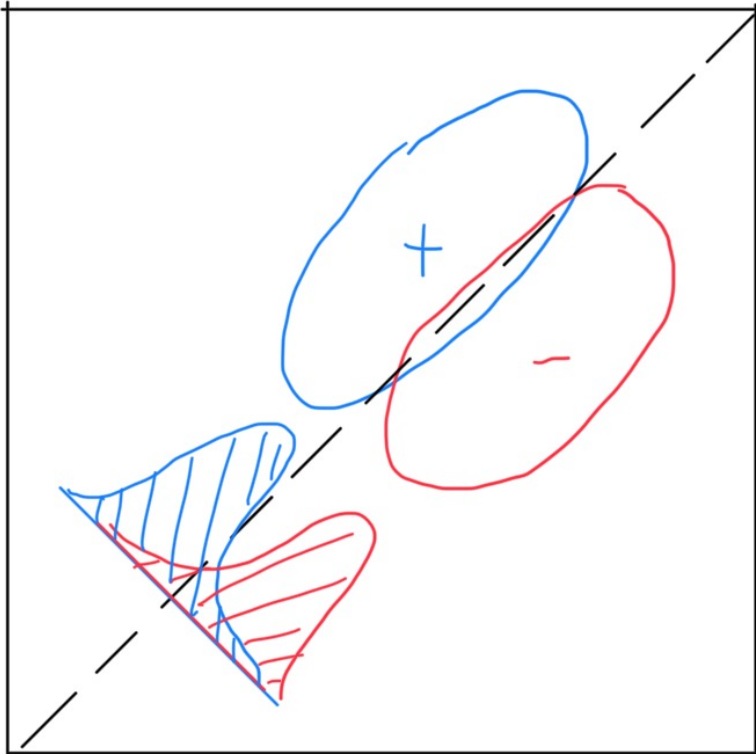
LDA를 활용한 차원 축소

- PCA는 비지도 방법으로 데이터의 전체적인 분포를 참고해 주성분을 찾고, 새로운 피쳐로 데이터를 투영시키는 것이 목적
- LDA는 지도적인 방법으로 축을 찾아, 데이터를 투영시키는 방법론

Dimension Reduction with LDA (2)

LDA를 활용한 차원 축소

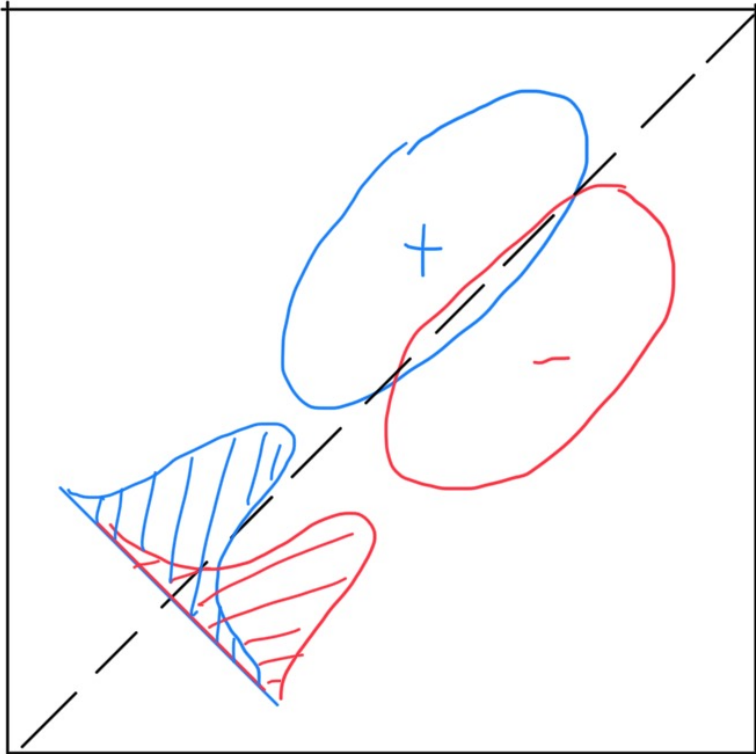
- 이전의 방법은 Bayes' rule의 확률모형으로부터 도출한 방법
- 분산의 관점에서 LDA를 똑같이 도출하고, 차원 축소에 사용할 수 있음



