AI6102: Machine Learning Methodologies & Applications

L11: Clustering Ensemble &Dimensionality Reduction

Sinno Jialin Pan

Nanyang Technological University, Singapore

Homepage: http://www.ntu.edu.sg/home/sinnopan

Outline

- Ensemble learning for clustering
- Dimensionality reduction
  - Feature selection
  - Feature extraction
    - Principal component analysis (PCA)

Ensemble Learning for Clustering

- Clustering ensembles:
  - Given an unlabeled data set $D = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_N\}$
  - A set of clustering solutions $\{f_1, f_2, \dots, f_T\}$, each of which assigns data to a set of clusters
  - An ensemble learner: a unified clustering solution f_M which combines base clustering solutions by their consensus

Cluster Ensembles

Index of a cluster

4 base clusterings

Ensemble clustering

instances

	f_1	f_2	f_3	f_4	f_M
$\boldsymbol{x}_1$	1	2	3	1	1
$\boldsymbol{x}_2$	1	3	3	3	1
$\boldsymbol{x}_3$	2	3	1	3	1
$\boldsymbol{x}_4$	2	2	1	4	2
$\boldsymbol{x}_5$	2	2	1	4	2
$\boldsymbol{x}_6$	3	1	2	2	3
$\boldsymbol{x}_7$	3	1	2	2	3

Challenges

- Unsupervised
  - No ground-truth
- The correspondence between the clusters in different clustering solutions is unknown
- Combinatorial optimization problem is NP-complete

The problem is at least as hard as any problem in **NP** (nondeterministic polynomial time), which is not solvable in polynomial time

Challenges (cont.)

Identical clustering results: $\{\{\boldsymbol{x}_1, \boldsymbol{x}_2\}, \{\boldsymbol{x}_3, \boldsymbol{x}_4, \boldsymbol{x}_5\}, \{\boldsymbol{x}_6, \boldsymbol{x}_7\}\}$

	f_1	f_2	f_3	f_4
$\boldsymbol{x}_1$	1	2	3	1
$\boldsymbol{x}_2$	1	3	3	3
$\boldsymbol{x}_3$	2	3	1	3
$\boldsymbol{x}_4$	2	2	1	4
$\boldsymbol{x}_5$	2	2	1	4
$\boldsymbol{x}_6$	3	1	2	2
$\boldsymbol{x}_7$	3	1	2	2

Numbers of clusters in different base clusterings can be different

They may not represent the same cluster!

A Similarity-based Method

- Input: data set $D = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_N\}$
  a set of clustering solutions: $\{f_1, f_2, \dots, f_T\}$
  a base clustering algorithm f for generating ensemble result
- Process:
  1. For $i = 1, \dots, T$
  2. Form a base clustering from D with f_i
  3. Derive an $N \times N$ similarity matrix $\mathbf{M}_{\boldsymbol{i}}$
  4. End
  5. Form the consensus similarity matrix $\mathbf{M} = \frac{1}{T}\sum_{i=1}^{T} \mathbf{M}_i$
  6. Perform f on $\mathbf{M}$ to generate K clusters
- Output: ensemble clustering results obtained by f

Similarity Matrix Construction

If $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ belong to the same cluster, then 1, otherwise 0.

	f_1
$\boldsymbol{x}_1$	1
$\boldsymbol{x}_2$	1
$\boldsymbol{x}_3$	2
$\boldsymbol{x}_4$	2
$\boldsymbol{x}_5$	2
$\boldsymbol{x}_6$	3
$\boldsymbol{x}_7$	3

Clustering

$\mathbf{M_1}$

	$\boldsymbol{x}_1$	$\boldsymbol{x}_2$	$\boldsymbol{x}_3$	$\boldsymbol{x}_4$	$\boldsymbol{x}_5$	$\boldsymbol{x}_6$	$\boldsymbol{x}_7$
$\boldsymbol{x}_1$	$\boxed{1}$	1	0	0	0	0	0
$\boldsymbol{x}_2$	1	1	0	0	0	0	0
$\boldsymbol{x}_3$	0	0	1	1	1	0	0
$\boldsymbol{x}_4$	0	0	1	1	1	0	0
$\boldsymbol{x}_5$	0	0	1	1	1	0	0
$\boldsymbol{x}_6$	0	0	0	0	0	1	1
$\boldsymbol{x}_7$	0	0	0	0	0	1	1

Consensus Similarity Matrix

	f_1
x_1	1
x_2	1
x_3	2
x_4	2
x_5	2
x_6	3
x_7	3

$\Rightarrow$

M_1

	x_1	x_2	x_3	x_4	x_5	x_6	x_7
x_1	1	1	0	0	0	0	0
x_2	1	1	0	0	0	0	0
x_3	0	0	1	1	1	0	0
x_4	0	0	1	1	1	0	0
x_5	0	0	1	1	1	0	0
x_6	0	0	0	0	0	1	1
x_7	0	0	0	0	0	1	1

	f_2
x_1	2
x_2	3
x_3	3
x_4	2
x_5	2
x_6	1
x_7	1

$\Rightarrow$

M_2

	x_1	x_2	x_3	x_4	x_5	x_6	x_7
x_1	1	0	0	1	1	0	0
x_2	0	1	1	0	0	0	0
x_3	0	1	1	0	0	0	0
x_4	1	0	0	1	1	0	0
x_5	1	0	0	1	1	0	0
x_6	0	0	0	0	0	1	1
x_7	0	0	0	0	0	1	1

	f_3
x_1	3
x_2	3
x_3	1
x_4	1
x_5	1
x_6	2
x_7	2

$\Rightarrow$

M_3

	x_1	x_2	x_3	x_4	x_5	x_6	x_7
x_1	1	1	0	0	0	0	0
x_2	1	1	0	0	0	0	0
x_3	0	0	1	1	1	0	0
x_4	0	0	1	1	1	0	0
x_5	0	0	1	1	1	0	0
x_6	0	0	0	0	0	1	1
x_7	0	0	0	0	0	1	1

	f_4
x_1	1
x_2	3
x_3	3
x_4	4
x_5	4
x_6	2
x_7	2

$\Rightarrow$

M_4

	x_1	x_2	x_3	x_4	x_5	x_6	x_7
x_1	1	0	0	0	0	0	0
x_2	0	1	1	0	0	0	0
x_3	0	1	1	0	0	0	0
x_4	0	0	0	1	1	0	0
x_5	0	0	0	1	1	0	0
x_6	0	0	0	0	0	1	1
x_7	0	0	0	0	0	1	1

