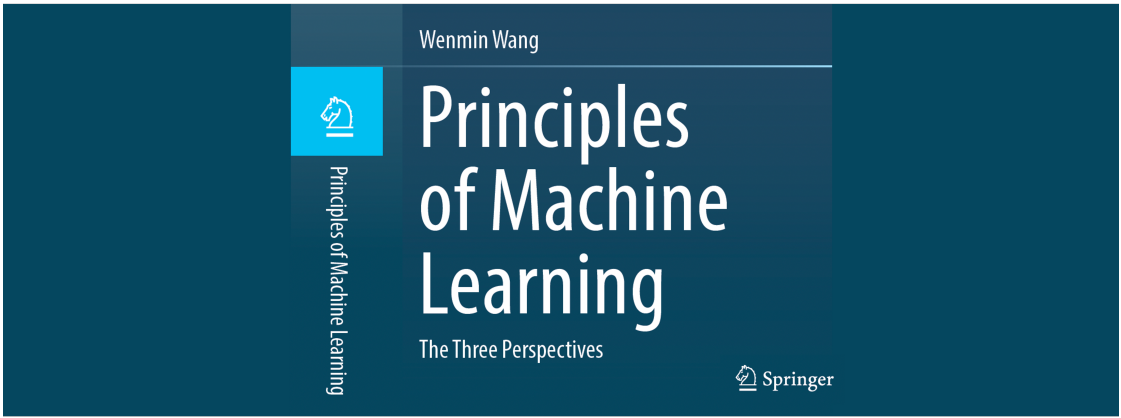


https://link.springer.com/book/10.1007/978-981-97-5333-8

Part IV Tasks

15 Dimensionality Reduction Task

15 Dimensionality Reduction Task

Dimensionality Reduction Problems

The dimensions involved in dimensionality reduction (DR) are mainly the dimensions of mathematical space and data space.

- Dimension of mathematical space:

  It refers to the minimum number of coordinates to represent any point.

  For Euclidean space, the point in one-, two-, and three-dimensional space are represented by one, two, and three coordinates.

- Dimension of data space:

  It refers to the number of features in the data, represented by vectors.

  The data is called high-dimensional if it is in high-dimensional data space.

Dimensionality Reduction Problems

Definition: (High-Dimensional Data)

Given a dataset with n data points and p features, if n is very large and $p \gg 3$, then the dataset is referred to as high-dimensional data.

What can the dimensionality reduction do:

- can be used for denoise of high-dimensional data.
- can reduce the number of features in high-dimensional data.
- can improve the accuracy of feature learning.
- can serve as a preprocessing for other machine learning tasks.

Definition

Definition: (Dimensionality Reduction)

Dimensionality reduction in machine learning is a task that maps the data in high-dimensional space to a low-dimensional space, while retaining the basic features of the original high-dimensional data.

High-dimensional data vs. Big data:

- High-dimensional data: the datasets with very large number of features.
- Big data: the datasets with very large sizes that is hard to process.

Why Dimensionality Reduction

(1) Curse of dimensionality:
as the dimensionality of data increases, the workload required for processing grows exponentially.

(2) Data sparsity:
the representation of high-dimensional data leads to inflation of useless information, with numerous meaningless features.

(3) Intrinsic dimension:
referring to the minimum number of feature required to represent data, but with numerous redundant features in the data.

(4) Data visualization:
two- or three-dimensional space can easy represent the relationships and patterns in data.

15 Dimensionality Reduction Task

Formal Description

Let $\mathbb{R}^h$ and $\mathbb{R}^l$ denote the sets of high- and low-dimensional real vectors, $h \gg l$ and $1 \leq l \leq 3$. Also, let $\mathcal{X} \subseteq \mathbb{R}^h$ and $\mathcal{Y} \subseteq \mathbb{R}^l$.

Given a dataset $D_{\text{high}} = \{\boldsymbol{x}_i | \ i = 1, 2, \ldots, m\}$, $\boldsymbol{x}_i \in \mathcal{X}$ and $D_{\text{high}} \subseteq \mathcal{X}$.