Dimension Reduction with LDA (3)

LDA를 활용한 차원 축소

- 데이터의 차원을 축소한 이후에, 같은 클래스의 데이터 안에서의 분산(within 분산)은 최소화, 다른 클래스간의 분산(between 분산)은 최대화



Dimension Reduction with LDA (4)

LDA를 활용한 차원 축소

- 클래스의 개수가 2개고, 2차원의 데이터 x 를 앞의 목적을 만족하는 단위벡터 w 로 투영
- 데이터 x 를 w 에 투영시켰을 때, 1차원에서의 좌표값을 p 로 정의

$$p_i = w^T x_i$$

- 각각의 클래스의 중심 벡터값 m_1, m_2 정의

$$m_1 = \frac{1}{N_1} \sum_{y_i \in C_1} x_i, m_2 = \frac{1}{N_2} \sum_{y_i \in C_2} x_i$$

- 중심 벡터 또한 단위벡터 w 로 투영

$$\bar{p}_k = w^T m_k$$

Dimension Reduction with LDA (5)

LDA를 활용한 차원 축소

- Within 분산:

$$s_1^2 + s_2^2 = \sum_{y_i \in \mathcal{C}_1} (p_i - \bar{p}_1)^2 + \sum_{y_i \in \mathcal{C}_2} (p_i - \bar{p}_2)^2$$

- Between 분산:

$$\sum_{k \in \{1,2\}} \left(\bar{p}_k - \frac{\bar{p}_1 + \bar{p}_2}{2} \right)^2 \approx (\bar{p}_1 - \bar{p}_2)^2$$

Dimension Reduction with LDA (6)

LDA를 활용한 차원 축소

- 두 분산을 이용한 목적식 $J(w)$ 를 만듦

$$\begin{aligned}
 J(w) &= \frac{(\bar{p}_1 - \bar{p}_2)^2}{s_1^2 + s_2^2} = \frac{(\bar{p}_1 - \bar{p}_2)^2}{\sum_{y_i \in C_1} (p_i - \bar{p}_1)^2 + \sum_{y_i \in C_2} (p_i - \bar{p}_2)^2} \\
 &= \frac{w^T (m_1 - m_2)(m_1 - m_2)^T w}{w^T \left(\sum_{y_i \in C_1} (x_i - m_1)(x_i - m_1)^T + \sum_{y_i \in C_2} (x_i - m_2)(x_i - m_2)^T \right) w} \\
 &= \frac{w^T S_B w}{w^T S_W w}
 \end{aligned}$$

Dimension Reduction with LDA (7)

LDA를 활용한 차원 축소

- 두 분산을 이용한 목적식 $J(w)$ 를 만듦

$$J(w) = \frac{w^T S_B w}{w^T S_W w}$$

- 목적함수를 미분했을 때 0이 되는 지점에서 최대화가 됨

$$2S_B w (w^T S_W w) - (w^T S_B w) 2S_W w = 0$$

$$\Rightarrow S_B w - \frac{w^T S_B w}{w^T S_W w} S_W w = 0$$

$$\Rightarrow S_B w - J(w) S_W w = 0$$

Dimension Reduction with LDA (8)

LDA를 활용한 차원 축소

- $S_B w - J(w) S_W w = 0$
- $(S_W^{-1} S_B) w = J(w) w$
- 결국 목적식을 최대화시키는 방향벡터는 $S_W^{-1} S_B$ 행렬의 첫번째 eigenvector임
- 고유값이 가장 클 때의 고유벡터로 데이터를 투영시켜 차원 축소
- 이 때의 고유벡터는 결국 확률 모형을 통해 얻었던 결정 결계와 수직의 관계



Part Five

t-SNE

Stochastic Neighbor Embedding (1)

SNE (stochastic neighbor embedding)

