Statistical Machine Learning

5주차: 차원 축소

담당: 18기 신인수

4주차 과제 우수자

박민지님☺

코드를화면공유해주시고, 간단히설명해주세요!

리마인드: **매주 수요일 22시까지** 과제를 제출해주셔야 합니다! (github)

과제제출시, 파일명은 기존.ipynb 파일이름을 유지하되, '_이름'만
추가해주세요 ☺ (ML_week5_HW_방서연)

1. Curse of Dimensionality

2. Vector and Matrices

3. PCA (Principal Component Analysis

4. EFA (Explanatory Factor Analysis

5. LDA, QDA (Discriminant Analysis)

참고할 만한 자료

3Blue1Brown 선형대수학

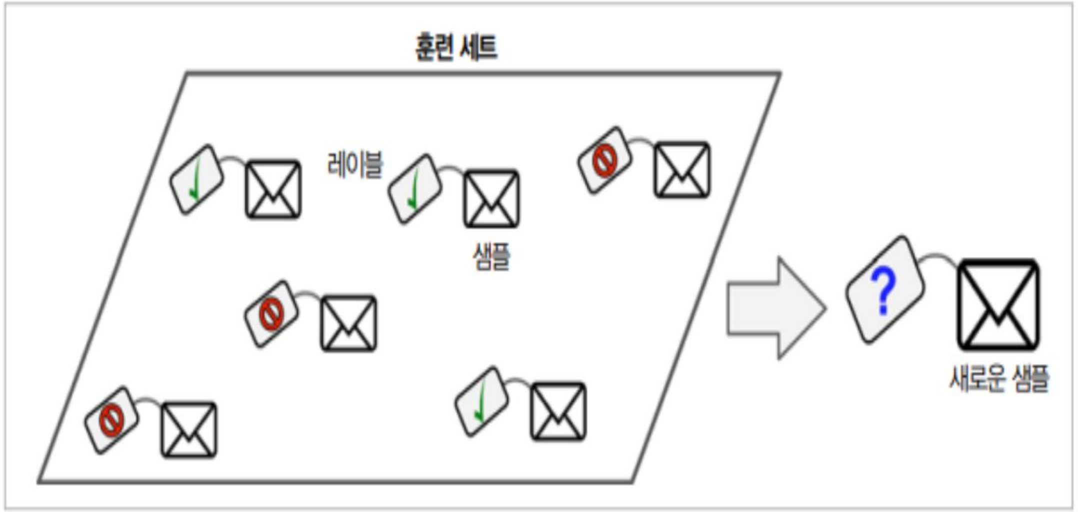
https://youtube.com/playlist?list=PLkoaxOTFHiqhVDo0nWybNmihCP_4BjOFR&feature=shared

StatQuest PCA

https://youtu.be/FgakZw6K1QQ?feature=shared

Dimension Reduction

Supervised Learning



Unsupervised Learning

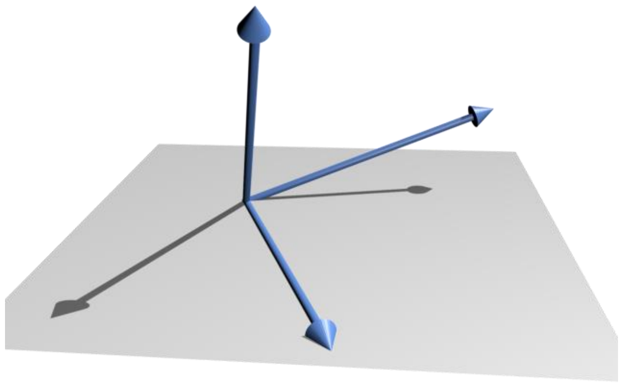
차원축소는 대표적인 비지도 학습법!

1. Curse of Dimensionality

What is Dimensionality?

- 사실 명확하게 정의하기는 어려움
- 분야마다(ex 수학, 물리학), 조건마다 (VC dimension 등 → take home) 정의가 다르기 때문
- 선형대수에서의 dimension 정의를 이용해보자.

→ **선형독립 (linearly independent) 한 벡터의 수**



Linearly independent → 3차원

Linearly dependent → 2차원
∵같은 평면에 존재

이미지: https://en.wikipedia.org/wiki/Linear_independence

Invertible Matrix Theorem (일부)

- **Linear Independence (선형독립)이란?**

$\boldsymbol{v_1}, \boldsymbol{v_2}, \cdots \boldsymbol{v_k}$: 벡터공간 V에 존재하는 벡터
$a_1, a_2 \cdots, a_k$: 스칼라
$\mathbf{0}$: 영벡터

$$a_1\boldsymbol{v_1} + a_2\boldsymbol{v_2} + \cdots + a_k\boldsymbol{v_k} = \mathbf{0}$$

$a_1, a_2 \cdots, a_k$ 중에서 적어도 1개는 0이 아닌 해가 있다.

Invertible Matrix Theorem (일부)

- **Invertible Matrix Theorem (일부)**

선형독립의 중요성을 보여주는 정리이자, 선형대수의 근간이 되는 정리

Invertible Matrix Theorem. *Let A be an $n \times n$ matrix, and let $T : \mathbf{R}^n \to \mathbf{R}^n$ be the matrix transformation $T(x) = Ax$. The following statements are equivalent:*

1. *A is invertible.*
2. *A has n pivots.*
3. $\mathrm{Nul}(A) = \{0\}$.
4. *The columns of A are linearly independent.* → 선형독립이 다양한 성질 보장
5. *The columns of A span $\mathbf{R}^n$.*
6. *$Ax = b$ has a unique solution for each b in $\mathbf{R}^n$.*
7. *T is invertible.*
8. *T is one-to-one.*
9. *T is onto.*

Curse of Dimensionality

When dimensionality increases

- data becomes increasingly sparse in the data space
- most training instances are likely to be far away from each other
- New instance will be likely be far away from training instance → overfitting

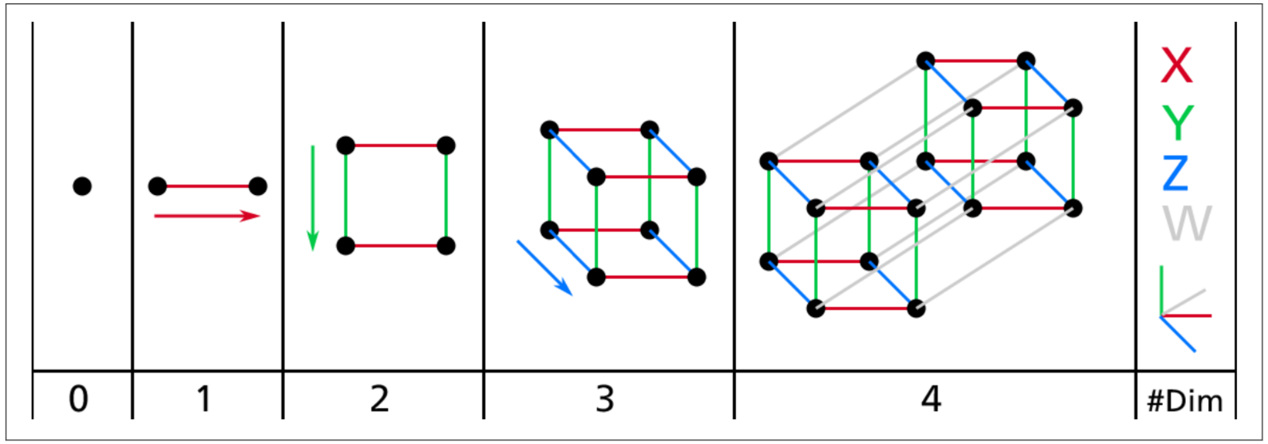
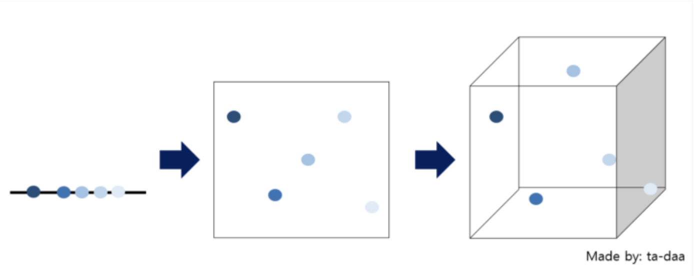


Figure 8-1. Point, segment, square, cube, and tesseract (0D to 4D hypercubes)[2]



Curse of Dimensionality

week5_curse_of_dim.R 참고

왜 일어나는가?

