차원 축소(Dimension Reduction)

데이터 분석 과정 중 전처리에 주로 사용된다

전처리에서 사용하는 기술: 정규화, 차원 축소, 영상처리

배경

이론적으로 차원의 증가는 모델 성능을 향상시킨다

* 가정: 모든 변수가 서로 독립이다

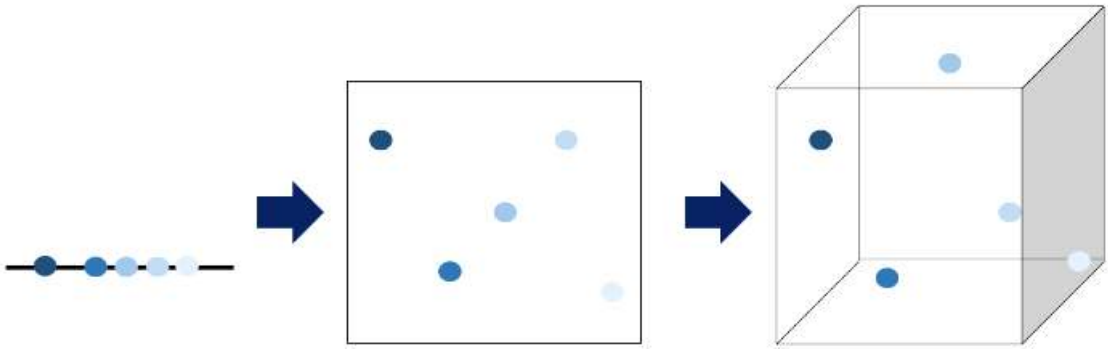
BUT 실제로는 차원의 증가는 모델 성능을 저하시킨다

* 실제: 모든 변수는 서로 상관관계가 있고 노이즈가 존재한다

차원의 저주(Curse of dimensionality)

차원이 증가할수록 동일 정보량 표현을 위한 필요 데이터 수가 지수적으로 증가

→ 차원이 증가하면서 학습 데이터 수가 차원 수보다 적어져 모델 성능이 저하된다



but 차원이 증가한다고 무조건 차원의 저주 발생하는 것은 아니다

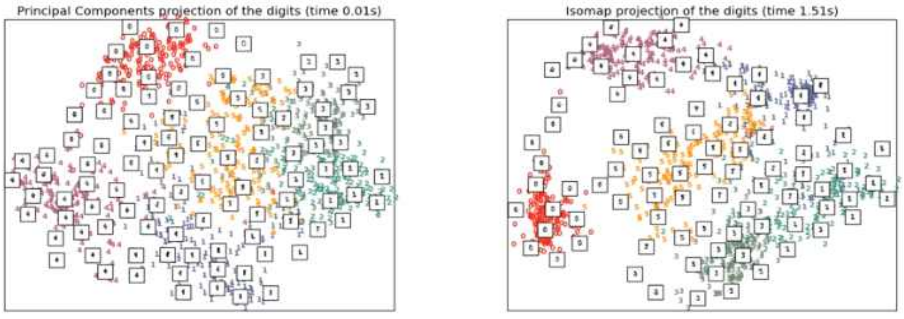
차원 증가로 변수의 수가 증가하는데,

변수의 수가 데이터(관측치)의 수보다 많아지면 차원의 저주가 발생한다

일반적으로 intrinsic dimension은 original dimension보다 상대적으로 작다

Intrinsic dimension: 데이터의 표현력을 상실하지 않는 최소 차원

MNIST(16×16=256차원) 데이터를 IPCA와 ISOMAP을 통해 2차원 데이터로 축소

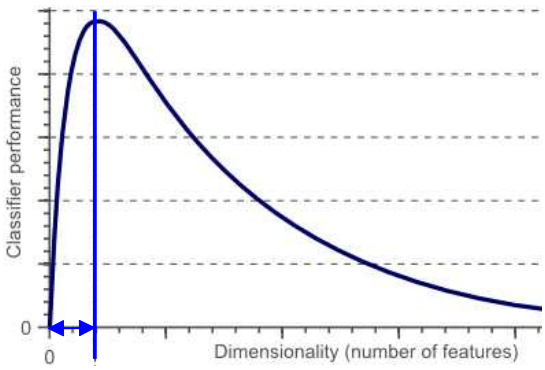


차원이 높을수록 발생하는 문제

1. 데이터에 포함될 노이즈의 비율도 높아진다 → 모델 성능 감소
2. 모델 학습과 추론의 계산 복잡도가 높아진다
3. 동일 성능을 얻기 위해 더 많은 데이터의 수가 필요하다

