11기 정규과정 4주차

ToBig's 10기 이민주

PCA & t-SNE

Contents

Unit 01 | Intro : Dimensionality Reduction

Dimensionality Reduction ??

차원 축소

Unit 01 | Intro : Dimensionality Reduction

Curse of Dimensionality ??

차원의 저주

Unit 01 | Intro : Dimensionality Reduction

- **Curse of Dimensionality 차원의 저주**

feature 수

: 변수가 늘어나고 차원*이 커지면서 발생하는 문제

* Dimension 차원 : 축(변수)의 개수

▪ Curse of Dimensionality 차원의 저주

: 변수가 늘어나고 차원이 커지면서 발생하는 문제

- 필요한 데이터의 수 기하급수적으로 증가

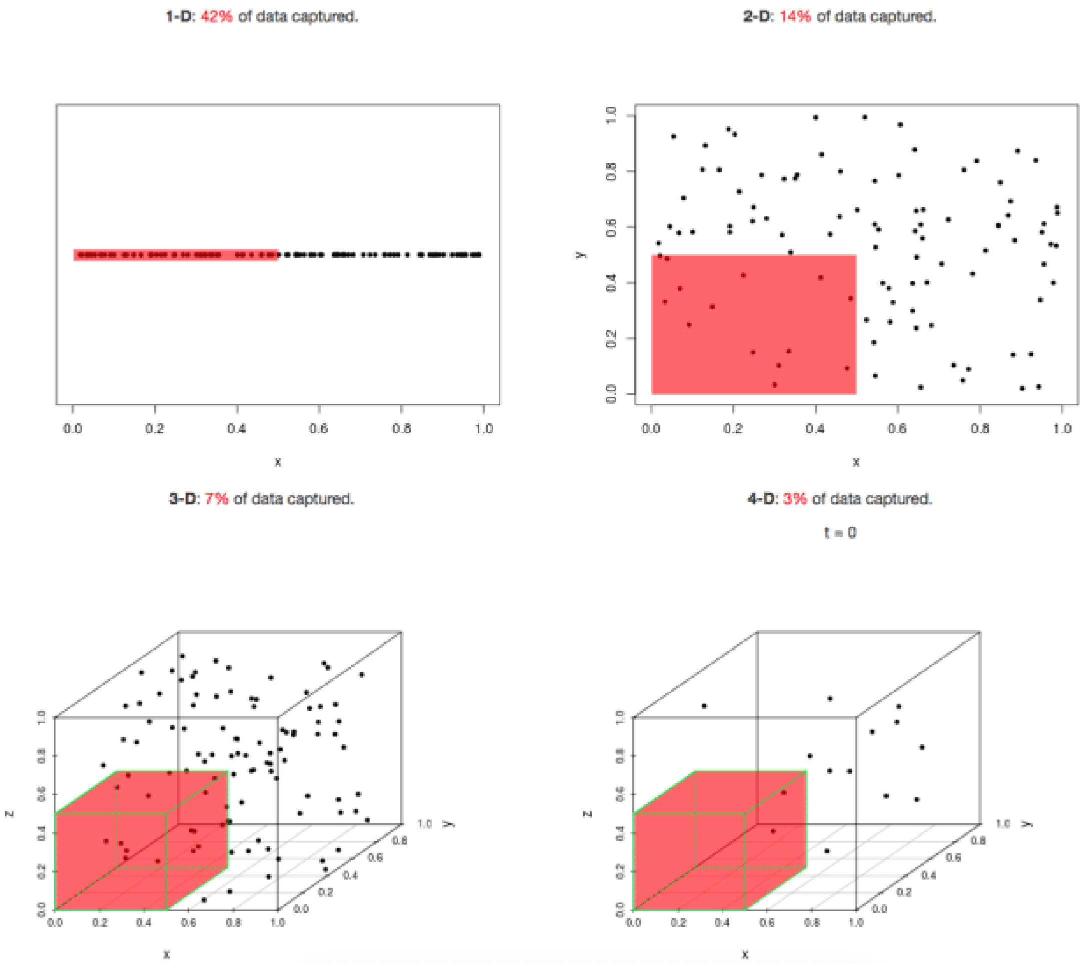
Unit 01 | Intro : Dimensionality Reduction

▪ Curse of Dimensionality 차원의 저주

: 변수가 늘어나고 차원이 커지면서 발생하는 문제

- 필요한 데이터의 수 기하급수적으로 증가

- 데이터의 크기가 정해진 경우,
  차원공간의 희소성이 증가 & 데이터의 밀도 감소



Unit 01 | Intro : Dimensionality Reduction

▪ Curse of Dimensionality 차원의 저주

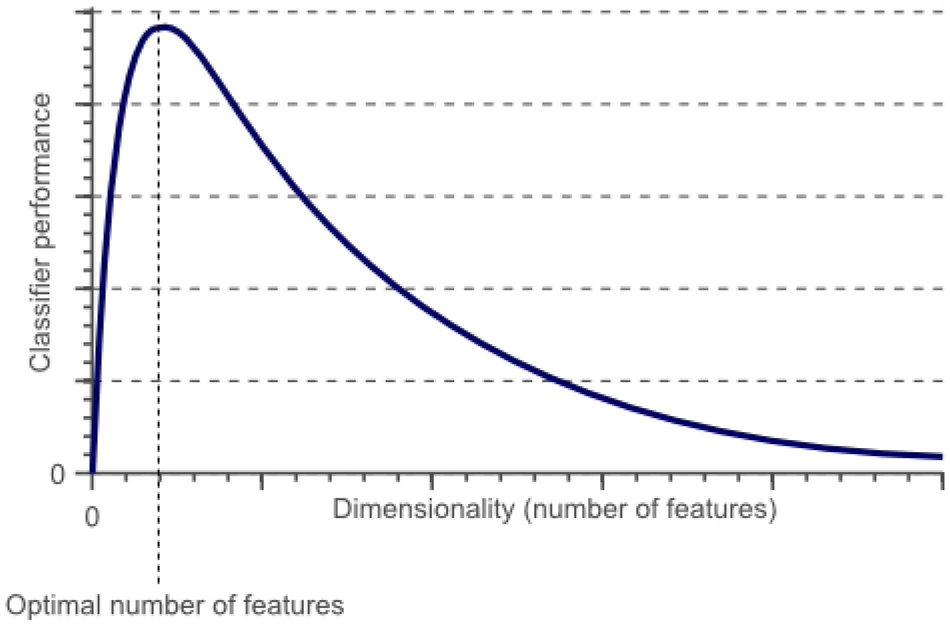
: 변수가 늘어나고 차원이 커지면서 발생하는 문제

- 필요한 데이터의 수 기하급수적으로 증가

- 데이터의 크기가 정해진 경우,
  차원공간의 희소성이 증가 & 데이터의 밀도 감소

- 오버피팅 확률 증가 & 성능 감소

- 연산량 급증



Dimensionality Reduction !!

차원 축소

▪ Dimensionality Reduction 차원축소

: 데이터의 특징을 잘 나타내는 변수만 사용해 차원 수를 줄이는 방법

➔ 차원의 저주를 해결 & 시각화 용이 & 데이터 특성 파악 용이 & 컴퓨팅파워 절약

1. Feature Selection 변수 선택

: 변수들 중 중요한 변수만 선택하고 나머지 변수는 버림

A, B, C, D, E ➔ A, D

정보손실이 많음

2. Feature Extraction 변수 추출

: 모든 변수를 조합해서 데이터를 잘 표현할 수 있는 새로운 변수 추출

A, B, C, D, E ➔ 가, 나

모든 정보 사용

가 : f(A,B,C,D,E)
나 : g(A,B,C,D,E)

Unit 01 | Intro : Dimensionality Reduction

▪ Dimensionality Reduction 차원축소

: 데이터의 특징을 잘 나타내는 변수만 사용해 차원 수를 줄이는 방법

➔ 차원의 저주를 해결 & 시각화 용이 & 데이터 특성 파악 용이 & 컴퓨팅파워 절약

1. Feature Selection 변수 선택

: 변수들 중 중요한 변수만 선택하고
나머지 변수는 버림

A, B, C, D, E ➔ A, D

2. Feature Extraction 변수 추출

: 모든 변수를 조합해서 데이터를
잘 표현할 수 있는 새로운 변수 추출

PCA

A, B, C, D, E ➔ 가, 나

Unit 02 | Eigen Decomposition

▪ Eigenvector 고유벡터 & Eigenvalue 고유값

정방대칭행렬 A에 대해,

- Eigenvector 고유벡터 : K　　$AK=\lambda K$

  : 선형변환 A를 했을 때, 방향이 변하지 않는 벡터

- Eigenvalue 고유값 : λ　　$\det(A-\lambda I)=0$

  : 벡터의 크기가 변하는 정도

A : n * n행렬

K : k1,k2,⋯,kn

n개의 n*1 행렬(벡터)

➔ 단위 길이(길이 = 1) & 서로 직교

λ : λ1, λ2,⋯, λn

n개의 스칼라값

- Eigenvector 고유벡터 & Eigenvalue 고유값

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

$n=2$

2×2 정방행렬 $A=\begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$

$\det|A-\lambda I| = \det\begin{vmatrix} 2-\lambda & 1 \\ 1 & 2-\lambda \end{vmatrix} = (2-\lambda)^2-1$ ad-bc

$= \lambda^2-4\lambda+3$

$= (\lambda-1)(\lambda-3)$

$\therefore \lambda_1=1, \ \lambda_2=3$

$\lambda=1$

$A\begin{pmatrix} k_1 \\ k_2 \end{pmatrix} = \lambda\begin{pmatrix} k_1 \\ k_2 \end{pmatrix}$

$\begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}\begin{pmatrix} k_1 \\ k_2 \end{pmatrix} = \begin{pmatrix} 2k_1+k_2 \\ k_1+2k_2 \end{pmatrix} = \begin{pmatrix} k_1 \\ k_2 \end{pmatrix}$

$\therefore k_1=-k_2$

$\begin{pmatrix} k_1 \\ k_2 \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$ * $1/\sqrt{2}$

$\lambda=3$

$A\begin{pmatrix} k_1 \\ k_2 \end{pmatrix} = \lambda\begin{pmatrix} k_1 \\ k_2 \end{pmatrix}$

$\begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}\begin{pmatrix} k_1 \\ k_2 \end{pmatrix} = \begin{pmatrix} 2k_1+k_2 \\ k_1+2k_2 \end{pmatrix} = \begin{pmatrix} 3k_1 \\ 3k_2 \end{pmatrix}$