Recap) 여러 가지 distance

- Distance measure

$$d(\mathbf{u},\mathbf{v}) = (\sum|u_i - v_i|^2)^{\frac{1}{2}} = ||\mathbf{u}-\mathbf{v}||_2 \qquad \textit{Euclidean (L2 norm)}$$

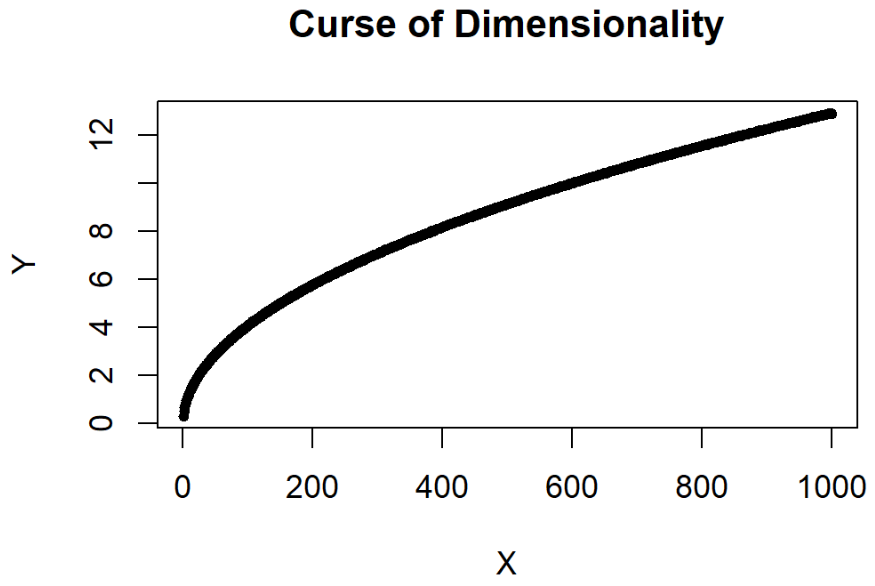
$$d(\mathbf{u},\mathbf{v}) = \sum|u_i - v_i| = ||\mathbf{u}-\mathbf{v}||_1 \qquad \textit{Manhattan (L1 norm)}$$

$$d(\mathbf{u},\mathbf{v}) = (\sum|u_i - v_i|^p)^{\frac{1}{p}} = ||\mathbf{u}-\mathbf{v}||_p \qquad \textit{Minkowski (Lp norm)}$$

$$d(\mathbf{u},\mathbf{v}) = \sqrt{(\mathbf{x}-\boldsymbol{\mu})^T \Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})} \qquad \textit{Mahalanobis Distance}$$

dimension 증가 → i 증가 → distance 증가

거리는 차원에 대해 단조증가



Curse of Dimensionality

Solution

Increase Size N

- the number of training instances required to reach a given density grows exponentially with the number of dimensions

Feature Selection

- Choosing $k<d$ important features, ignoring the remaining $d - k$
- Subset selection algorithms

Feature Extraction

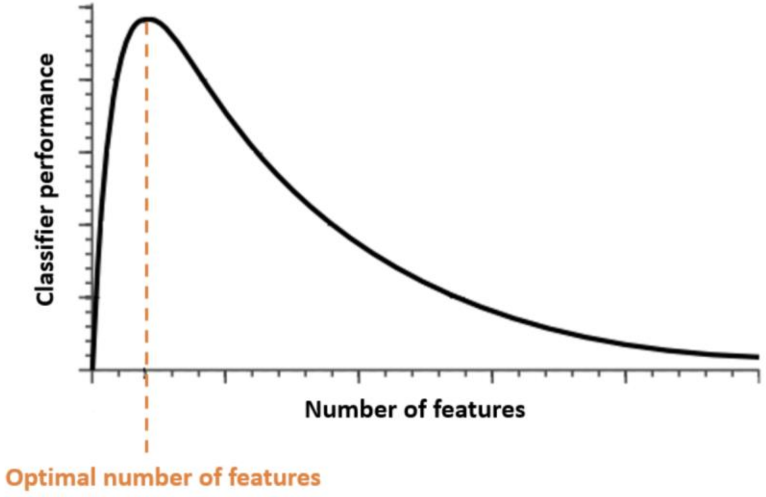
- Project the original x_i , i =1,...,d dimensions to new $k<d$ dimensions, z_j , j =1,...,k
- Ex) PCA

Regularization

- Lasso, Ridge regression

Why Reduce Dimensionality?

- Reduces time complexity : Less computation
- Reduces space complexity : Fewer parameters
- Saves the cost of observing the feature
- Simpler models are more robust on small datasets : more General model
- More interpretable : simpler explanation
- Data visualization (structure, groups, outliers, etc) if plotted in 2 or 3 dimensions



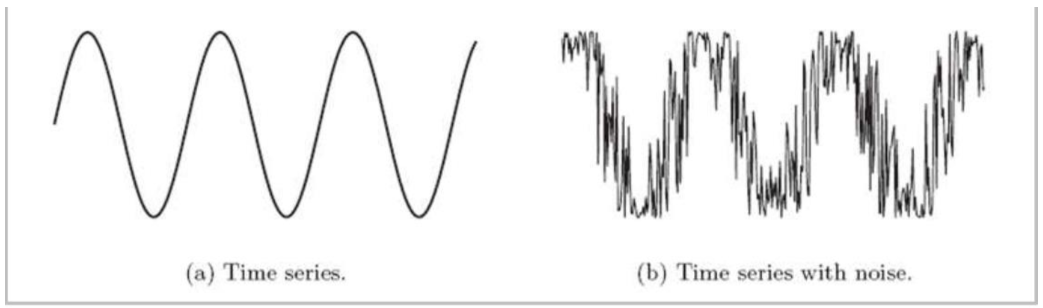


Figure 2.5. Noise in a time series context.

결정적으로

차원축소를 하는 이유는

1. Curse of Dimensionality를 완화하기 위함
2. 우리는 3차원보다 높은 차원을 이해하기 어렵기 때문에
3. 단순선형회귀의 경우 N < p (데이터수 < 변수 개수)면 다음의 해는 존재하지 않음

∵ Invertible matrix theorem와 $rank(X^TX) = rank(X)$ 에 의해

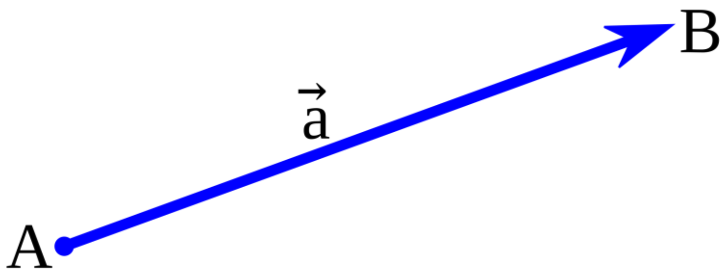
$$\hat{\beta} = \left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top \mathbf{y}$$

$$\hat{\mathbf{y}} = \mathbf{X}\,\hat{\beta}$$

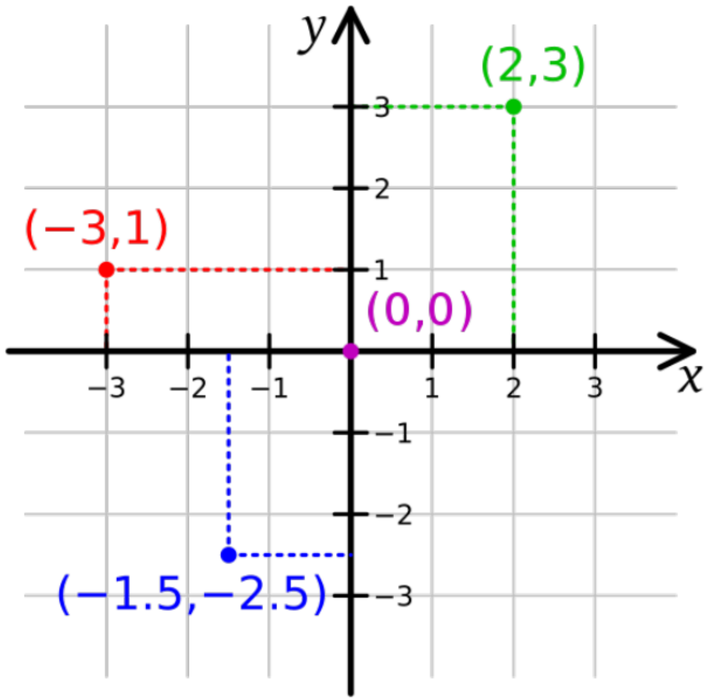
2. Vector and Matrices

Vector

- Vector(벡터): 크기와 방향이 있는 물리량
- 화살표, 혹은 좌표 평면 (좌표 공간)의 점으로 표현
- 주로 column vector로 표현



$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}$$



Matrix

- Matrix(행렬): 수 또는 다항식을 직사각형 모양으로 배열한 것
- **“열 벡터의 모임”**
- A는 n x m 행렬 → m개의 열벡터

$$\boldsymbol{A}_{n\times m} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{bmatrix} = [\boldsymbol{v_1} \quad \boldsymbol{v_2} \quad \cdots \quad \boldsymbol{v_m}]$$

Matrix Multiplication

- 벡터 x, y, 행렬 A에 대해 → $\mathbf{y} = \mathbf{A}\mathbf{x}$
- x 벡터는 A 행렬의 열벡터로 표현하면 어디로 갈까? → y 벡터로 간다
  - Linear Map (선형 변환)
  - Rotation (회전 변환)

Ex: 선형변환 예시 $\boldsymbol{A} = \begin{bmatrix} \mathbf{2} & \mathbf{-1} \\ \mathbf{3} & \mathbf{2} \end{bmatrix}$, $\boldsymbol{x} = \begin{bmatrix} \mathbf{4} \\ \mathbf{2} \end{bmatrix}$일 때, $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$는 어디로 갈까? → 다음 슬라이드