Design a hypothesis set H that maps the data in $\mathcal{X}$ into $\mathcal{Y}$:

$$H : \mathcal{X} \rightarrow \mathcal{Y} \ .$$

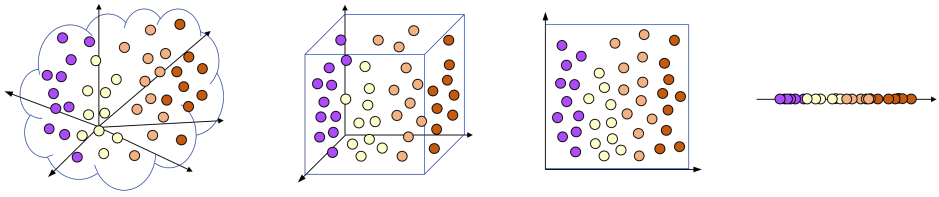
The goal is to find a best hypothesis $h \in H$, mapping D_{high} to $D_{\text{low}} \subseteq \mathcal{Y}$, i.e.,

$$h : D_{\text{high}} \rightarrow D_{\text{low}},$$

while preserving the basic properties of the high-dimensional data.

Illustrated Description

Schematic diagrams of high-dimensional and low-dimensional space.



(a) is high-dimensional space $\mathcal{X} \subseteq \mathbb{R}^h$ before dimensionality reduction, (b), (c), and (d) are three-, two-, and one-dimensional space respectively.

15 Dimensionality Reduction Task

Feature Selection and Extraction

Feature selection:

- Selecting a subset from the original feature pool.
- Filter out the irrelevant but retain the highly correlated features from data.

E.g. filter, wrapper, and embedded methods.

Feature extraction:

- Extracting useful features from existing data.
- Transform and map the features of input data to low-dimensional space.

E.g. principal component analysis (PCA), linear discriminant analysis (LDA).

Linear and Nonlinear

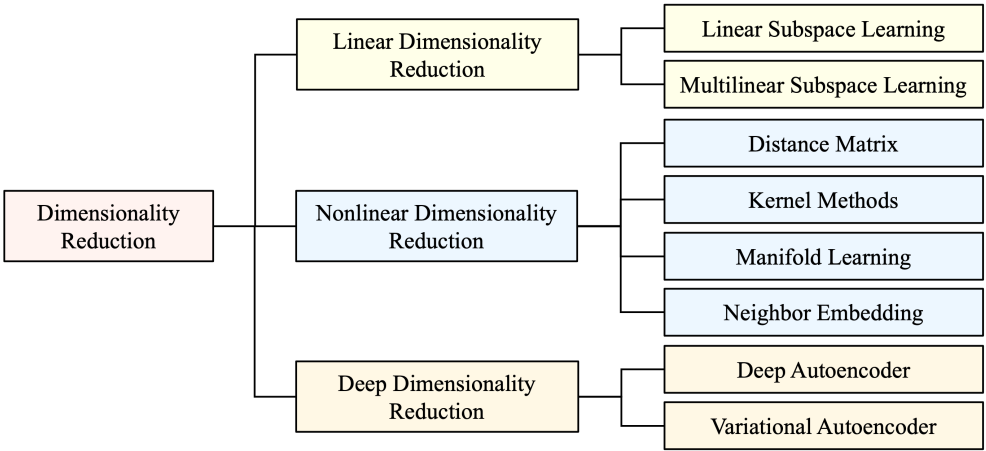
Linear dimensionality reduction:

- Using linear mapping to map the data in high-dimensional space to a low-dimensional space.
- Two subtypes: linear subspace learning, and multilinear subspace learning.

Nonlinear dimensionality reduction:

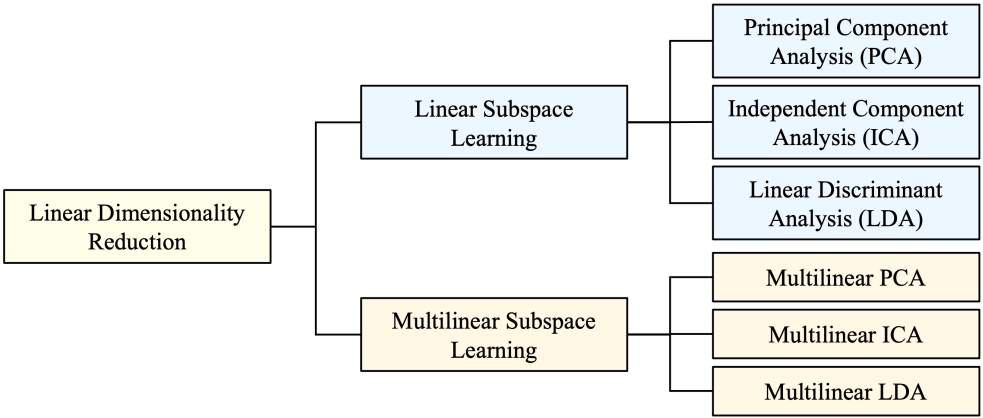
- Using nonlinear mapping to map the data in high-dimensional space to a low-dimensional space.
- Four subtypes: distance matrix, kernel methods, manifold learning, and neighborhood embedding methods.