해결 방법

1. 도메인 지식을 이용한다: 중요한 특성만 사용한다
2. 목적 함수에 Regularization term을 추가한다
3. 차원 축소 기술을 전처리에 사용한다



목적

: 모델의 성능을 최대로 해주는 변수의 일부 셋(subset)을 찾는다

효과

1. 변수 간 상관관계 제거한다
2. 단순한 후처리(post-processing)
3. 적절한 정보를 유지하면서 중복되거나 불필요한 변수를 제거한다
4. 시각화가 가능하다

지도학습 기반 차원 축소

학습 결과가 피드백되어 Feture Selection을 반복한다

→ 정답과 예측값 비교를 통해 에러 보정

비지도학습 기반 차원 축소

지도학습처럼 피드백을 통한 Feature Selection 반복이 없다

차원 축소 방법

1. 변수/특성 선택(Feature selection)

: 유의미한 변수만 선택한다 $x_1, x_2, \ldots, x_n \rightarrow x_1, x_k$

장점: 선택한 변수 해석이 용이하다

단점: 변수간 상관관계 고려가 어렵다

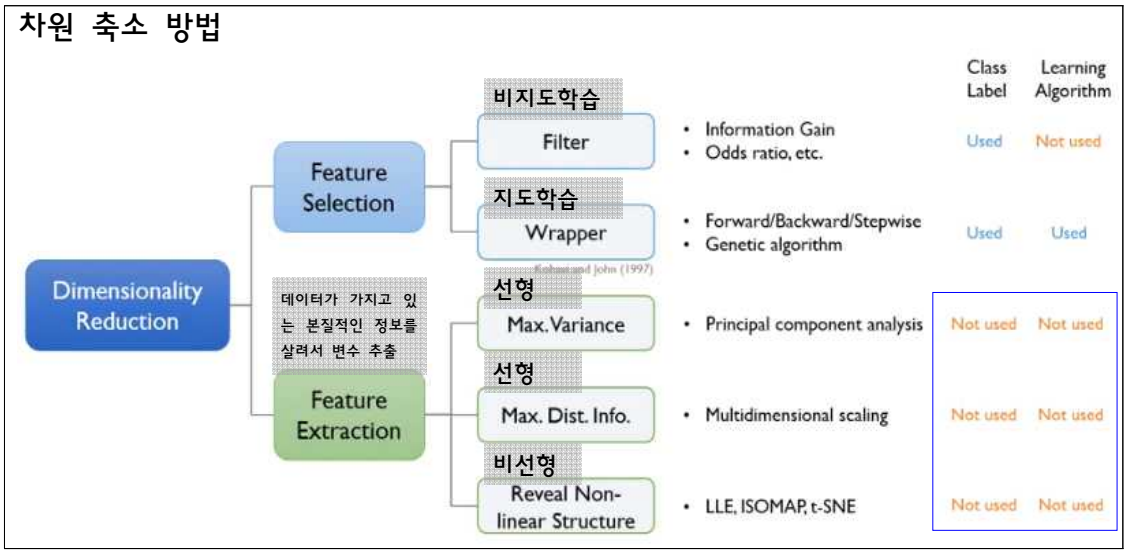
2. 변수/특성 추출(Feature extraction)