Column Vector

$$\begin{pmatrix} x \\ y \end{pmatrix} = x\boldsymbol{i} + y\boldsymbol{j}$$

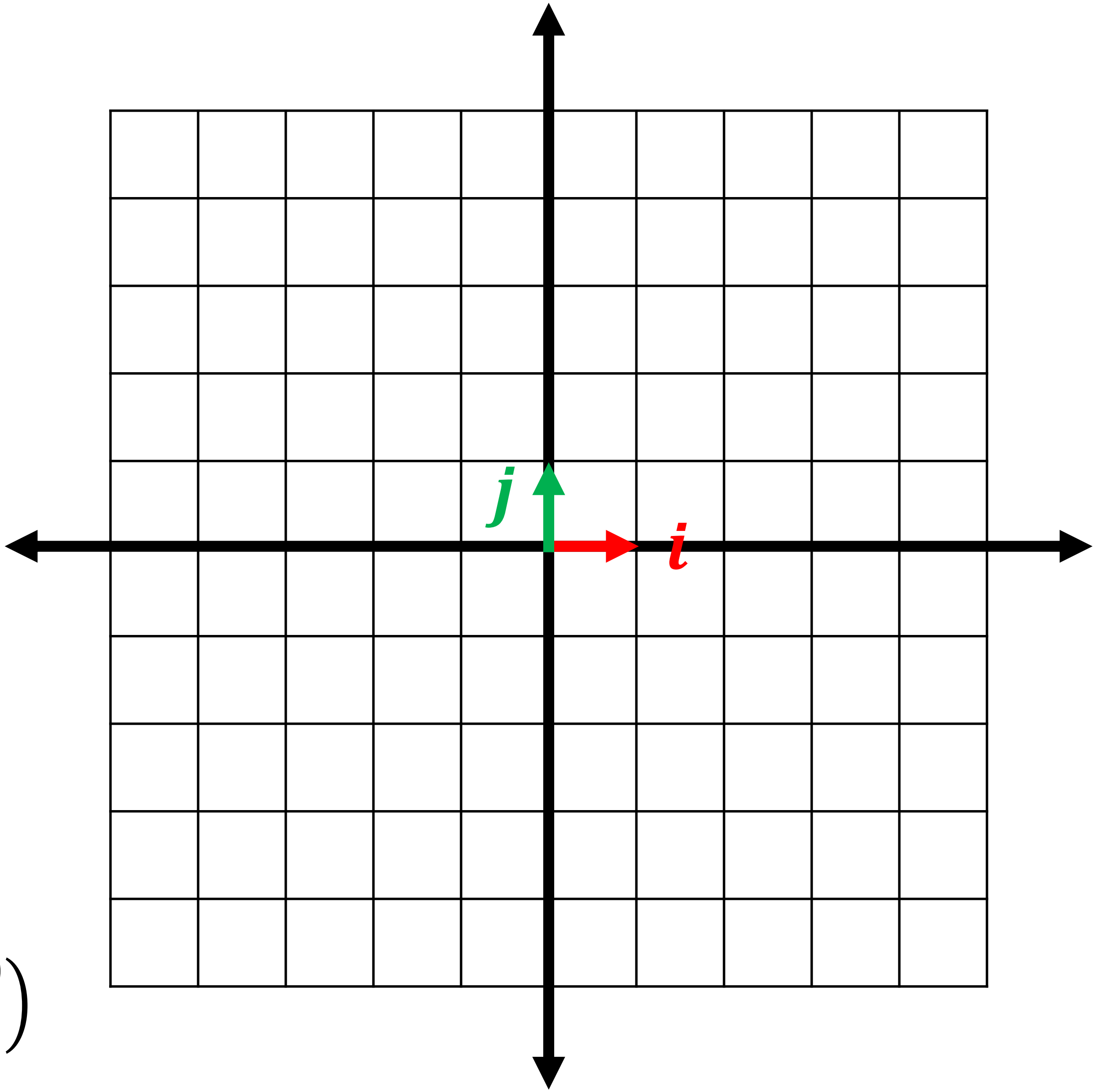
Unit Vector

$$\boldsymbol{i} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$$\boldsymbol{j} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Identity Matrix

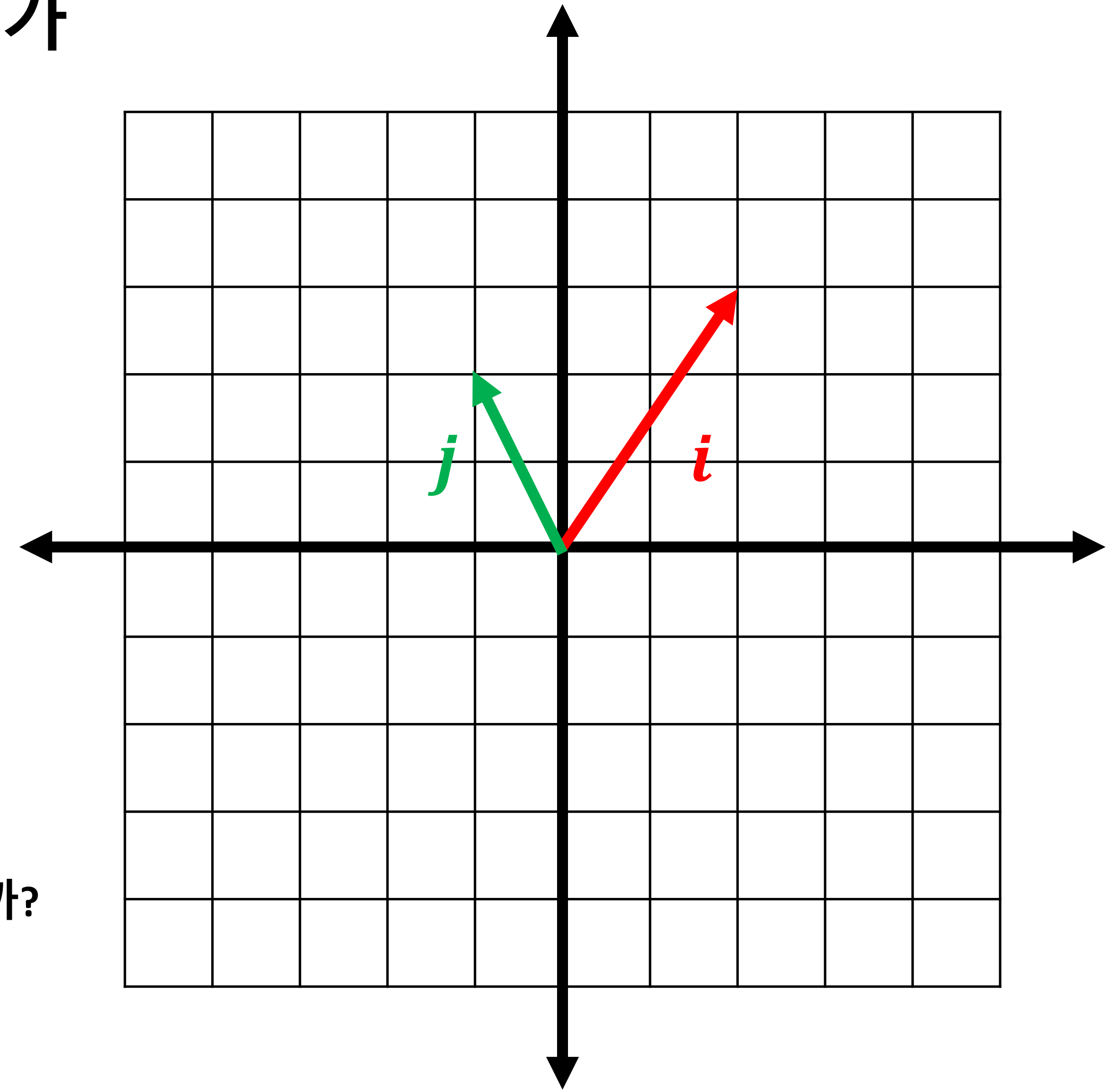
$$\boldsymbol{I}_2 = (\boldsymbol{i} \quad \boldsymbol{j}) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$



어떤 변환으로 i, j가 옮겼다면?