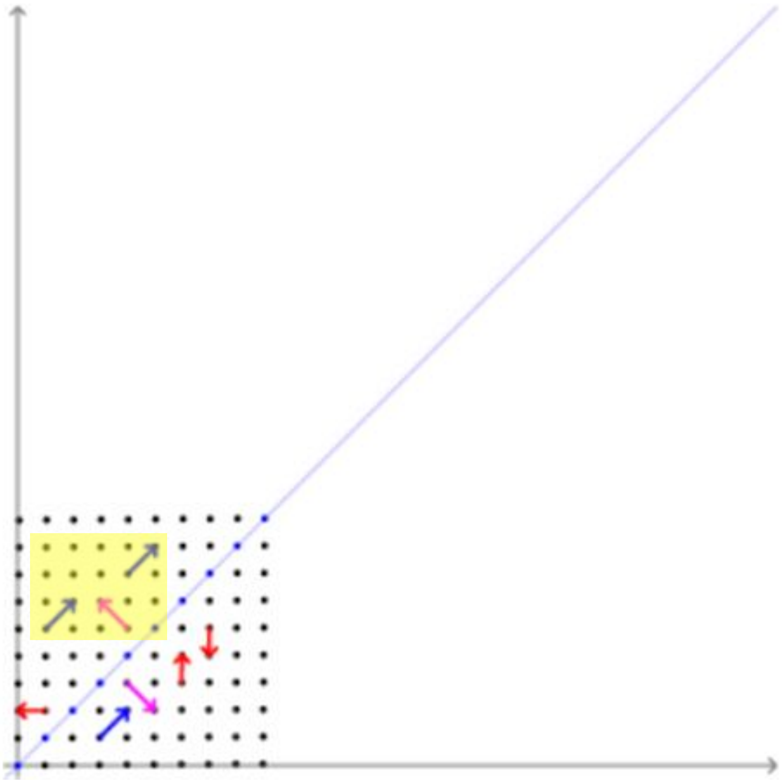
$\therefore k_1=k_2$

$\begin{pmatrix} k_1 \\ k_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ * $1/\sqrt{2}$

Unit 02 | Eigen Decomposition

▪ Eigenvector 고유벡터 & Eigenvalue 고유값

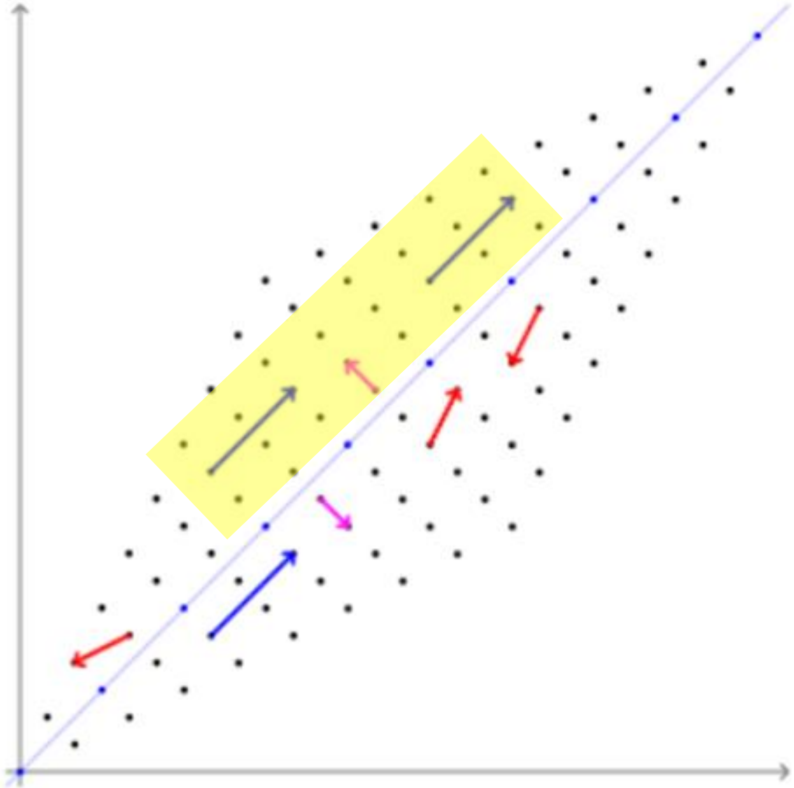
- Eigenvector 선형변환 A를 했을 때, 방향이 변하지 않는 벡터
- Eigenvalue 벡터의 크기가 변하는 정도



$A = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$ 변환

$\lambda=1 \Rightarrow \begin{pmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{pmatrix}$

$\lambda=3 \Rightarrow \begin{pmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{pmatrix}$



Unit 02 | Eigen Decomposition

▪ Spectral Decomposition 스펙트럼 분해

n차 대칭행렬 A

$$A = \lambda 1 * k1 * k1' + \lambda 2 * k2 * k2' + \cdots + \lambda n * kn * kn'$$

$$= \Sigma\, \lambda i * ki * ki'$$

$$= [k1\ k2 \cdots kn] \begin{bmatrix} \lambda 1 & \cdots & 0 \\ 0 & \lambda 2 & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda n \end{bmatrix} \begin{bmatrix} k1' \\ k2' \\ \vdots \\ kn' \end{bmatrix}$$

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

P : 고유벡터의 열로 구성된 행렬
Λ : 고유값을 대각원소로 하는 대각행렬

n차 양정치행렬* A의 성질

$$**tr(A) = tr(P\Lambda P') = tr(P'P\Lambda)$$

$$= tr(\Lambda) = \Sigma\, \lambda i$$

*양의 정부호 행렬 == 양정치행렬

**tr : trace, 행렬의 대각합

***공분산행렬은 언제나 양정치행렬

Unit 02 | Eigen Decomposition

▪ Spectral Decomposition 스펙트럼 분해

n차 대칭행렬 A

$$A = \lambda 1{*}k1{*}k1' + \lambda 2{*}k2{*}k2' + \cdots + \lambda n{*}kn{*}kn'$$

$$= \Sigma\ \lambda i{*}ki{*}ki'$$

$$= [k1\ k2 \cdots kn]\begin{bmatrix} \lambda 1 & \cdots & 0 \\ 0 & \lambda 2 & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda n \end{bmatrix}\begin{bmatrix} k1' \\ k2' \\ \vdots \\ kn' \end{bmatrix}$$

$= P\Lambda P$ (단, PP' = P'P = I)

코드로 !

P : 고유벡터의 열로 구성된 행렬
Λ : 고유값을 대각원소로 하는 대각행렬

n차 양정치행렬* A의 성질

$$**tr(A) = tr(P\Lambda P') = tr(P'P\Lambda)$$

$$= tr(\Lambda) = \Sigma\ \lambda i$$

*양의 정부호 행렬 == 양정치행렬
**tr : trace, 행렬의 대각합
***공분산행렬은 언제나 양정치행렬

Unit 03 | PCA : Principal Component Analysis

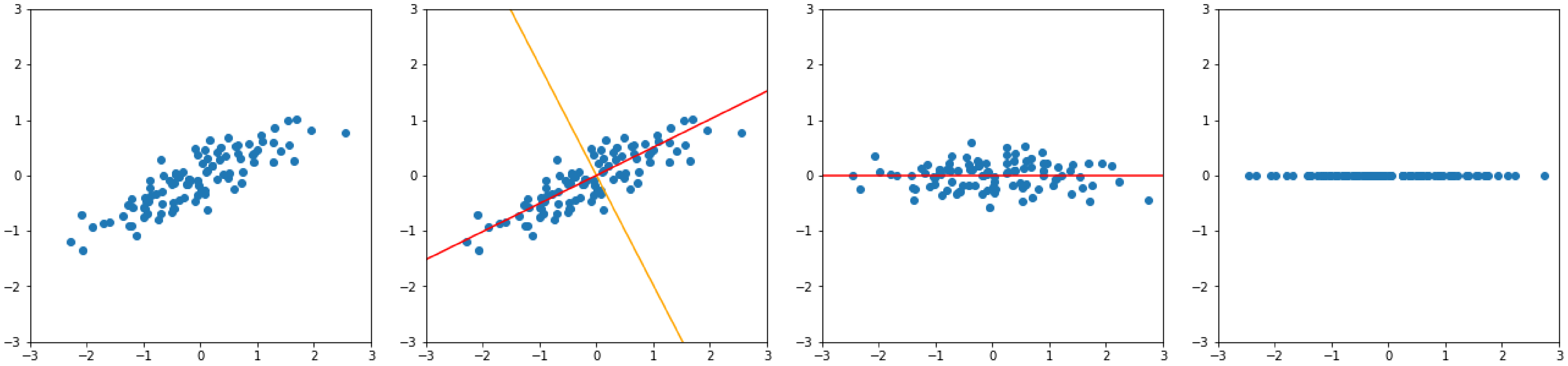
- ## Principal Component Analysis 주성분분석

: 변수들의 전체 분산을 최대한 설명하는 소수의 주성분을 통해 분석하는 방법

➔ 데이터의 구조 최대한 유지

➔ 서로 상관관계가 있는 변수들 사이의 복잡한 구조를 좀 더 간편하게 변환

➔ 기하학적 측면, 좌표축을 회전시켜 얻어진 새로운 좌표축을 선택

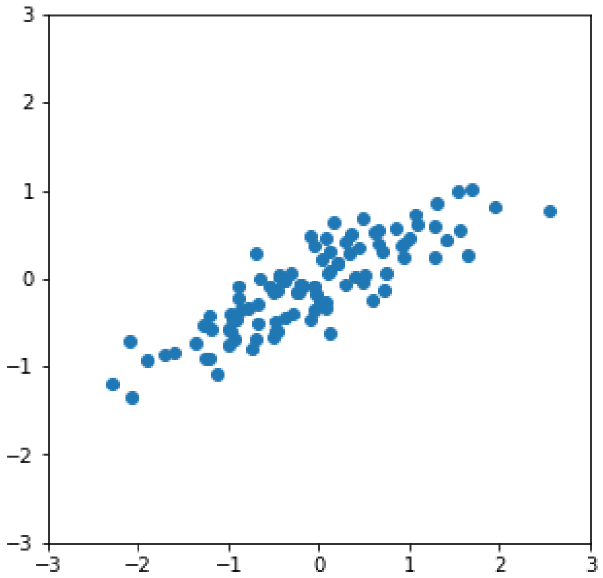


Unit 03 | PCA : Principal Component Analysis

▪ Principal Component Analysis 주성분분석

공분산행렬

$$\Sigma = \begin{pmatrix} \mathbf{0.879} & \mathbf{0.415} \\ \mathbf{0.415} & \mathbf{0.268} \end{pmatrix}$$