: 예측 변수의 변환(선형 결합)을 통해 새로운 변수 추출(생성)한다

= 변수/특성 생성(Feature construction) $z = f(x_1, x_2, ..., x_n)$

장점: 변수가 상관관계를 고려한다, 변수의 개수를 많이 줄일 수 있다

단점: 추출된 변수의 해석이 어렵다

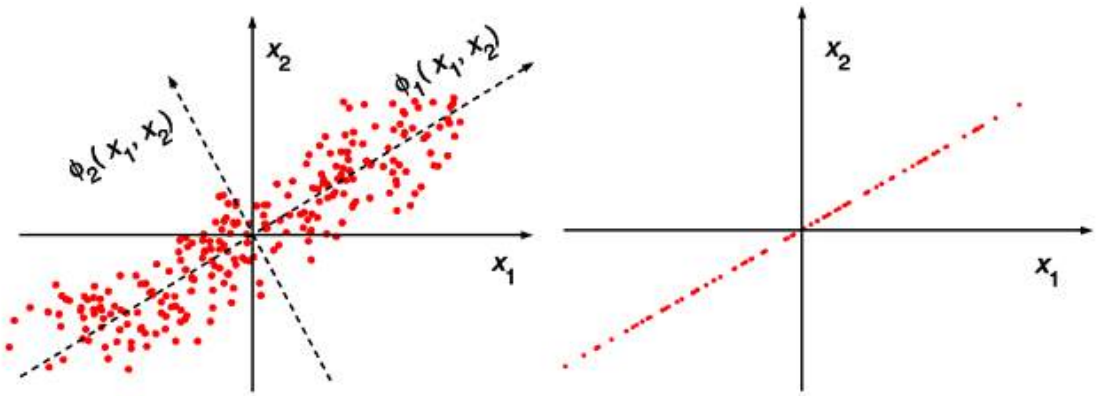


주성분 분석(PCA. Principal Component Analysis)

차원을 줄이는 비지도 학습 방법

사영한 다음 원 데이터의 분산(variance)을 최대한 보존할 수 있는 기저(축)를 찾아 차원을 줄이는 방법

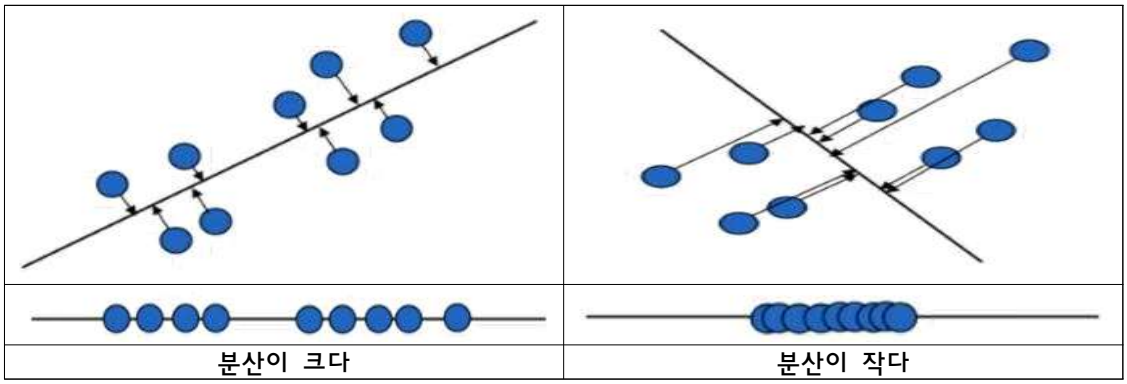
기저(축): Principal Axis → PC(Principal Components)



(a) PCA basis (b) PCA reduction to 1D

데이터를 사영(projection)할 때는 손실되는 정보의 양이 적은 쪽의 기저를 선택

분산이 보존되는(큰) 기저를 선택한다



분산이 크다 | 분산이 작다

주성분 분석 → 선형 결합

데이터(X) 사영 변환 후 새로운 변수(Z)에도 분산이 보존되는 기저(α)를 찾는다

$X_1, X_2, ..., X_p$: 원데이터 p개의 변수

$\alpha_i = [\alpha_{i1}, \alpha_{i2}, ..., \alpha_{ip}]$: i번째 기저($basis$) 또는 계수($loading$)

$Z_1, Z_2, ..., Z_p$: 각 기저로 사영 변환 후 변수(주성분). PC

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

P차원 데이터 → PCA → P개의 축 → P차원 변수 추출(생성)

공분산(Covariance): 변수의 상관 정도

X: 입력 데이터(n개 데이터, d개 변수)

$\overline{X}$: X의 평균

$$\underset{[d\times d]}{Cov(X)} = \frac{1}{n}(\underset{[d\times n]}{X} - \overline{X})(\underset{[n\times d]}{X} - \overline{X})^T$$