$$\boldsymbol{i} = \begin{pmatrix} 2 \\ 3 \end{pmatrix}$$

$$\boldsymbol{j} = \begin{pmatrix} -1 \\ 2 \end{pmatrix}$$



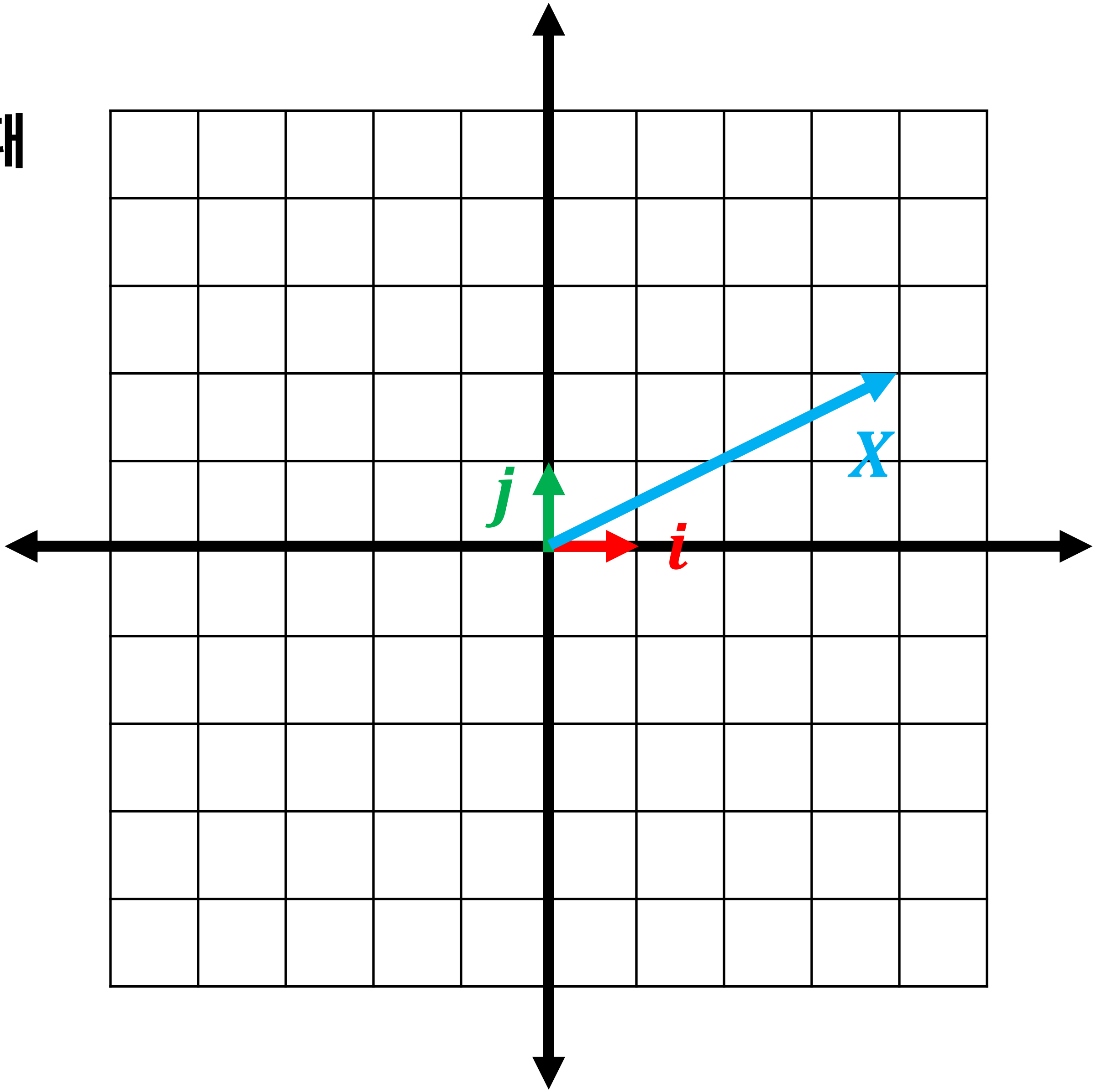
$\begin{pmatrix} x \\ y \end{pmatrix} = x\boldsymbol{i} + y\boldsymbol{j}$ 는 어디로 갈까?

예시

$X = \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 4 \\ 2 \end{pmatrix}$ 일 때

$i = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$

$j = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$



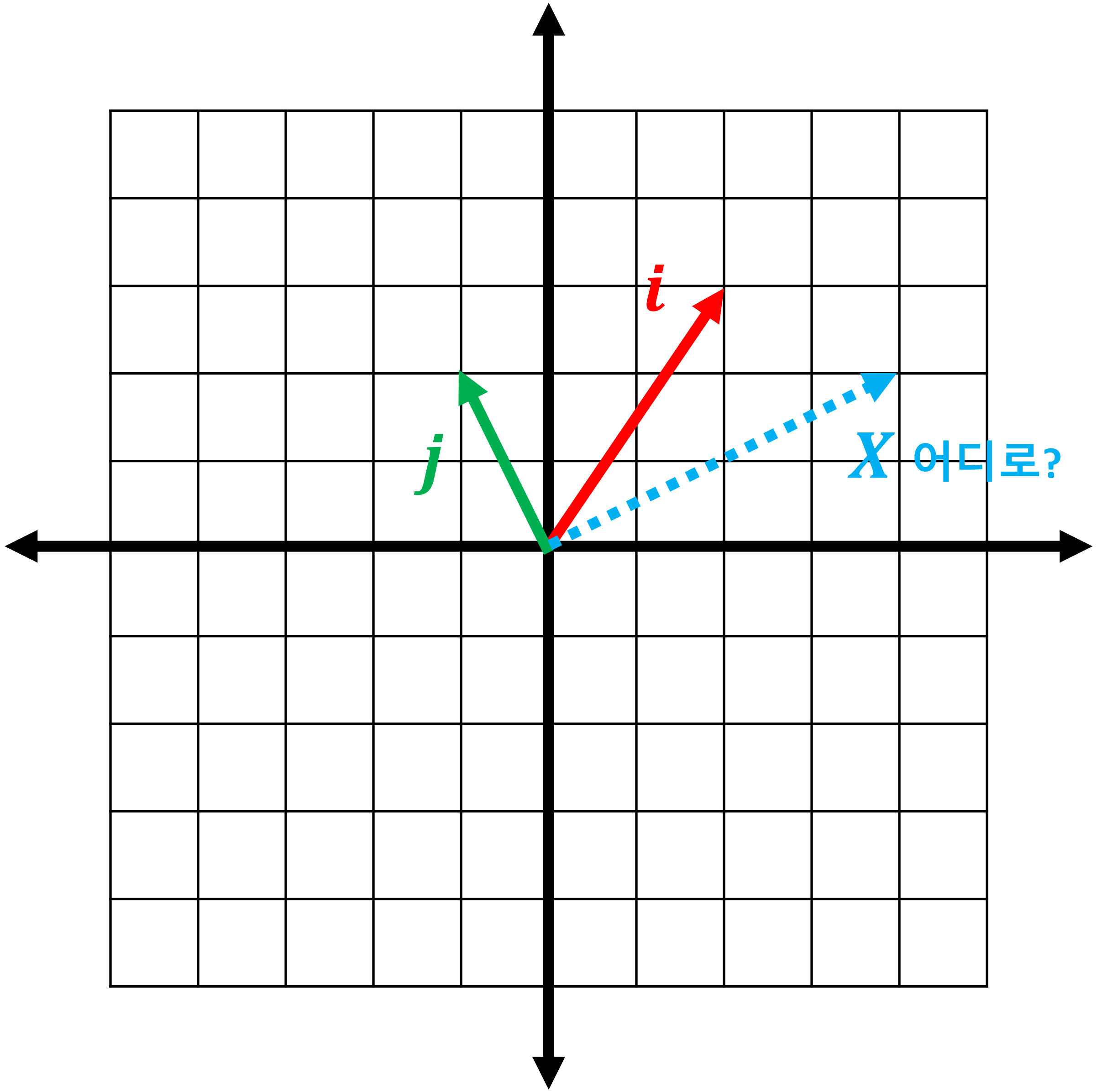
예시

$$i = \begin{pmatrix} 2 \\ 3 \end{pmatrix}$$

$$j = \begin{pmatrix} -1 \\ 2 \end{pmatrix}$$

$$X = \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 4 \\ 2 \end{pmatrix}$$

어디로?