Hierarchical Relationship



15 Dimensionality Reduction Task

Hierarchical Relationship



Linear Subspace Learning

- Principal component analysis (PCA)

  It uses an orthogonal transformation to linearly transform high-dimensional data to a new coordinate system, describing it with fewer dimensions.

- Independent component analysis (ICA)

  It decomposes high-dimensional data into independent non-Gaussian distributions, and identifies the components with least correlation with other components.

- Linear discriminant analysis (LDA)

  It finds the projection matrix in dataset, minimize the difference between same class data and maximize the difference between different class data, in low-dimensional space.

Multilinear Subspace Learning

When dealing with a large amount of high-dimensional data, linear subspace learning needs to estimate numerous parameters.

Multilinear subspace learning uses different types of data tensor analysis tools to reduce dimensions.

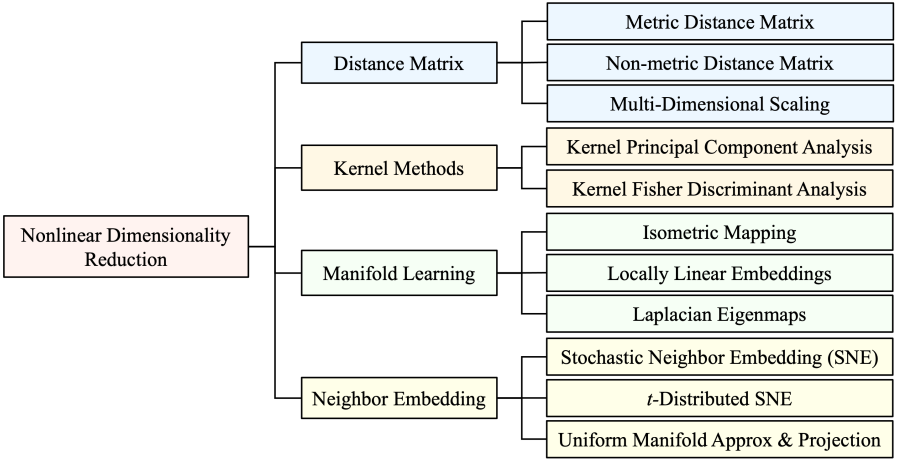
It can be used for such high-dimensional data, e.g,, its metrics are vectorized and organized into tensors, or viewed as matrices and connected into tensors.

Linear subspace learning can be generalized to multilinear space learning:

- Multilinear principal component analysis (MPCA).
- Multilinear independent component analysis (MICA).
- Multilinear linear discriminant analysis (MLDA).

15 Dimensionality Reduction Task

Hierarchical Relationship



Distance Matrix Method

Distance matrix, also known as dissimilarity matrix, is a square matrix that contains the pairwise distances between data points in the dataset.

Given n data points $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$, distance matrix $D = [d_{ij}]$, where $d_{ij} = d(\boldsymbol{x}_i, \boldsymbol{x}_j)$, $i, j = 1, \ldots, n$, and $d_{ij} = 0$ if $i = j$. It can be divided into:

- metric distance matrix,
- non-metric distance matrix,
- multidimensional scaling.

E.g., in metric distance matrix, each of non-diagonal elements represents the distance between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$.

0	$\boldsymbol{x}_1$	$\boldsymbol{x}_2$	$\boldsymbol{x}_3$	$\cdots$	$\boldsymbol{x}_n$
$\boldsymbol{x}_1$	0	d_{12}	d_{13}	$\cdots$	d_{1n}
$\boldsymbol{x}_2$	d_{21}	0	d_{23}	$\cdots$	d_{2n}
$\boldsymbol{x}_3$	d_{31}	d_{32}	0	$\cdots$	d_{3n}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$\boldsymbol{x}_n$	d_{n1}	d_{n2}	d_{n3}	$\cdots$	0

Kernel Methods

Kernel methods have been introduced in Chapter 4, can also be used in nonlinear dimensionality reduction.