데이터셋의 전체 분산(Total Variance)

$$tr[Cov(X)] = Cov(X)_{11} + Cov(X)_{22} + \cdots + Cov(X)_{dd}$$

ex) $A = \begin{bmatrix} 1 & 3 & 5 \\ 5 & 4 & 1 \\ 3 & 8 & 6 \end{bmatrix}$ $cov(A) = \begin{bmatrix} 2.67 & 0.67 & -2.67 \\ 0.67 & 4.67 & 2.33 \\ -2.67 & 2.33 & 4.67 \end{bmatrix}$

$$tr[cov(A)] = cov(A)_{11} + cov(A)_{22} + cov(A)_{33} = 2.67 + 4.67 + 4.67 = 12.01$$

사영(Projection)

$\vec{x}$: 사영 후 벡터
p: 직교사영을 위한 스케일러
$\vec{b}$: 데이터, $\vec{a}$: 기저축(PC)

$$(\vec{b} - p\vec{a})^T\vec{a} = 0 \Rightarrow \vec{b}^T\vec{a} - p\vec{a}^T\vec{a} = 0 \Rightarrow p = \frac{\vec{b}^T\vec{a}}{\vec{a}^T\vec{a}}$$

$$\vec{x} = p\vec{a} = \frac{\vec{b}^T\vec{a}}{\vec{a}^T\vec{a}}\vec{a}$$

$$If\ \vec{a}\ is\ unit\ vector, p = \vec{b}^T\vec{a} \Rightarrow \vec{x} = p\vec{a} = (\vec{b}^T\vec{a})\vec{a}$$

고유값(eigenvalue)과 고유벡터(eigenvector)

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

벡터에 행렬을 곱하는 것은 선형 변환의 의미를 가진다
1. 고유 벡터는 변환에 의해 방향 변화가 발생하지 않는다
2. 고유벡터의 크기 변화는 λ 만큼 변한다

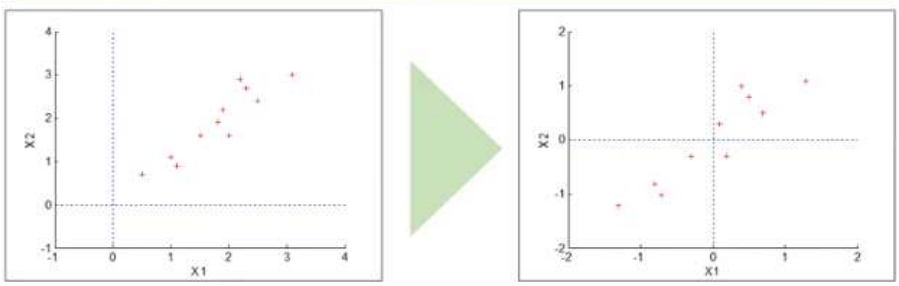
행렬 A가 non-singular 하다면, d개의 고유값과 고유벡터가 존재한다
고유 벡터는 서로 직교한다(Orthogonal)

$$tr(A) = \lambda_1 + \lambda_2 + \cdots + \lambda_d$$

주성분 분석 알고리즘

1. 데이터 센터링(Data Centering)

정규화를 통해 데이터 평균을 0으로 변경한다



2. 최적화 문제 정의

데이터 X를 기저 벡터 W에 사영하면, 사영 후 분산은

$$V = \frac{1}{n}(w^T X)(w^T X)^T = \frac{1}{n} w^T X X^T w = w^T S w$$

S: X의 covariance matrix

PCA의 방향성: 사영된 데이터 분산의 최대화

$max\, w^T S w \;\; s.t.\; w^T w = 1$을 만족하는 S

3. 최적화 문제 솔루션

라그랑지 승수(Largrangian multiplier)를 적용한다

S: X의 $covariance\, matrix$
w : S의 $eigenvector$
λ : S의 $eigenvalue$

$max\, w^T S w \;\; s.t.\; w^T w = 1$

$L = w^T S w - \lambda(w^T w - 1)$

$\frac{\partial L}{\partial w} = 0 \implies Sw - \lambda w = 0 \implies (S - \lambda I) w = 0$

을 만족하는 $Eigenvalue(\lambda)$와 $Eigenvector(w)$

4. 주축 정렬

Eigenvectors를 대응하는 Eigenvalue가 큰 순서대로 정렬한다

w_1은 eigenvector이고 대응되는 eigenvalue는 λ_1이다

$v = (w_1^T X)(w_1^T X)^T = w_1^T X X^T w_1 = w_1^T S w_1$

$since \;\; Sw_1 = \lambda_1 w_1, \;\; w_1^T S w_1 = w_1^T \lambda_1 w_1 = \lambda_1 w_1^T w_1 = \lambda_1$

이므로 λ_1은 w_1에 사영된 데이터의 분산이다.