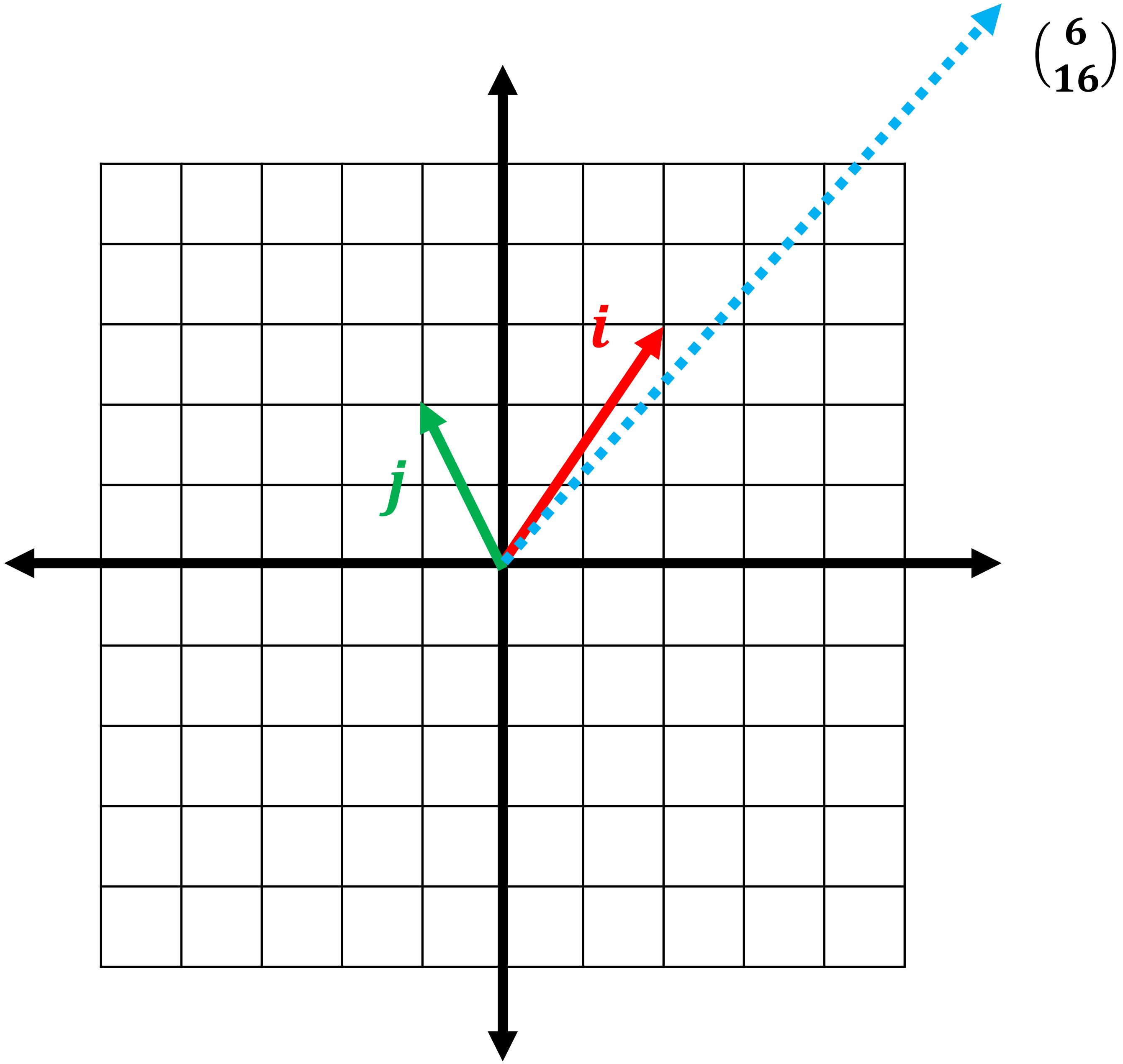
관계식

$$x = \begin{pmatrix} 4 \\ 2 \end{pmatrix} = 4i + 2j$$

$$y = 4\begin{pmatrix} 2 \\ 3 \end{pmatrix} + 2\begin{pmatrix} -1 \\ 2 \end{pmatrix} = \begin{pmatrix} 6 \\ 16 \end{pmatrix}$$

행렬표기

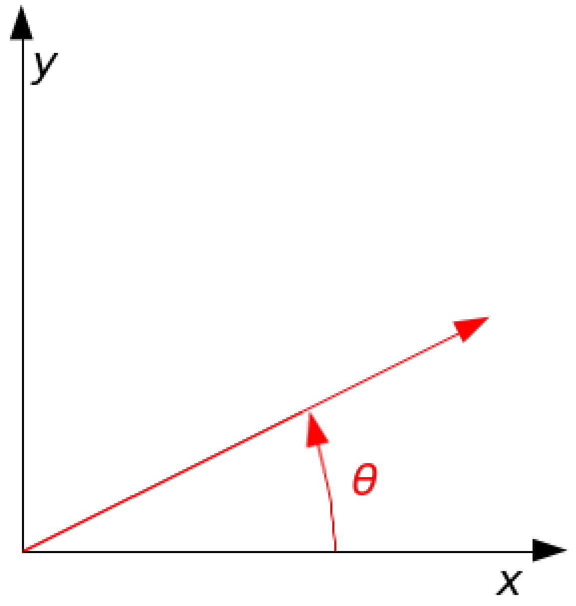
$$y = Ax = \begin{bmatrix} 2 & -1 \\ 3 & 2 \end{bmatrix}\begin{pmatrix} 4 \\ 2 \end{pmatrix} = \begin{pmatrix} 6 \\ 16 \end{pmatrix}$$



Matrix Multiplication

- 회전 변환

$$R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$



$\boldsymbol{y} = \boldsymbol{R}(\boldsymbol{\theta})\boldsymbol{x}$ 에 의해 회전됨

Eigenvalue and Eigenvector

Matrix we are finding the eigenvector/eigenvalue of

eigenvalue

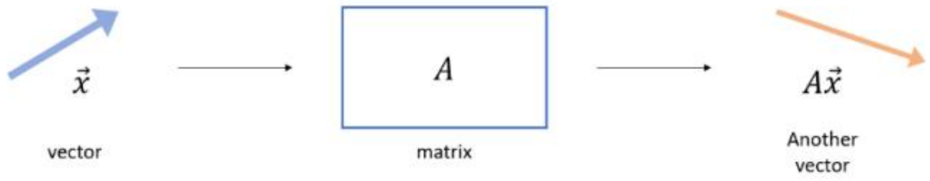
$$Ax = \lambda x$$

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

identity matrix

For matrix A, scalar lambda which does not change the direction of x, is called "eigenvalue" of A

corresponding vector is called "eigenvector" x



→ $\det(A - \lambda I) = 0$의 해가 eigenvalue →λ 구함

→ $Ax = \lambda x$에서 0이 아닌 해가 eigenvector → x 구함

Eigendecomposition

- $A_{n \times n}$: n x n 행렬
- Eigenvalues: $\lambda_1, \lambda_2 \cdots, \lambda_n$
- Eigenvectors: $\boldsymbol{v_1}, \boldsymbol{v_2}, \cdots, \boldsymbol{v_n}$
- $P = [\boldsymbol{v_1} \quad \boldsymbol{v_2} \quad \cdots \quad \boldsymbol{v_n}]$

$$A = PDP^{-1}$$

- $D = \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \lambda_n \end{bmatrix}$

- A가 symmetric matrix라면 P는 orthogonal matrix → 이 경우
  - 이 경우를 spectral decomposition이라고 함

$$A = \sum_{i=1}^{n} \lambda_i \boldsymbol{v_i} \boldsymbol{v_i^T}$$

Singular Value Decomposition (SVD)

$$A = U\Sigma V^T$$

- Eigendecomposition의 단점:
  - 정방행렬일 때만 가능
  - Symmetric해야 P가 orthogonal
- SVD는 아무 행렬이 와도 할 수 있다!
- $A_{m\times n}$: m x n 행렬 → AA^T, A^TA는 항상 symmetric!
- A^TA 의 Eigenvalues: $\lambda_1, \lambda_2 \cdots, \lambda_n$ → Singular values: $\sigma_i = \sqrt{\lambda_i}$
- A^TA 의 Eigenvectors: $\boldsymbol{v_1}, \boldsymbol{v_2}, \cdots, \boldsymbol{v_n}$
- $V_{n\times n} = [\boldsymbol{v_1} \quad \boldsymbol{v_2} \quad \cdots \quad \boldsymbol{v_n}]$ → $U_{m\times \mathrm{m}} = [\boldsymbol{u_1} \quad \boldsymbol{u_2} \quad \cdots \quad \boldsymbol{u_n}] = \left[\frac{\boldsymbol{Av_1}}{\sigma_i} \quad \frac{\boldsymbol{Av_2}}{\sigma_2} \quad \cdots \quad \frac{\boldsymbol{Av_n}}{\sigma_n}\right]$
- $D = \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \lambda_n \end{bmatrix}$ → $\Sigma = \begin{bmatrix} D & 0 \\ 0 & 0 \end{bmatrix}$

3. PCA (Principal Component Analysis)

Algorithm 1. PCA

Covariance and Correlation Matrix

변수1 (X1)	변수2 (X2)	변수3 (X3)	변수4 (X4)
x11	x12	x13	x14
x21	x22	x23	x24
x31	x32	x33	x34
x41	x42	x43	x44

Covariance Matrix

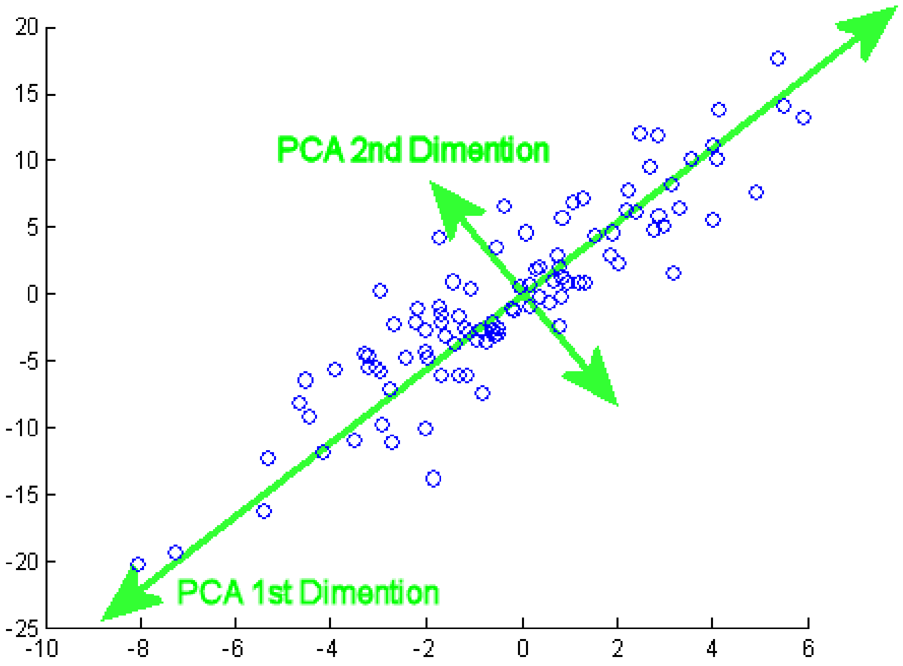
$$\begin{bmatrix} Cov(X_1,X_1) & Cov(X_1,X_2) & Cov(X_1,X_3) & Cov(X_1,X_4) \\ Cov(X_2,X_1) & Cov(X_2,X_2) & Cov(X_2,X_3) & Cov(X_2,X_4) \\ Cov(X_3,X_1) & Cov(X_3,X_2) & Cov(X_3,X_3) & Cov(X_3,X_4) \\ Cov(X_4,X_1) & Cov(X_4,X_2) & Cov(X_4,X_3) & Cov(X_4,X_4) \end{bmatrix}$$