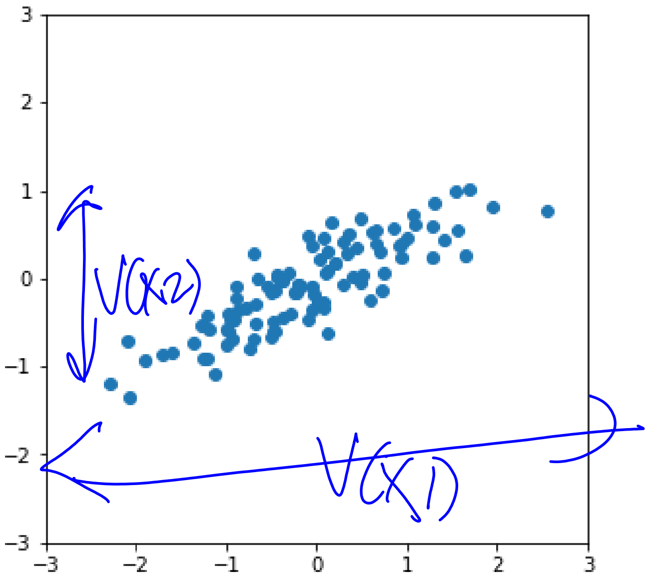


- Principal Component Analysis 주성분분석

공분산행렬

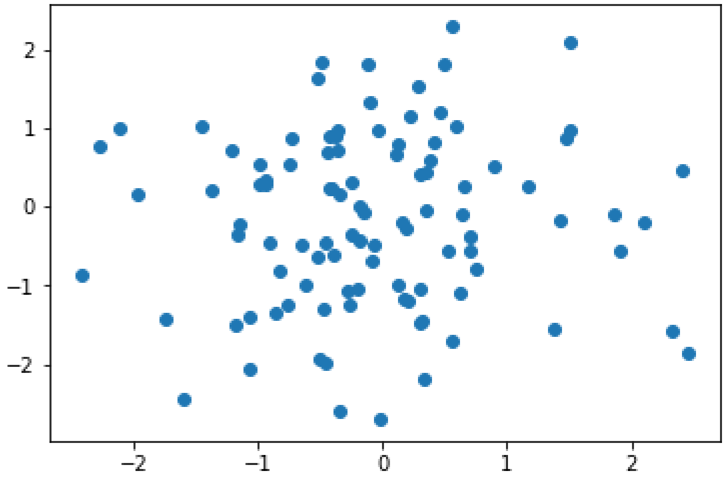
$$\Sigma = \begin{pmatrix} 0.879 & 0.415 \\ 0.415 & 0.268 \end{pmatrix}$$

$$\begin{matrix} V(X_1) & COV(X_1, X_2) \\ COV(X_1, X_2) & V(X_2) \end{matrix}$$



곱해주면됨

공분산행렬 Σ 변환



0.879 0.268

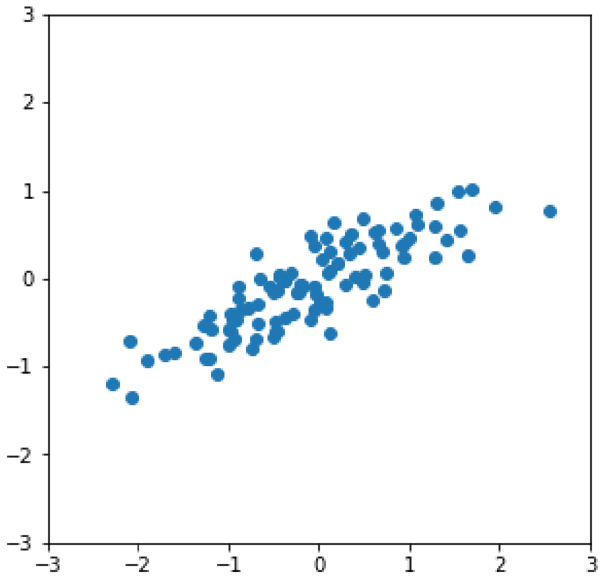
$V(X_1) > V(X_2)$

Unit 03 | PCA : Principal Component Analysis

▪ Principal Component Analysis 주성분분석

공분산행렬

$$\Sigma = \begin{pmatrix} 0.879 & 0.415 \\ 0.415 & 0.268 \end{pmatrix}$$



공분산행렬 스펙트럼 분해

$$\Sigma = \begin{pmatrix} 0.879 & 0.415 \\ 0.415 & 0.268 \end{pmatrix}$$

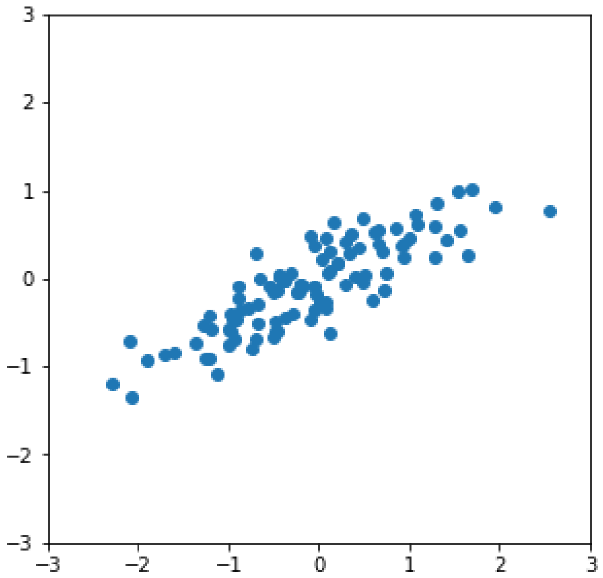
$$= P\Lambda P'$$

$$= \begin{pmatrix} 0.8923 & -0.4514 \\ 0.4513 & 0.8923 \end{pmatrix} \begin{pmatrix} 1.089 & 0 \\ 0 & 0.0058 \end{pmatrix} \begin{pmatrix} 0.8923 & 0.4513 \\ -0.4514 & 0.8923 \end{pmatrix}$$

▪ Principal Component Analysis 주성분분석

공분산행렬

$$\Sigma = \begin{pmatrix} 0.879 & 0.415 \\ 0.415 & 0.268 \end{pmatrix}$$



공분산행렬 스펙트럼 분해

$$\Sigma = \begin{pmatrix} 0.879 & 0.415 \\ 0.415 & 0.268 \end{pmatrix}$$

$$= P\Lambda P'$$

$$= \left(\underset{k1}{\boxed{\begin{matrix} 0.8923 \\ 0.4513 \end{matrix}}} \quad \underset{k2}{\boxed{\begin{matrix} -0.4514 \\ 0.8923 \end{matrix}}} \right) \begin{pmatrix} \overset{\lambda 1}{1.089} & 0 \\ 0 & \underset{\lambda 2}{0.0058} \end{pmatrix} \begin{pmatrix} 0.8923 & 0.4513 \\ -0.4514 & 0.8923 \end{pmatrix}$$

Unit 03 | PCA : Principal Component Analysis

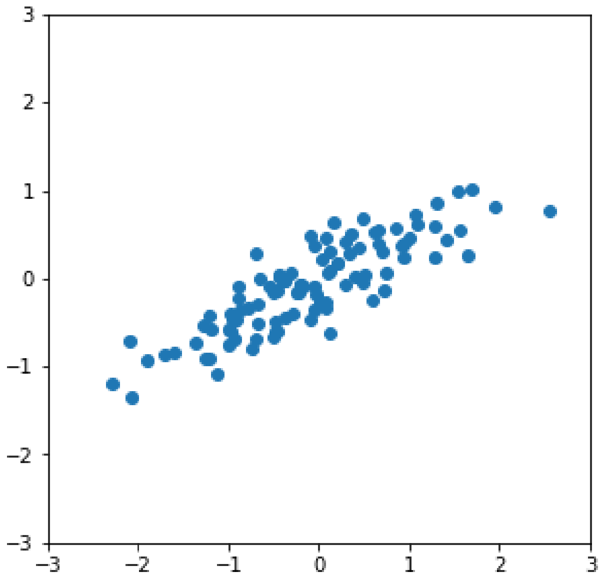
- Principal Component Analysis 주성분분석

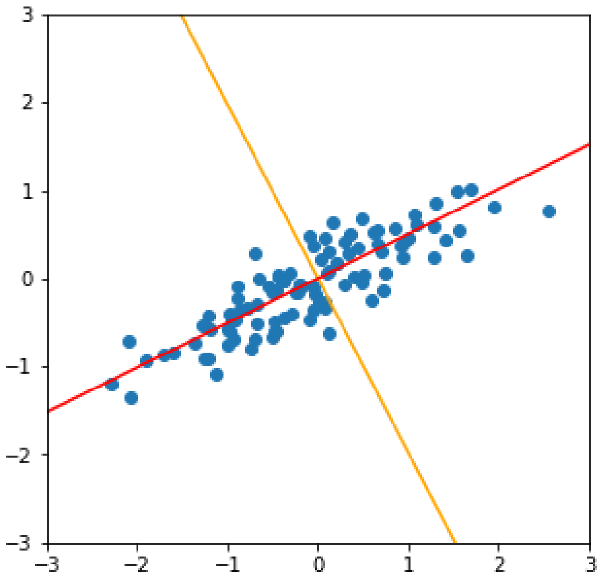
공분산행렬

$$\Sigma = \begin{pmatrix} 0.879 & 0.415 \\ 0.415 & 0.268 \end{pmatrix}$$

$$\lambda 1 = 1.089 \ \& \ k1 = \begin{pmatrix} 0.8923 \\ 0.4513 \end{pmatrix}$$

$$\lambda 2 = 0.058 \ \& \ k2 = \begin{pmatrix} -0.4514 \\ 0.8923 \end{pmatrix}$$