- **현 차원**에서의 데이터에 대한 거리 정보를, **축소한 차원**에서도 유지시키고 싶음
- 데이터 샘플 x_i 를 기준으로 다른 샘플들에 대해 **가우시안 분포**를 활용해 가중치(거리) 부여

$$p_{j|i} = \frac{e^{-\frac{\|x_i - x_j\|^2}{2\sigma_i^2}}}{\sum_{k \neq i} e^{-\frac{\|x_i - x_k\|^2}{2\sigma_i^2}}}, \quad q_{j|i} = \frac{e^{-\|y_i - y_j\|^2}}{\sum_{k \neq i} e^{-\|y_i - y_k\|^2}}$$

- 각 샘플마다 다른 샘플과의 거리에 대한 분산은 매우 다를 것임 (σ_i 사용)

Stochastic Neighbor Embedding (2)

SNE (stochastic neighbor embedding)

- 원형 데이터에서의 거리와 축소된 차원에서의 거리를 비슷하게 만들어주는 손실 함수 정의

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

- 분포 사이의 거리를 정량화해주는 KL divergence (kullback leibler divergence) 사용
- 경사 하강법을 통해 y_i 를 이동시킴

$$\frac{\partial L}{\partial y_i} = 2 \sum_j (y_j - y_i)(p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j})$$

Stochastic Neighbor Embedding (3)

SNE (stochastic neighbor embedding)

- 두 샘플 간의 가중치여도, 어떤 샘플이 기준인지에 따라 차이가 있음 (σ 가 다르기 때문)
- **Symmetric한 가중치**를 갖기 위해 다음과 같이 정의

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$$

t-SNE (1)

t-SNE

- 가우시안 분포는 확률 값 변화의 경사가 가파른 특성이 있음
- 따라서, 일정 거리 이상부터는 값이 매우 작아지는 **crowding 문제**가 있음
- 먼 거리의 샘플에 대한 가중치 값이 의미를 잃어버리는 문제가 발생
- 가우시안 분포를 t 분포로 대체해 사용한 것이 t-SNE 알고리즘임

t-SNE (2)

t Distribution

- 정규 분포로부터 정의되었기 때문에 매우 유사한 분포의 형태를 가지고 있음
- 표본의 수가 적고 모집단의 표준편차를 모를 때, 모집단의 평균을 추정하기 위해 사용됨
- 자유도($n - 1$)에 따라 모양이 달라지지만, 작은 자유도에서는 정규분포보다 꼬리가 두터운 형태를 가짐

t-SNE (3)

t-SNE

- $f(x) = \frac{\Gamma(\frac{v+1}{2})}{\sqrt{v\pi}\Gamma(\frac{v}{2})} \left(1 + \frac{x^2}{v}\right)^{-\frac{v+1}{2}}$, where $\Gamma(n) = (n-1)!$
- 축소된 차원에서의 가중치를 자유도 1의 t 분포를 사용해 표현

$$q_{j|i} = \frac{\left(1 + \|y_i - y_j\|^2\right)^{-1}}{\sum_{k \neq i} \left(1 + \|y_i - y_k\|^2\right)^{-1}}$$

t-SNE (4)

t-SNE

- t-SNE 학습에는 **perplexity**라는 하이퍼파라미터가 필요
- 설정된 Perplexity를 만족하는 σ_i 를 찾아서 알고리즘에 사용함
- $\text{perplexity} = 2^{\text{entropy}} = 2^{\sum -p_{j|i} \log p_{j|i}}$
- Entropy는 클수록 불확실성 증가(uniform 분포), 작을수록 불확실성 감소(one-hot 분포)