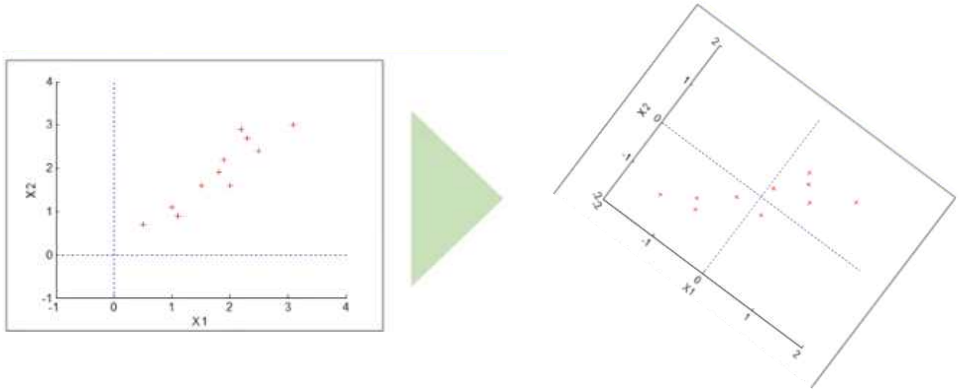
λ 크다 → 분산이 크다 → 손실되는 데이터의 양이 적다 → w는 더 좋은 기저

첫 번째 주성분으로 설명 가능한 데이터 비율 $= \frac{\lambda_1}{\lambda_1 + \lambda_2}$

⇒ 전체 분산 중 해당 기저가 설명할 수 있는 데이터의 비율
n차원에서 어떤 축을 몇 개 결정할지 도움을 준다

5. PCA로 변환된 데이터

Principle Component 1= $Z_1 = W_1^T X$

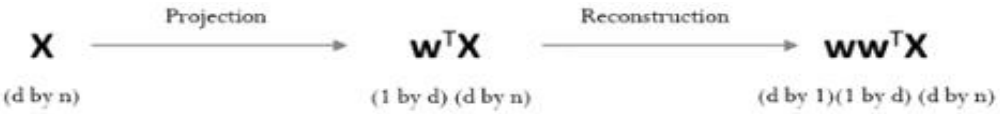


6. 원데이터로 복원

2D → 1D PCA → 2D 데이터 복원(Reconstruction)

└ loss가 발생하여 원데이터와 복원데이터가 같지는 않다

* 원래대로 복원할 수는 없다



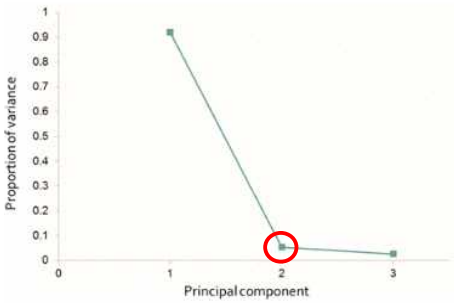
x_1	0.69	−1.31	0.39	0.09	1.29	0.49	0.19	−0.81	−0.31	−0.71
x_2	0.49	−1.21	0.99	0.29	1.09	0.79	−0.31	−0.81	−0.31	−1.01

z_1	0.83	−1.78	0.99	0.27	1.68	0.91	−0.10	−1.14	−0.44	−1.22

x_1'	0.56	−1.21	0.67	0.19	1.14	0.62	−0.07	−0.78	−0.30	−0.83
x_2'	0.61	−1.31	0.73	0.20	1.23	0.67	−0.07	−0.84	−0.32	−0.90

주성분 개수 선정법

1. 고유값 감소율이 유의미하게 낮아지는 elbow point의 주성분을 선택한다



2. 일정 수준 이상의 분산비를 보존하는 최소 주성분을 선택한다(보통 70% 이상)

ex) 전체 분산의 70%를 포함하기 위해서는 몇 개의 주성분을 사용해야 할까?