Unit 03 | PCA : Principal Component Analysis

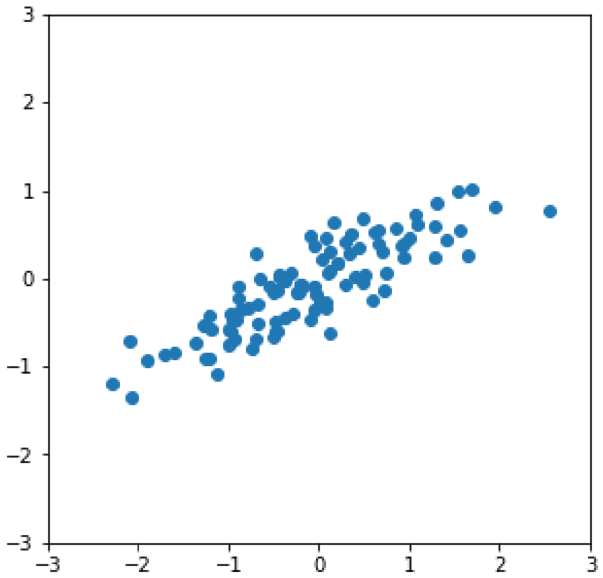
▪ Principal Component Analysis 주성분분석

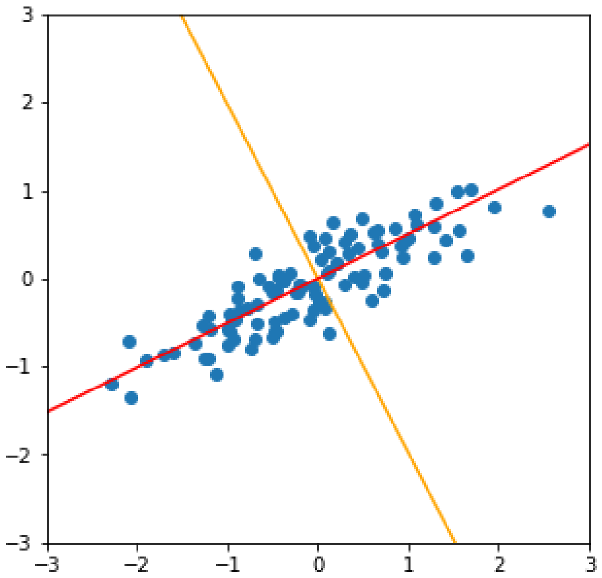
공분산행렬

$$\Sigma = \begin{pmatrix} 0.879 & 0.415 \\ 0.415 & 0.268 \end{pmatrix}$$

$$\lambda 1 = 1.089 \ \& \ k1 = \begin{pmatrix} 0.8923 \\ 0.4513 \end{pmatrix}$$

$$\lambda 2 = 0.058 \ \& \ k2 = \begin{pmatrix} -0.4514 \\ 0.8923 \end{pmatrix}$$





고유벡터 : 데이터가 분산된 방향

고유값 : 전체분산에 대한
주성분 PCi의 설명비율

total variance = tr(Σ) = tr(Λ) = λ1 + … + λn

var(PCi) = λi

➔ 고유값이 큰 것부터 사용

➔ 처음 k개 주성분의 설명비율
= (λ1+⋯+ λk) / total variance

Unit 03 | PCA : Principal Component Analysis

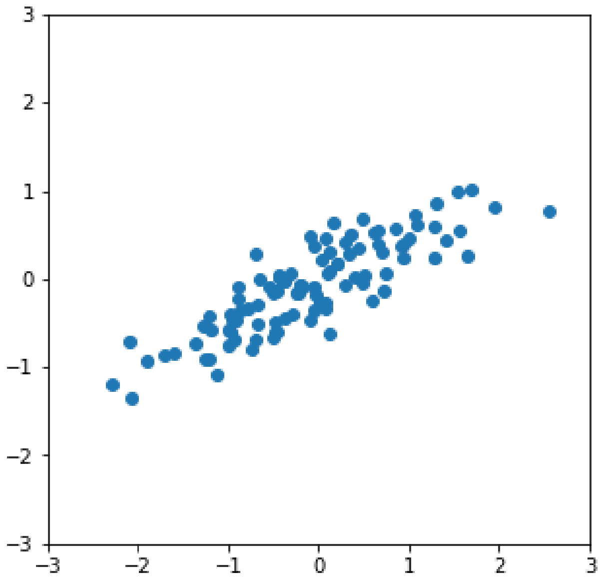
▪ Principal Component Analysis 주성분분석

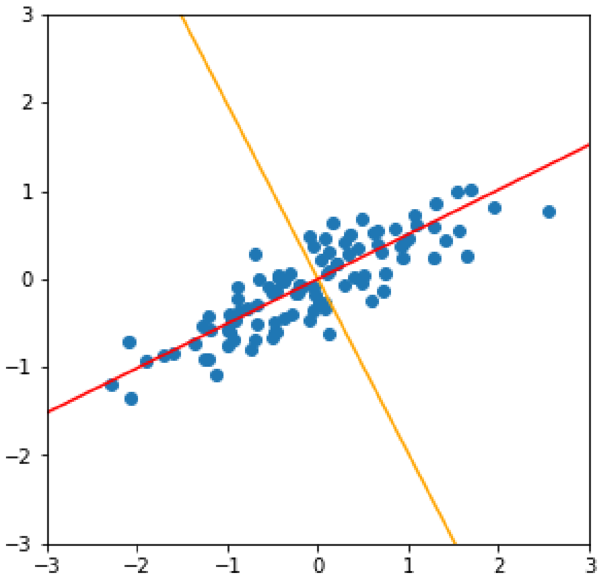
공분산행렬

$$\Sigma = \begin{pmatrix} 0.879 & 0.415 \\ 0.415 & 0.268 \end{pmatrix}$$

$$\lambda 1 = 1.089 \ \& \ k1 = \begin{pmatrix} 0.8923 \\ 0.4513 \end{pmatrix}$$

$$\lambda 2 = 0.058 \ \& \ k2 = \begin{pmatrix} -0.4514 \\ 0.8923 \end{pmatrix}$$





고유벡터 : 데이터가 분산된 방향

고유값 : 전체분산에 대한 주성분 PCi의 설명비율

total variance = tr(Σ) = tr(Λ) = λ1 + … + λn

var(PCi) = λi

0.879(σ1^2) + 0.268(σ2^2)