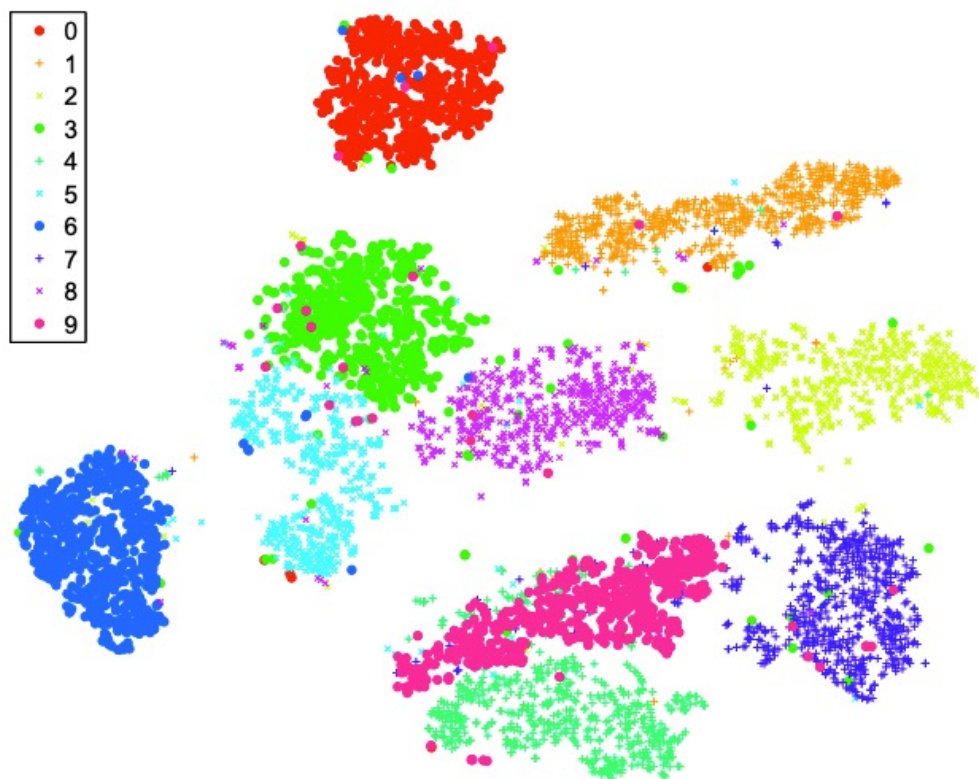
t-SNE (5)

t-SNE

- $perplexity = 2^{entropy} = 2^{\sum -p_{j|i} \log p_{j|i}}$
- 큰 값으로 설정하면, 모든 샘플에 대한 가중치가 비슷해지도록 큰 값의 σ_i 사용
- 큰 값의 perplexity는 차원 축소에 있어서 모든 샘플의 영향력이 커짐
- 반대의 경우 근처 샘플의 영향력이 매우 커짐
- 일반적으로 5 ~ 50 사이로 설정함

t-SNE (6)

t-SNE





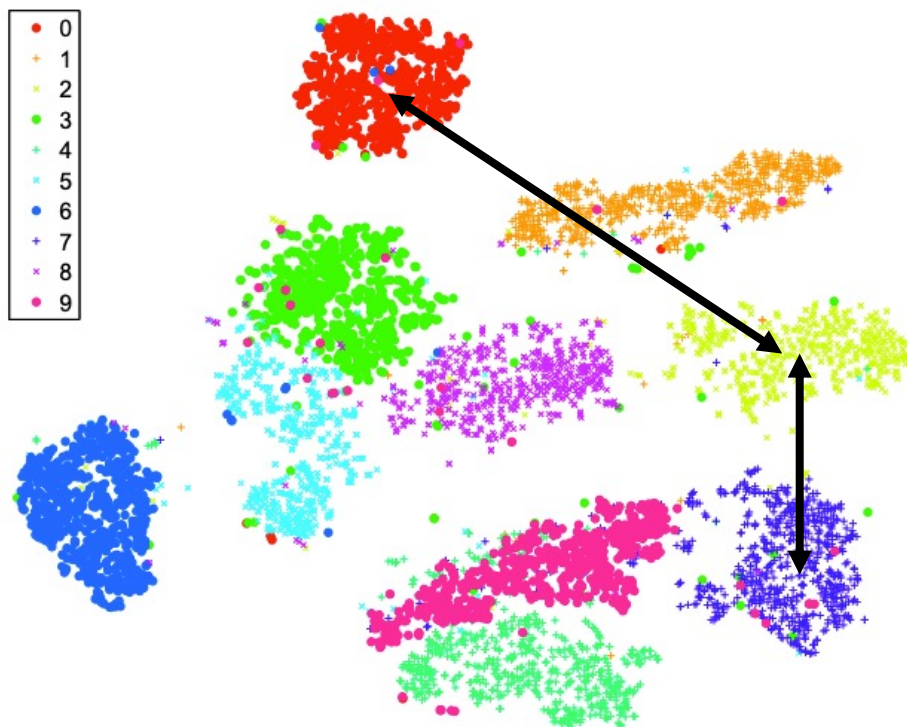
Part Six

UMAP

UMAP (1)