Consensus Similarity Matrix

$\mathbf{M_1}$

	x_1	x_2	x_3	x_4	x_5	x_6	x_7
x_1	1	1	0	0	0	0	0
x_2	1	1	0	0	0	0	0
x_3	0	0	1	1	1	0	0
x_4	0	0	1	1	1	0	0
x_5	0	0	1	1	1	0	0
x_6	0	0	0	0	0	1	1
x_7	0	0	0	0	0	1	1

$\mathbf{M_2}$

	x_1	x_2	x_3	x_4	x_5	x_6	x_7
x_1	1	0	0	1	1	0	0
x_2	0	1	1	0	0	0	0
x_3	0	1	1	0	0	0	0
x_4	1	0	0	1	1	0	0
x_5	1	0	0	1	1	0	0
x_6	0	0	0	0	0	1	1
x_7	0	0	0	0	0	1	1

$\mathbf{M_3}$

	x_1	x_2	x_3	x_4	x_5	x_6	x_7
x_1	1	1	0	0	0	0	0
x_2	1	1	0	0	0	0	0
x_3	0	0	1	1	1	0	0
x_4	0	0	1	1	1	0	0
x_5	0	0	1	1	1	0	0
x_6	0	0	0	0	0	1	1
x_7	0	0	0	0	0	1	1

$\mathbf{M_4}$

	x_1	x_2	x_3	x_4	x_5	x_6	x_7
x_1	1	0	0	0	0	0	0
x_2	0	1	1	0	0	0	0
x_3	0	1	1	0	0	0	0
x_4	0	0	0	1	1	0	0
x_5	0	0	0	1	1	0	0
x_6	0	0	0	0	0	1	1
x_7	0	0	0	0	0	1	1

$$\mathbf{M} = \frac{1}{4}\left((\mathbf{M}_1 + \mathbf{M}_2 + \mathbf{M}_3 + \mathbf{M}_4\right)$$

	x_1	x_2	x_3	x_4	x_5	x_6	x_7
x_1	1	$\frac{1}{2}$	0	$\frac{1}{4}$	$\frac{1}{4}$	0	0
x_2	$\frac{1}{2}$	1	$\frac{1}{2}$	0	0	0	0
x_3	0	$\frac{1}{2}$	1	$\frac{1}{2}$	$\frac{1}{2}$	0	0
x_4	$\frac{1}{4}$	0	$\frac{1}{2}$	1	1	0	0
x_5	$\frac{1}{4}$	0	$\frac{1}{2}$	1	1	0	0
x_6	0	0	0	0	0	1	1
x_7	0	0	0	0	0	1	1

Soft Clustering

Each cluster C_l

$$\mathbf{M}_1(i,j) = \sum_{l=1}^{3} P(C_l|\boldsymbol{x}_i) \times P(C_l|\boldsymbol{x}_j)$$

f_1

$P(C_1|\boldsymbol{x}_2)$

	1	2	3
$\boldsymbol{x}_1$	1	0	0
$\boldsymbol{x}_2$	0.5	0.5	0
$\boldsymbol{x}_3$	0.33	0.34	0.33
$\boldsymbol{x}_4$	0.25	0.5	0.25
$\boldsymbol{x}_5$	0.6	0.2	0.2
$\boldsymbol{x}_6$	0.4	0.4	0.2
$\boldsymbol{x}_7$	0	1	0

$\mathbf{M}_1$

	$\boldsymbol{x}_1$	$\boldsymbol{x}_2$	$\boldsymbol{x}_3$	$\boldsymbol{x}_4$	$\boldsymbol{x}_5$	$\boldsymbol{x}_6$	$\boldsymbol{x}_7$
$\boldsymbol{x}_1$	1						
$\boldsymbol{x}_2$	0.5	1		0.375			
$\boldsymbol{x}_3$							
$\boldsymbol{x}_4$							
$\boldsymbol{x}_5$							
$\boldsymbol{x}_6$							
$\boldsymbol{x}_7$							