= 1.089(λ1) + 0.058(λ2) = 1.147

PC1의 설명력 = 1.089/1.147 = 0.9494

λ total V

PC2의 설명력 = 0.058/1.147 = 0.0506

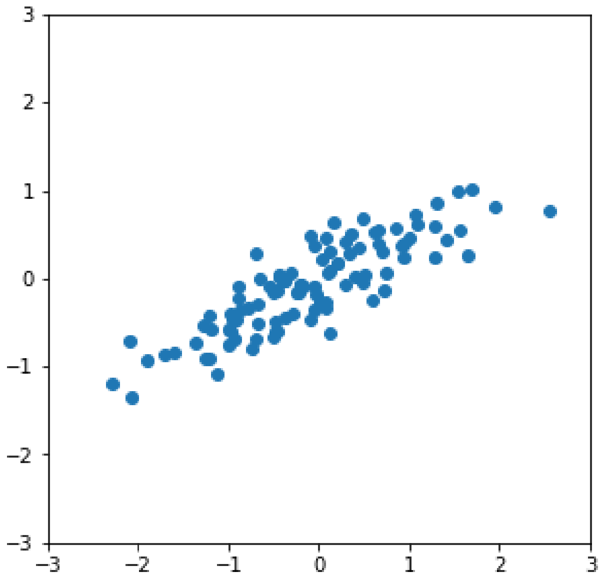
▪ Principal Component Analysis 주성분분석

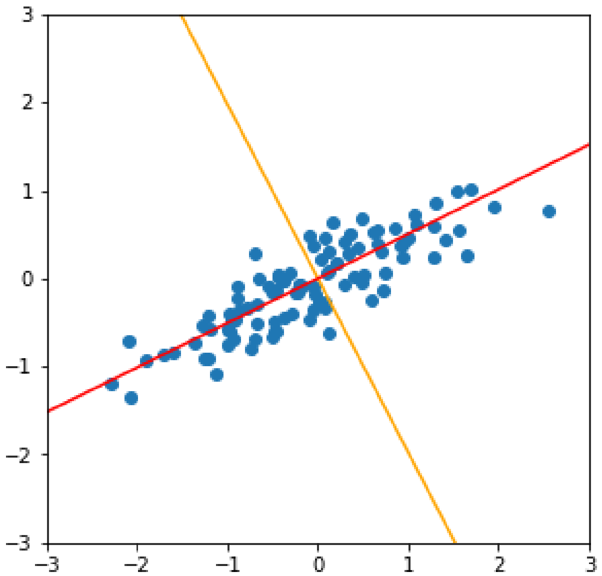
공분산행렬

$$\Sigma = \begin{pmatrix} 0.879 & 0.415 \\ 0.415 & 0.268 \end{pmatrix}$$

$$\lambda1 = 1.089 \ \& \ k1 = \begin{pmatrix} 0.8923 \\ 0.4513 \end{pmatrix}$$

$$\lambda2 = 0.058 \ \& \ k2 = \begin{pmatrix} -0.4514 \\ 0.8923 \end{pmatrix}$$





단! 공분산의 문제점 존재

- 측정단위에 영향을 받음
- 정규화 사용 → 아예 다른 값 나오므로 주의해야

$$\rho = \begin{pmatrix} 1.0 & 0.858 \\ 0.858 & 1.0 \end{pmatrix}$$

$$\lambda1 = 1.858 \rightarrow \begin{pmatrix} 0.707 \\ 0.707 \end{pmatrix}$$

$$\lambda2 = 0.142 \rightarrow \begin{pmatrix} -0.707 \\ 0.707 \end{pmatrix}$$

▪ Principal Component Analysis 주성분분석

$PC_i = k_i' \cdot X =$

ith주성분

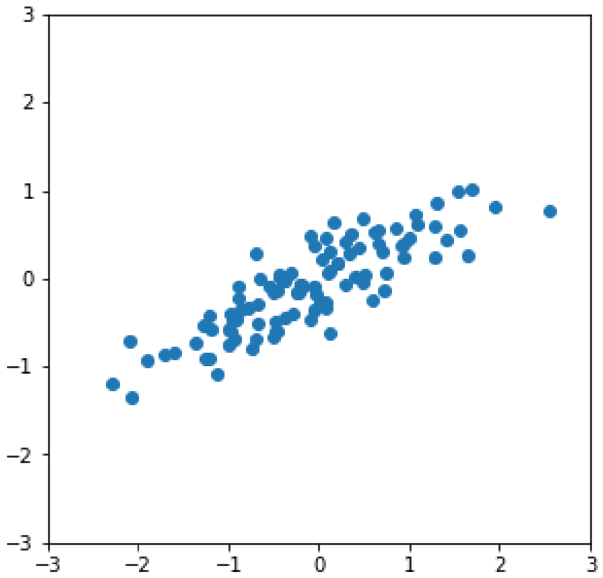
공분산행렬

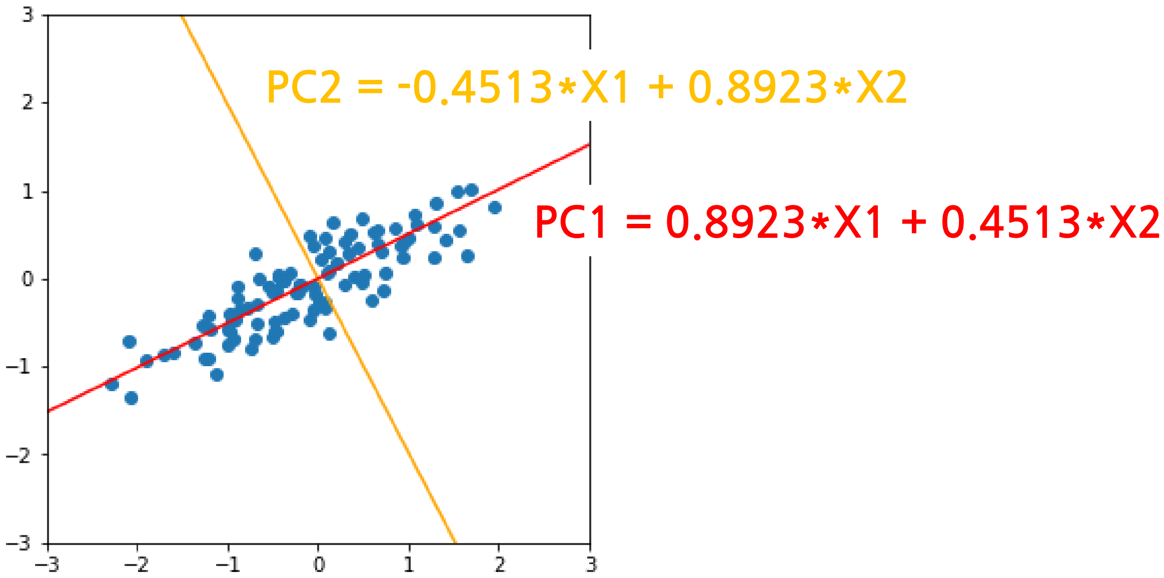
$$\Sigma = \begin{pmatrix} 0.879 & 0.415 \\ 0.415 & 0.268 \end{pmatrix}$$

$$\lambda 1 = 1.089 \;\&\; k1 = \begin{pmatrix} 0.8923 \\ 0.4513 \end{pmatrix}$$

고유벡터
직교

$$\lambda 2 = 0.058 \;\&\; k2 = \begin{pmatrix} -0.4514 \\ 0.8923 \end{pmatrix}$$





Unit 03 | PCA : Principal Component Analysis

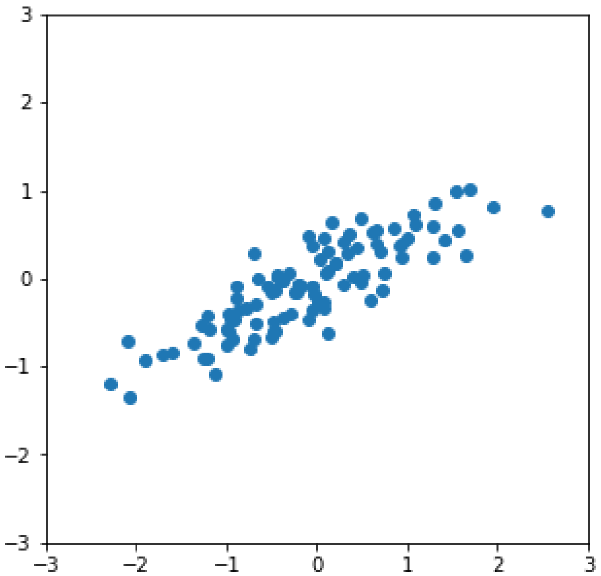
- Principal Component Analysis 주성분분석

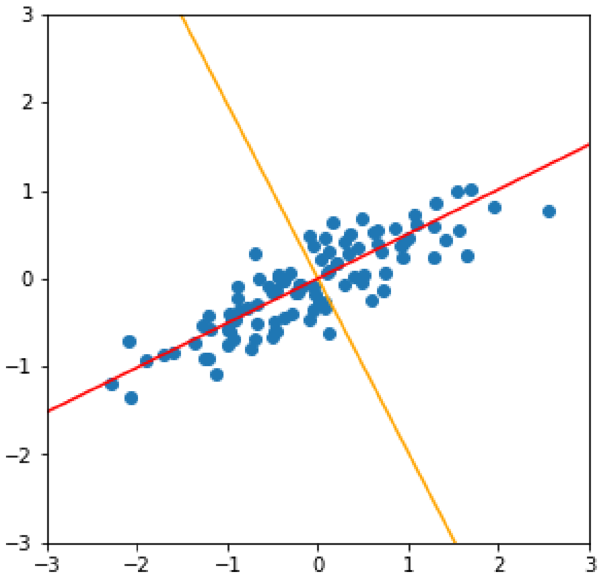
공분산행렬

$$\Sigma = \begin{pmatrix} 0.879 & 0.415 \\ 0.415 & 0.268 \end{pmatrix}$$

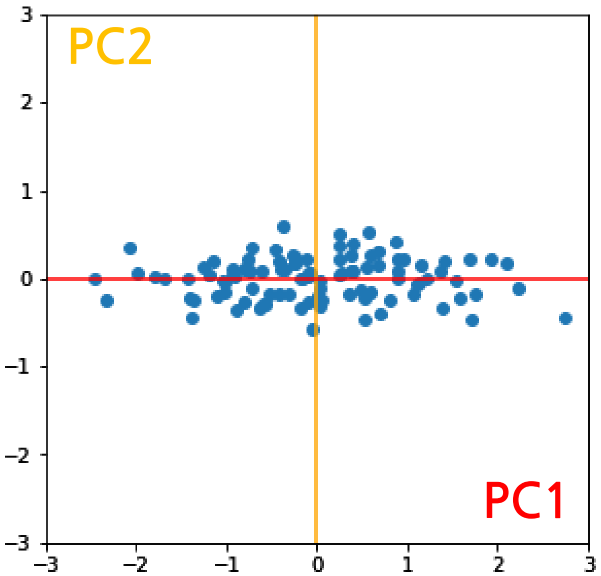
$$\lambda 1 = 1.089 \ \& \ k1 = \begin{pmatrix} 0.8923 \\ 0.4513 \end{pmatrix}$$

$$\lambda 2 = 0.058 \ \& \ k2 = \begin{pmatrix} -0.4514 \\ 0.8923 \end{pmatrix}$$

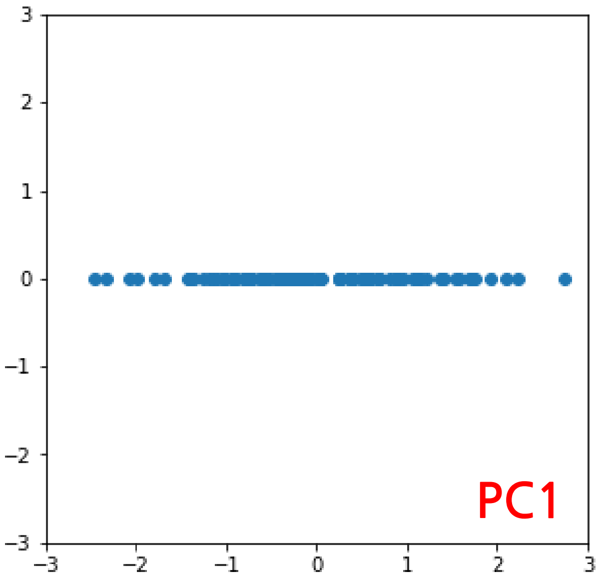




회전



주성분1만 사용



Unit 03 | PCA : Principal Component Analysis

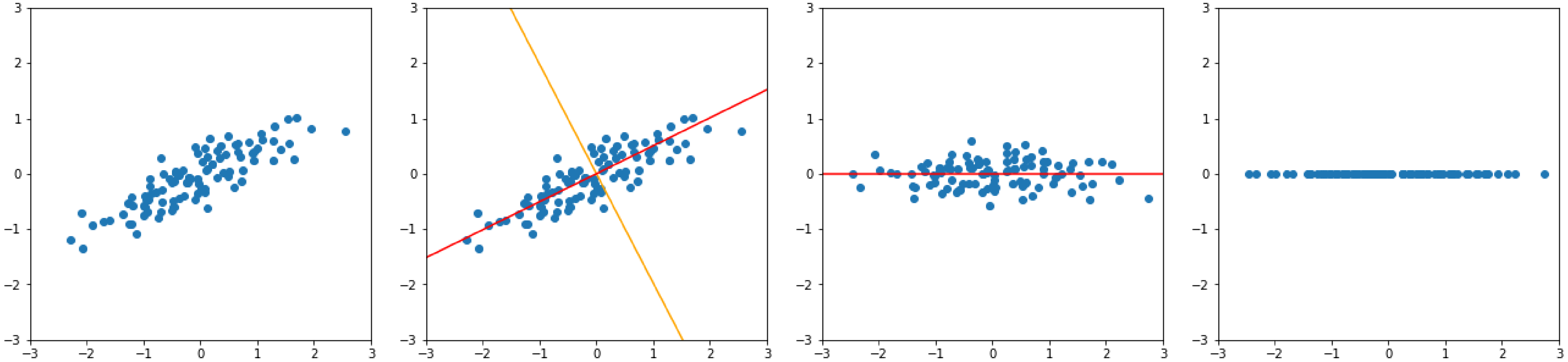
- **Principal Component Analysis 주성분분석**