Typical kernel-based algorithms for nonlinear dimensionality reduction:

- Kernel principal component analysis (kernel PCA)
  It is an extension of principal component analysis (PCA) based on kernel methods. After using kernel methods, the original linear operations of PCA run in the reproducing kernel Hilbert space.
- Kernel Fisher discriminant analysis (kernel FDA)
  It is also known as generalized discriminant analysis and kernel discriminant analysis, is an extension of Fisher discriminant analysis (FDA) based on kernel methods.

Manifold Learning

It is a type of machine learning method for nonlinear dimensionality reduction based on manifold.

Definition: (Manifold)

An n-dimensional manifold is a topological space with the property that each point has a neighborhood that is homeomorphic to an open subset of n-dimensional Euclidean space.

Manifolds in mathematics are used to describe geometric bodies and can be equipped with additional structures.

Manifold Learning

Manifold hypothesis:

many high-dimensional datasets in the real world actually lie along low-dimensional latent manifolds in high-dimensional space.

Representative algorithms:

- Isometric mapping (Isomap)
- Locally linear embedding (LLE)
- Laplacian eigenmaps (LE)
- Local tangent space alignment (LTSA)
- Inductive manifold learning
- Symmetric positive definite manifold (SPD manifold)

Neighborhood Embedding

It is a class of probabilistic methods evolved from neighborhood similarities:

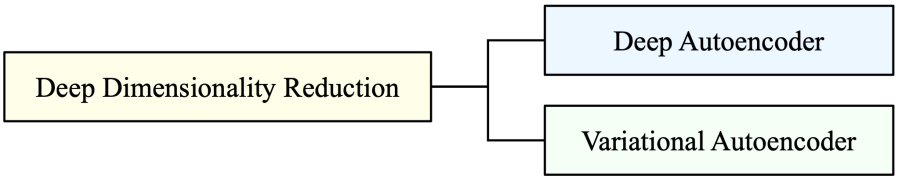
- Stochastic neighbor embedding (SNE):
  converting high-dimensional data into low-dimensional space that preserves neighbor identities, using Gaussian distribution for both of high- and low-dimensional space.
- t-distributed stochastic neighbor embedding (t-SNE):
  a variant of SNE in response to crowding problem, using t-distribution for the low-dimensional space.
- Uniform manifold approximation and projection (UMAP):
  many similarities with t-SNE in terms of data visualization, but lies in three assumptions about the data.

15 Dimensionality Reduction Task

About Deep Dimensionality Reduction

It is a class of dimensionality reduction methods based on deep learning.

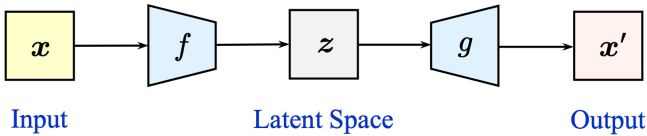
Representative algorithms for deep dimensionality reduction include:



Autoencoder vs. Variational Autoencoder

- Deep Autoencoder

  The one for dimensionality reduction is the undercomplete autoencoder (UAE). In which, the dimension of latent space is less than input and output spaces, thereby achieving dimensionality reduction.



- Variational Autoencoder

  Its latent space is composed of probability distributions rather than fixed vectors, can also be used for dimensionality reduction.

15 Dimensionality Reduction Task

Principal Component Analysis

Principal component analysis (PCA) is to map the data from high-dimensional dataset to low-dimensional linear subspace, so as to form a new coordinate system, where each coordinate axis represents a principal component (PC).

These PCs are orthogonal linear transformations of high-dimensional data, where the first PC is orthogonal axis with largest variance.

The kth row $(k = 1, \ldots, n)$ of data and linear combination of coefficients are:

$$z_k = \sum_{j=1}^{n} w_{kj} x_j = \mathbf{X} w_k,$$

where $\mathbf{X} = [x_1 x_2 \cdots x_n]$ is a matrix, and $w_k = [w_{k1} w_{k2} \cdots w_{kn}]^\mathsf{T}$ is a weight vector.