$0.5 \times 0.25 + 0.5 \times 0.5 + 0 \times 0.25 = 0.375$

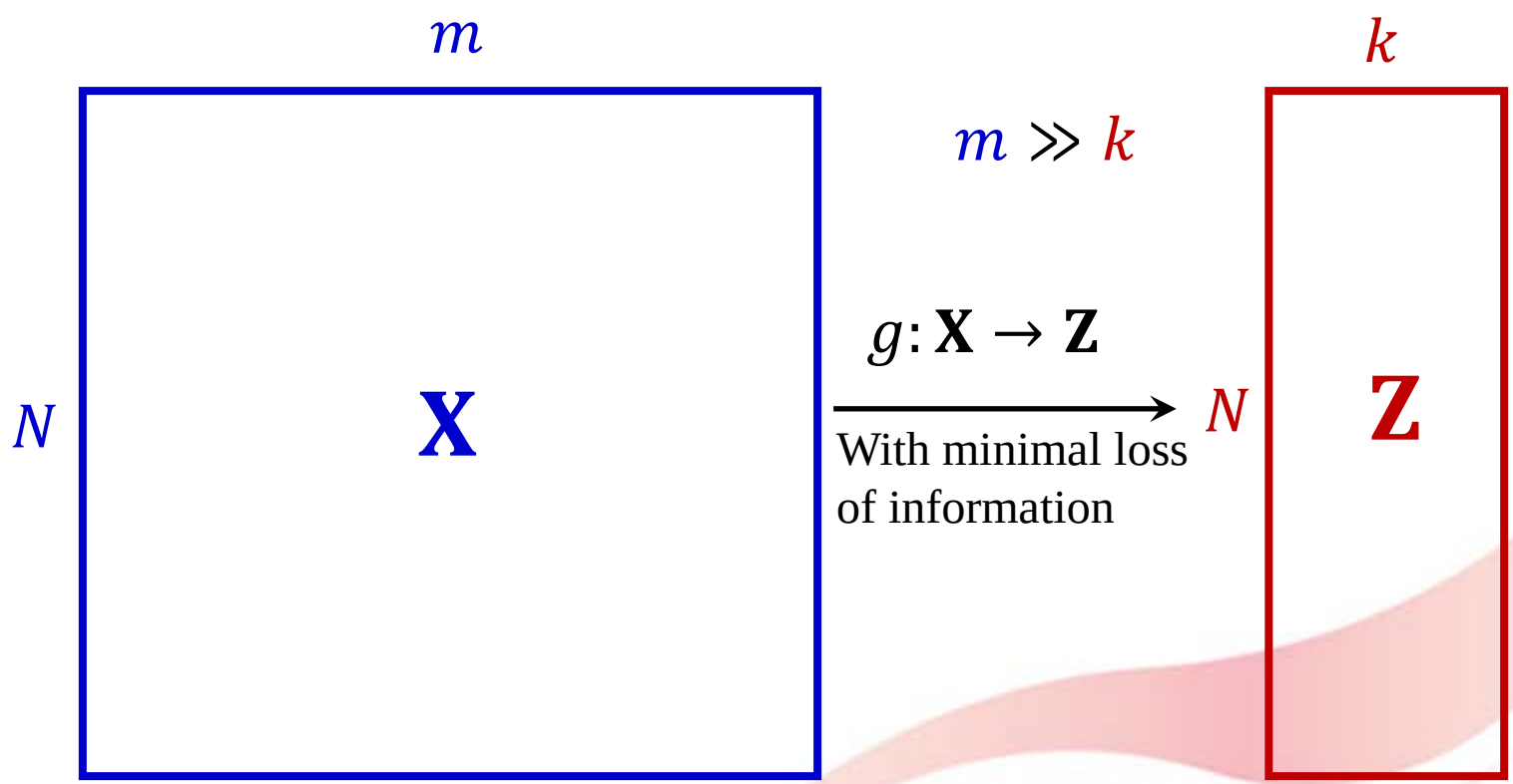
$1 \times 0.5 + 0 \times 0.5 + 0 \times 0 = 0.5$

Outline

- Ensemble learning for clustering
- Dimensionality reduction
  - Feature selection
  - Feature extraction

High-level Idea

- To summarize observed high-dimensional data instances with low-dimensional vectors



Why Dimensionality Reduction

- Avoid curse of dimensionality
  - Distance-based methods, e.g., *K*NN classifiers
- Reduce amount of time and memory for downstream machine learning tasks
- Make visualization possible
  - 2D or 3D
- Help to reduce noise, and thus improve the performance of downstream machine learning tasks

Dim Reduction Approaches

- Feature Selection
  - To select a subset of k features from the original m features to represent each data instance
- Feature Extraction
  - To extract or learn k new features from the original m features to represent each data instance

Feature Selection

- To select a subset of k features from the original m features to represent each data instance
  - Brute-force approach
    - Try all possible feature subsets and select the best subset based on some criterion
  - Greedy search
    - Start from an empty set and iteratively add the “best” features to the set based on some criterion
    - Start from the whole set of features and iteratively remove the “worst” features from the set based on some criterion

Feature Extraction

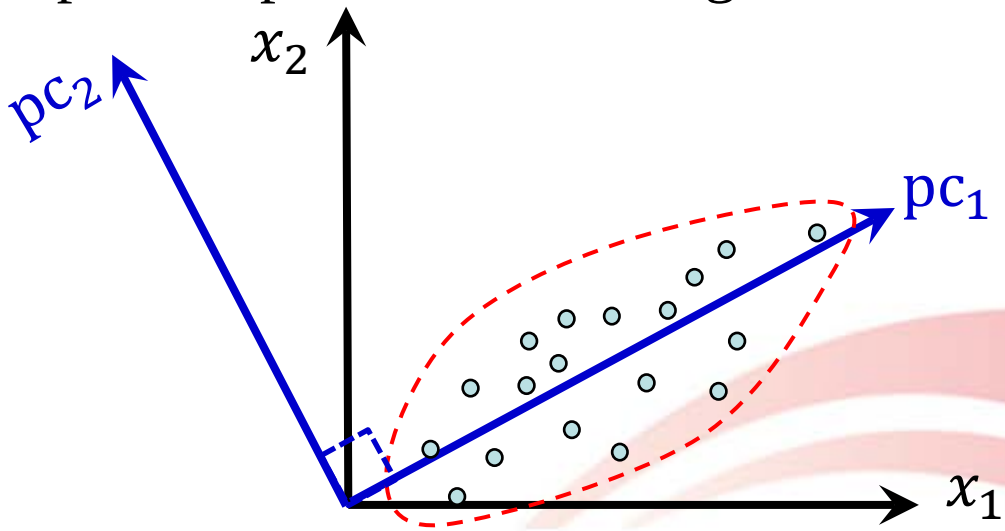
- To extract or learn k new features from the original m features to represent each data instance
  - Linear combination of original features
    - Principal component analysis
  - Nonlinear combination of original features

Principal Component Analysis

- One of the most widely-used (unsupervised) dimensionality reduction methods
- Takes a data matrix of N data points by m features, and summarizes it by principal components that are linear combinations of the original m features
- The first k components display as much as possible of the variation among data instances

Geometric Rationale

- Goal: to find a projection or rotation of the original m-dimensional coordinate system to capture the largest amount of variation in data
  - Ordered s.t. the 1st principal component has the highest variance, the 2nd component has the next highest variance, …, the m-th component has the lowest variance
  - Principal components are orthogonal to each other



PCA Algorithm

Input: $\mathcal{D} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_N\}$ a set of observed data

1. Centering the data instances s.t. the mean is 0

$$\widehat{\boldsymbol{\mu}} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{x}_i \longrightarrow \boldsymbol{x}_i = \boldsymbol{x}_i - \widehat{\boldsymbol{\mu}}$$