  : 변수들의 전체 분산을 최대한 설명하는 소수의 주성분을 통해 분석하는 방법

  ➔ 데이터의 구조 최대한 유지

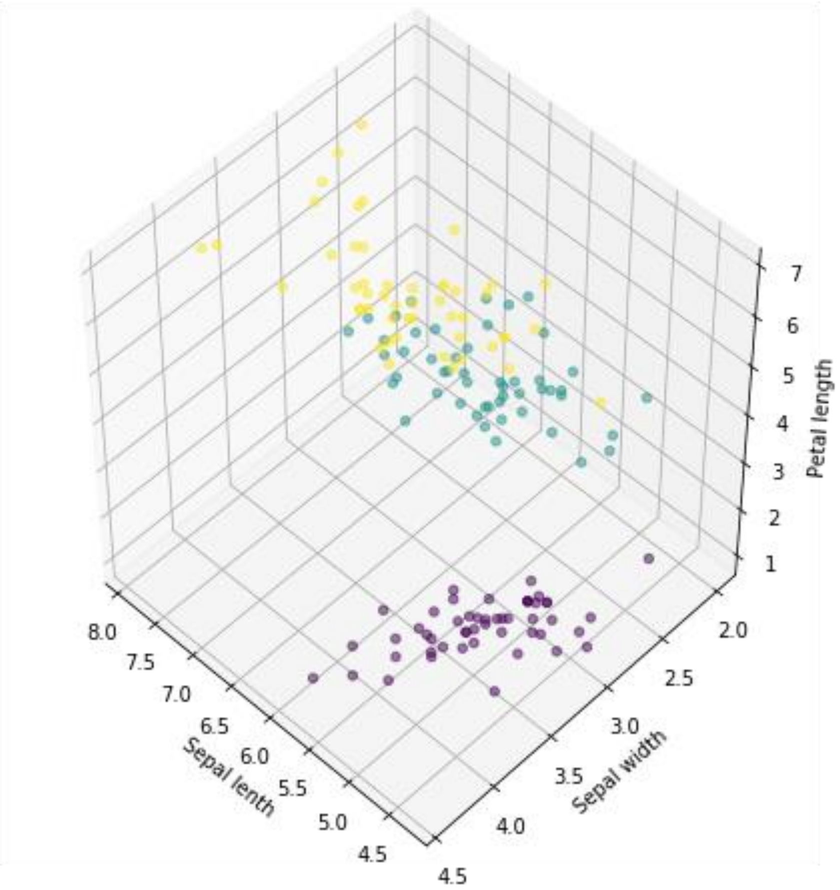
  ➔ 서로 상관관계가 있는 변수들 사이의 복잡한 구조를 좀 더 간편하게 변환

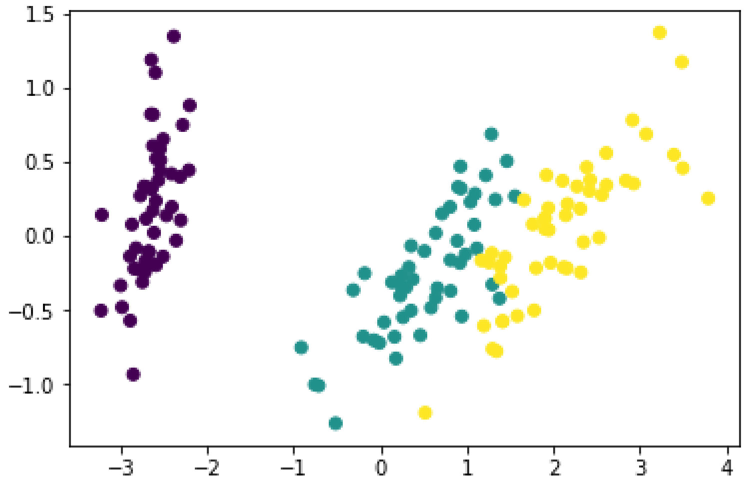
  ➔ 기하학적 측면, 좌표축을 회전시켜 얻어진 새로운 좌표축을 선택

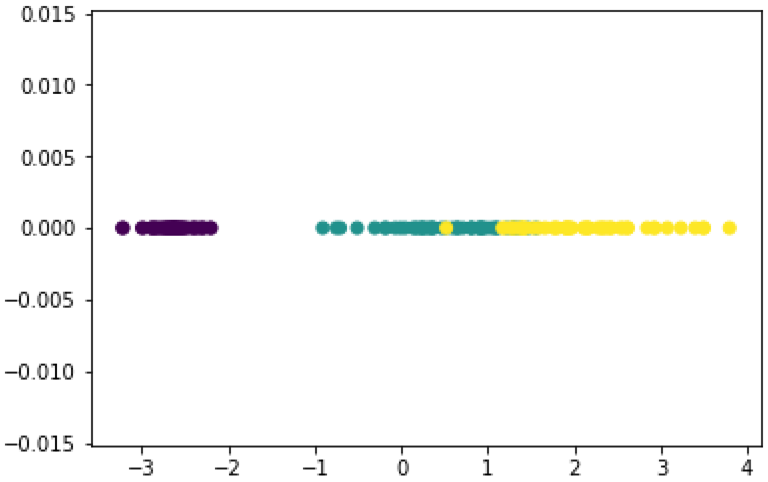


Unit 03 | PCA : Principal Component Analysis

- **Principal Component Analysis 주성분분석**



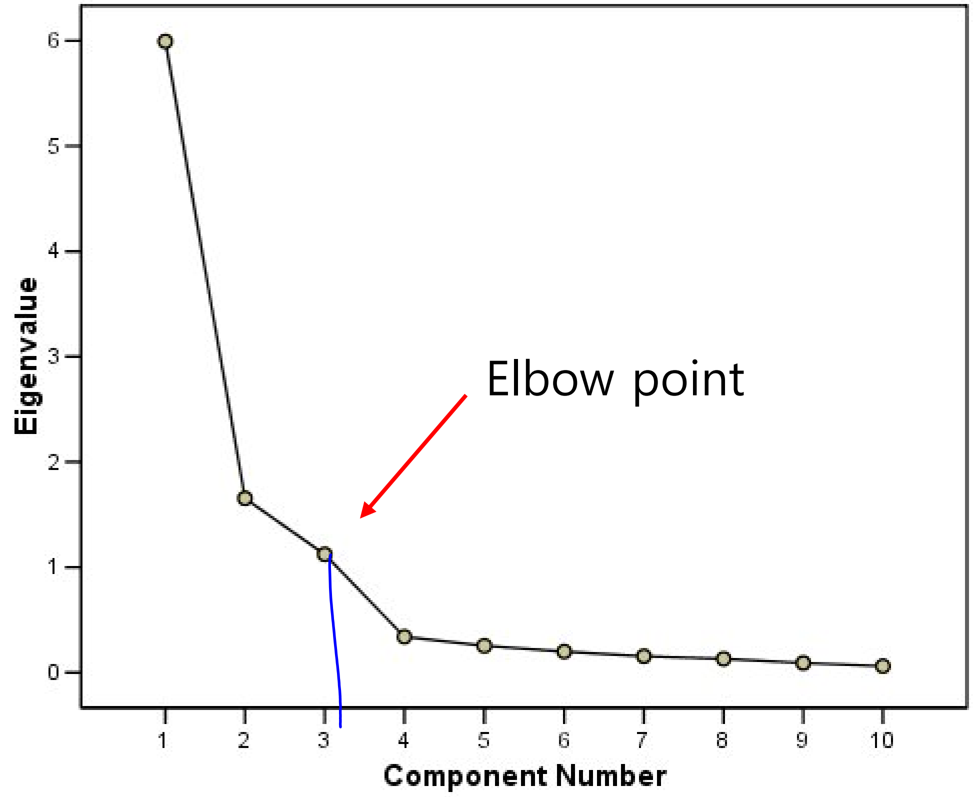




Unit 03 | PCA : Principal Component Analysis

▪ 주성분의 개수 k 결정

<Scree plot>



시각화 했을때 유용하다!!

1. Elbow point

   : 곡선의 기울기가 급격히 감소하는 지점

2. Kaiser's Rule

   : 고유값이 1 이상

3. 누적설명률이 70%~80% 이상인 지점

Unit 03 | PCA : Principal Component Analysis

▪ PCA Regression 주성분 회귀분석

PC1 = a11*x1 + a12*x2 + … + a1n*xn

PC2 = a21*x1 + a22*x2 + … + a2n*xn

다중공선성을 해결해준다.

y* = b0 + b1PC1 + b2PC2
= b0 + b1(a11*x1 + a12*x2 + … + a1n*xn) + b2(a21*x1 + a22*x2 + … + a2n*xn)
= b0 + (b1*a11+b2*a21)x1 + (b1*a12+b2*a22)x2 + (b1*a1n+b2*a2n)xn

➔ X에 대한 선형 결합

➔ 범주형자료는 사용 X

➔ 다른 알고리즘에 PC이용

※주의사항

Train dataset을 정규화 했으면,
Test dataset을 정규화 할 때는 Train의 mean & std 사용

Unit 03 | PCA : Principal Component Analysis

▪ PCA Regression 주성분 회귀분석

PC1 = a11*x1 + a12*x2 + … + a1n*xn

PC2 = a21*x1 + a22*x2 + … + a2n*xn

y* = b0 + b1PC1 + b2PC2
= b0 + b1(a11*x1 + a12*x2 + … + a1n*xn) + b2(a21*x1 + a22*x2 + … + a2n*xn)
= b0 + (b1*a11+b2*a21)x1 + (b1*a12+b2*a22)x2 + (b1*a1n+b2*a2n)xn

코드로!

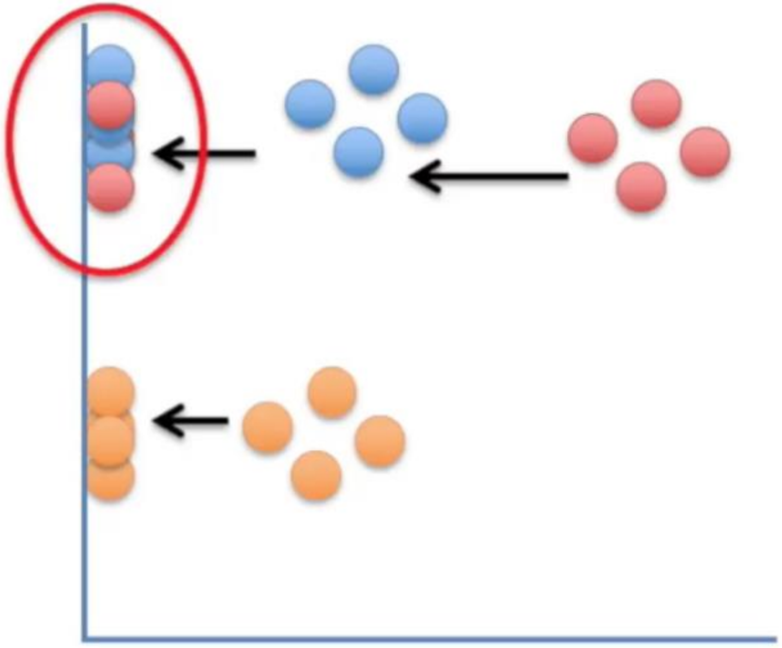
➔ X에 대한 선형 결합

➔ 범주형자료는 사용 X

➔ 다른 알고리즘에 PC이용

Unit 03 | PCA : Principal Component Analysis

▪ PCA의 문제점



PC + 원래변수

└> 성능 올라가기도 함.