UMAP (uniform manifold approximation and projection)

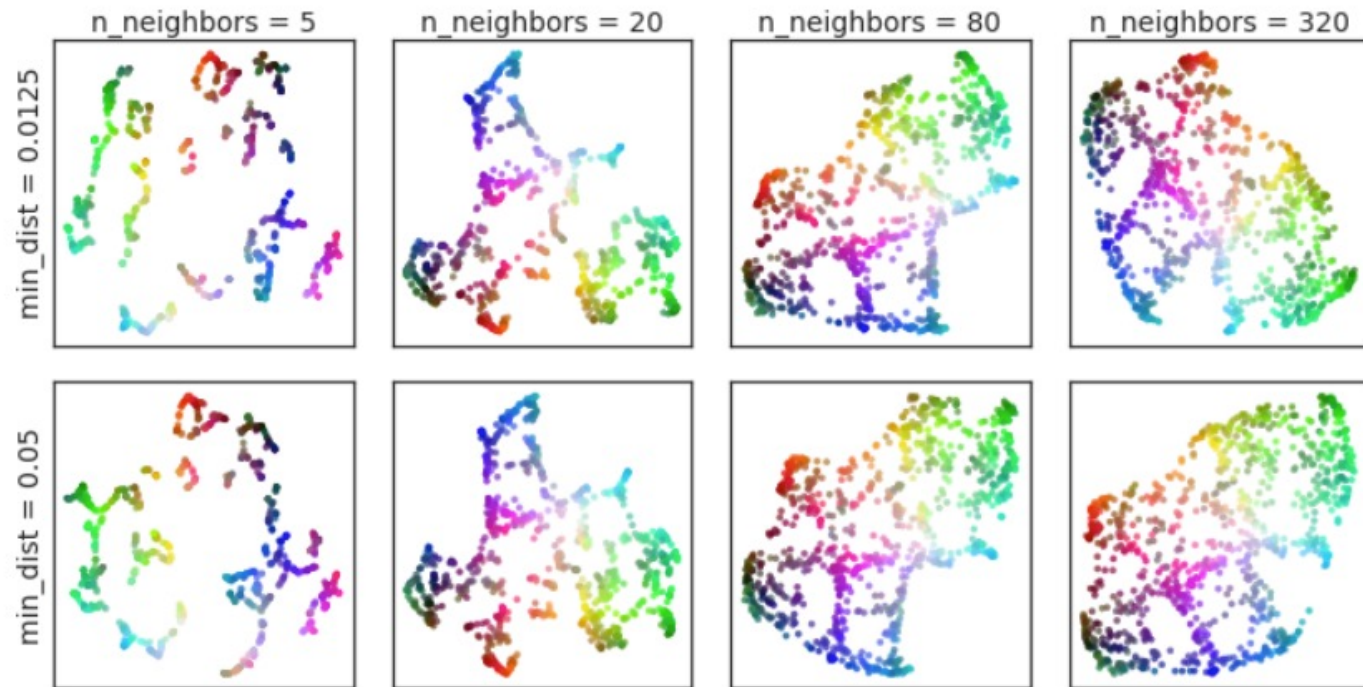
- t-SNE는 학습 속도가 매우 느린 단점이 있음
- 또한 클러스터 간의 유사성은 보장되지 않는 단점이 있음



UMAP (2)

UMAP (uniform manifold approximation and projection)

- 2018년도의 최신 알고리즘
- t-SNE보다 속도가 매우 빠르며 보다 수학적 내용을 기반으로 만들어짐





Thank you

Dimension Reduction