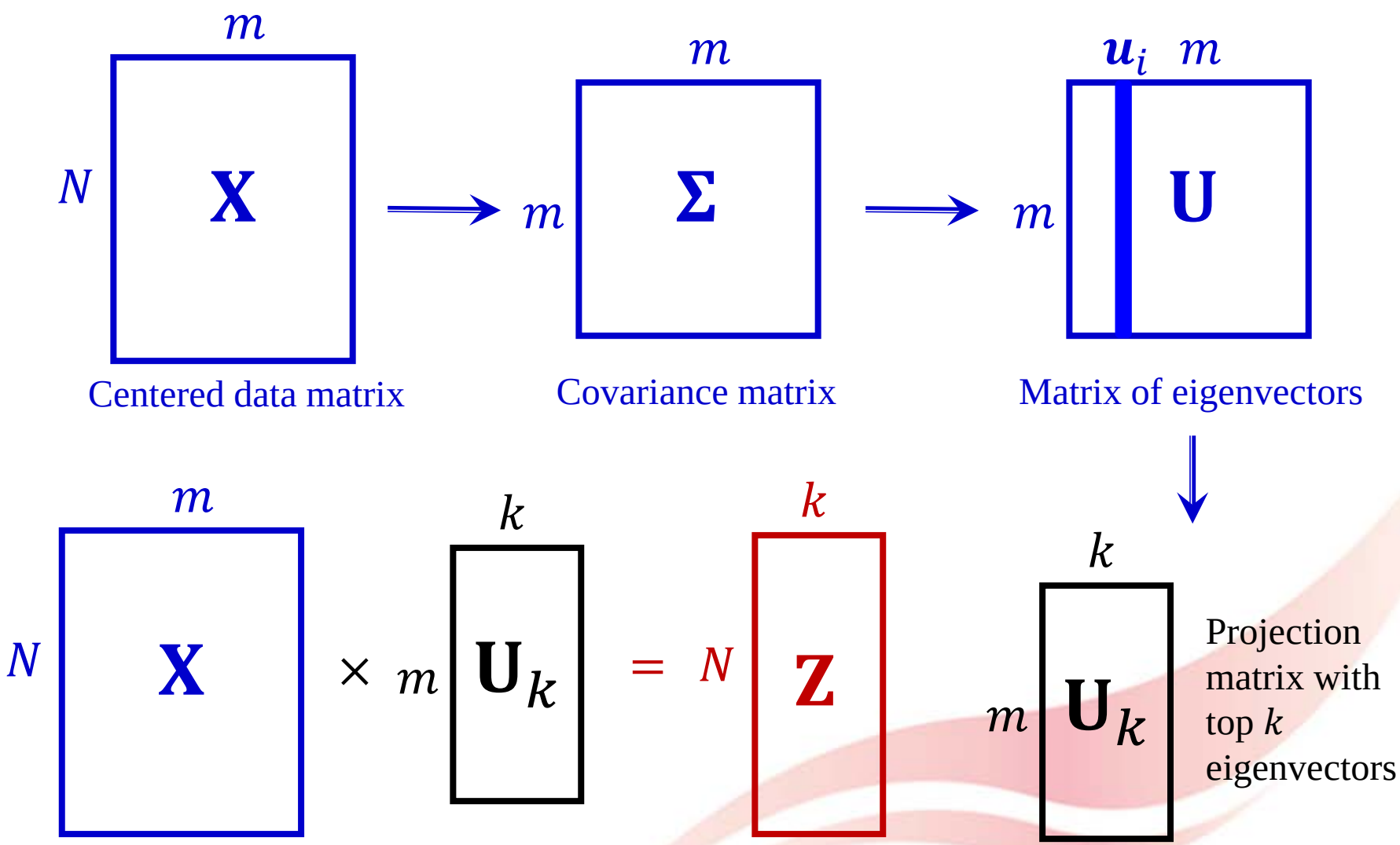
2. Compute sample covariance matrix

$$\widetilde{\boldsymbol{\Sigma}} = \frac{1}{N-1}\sum_{i=1}^{N} \boldsymbol{x}_i \boldsymbol{x}_i^T$$

Each $\boldsymbol{u}_i$ is of m dimensions

3. Compute eigenvectors of $\widetilde{\boldsymbol{\Sigma}}$: $\boxed{\{\boldsymbol{u}_1, \boldsymbol{u}_2, \dots, \boldsymbol{u}_m\}}$, which are sorted based on their eigenvalues in non-increasing order, i.e., $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_m$
4. Select the first k eigenvectors to construct principal components

Illustration



Derivation of PCA

- The variance preservation view
  - The first k components display as much as possible of the variation among data instances
- The minimum reconstruction view
  - The first k components convey maximum useful information of original data instances

Eigenvalues & Eigenvectors

- Given a m-by-m square matrix $\mathbf{A}$, if there exist a non-zero m-dimensional column vector $\boldsymbol{u}$, s.t.

scalar

$$\mathbf{A}\boldsymbol{u} = \boxed{\lambda}\boldsymbol{u}$$

then $\boldsymbol{u}$ is the eigenvector, and λ is called the corresponding eigenvalue

- Notes:
  - There are m eigenvectors and eigenvalues
  - An eigenvalue can be positive, negative or zero
  - An eigenvector cannot be a zero vector
  - Eigenvectors are orthogonal to each other

Orthogonal Vectors

- Two vectors $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ are said to be orthogonal if they are perpendicular to each other, i.e., the inner or dot product of two vectors is 0
  - $\boldsymbol{v}_1 \cdot \boldsymbol{v}_2 = 0$
- A set of vectors $\{\boldsymbol{v}_1, \dots, \boldsymbol{v}_k\}$ are mutually orthogonal if every pair of vectors is orthogonal
  - $\boldsymbol{v}_i \cdot \boldsymbol{v}_j = 0$, for any $i \neq j$

$$\boldsymbol{v}_1 = \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix} \quad \boldsymbol{v}_2 = \begin{pmatrix} 1 \\ \sqrt{2} \\ 1 \end{pmatrix} \quad \boldsymbol{v}_3 = \begin{pmatrix} 1 \\ -\sqrt{2} \\ 1 \end{pmatrix}$$

$$\boldsymbol{v}_1 \cdot \boldsymbol{v}_2 = \boldsymbol{v}_1 \cdot \boldsymbol{v}_3 = \boldsymbol{v}_2 \cdot \boldsymbol{v}_3 = 0$$