선형으로 데이터를 투영

➔ 군집화 되어있는 데이터들이 뭉게짐

➔ 데이터 사이의 구별이 어려움

➔ 군집의 변별력이 사라짐

공간이 사라지는 문제점

Unit 04 | t-SNE : t - Stochastic Neighbor Embedding

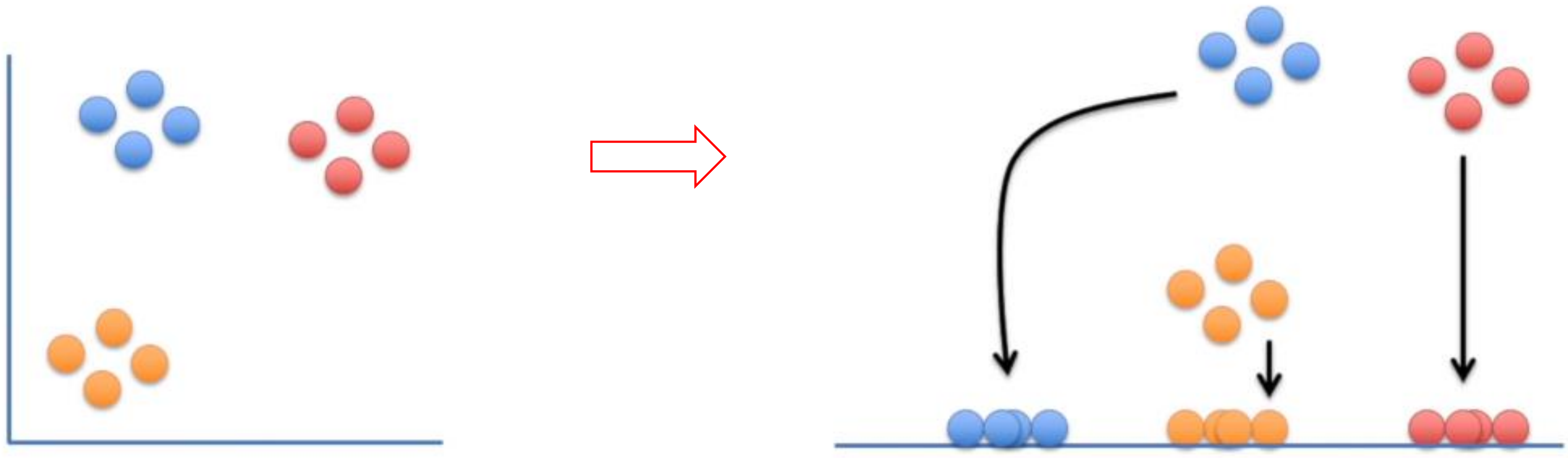
- **t - Stochastic Neighbor Embedding**

  : 고차원에서 데이터 x와 이웃 간의 거리를 최대한 보존하는 저차원의 y를 학습시켜,
  직관적으로 데이터 구조를 확인할 수 있도록 시각화 하는 기법

Unit 04 | t-SNE : t - Stochastic Neighbor Embedding

▪ t - Stochastic Neighbor Embedding

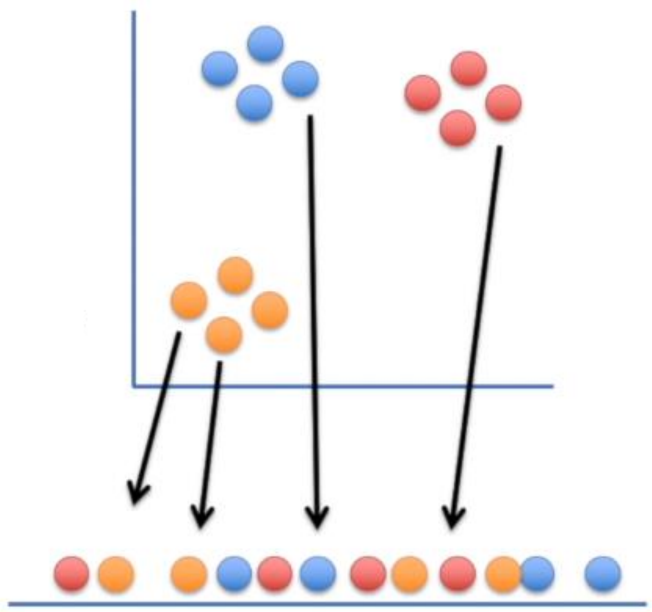
: 고차원에서 데이터 x와 이웃 간의 거리를 최대한 보존하는 저차원의 y를 학습시켜,
직관적으로 데이터 구조를 확인할 수 있도록 시각화 하는 기법



Unit 04 | t-SNE : t - Stochastic Neighbor Embedding

▪ t - Stochastic Neighbor Embedding

: 고차원에서 데이터 x와 이웃 간의 거리를 최대한 보존하는 저차원의 y를 학습시켜, 직관적으로 데이터 구조를 확인할 수 있도록 시각화 하는 기법



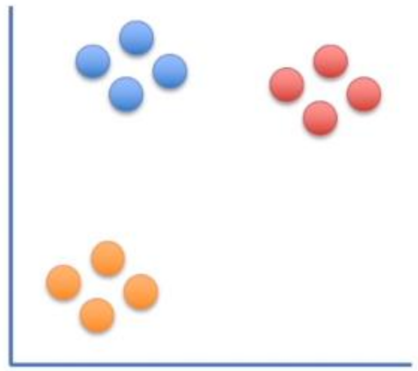
<t-SNE 과정>

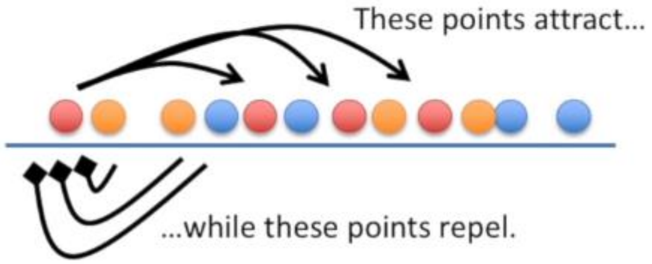
1. 랜덤하게 배열

Unit 04 | t-SNE : t - Stochastic Neighbor Embedding

▪ t - Stochastic Neighbor Embedding

: 고차원에서 데이터 x와 이웃 간의 거리를 최대한 보존하는 저차원의 y를 학습시켜,
직관적으로 데이터 구조를 확인할 수 있도록 시각화 하는 기법





<t-SNE 과정>

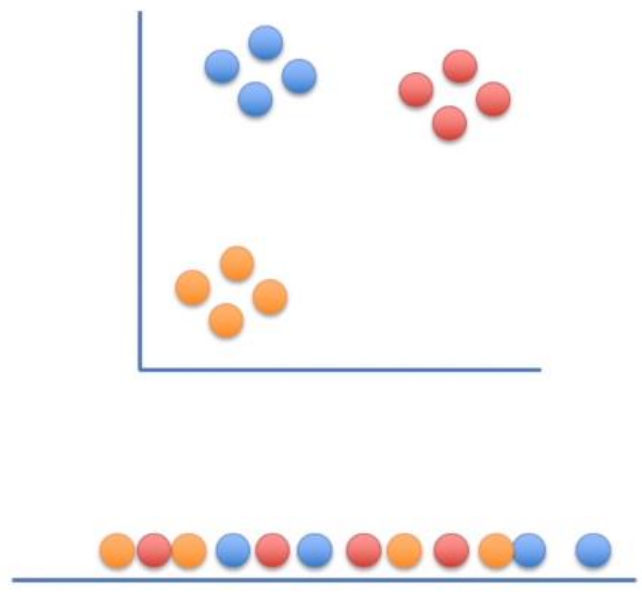
1. 랜덤하게 배열

2. 동일 군집의 값은 가까이로,
   다른 군집의 값은 멀리

Unit 04 | t-SNE : t - Stochastic Neighbor Embedding

▪ t - Stochastic Neighbor Embedding

: 고차원에서 데이터 x와 이웃 간의 거리를 최대한 보존하는 저차원의 y를 학습시켜,
직관적으로 데이터 구조를 확인할 수 있도록 시각화 하는 기법



<t-SNE 과정>

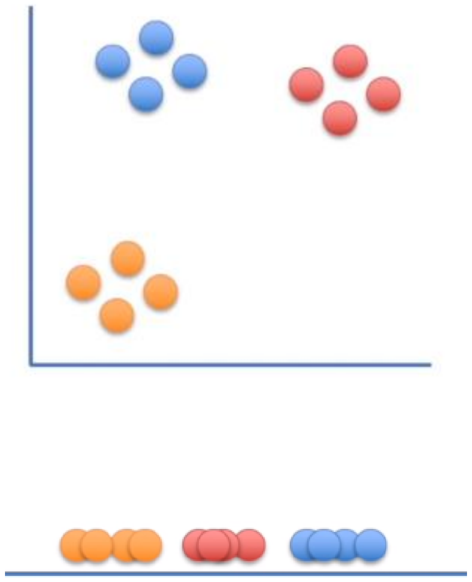
1. 랜덤하게 배열

2. 동일 군집의 값은 가까이로,
   다른 군집의 값은 멀리

3. 균형이 되는 지점으로 이동

▪ t - Stochastic Neighbor Embedding

: 고차원에서 데이터 x와 이웃 간의 거리를 최대한 보존하는 저차원의 y를 학습시켜,
직관적으로 데이터 구조를 확인할 수 있도록 시각화 하는 기법



<t-SNE 과정>

1. 랜덤하게 배열 → 할때마다 결과가 바뀜

2. 동일 군집의 값은 가까이로,
   다른 군집의 값은 멀리

3. 균형이 되는 지점으로 이동

4. 2&3 반복 후 군집끼리 가까이 이동

Unit 04 | t-SNE : t - Stochastic Neighbor Embedding

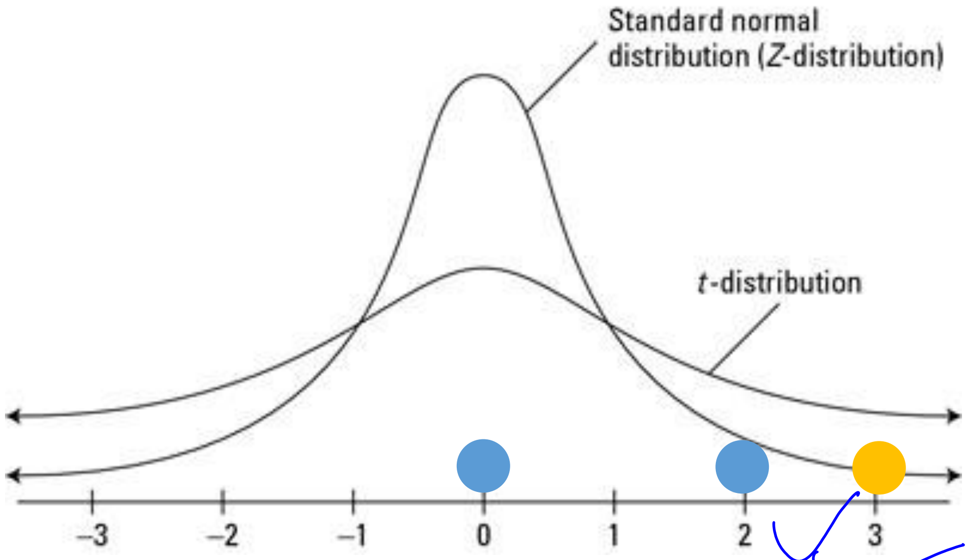
거리정보를 확률로 계산

- **t - Stochastic Neighbor Embedding**

: 고차원에서 데이터 x와 이웃 간의 거리를 최대한 보존하는 저차원의 y를 학습시켜, 직관적으로 데이터 구조를 확인할 수 있도록 시각화 하는 기법

