<https://towardsdatascience.com/pca-and-svd-explained-with-numpy-5d13b0d2a4d8>

PCA와 SVD는 linear 차원 축소방법으로 high dimensional data matrix에서 features의 linear combination을 찾아낸다.

PCA는 종종 생물학자들이 population genetics, transcriptomics, proteomic, microbiome의 dataset에서 source variances를 분석하고 시각화하는데 사용됨

SVD(특히 reduced SVD)는 자연어 처리과정 분야에서 널리 사용됨

PCA

* To find linearly uncorrelated orthogonal axes, which are also known as principle components (PCs) in the m-dimensional space to project the data points onto those PCs
* The first PC captures the largest variance in the data
* All the PCs are orthogonal to each other
* Use a pair of perpendicular lines in the 2D space as the two PCs
* To make the first PC capture the largest variance, we rotate our pair of PCs to make one of them optimally align with the spread of the data points
* all the data points can be projected onto the PCs -> **dimensionality-reduced representation of the dataset**
* PCs can be determined via eigen-decomposition of the covariance matrix C
* The geometrical meaning of eigen-decomposition is to find a new coordinate system of the eigenvectors for C through rotations.
* https://miro.medium.com/max/163/1*c1S0_26A8RxEQQUVyMp5Vw.png (covariance matrix : C=mxm, matrix of eigenvector : W=mxm, diagonal matric of m eigenvalues : Λ)

\*PCA 구하는 과정 요약

1.데이터 집합에 대해 공분산 행렬 구함

- 데이터의 집합의 각 열을 데이터의 속성이라 가정

2. 공분산 행렬에서 eigenvalue, eigenvector 구함

3. eigenvalue 목록에서 값이 높은 k개 인덱스 구함

- 인덱스 값이 i1, i2, …, ik인 경우 가 주요 구성요소가 됨

Python

import numpy as np

def pca(X):

#data matrix X, assumes 0=centered

n,m=X.shape

assert np.allclose(X.mean(axis=0), np.zeros(m))

#compute covariance matrix

C=np.dot(X.T,X)/(n-1)

#Eigen decomposition

eigen\_vals, eigen\_vecs=np.linalg.eig(C)

#project X onto PC space

X\_pca=np.dot(X,eigen\_vecs)

return X\_pca

SVD

* Another decomposition method for both real and complex matrices
* decomposes a matrix into the product of two unitary matrices (U, V\*) and a rectangular diagonal matrix of singular values (Σ)
* X=UΣV\*
* Unitary matrices U,V = real matrices
* SVD는 np.linalg.svd로 구현가능

Python

def svd(X):

#data matrix X, X doesn't need to be 0-centered

n,m=X.shape

#compute full SVD

U,Sigma, Vh=np.linalg.svd(X,

full\_matrices=False,

compute\_uv=True)

#transform X with SVD components

X\_svd=np.dot(U, np.diag(Sigma))

return X\_svd

PCA

: 행렬 A의 공분산 행렬에 대한 eigenvalue 와 eigenvector를 구하고 최종적으로는 상위 n개의 eigenvalue에 해당하는 eigenvector만을 추려내서 차원 축소하는 것

각 데이터의 분산을 최대치로 하는 직교 좌표들을 구하고 이 직교 좌표들 중 분산을 보다 크게 하는 상위권 n개의 좌표들을 제외한 나머지 좌표들을 생략하는 것, 데이터를 충분히 잘 표현하는 범위내에서 차원을 축소하는 것

SVD

: 행렬 A 자체에 대해서 분해를 수행하여  UΛVt (Vt는 행렬 V의 전치행렬)로 표현하고 Λ는 U와 Vt에 대한 eigenvalues의 양의 제곱근으로 이루어져 있는데 마찬가지로 eigenvalue가 낮은 애들부터 순차적으로 제거해가는, 즉 PCA처럼 상위 n개의 eigenvalue에 해당하는 eigenvector만을 추려내어 A를 다시 표현