Orthonormal Vectors

- A set of vectors $\{v_1, \dots, v_k\}$ are mutually orthonormal if every pair of vectors is orthogonal, and the L_2 norm of each vector is 1
  - $\boldsymbol{v}_i \cdot \boldsymbol{v}_j = 0$, for any $i \neq j$
  - $\|\boldsymbol{v}_i\|_2 = \sqrt{\boldsymbol{v}_i \cdot \boldsymbol{v}_i} = 1$
- A set of orthogonal vectors $\{\boldsymbol{v}_1, \dots, \boldsymbol{v}_k\}$ can be normalized to orthonormal via $\left\{\frac{\boldsymbol{v}_1}{\|\boldsymbol{v}_1\|_2}, \dots, \frac{\boldsymbol{v}_d}{\|\boldsymbol{v}_k\|_2}\right\}$

$$\boldsymbol{v}_1 = \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix} \quad \boldsymbol{v}_2 = \begin{pmatrix} 1 \\ \sqrt{2} \\ 1 \end{pmatrix} \quad \boldsymbol{v}_3 = \begin{pmatrix} 1 \\ -\sqrt{2} \\ 1 \end{pmatrix} \longrightarrow \begin{array}{l} \|\boldsymbol{v}_1\|_2 = \sqrt{2} \\ \|\boldsymbol{v}_2\|_2 = 2 \\ \|\boldsymbol{v}_3\|_2 = 2 \end{array}$$

$$\boldsymbol{v}_1' = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix} \quad \boldsymbol{v}_2' = \frac{1}{2}\begin{pmatrix} 1 \\ \sqrt{2} \\ 1 \end{pmatrix} \quad \boldsymbol{v}_3' = \frac{1}{2}\begin{pmatrix} 1 \\ -\sqrt{2} \\ 1 \end{pmatrix} \longrightarrow \begin{array}{l} \|\boldsymbol{v}_1'\|_2 = 1 \\ \|\boldsymbol{v}_2'\|_2 = 1 \\ \|\boldsymbol{v}_3'\|_2 = 1 \end{array}$$

Orthonormal Vectors (cont.)

- Given a matrix $\mathbf{V} = (\boldsymbol{v}_1, \dots, \boldsymbol{v}_k)$, where $\boldsymbol{v}_i$ is a m-dimensional column vector, and $m \geq k$
- If the columns of $\mathbf{V}$ is orthonormal, then we have

$$\mathbf{V}^T\mathbf{V} = \mathbf{I}_d$$

$$\mathbf{I}_d = \begin{array}{c} \overbrace{\hphantom{1 \quad 0 \quad \cdots \quad 0}}^{d} \\ \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 1 \end{pmatrix} \end{array} \left.\vphantom{\begin{pmatrix} 1 \\ 0 \\ \cdots \\ 0 \end{pmatrix}}\right\} d$$

Orthonormal Basis

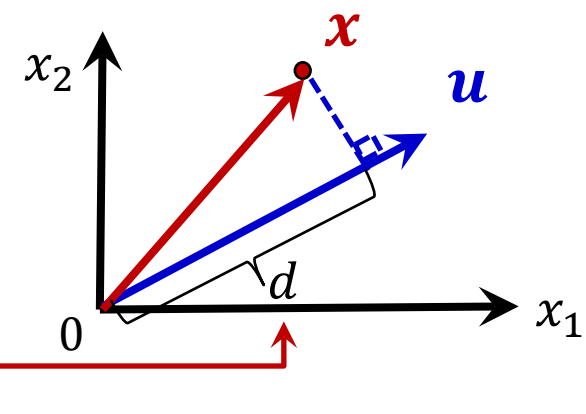
- Given a set of m-dimensional vectors $\{\boldsymbol{v}_1, \dots, \boldsymbol{v}_m\}$, if $\{\boldsymbol{v}_1, \dots, \boldsymbol{v}_m\}$ are orthonormal, then $\{\boldsymbol{v}_1, \dots, \boldsymbol{v}_m\}$ are referred to as an orthonormal basis for a m-dimensional (Euclidean) space
- For any m-dimensional vector $\boldsymbol{x}$, we have

$$\boldsymbol{x}_i = \sum_{j=1}^{m} \langle \boldsymbol{x}, \boldsymbol{v}_j \rangle \boldsymbol{v}_j$$

Variance Preservation

- Given a vector, represented by $\boldsymbol{u}$, going through the origin, the length of the projection of a data instance $\boldsymbol{x}$ onto $\boldsymbol{u}$ is

$$\frac{\boldsymbol{u}^T\boldsymbol{x}}{\|\boldsymbol{u}\|_2}$$



- This is because

$$\boldsymbol{u}\cdot\boldsymbol{x} = \boldsymbol{u}^T\boldsymbol{x} = \|\boldsymbol{u}\|_2 \times \boxed{\|\boldsymbol{x}\|_2 \times \cos(\theta)}$$

the length of the projection of $\boldsymbol{x}$ onto $\boldsymbol{u}$

$$\frac{\boldsymbol{u}^T\boldsymbol{x}}{\|\boldsymbol{u}\|_2} = d$$

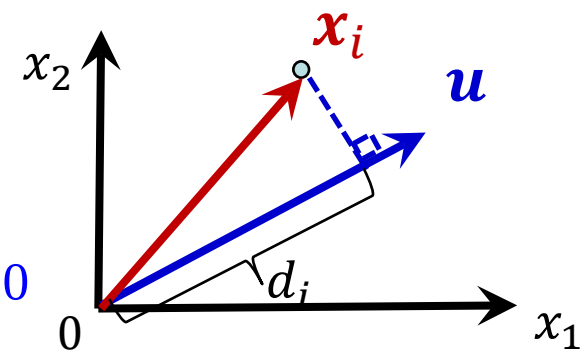
- If $\boldsymbol{u}$ is of unit length, i.e., $\|\boldsymbol{u}\|_2 = 1$, then $\boldsymbol{u}^T\boldsymbol{x} = d$

Variance Preservation (cont.)