PC	1	2	3	4
$\frac{PC\ i\text{의 분산}}{\sum\text{분산}}$	0.35	0.30	0.20	0.15

PC 1: 0.35 ⟨ 0.70

PC 1 + PC 2: p.35 + 0.30 = 0.65 ⟨ 0.70

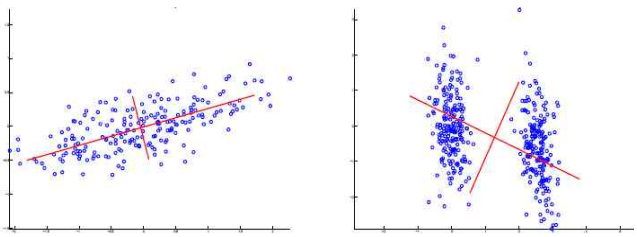
PC 1 + PC 2 + PC 3 = 0.35 + 0.30 + 0.20 = 0.85 ⟩ 0.70

→ PC1, 2, 3 선택

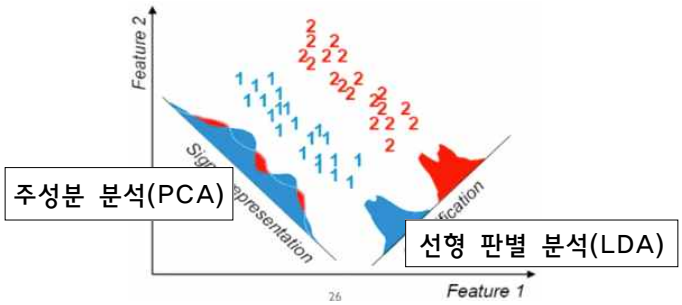
한계

1. 데이터 분포가 가우시안이 아니거나 다중 가우시안에 대해서는 적용하기 어렵다

주성분 분석의 가정: 데이터 분포가 단일 가우시안이다



2. 분류 문제를 위해 디자인되지 않아, 분류 모델의 성능 향상을 보장하지 못한다



랜덤 PCA(Randomized PCA)

: Randomized PCA는 QR 분해를 이용해서 행렬의 SVD를 수행하는 방법

자료의 크기 또는 특성 변수의 크기가 매우 크면 주성분 W를 구하기 위한 특이값 분해(SVD) 계산이 불가능하거나 시간이 많이 소요된다

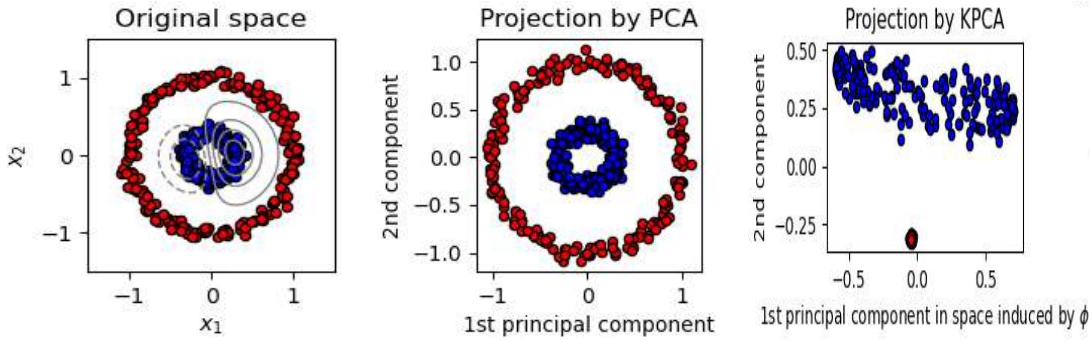
→ 이때 Randomized PCA가 유용하다

커널 PCA(Kernelized PCA)

: 특성 변수 x를 비선형 h(x)로 변환한 후, 이를 PCA 하여 차원 축소하는 방법

PCA는 선형 변환이고 Kernelized PCA는 비선형 변환이다

SVM의 커널 트릭을 PCA에서도 사용한다



차원 축소 api

sklearn.decomposition.PCA(n_components=None)

n_components: 주성분 개수

int, float or 'mle', default = None

sklearn.decomposition.KernelPCA(n_components=None, kernel='linear', gamma=None)

n_components: int, default = None

kernel: {'linear', 'poly', 'rbf', 'sigmoid', 'cosine',

precomputed'}, default = 'linear'

gamma: float, default = None