Correlation Matrix

$$\begin{bmatrix} Cor(X_1,X_1) & Cor(X_1,X_2) & Cor(X_1,X_3) & Cor(X_1,X_4) \\ Cor(X_2,X_1) & Cor(X_2,X_2) & Cor(X_2,X_3) & Cor(X_2,X_4) \\ Cor(X_3,X_1) & Cor(X_3,X_2) & Cor(X_3,X_3) & Cor(X_3,X_4) \\ Cor(X_4,X_1) & Cor(X_4,X_2) & Cor(X_4,X_3) & Cor(X_4,X_4) \end{bmatrix}$$

Algorithm 1. PCA

- 데이터 행렬 X 가 n x p 행렬이라고 하자
- Correlation matrix, Covariance matrix 둘 중 하나를 고른다
  - 주로 correlation matrix를 고름 → Σ 라고 하자. (p x p)
- Σ의 eigenvalue와 eigenvector를 구함. → 각각 p개
- $\lambda_1 \geq \lambda_2 \cdots \geq \lambda_p$로 정렬, 각 eigenvalue에 대응하는 eigenvector를 $v_1, v_2, \cdots v_p$라고 하자.
- $X = [x_1 \quad x_2 \quad \cdots \quad x_p]$ 데이터를 변수들의 벡터로 생각할 때:
- PC1 = $v_1^T X$, PC2 = $v_2^T X$ …



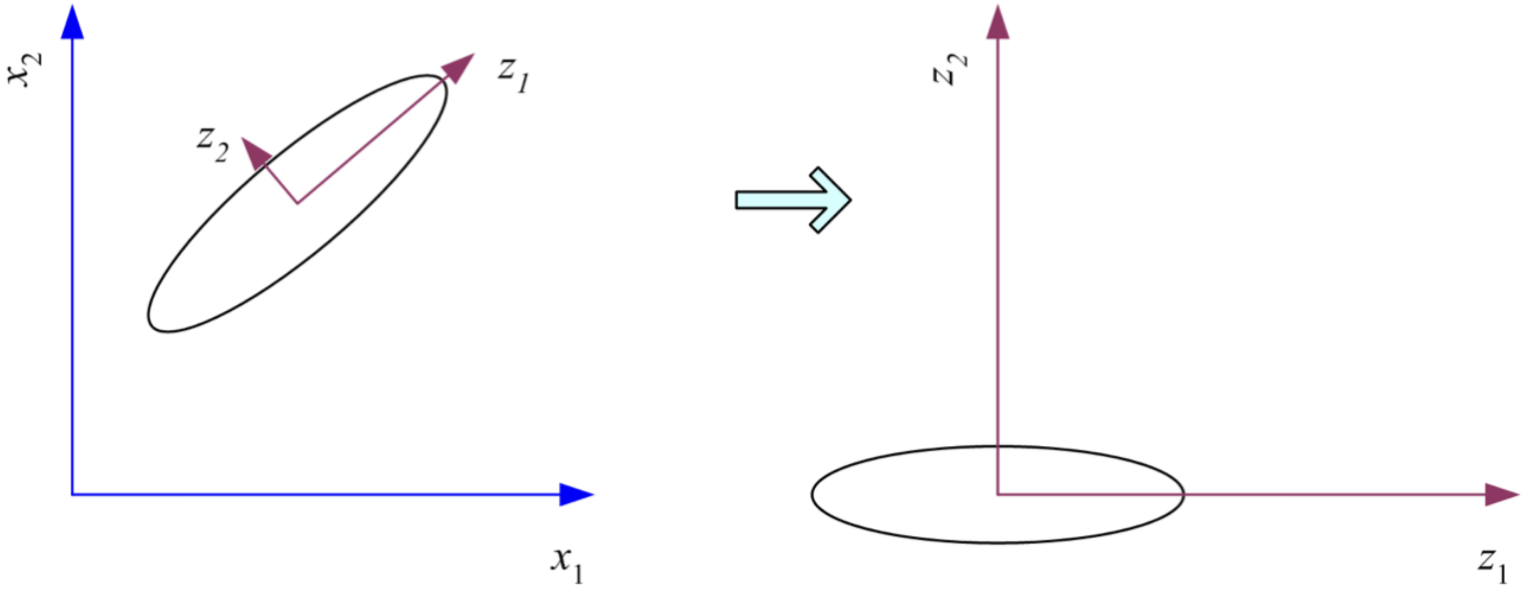
What PCA does

By centering, x-m → x’

Z = x’W

Where the columns of W are the eigenvector of Covariance matrix.

Centers the data at the origin and rotates the axes



How Many dimension K?

W : eigenvector of Σ (covariance matrix of x)
α : eigenvalue of Σ (covariance matrix of x)

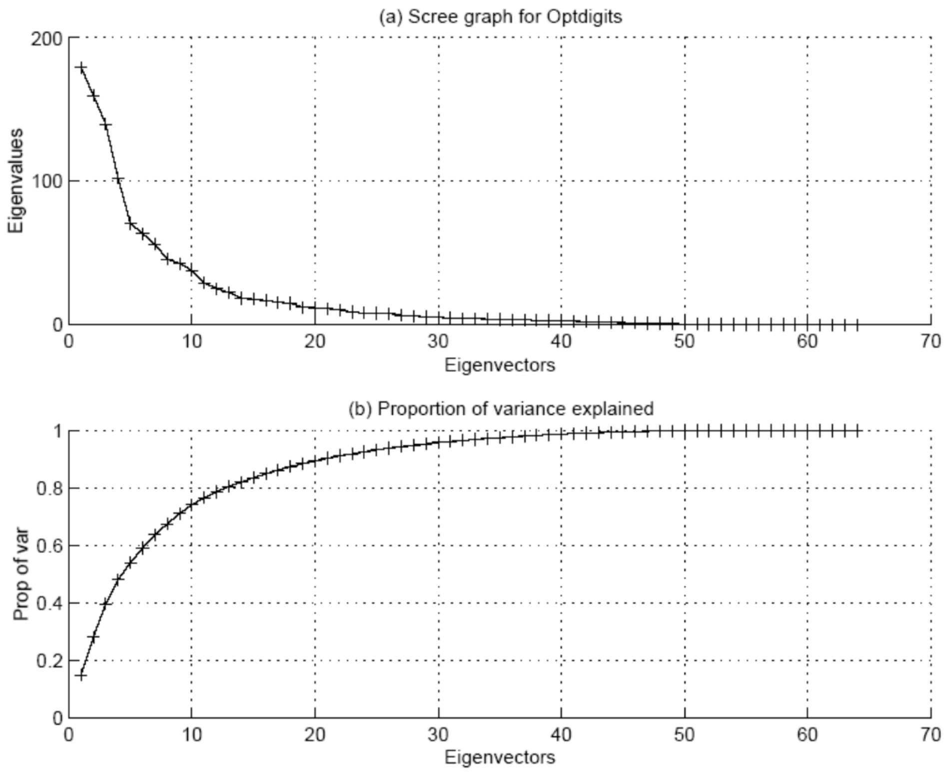
→ Σ : dxd matrix → we can find w & α for all d vectors

- Proportion of Variance (PoV)