- Given $\mathcal{D} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$ with mean being 0: $\boldsymbol{x}_i = \boldsymbol{x}_i - \widehat{\boldsymbol{\mu}}$
- For each data instance $\boldsymbol{x}_i$, the length of its projection onto $\boldsymbol{u}$ is

$$\boldsymbol{u}^T \boldsymbol{x}_i = d_i$$

- On the other hand, the variance of the data instances projected onto $\boldsymbol{u}$ is



$$\frac{1}{N-1}\sum_{i=1}^{N}(\boxed{u_i} - \boxed{0})^2$$

The mean is 0

The coordinate of $\boldsymbol{x}_i$ onto $\boldsymbol{u}$

$$\Downarrow$$

$$\frac{1}{N-1}\sum_{i=1}^{N} d_i^2 = \frac{1}{N-1}\sum_{i=1}^{N}(\boldsymbol{u}^T \boldsymbol{x}_i)^2 = \boldsymbol{u}^T \widetilde{\boldsymbol{\Sigma}} \boldsymbol{u}$$

$$\widetilde{\boldsymbol{\Sigma}} = \frac{1}{N-1}\sum_{i=1}^{N} \boldsymbol{x}_i \boldsymbol{x}_i^T$$

Variance Preservation (cont.)

- The goal of PCA (for simplicity, projected on 1D space) is to find the $\boldsymbol{u}$ that maximizes the variance, expecting it maximally preserves the distinction among data points
- The resultant optimization problem is

$$\max_{\boldsymbol{u}} \quad \boldsymbol{u}^T \widetilde{\boldsymbol{\Sigma}}\, \boldsymbol{u}$$
$$\text{s.t.} \quad \|\boldsymbol{u}\|_2 = 1$$

- It can be solved by forming the Lagrangian

$$\boldsymbol{u}^T \widetilde{\boldsymbol{\Sigma}} \boldsymbol{u} + \lambda(1 - \boldsymbol{u}^T \boldsymbol{u})$$

- By setting the gradient w.r.t. $\boldsymbol{u}$ to zero, we have

$$2\widetilde{\boldsymbol{\Sigma}}\boldsymbol{u} - 2\lambda\boldsymbol{u} = \boldsymbol{0} \longrightarrow \boxed{\widetilde{\boldsymbol{\Sigma}}\boldsymbol{u} = \lambda\boldsymbol{u}}$$

The desired direction $\boldsymbol{u}$ is an eigenvector of $\widetilde{\boldsymbol{\Sigma}}$

$\widetilde{\boldsymbol{\Sigma}}$ has m eigenvectors, which one?

Variance Preservation (cont.)

- Recall that the variance of the projected dataset $\mathcal{D} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_N\}$ is $\boldsymbol{u}^T \widetilde{\boldsymbol{\Sigma}}\, \boldsymbol{u}$
- By substituting $\widetilde{\boldsymbol{\Sigma}}\boldsymbol{u} = \lambda \boldsymbol{u}$ into the above formula, the projected variance becomes

$\|\boldsymbol{u}\|_2^2 \quad (\|\boldsymbol{u}\|_2 = 1)$

$$\boldsymbol{u}^T \widetilde{\boldsymbol{\Sigma}}\, \boldsymbol{u} = \boldsymbol{u}^T \lambda \boldsymbol{u} = \lambda \boxed{\boldsymbol{u}^T \boldsymbol{u}} = \lambda$$

- To find a direction that maximizes the projected variance is to find the eigenvector $\boldsymbol{u}$ of $\widetilde{\boldsymbol{\Sigma}}$ with the largest eigenvalue
- Generalized to multivariate case: let $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_m \geq 0$ be the eigenvalues of $\widetilde{\boldsymbol{\Sigma}}$, and $\boldsymbol{u}_1, \boldsymbol{u}_2, \dots, \boldsymbol{u}_m$ be the corresponding eigenvectors, and choose the top k eigenvectors as the principal components

Derivation of PCA (cont.)

- The variance preservation view
  - The first k components display as much as possible of the variation among data instances
- The minimum reconstruction view
  - The first k components convey maximum useful information of original data instances

Minimum Reconstruction Error

- Given any <u>orthonormal</u> basis $\boldsymbol{u}_1, \boldsymbol{u}_2, \dots, \boldsymbol{u}_m$, a data point $\boldsymbol{x}_i$ (has been centered) can be written as

$$\boldsymbol{x}_i = \sum_{j=1}^{m} \boxed{\alpha_{ij}} \boldsymbol{u}_j \qquad \alpha_{ij} = \langle \boldsymbol{x}_i, \boldsymbol{u}_j \rangle = \boldsymbol{u}_j^T \boldsymbol{x}_i$$

- Consider the k-term approximation of $\boldsymbol{x}_i$:

$$\hat{\boldsymbol{x}}_i \approx \sum_{j=1}^{k} \alpha_{ij} \boldsymbol{u}_j$$

- The error of the approximate over all data points is

$$E = \frac{1}{N} \sum_{i=1}^{N} \|\hat{\boldsymbol{x}}_i - \boldsymbol{x}_i\|_2^2 = \frac{1}{N} \sum_{i=1}^{N} \left\| \sum_{j=k+1}^{m} \alpha_{ij} \boldsymbol{u}_j \right\|_2^2 = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=k+1}^{m} \alpha_{ij}^2$$

Minimum Reconstruction Error (cont.)