➔ 데이터의 군집성 유지

➔ 가까운 이웃(동일 군집 데이터)의 거리만을 유지



거리 정보를 확률적으로 계산

➔ 저차원의 y를 조금씩 업데이트

정규분포 : 꼬리가 얇아 거리가 일정 수준을 넘으면
매우 작은 값으로 다 확률이 같아짐

➔ 꼬리가 두꺼운 t - 분포를 사용

꼬리가 두꺼워야 애매 구별이 잘됨 (P값의 차이)

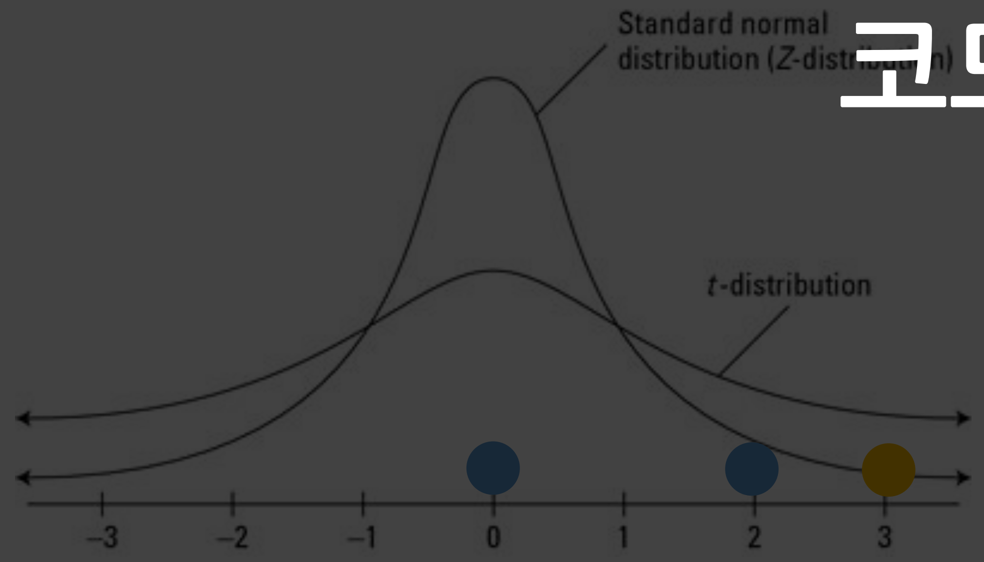
Unit 04 | t-SNE : t - Stochastic Neighbor Embedding

- t - Stochastic Neighbor Embedding

: 고차원에서 데이터 x와 이웃 간의 거리를 최대한 보존하는 저차원의 y를 학습시켜,
직관적으로 데이터 구조를 확인할 수 있도록 시각화 하는 기법

➔ 데이터의 군집성 유지

➔ 가까운 이웃(동일 군집 데이터)의 거리만을 유지



거리 정보를 확률적으로 계산

코드로 !

➔ 저차원의 y를 조금씩 업데이트

정규분포 : 꼬리가 얇아 거리가 일정 수준을 넘으면
매우 작은 값으로 다 확률이 같아짐

➔ 꼬리가 두꺼운 t - 분포를 사용

Unit 04 | t-SNE : t - Stochastic Neighbor Embedding

▪ t - Stochastic Neighbor Embedding

: 고차원에서 데이터 x와 이웃 간의 거리를 최대한 보존하는 저차원의 y를 학습시켜,
직관적으로 데이터 구조를 확인할 수 있도록 시각화 하는 기법

➔ 데이터의 군집성 유지

➔ 가까운 이웃(동일 군집 데이터)의 거리만을 유지

※ 문제점

계속 업데이트하는 방식으로 변형된 <u>값이 매번 바뀜</u>

➔ t-SNE 결과를 새로운 feature로 사용 불가능 시각화용으로 사용

군집만 유지하고 값이 완전히 변화

Unit 05 | Assignment

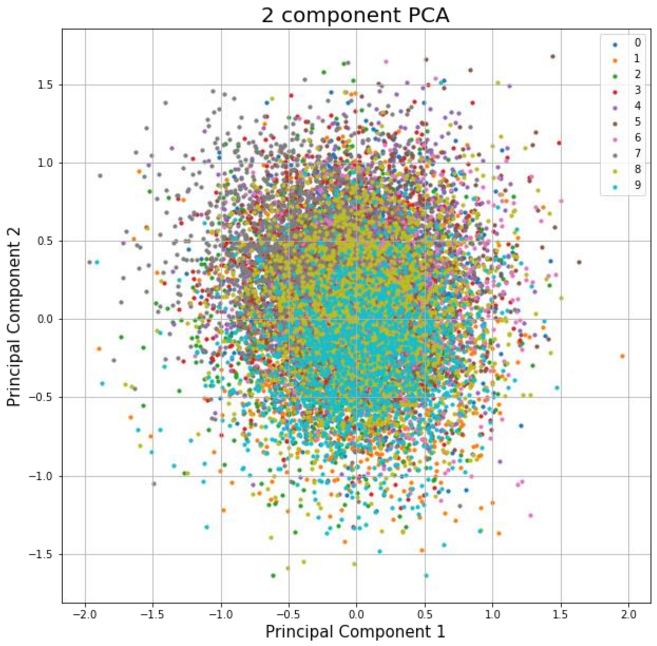
Q1. PCA를 직접 구현한 myPCA (Class 이용해서 만들기!)

Q2. 단어 임베딩 벡터 시각화
➔ kmeans로 n=10 군집화
➔ 차원축소 전 데이터로 시각화
➔ PCA & TSNE로 시각화

Q3. 축소 전 데이터와 PCA축소 데이터 training 시간 비교
➔ train 데이터 로드 후 train과 test(20%) 로 split
➔ 원본 데이터로 knn과 svm 적용 + 타임스탬프 걸린시간
➔ 축소 데이터로 knn과 svm 적용 + 타임 스템프
➔ training 시간 비교 & test accuracy 비교

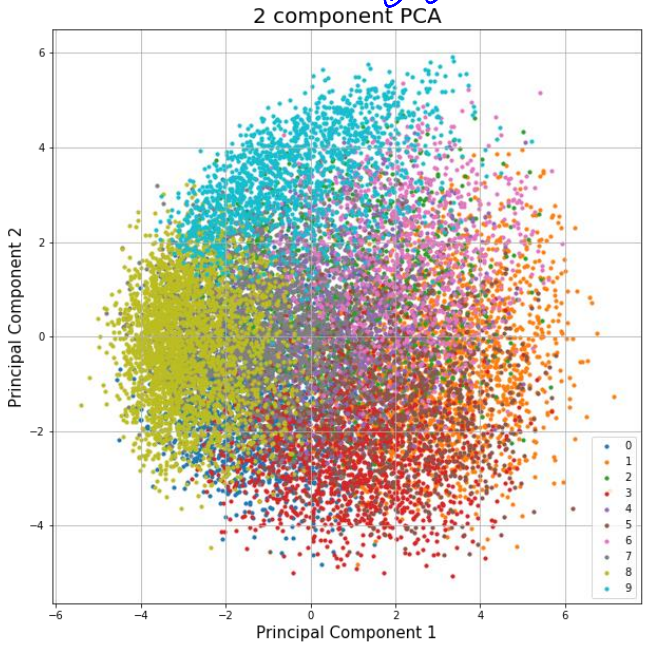
Unit 05 | Assignment

Q2 결과



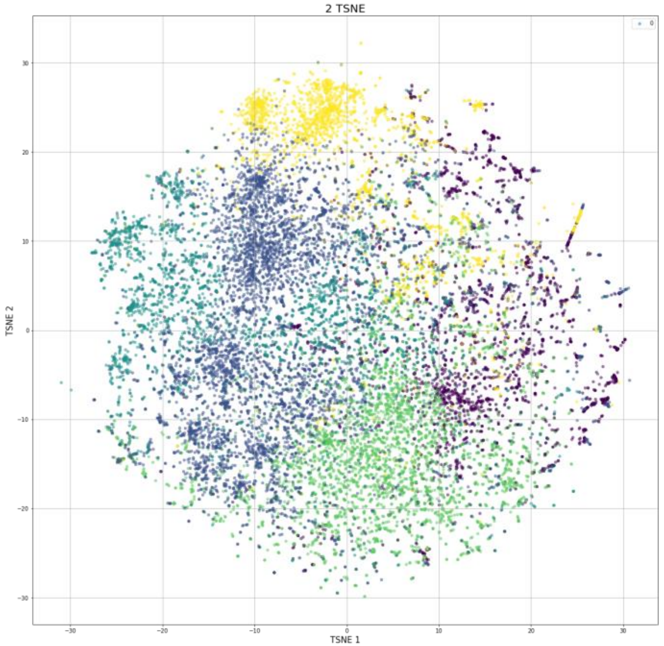
원본

10개군집



PCA

5개군집



차원 상관은 없음

TSNE (비슷한 느낌)

Unit 00 | 참고자료

\# 공돌이의 수학정리노트 ➔ 자세한 수학적 접근
PCA : https://wikidocs.net/7646
Eigen : https://wikidocs.net/4050

\# ratsgo's blog ➔ 상세한 예시와 R코드
PCA : https://ratsgo.github.io/machine%20learning/2017/04/24/PCA/
t-SNE : https://ratsgo.github.io/machine%20learning/2017/04/28/tSNE/

\# 조대협의 블로그 ➔ Python 코드
PCA : https://bcho.tistory.com/1209?category=555440
t-SNE : https://bcho.tistory.com/1210

\# 꼬깔콘의 분석일지 ➔ 차원의 저주
https://kkokkilkon.tistory.com/127

\# t-SNE 관련 youtube
https://www.youtube.com/watch?v=NEaUSP4YerM

\# 책 : 응용다변량분석 (성웅현)

Q & A

들어주셔서 감사합니다.