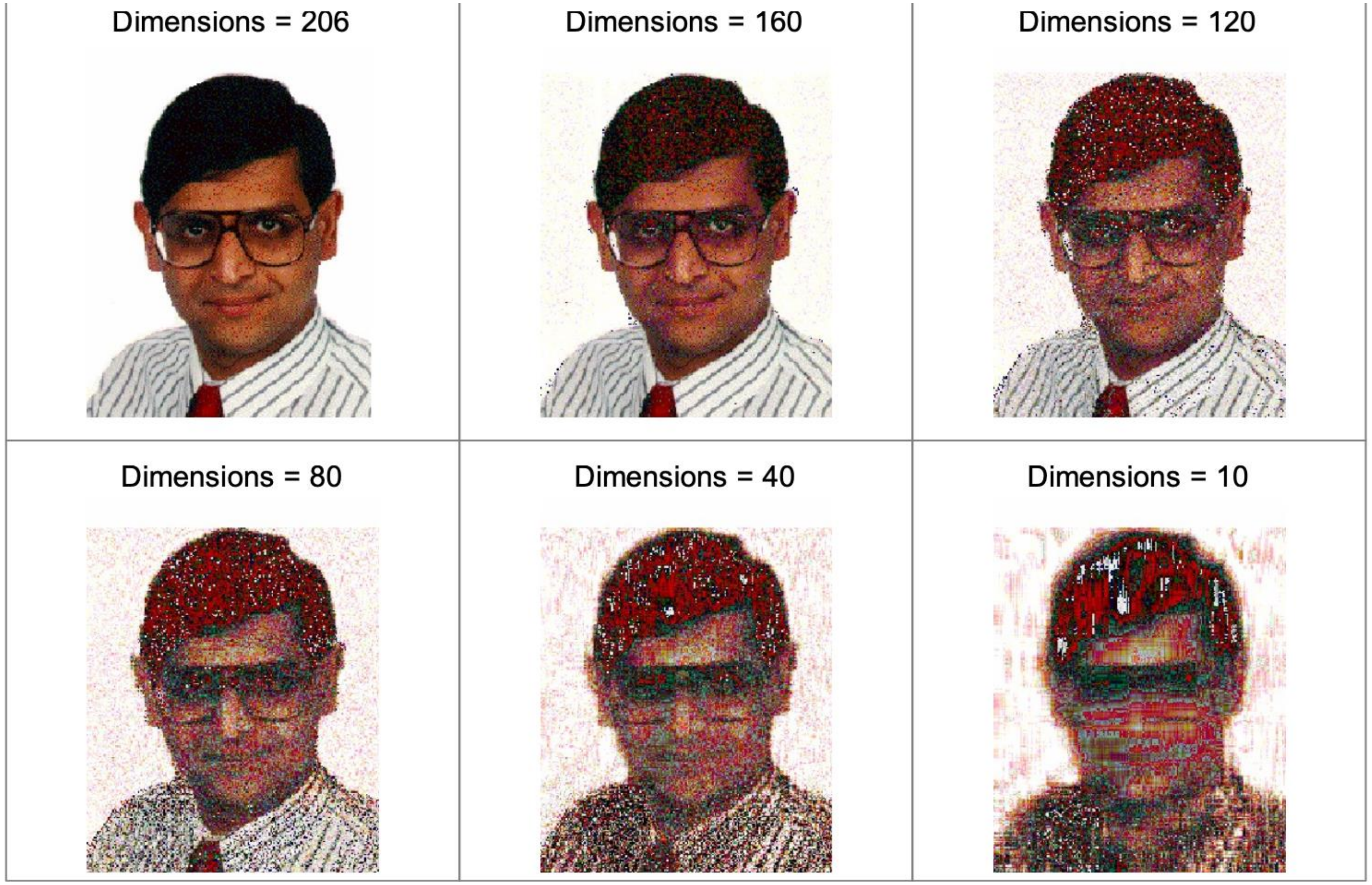
$$\frac{\alpha_1 + \alpha_2 + \cdots + \alpha_k}{\alpha_1 + \alpha_2 + \cdots + \alpha_k + \cdots + \alpha_d}$$

$$\alpha_1 \geq \alpha_2 \geq \cdots \geq \alpha_k \cdots \geq \alpha_d$$

- Typically, stop at PoV >0.9
- Scree graph plots of PoV vs k, stop at ellbow



PCA


Dimensions = 206
Dimensions = 160
Dimensions = 120
Dimensions = 80
Dimensions = 40
Dimensions = 10

4. EFA (Explanatory Factor Analysis)

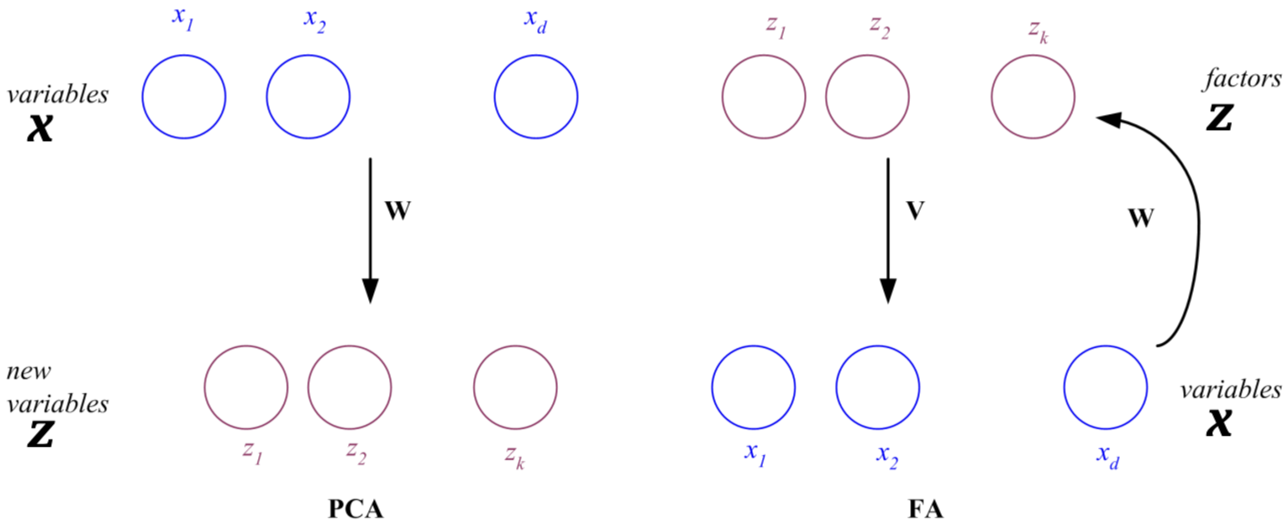
Algorithm 2. FA (Factor Analysis)

- Find a small number of factors z, which when combined generate x:

$$x - m = Vz + \varepsilon$$

- z : $latent\ vector, \varepsilon : noise\ sources, V : factor\ loadings$

$factor$ analysis ~ $\boldsymbol{Matrix\ Factorization}$

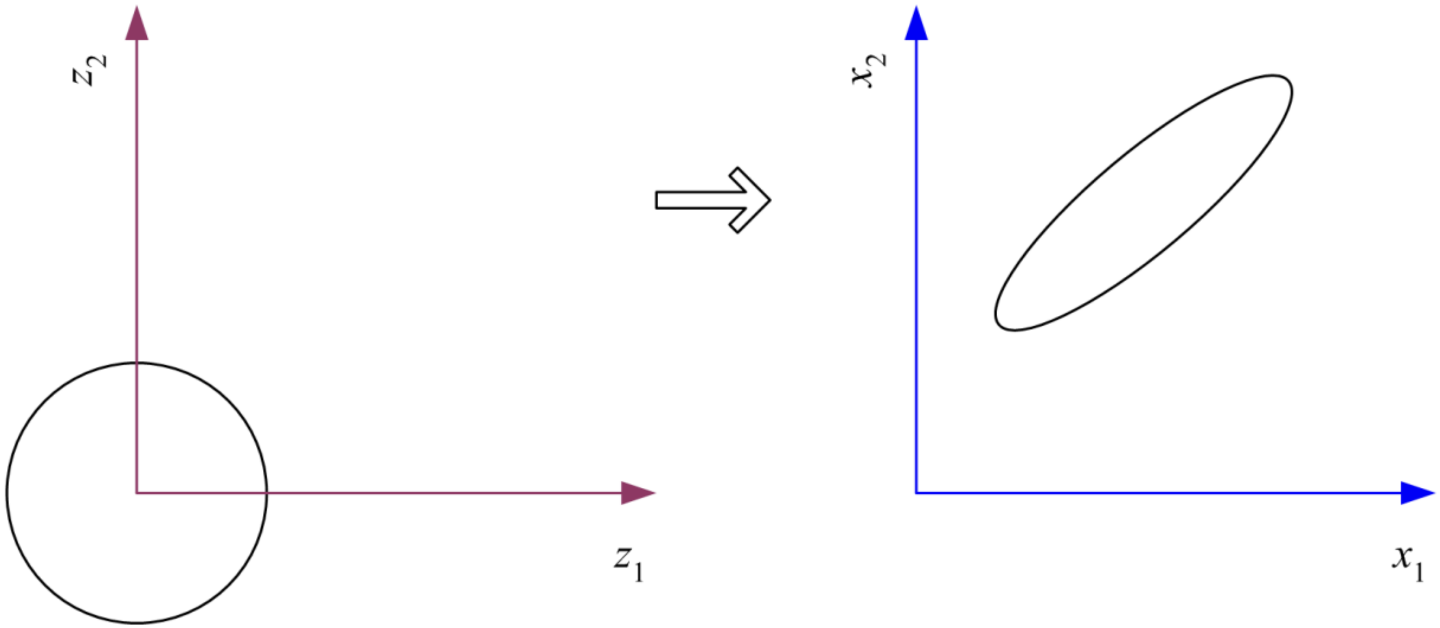


→ Factor (=latent variable): 측정되지 않는 변수 (unobservable), ex 행복도

→ Manifest variable: 측정 가능한 변수, ex 설문조사 결과

→ Manifest variable을 factor의 선형결합으로 나타내는 것이 목표

FA



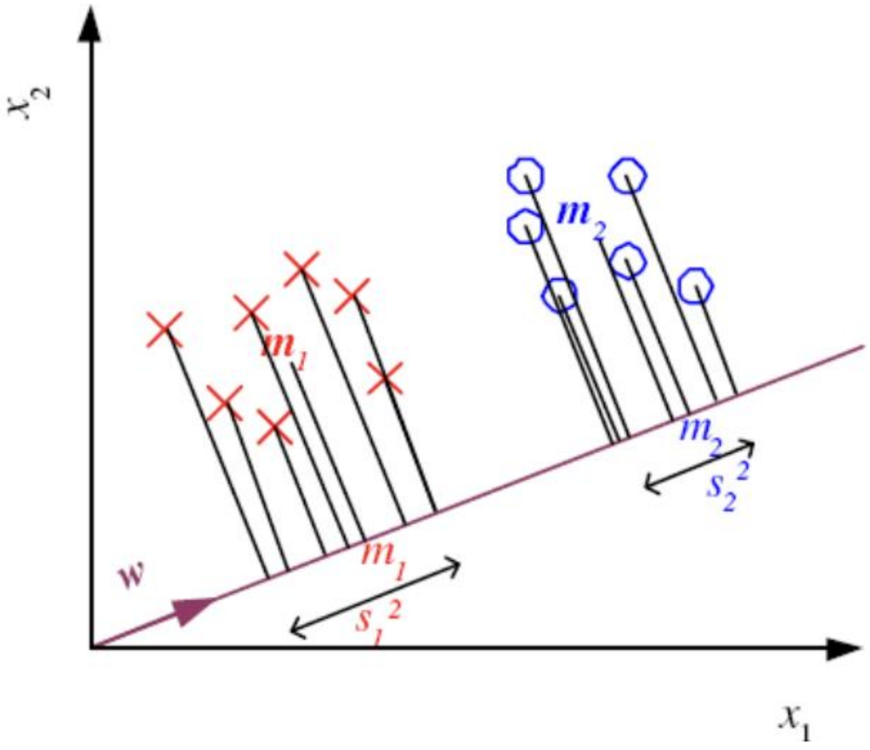
5. LDA, QDA (Discriminant Analysis)

Algorithm 3. LDA (Linear Discriminant Analysis)

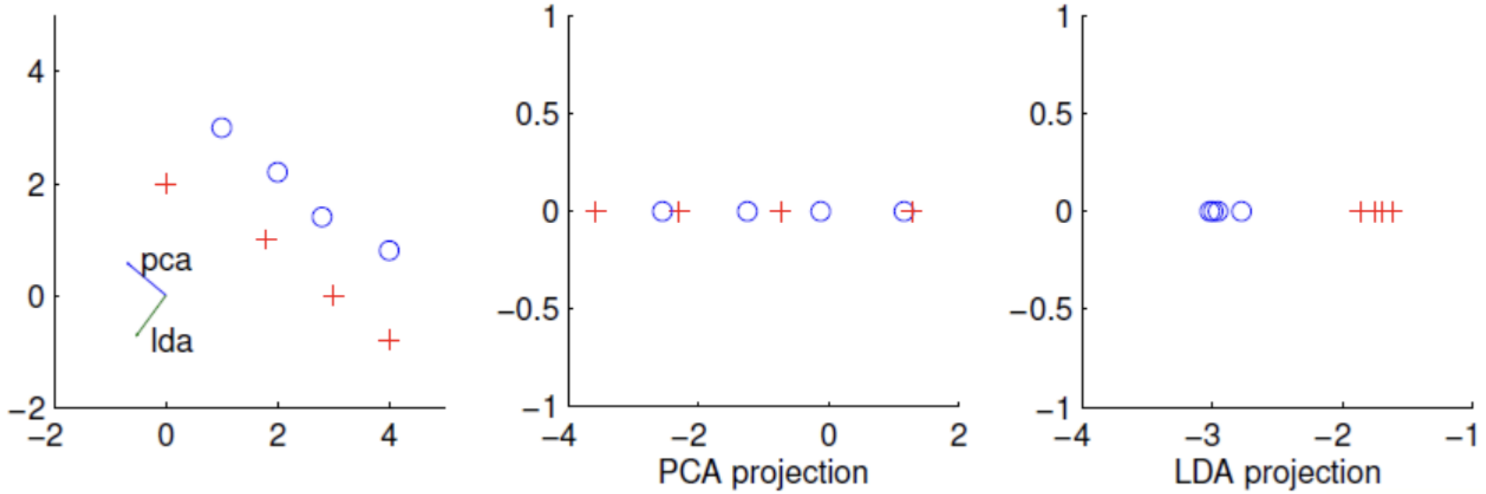
- Find a low-dimensional space such that when x is projected, classes are well-separated
- Find W that maximizes

$$J(w) = \frac{(m_1 - m_2)^2}{s_1^2 + s_2^2}$$

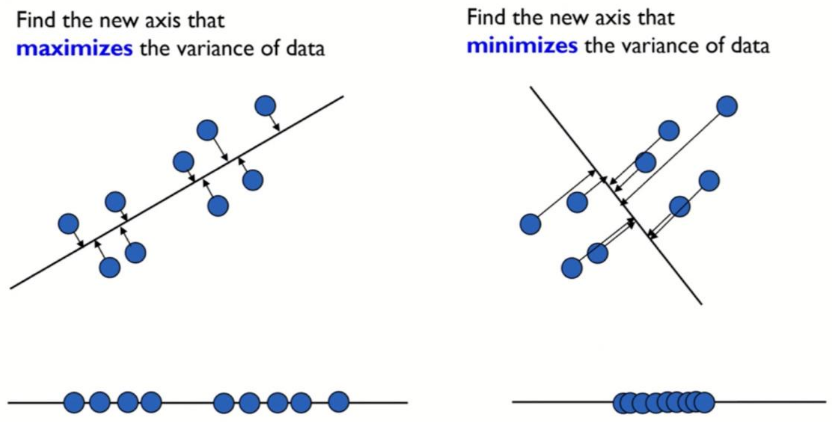
→ 정규분포를 따르는 두 개의 집단으로 나누기
→ 두 집단의 공분산 행렬이 같다고 가정
→ 그룹 수를 미리 정한다.



PCA vs LDA

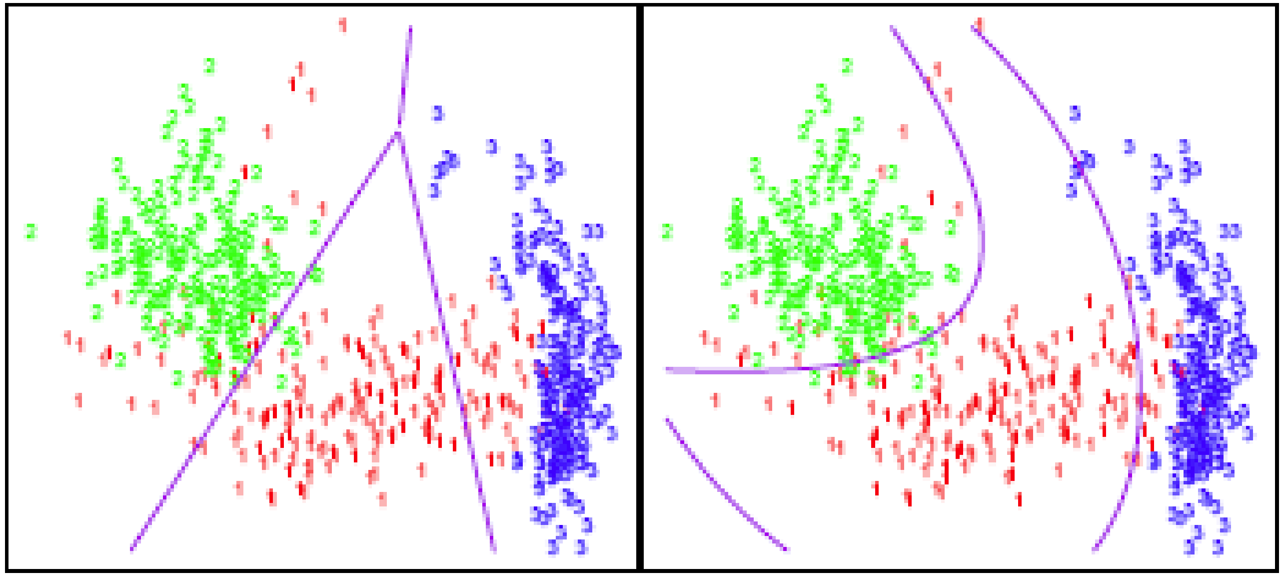


pca
lda
PCA projection
LDA projection



Find the new axis that **maximizes** the variance of data
Find the new axis that **minimizes** the variance of data

QDA (Quadratic Discriminant Analysis)



LDA **QDA**

→ 두 집단의 공분산 행렬이 같지 않을 때 이용!
→ 직선으로 나누는 LDA와 달리 2차식으로 나누기

Other Algorithms

- **tSNE, UMAP** →생명과학 분야에 많이 쓰임
- Canonical Correlation Analysis (CCA)
- Isometric feature mapping(Isomap)
- **multi-dimensional scaling (MDS)**
- Locally linear Embedding(LLE)
- **Sufficient Dimension Reduction (SDR)**

- **kPCA(kernel PCA)**
- **FPCA (Functional PCA)**

수고하셨습니다!

해당 세션자료는 KUBIG Github에서 보실 수 있습니다!