- The error of the approximate over all data points

$$E = \frac{1}{N}\sum_{i=1}^{N}\|\widehat{\boldsymbol{x}}_i - \boldsymbol{x}_i\|_2^2 = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=k+1}^{m}\alpha_{ij}^2$$

$$= \frac{1}{N}\sum_{i=1}^{N}\sum_{j=k+1}^{m}\boldsymbol{u}_j^T\boldsymbol{x}_i{\boldsymbol{x}_i}^T\boldsymbol{u}_j \approx \sum_{j=k+1}^{m}\boldsymbol{u}_j^T\widetilde{\boldsymbol{\Sigma}}\boldsymbol{u}_j$$

- Suppose $k = m - 1$, i.e., we aim to remove a single dimension, then resultant optimization problem is

$$\begin{aligned} \min_{\boldsymbol{u}} \quad & \boldsymbol{u}^T\widetilde{\boldsymbol{\Sigma}}\boldsymbol{u} \\ \text{s.t.} \quad & \|\boldsymbol{u}\|_2 = 1 \end{aligned}$$

Minimum Reconstruction Error (cont.)

- By setting the gradient of the Lagrangian w.r.t. $\boldsymbol{v}$ to zero, we have

$$2\widetilde{\boldsymbol{\Sigma}}\boldsymbol{u} - 2\lambda\boldsymbol{u} = \boldsymbol{0} \longrightarrow \boxed{\widetilde{\boldsymbol{\Sigma}}\boldsymbol{u} = \lambda\boldsymbol{u}}$$

The desired direction $\boldsymbol{u}$ is an eigenvector of $\widetilde{\boldsymbol{\Sigma}}$ with a corresponding eigenvalue λ

$\widetilde{\boldsymbol{\Sigma}}$ has m eigenvectors, which one?

- Our goal is to minimize the reconstruction error $\boldsymbol{u}^T\widetilde{\boldsymbol{\Sigma}}\boldsymbol{u}$

$$\boldsymbol{u}^T\widetilde{\boldsymbol{\Sigma}}\boldsymbol{u} = \boldsymbol{u}^T\lambda\boldsymbol{u} = \lambda\boldsymbol{u}^T\boldsymbol{u} = \lambda$$

- Therefore, $\boldsymbol{u}$ should be the eigenvector of $\widetilde{\boldsymbol{\Sigma}}$ with the smallest eigenvalue
- Similarly, the other dimensions to remove are subsequently the eigenvectors corresponding to the least eigenvalues

Determine the Value of k

- Wrapper approaches
  - Dimensionality reduction is usually an intermedia step for some final tasks, such as classification, regression, clustering
  - Use cross-validation based on the performance of the final task to tune the value of k
- Based on the percentage of variance preserved

$$p_{\text{var}} = \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{m} \lambda_i} \times 100$$

  - Predefine a value for the percentage of variance to determine the value of k

How to Obtain Eigenvectors

- If m-by-m square matrix $\mathbf{A}$ can be written as

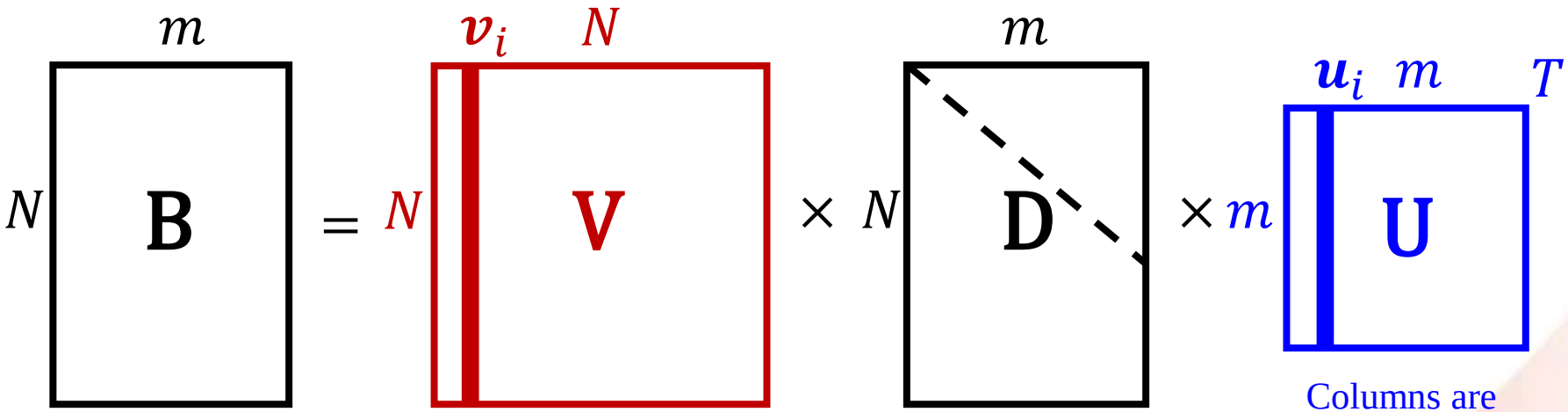
$$\mathbf{A} = \mathbf{B}^T\mathbf{B}, \text{ where } \mathbf{B} \text{ is } N\text{-by-}m$$

  then all the eigenvalues of $\mathbf{A}$ are non-negative
- The eigenvectors of $\mathbf{A} = \mathbf{B}^T\mathbf{B}$ can be computed by performing Singular Value Decomposition (SVD) on $\mathbf{B}$

Singular Value Decomposition (SVD)

- The SVD of any N-by-m ($N > m$) matrix $\mathbf{B}$ has the form

$$\mathbf{B} = \mathbf{V}\mathbf{D}\mathbf{U}^T$$



Columns are mutually orthonormal

$$\mathbf{V}^T\mathbf{V} = \mathbf{I}_N$$

All elements are zero expect for the first m-by-m submatrix diagonal being $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_m$

Columns are mutually orthonormal

$$\mathbf{U}^T\mathbf{U} = \mathbf{I}_m$$

Eigen Vectors via SVD

m-by-m diagonal matrix with elements $\lambda_1^2 \geq \lambda_2^2 \geq \cdots \geq \lambda_m^2 \geq 0$