Principal Component Analysis

Let U_k be the kth principal component, its maximum variance is:

$$U_k = \max\left(\mathrm{var}\left(\boldsymbol{z}_k\right)\right) = \max\left(\mathrm{var}\left(\mathbf{X}\boldsymbol{w}_k\right)\right) = \max\left(\boldsymbol{w}_k{}^{\mathsf{T}}\mathbf{S}\boldsymbol{w}_k\right),$$

where $\mathbf{S}$ represents the $n \times n$ covariance matrix.

To maximize $\mathrm{var}\left(\boldsymbol{w}_k{}^{\mathsf{T}}\mathbf{S}\boldsymbol{w}_k\right)$, which can be achieved by increasing the value of $\boldsymbol{w}_k$, while restricting $\boldsymbol{w}_k$ to be a unit-norm vector, i.e., $\boldsymbol{w}_k{}^{\mathsf{T}}\boldsymbol{w}_k = 1$.

Therefore, the maximization of U_k can be rewritten as:

$$U_k = \max\left(\boldsymbol{w}_k{}^{\mathsf{T}}\mathbf{S}\boldsymbol{w}_k\right) = \underset{\boldsymbol{w}_k{}^{\mathsf{T}}\boldsymbol{w}_k=1}{\arg\max}\left(\boldsymbol{w}_k{}^{\mathsf{T}}\mathbf{S}\boldsymbol{w}_k - \lambda_k\left(\boldsymbol{w}_k{}^{\mathsf{T}}\boldsymbol{w}_k - 1\right)\right),$$

where λ is the Lagrange multiplier.

Principal Component Analysis

For $\boldsymbol{w}_k$ in brackets of $\arg\max(\cdot)$, differentiate it and set it to zero, that is:

$$\frac{d}{d\boldsymbol{w}_k}(\boldsymbol{w}_k{}^{\mathsf{T}}\mathbf{S}\boldsymbol{w}_k - \lambda_k(\boldsymbol{w}_k{}^{\mathsf{T}}\boldsymbol{w}_k - 1)) = 0.$$

Hence, we have: $\mathbf{S}\boldsymbol{w}_k - \lambda_k\boldsymbol{w}_k = 0$, and $\mathbf{S}\boldsymbol{w}_k = \lambda_k\boldsymbol{w}_k$, where $\boldsymbol{w}_k$ is unit norm eigenvector of $\mathbf{S}$, and λ_k is corresponding eigenvalue.

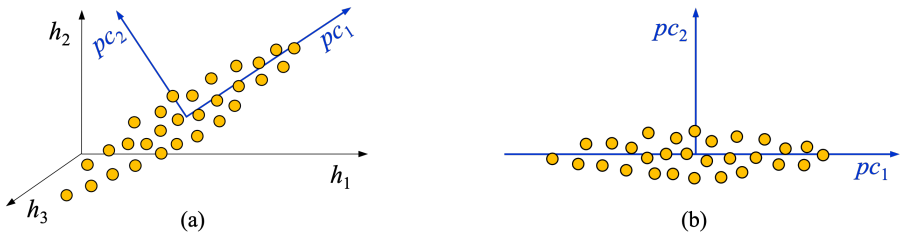
Therefore, the kth principal component is:

$$U_k = \max\left(\boldsymbol{w}_k{}^{\mathsf{T}}\mathbf{S}\boldsymbol{w}_k\right) = \max\left(\boldsymbol{w}_k{}^{\mathsf{T}}\lambda_k\boldsymbol{w}_k\right) = \max\left(\lambda_k\boldsymbol{w}_k{}^{\mathsf{T}}\boldsymbol{w}_k\right) = \max\left(\lambda_k\right).$$

That is the kth principal component U_k equals the maximum value of λ_k.

Principal Component Analysis

The first PC is $U_1 = \max(\text{var}(X\boldsymbol{w}_1))$, while second PC is $U_2 = \max(\text{var}(X\boldsymbol{w}_2))$, but the U_2 is not related to U_1, and so on.



The first two principal components are usually used to draw a two-dimensional data graph for visualizing the clusters of data points after dimension reduction.

Multi-Dimensional Scaling

Four types of multi-dimensional scaling (MDS), depending on the differences in the input matrix.

- Classic MDS: based on metric distance matrix, and preserve the distances in the dimensions after dimension reduction.
- Metric MDS: similar to classic MDS, but use iterative algorithm, dealing with ratio data and interval data at same time.
- Non-metric MDS: focus on the rank order between data points, but not the actual distance between data points.
- Generalized MDS: an extension of metric MDS, and its target space is any smooth non-Euclidean space.