- Suppose $\mathbf{A} = \mathbf{B}^T\mathbf{B}$ and $\mathbf{B}$ is a N-by-m matrix
- Perform SVD on $\mathbf{B}$ via $\mathbf{B} = \mathbf{V}\mathbf{D}\mathbf{U}^T$
- Then $\mathbf{A}$ can be rewritten as

$$\widetilde{\mathbf{D}} = \mathbf{D}^T\mathbf{D} = \begin{pmatrix} \lambda_1^2 & & \\ & \cdots & \\ & & \lambda_m^2 \end{pmatrix}$$

$$\mathbf{V}^T\mathbf{V} = \mathbf{I}_N$$

$$\mathbf{A} = \mathbf{B}^T\mathbf{B} = (\mathbf{V}\mathbf{D}\mathbf{U}^T)^T\mathbf{V}\mathbf{D}\mathbf{U}^T = \mathbf{U}\mathbf{D}^T\boxed{\mathbf{V}^T\mathbf{V}}\mathbf{D}\mathbf{U}^T = \mathbf{U}\boxed{\widetilde{\mathbf{D}}}\mathbf{U}^T$$

- As $\mathbf{U}^T\mathbf{U} = \mathbf{I}_m$, we have

$$\mathbf{A}\mathbf{U} = \mathbf{U}\widetilde{\mathbf{D}}$$

$$[\mathbf{A}\boldsymbol{u}_1, \mathbf{A}\boldsymbol{u}_2, \ldots, \mathbf{A}\boldsymbol{u}_m] = [\lambda_1^2\boldsymbol{u}_1, \lambda_2^2\boldsymbol{u}_2, \ldots, \lambda_m^2\boldsymbol{u}_m]$$

$$\mathbf{A}\boldsymbol{u}_i = \lambda_i^2\boldsymbol{u}_i$$

Each column $\boldsymbol{u}_i$ of $\mathbf{U}$ is an eigenvector of $\mathbf{A}$ with an eigenvalue of λ_i^2

Therefore, eigenvectors of $\mathbf{A}$ can be obtained by performing SVD on $\mathbf{B}$

Eigen Vectors via SVD (cont.)

- Given a data matrix $\mathbf{X}$ $(N \times m)$, which has been centered, then the covariance matrix can be computed

$$\widetilde{\boldsymbol{\Sigma}} = \frac{1}{N-1}\mathbf{X}^T\mathbf{X}$$

- Perform SVD on $\mathbf{X}$ via $\mathbf{X} = \mathbf{V}\mathbf{D}\mathbf{U}^T$
- Then $\widetilde{\boldsymbol{\Sigma}}$ can be rewritten as

$$\widetilde{\boldsymbol{\Sigma}} = \frac{1}{N-1}\mathbf{X}^T\mathbf{X} = \frac{1}{N-1}(\mathbf{V}\mathbf{D}\mathbf{U}^T)^T\mathbf{V}\mathbf{D}\mathbf{U}^T$$

$$= \frac{1}{N-1}\mathbf{U}\mathbf{D}\overset{\mathbf{I}}{\boxed{\mathbf{V}^T\mathbf{V}}}\mathbf{D}\mathbf{U}^T = \mathbf{U}\widetilde{\mathbf{D}}\mathbf{U}^T$$

Denote $\widetilde{\mathbf{D}} = \frac{1}{N-1}\mathbf{D}^T\mathbf{D}$

$$\widetilde{\boldsymbol{\Sigma}}\mathbf{U} = \widetilde{\mathrm{D}}\mathbf{U}$$

Each column $\boldsymbol{u}_i$ of $\mathbf{U}$ is an eigenvector of $\widetilde{\boldsymbol{\Sigma}}$, whose eigenvalue is the i-th diagonal element in $\widetilde{\mathrm{D}}$

Further Readings

- For feature subset selection:
  - An Introduction to Variable and Feature Selection, Isabelle Guyon, Andre Elisseeff, in JMLR 2003
- For dimensionality reduction:
  - Dimensionality Reduction: A Comparative Review, L.J.P. van der Maaten and E. O. Postma and H. J. van den Herik, Technical Report, 2008

Implementation using scikit-learn

- API: sklearn.decomposition.PCA
  https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html#sklearn.decomposition.PCA

sklearn.decomposition.PCA

class `sklearn.decomposition.PCA`(*n_components=None, *, copy=True, whiten=False, svd_solver='auto', tol=0.0, iterated_power='auto', random_state=None*) [source]

Principal component analysis (PCA).

Linear dimensionality reduction using Singular Value Decomposition of the data to project it to a lower dimensional space. The input data is centered but not scaled for each feature before applying the SVD.

It uses the LAPACK implementation of the full SVD or a randomized truncated SVD by the method of Halko et al. 2009, depending on the shape of the input data and the number of components to extract.

It can also use the scipy.sparse.linalg ARPACK implementation of the truncated SVD.

Notice that this class does not support sparse input. See `TruncatedSVD` for an alternative with sparse data.

Read more in the User Guide.

Parameters:

n_components : *int, float, None or str*

Number of components to keep. if n_components is not set all components are kept:

```
n_components == min(n_samples, n_features)
```

If `n_components == 'mle'` and `svd_solver == 'full'`, Minka's MLE is used to guess the dimension. Use of `n_components == 'mle'` will interpret `svd_solver == 'auto'` as `svd_solver == 'full'`.

If `0 < n_components < 1` and `svd_solver == 'full'`, select the number of components such that the amount of variance that needs to be explained is greater than the percentage specified by n_components.

If `svd_solver == 'arpack'`, the number of components must be strictly less than the minimum of n_features and n_samples.

Hence, the None case results in:

```
n_components == min(n_samples, n_features) - 1
```

Example

```
>>> from sklearn.decomposition import PCA
>>> from sklearn.preprocessing import StandardScaler

>>> import pandas as pd
>>> X = pd.read_csv("xxxx")
>>> X_t = pd.read_csv("yyyy")

>>> sc = StandardScaler()
>>> X = sc.fit_transform(X)
>>> X_t = sc.transform(X_t)

>>> pca = PCA(n_components=2)
>>> pca.fit(X)
>>> X_pca = pca.transform(X)
>>> X_t_pca = pca.transform(X_t)
```

Thank you!