Classic Multi-Dimensional Scaling

The Euclidean distances between several cities by a matrix $\mathbf{D} = [d_{ij}]$, where:

$$d_{ij} = \sqrt{(\boldsymbol{x}_i - \boldsymbol{x}_j)^2 + (\boldsymbol{y}_i - \boldsymbol{y}_j)^2}.$$

The goal is finding the coordinates between cities. Its loss function $\mathrm{Strain}_D(\cdot)$ is:

$$\mathrm{Strain}_D\left(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\right) = \left(\frac{\sum_{i,j} \left(b_{ij} - \boldsymbol{x}_i^{\mathsf{T}} \boldsymbol{x}_j\right)^2}{\sum_{i,j} b_{ij}^2} \right)^{1/2},$$

where $\boldsymbol{x}_i^{\mathsf{T}} \boldsymbol{x}_j$ is the inner product of $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, and b_{ij} is an element in matrix $\mathbf{B}$.

Classic Multi-Dimensional Scaling

The matrix $\mathbf{B}$ can be derived from eigenvalue decomposition of $\mathbf{B} = \mathbf{X}\mathbf{X}'$.

The algorithm of classical MDS is summarized as below:

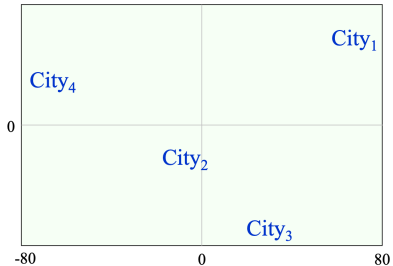
1) Set up the squared matrix: $\mathbf{D}^{(2)} = [d_{ij}]$.
2) Apply double centering: $\mathbf{B} = -\frac{1}{2}\mathbf{J}\mathbf{D}^{(2)}\mathbf{J}$, using centering matrix $\mathbf{J} = \mathbf{I} - \frac{1}{n}\mathbf{J_n}$, where $\mathbf{I}$ is an identity matrix and $\mathbf{J_n}$ is a $n \times n$ matrix of all ones.
3) Extract m largest eigenvalues $\lambda_1, \ldots, \lambda_m$, and corresponding eigenvectors $e_1, \ldots, e_m$, where m is the number of dimensions.
4) And the $\mathbf{X} = \mathbf{E}_m \mathbf{\Lambda}_m^{1/2}$, where $\mathbf{E}_m = [e_k]$ and $\mathbf{\Lambda}_m = [\lambda_k]$.

Classic Multi-Dimensional Scaling

Given the four cities City_1, City_2, City_3 and City_4,

	City_1	City_2	City_3	City_4
City_1	0	93	82	133
City_2	93	0	52	60
City_3	82	52	0	111
City_4	133	60	111	0

(a)



(b)

$n = 4$, and the distances matrix $\mathbf{D}$ between the four cities is given in (a).

Classic Multi-Dimensional Scaling

$$\mathbf{D}^{(2)} = \begin{bmatrix} 0 & 8649 & 6724 & 17689 \\ 8649 & 0 & 2704 & 3600 \\ 6724 & 2704 & 0 & 12321 \\ 17689 & 3600 & 12321 & 0 \end{bmatrix}, \quad \mathbf{J} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} - \frac{1}{4} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}.$$

Applying $\mathbf{J}$ to $\mathbf{D}^{(2)}$ results in the double-centered matrix B:

$$\mathbf{B} = -\frac{1}{2}\mathbf{J}\mathbf{D}^{(2)}\mathbf{J} = \begin{bmatrix} 5035.0625 & -1553.0625 & 258.9375 & -3740.938 \\ -1553.0625 & 507.8125 & 5.3125 & 1039.938 \\ 258.9375 & 5.3125 & 2206.8125 & -2471.062 \\ -3740.9375 & 1039.9375 & -2471.0625 & 5172.062 \end{bmatrix}.$$

Classic Multi-Dimensional Scaling

For the two-dimensional representation, $m = 2$, to extract first two largest eigenvalues and eigenvectors from $\mathbf{B}$, and then we can get $\mathbf{X}$ as below:

$$\lambda_1 = 9724.168, \quad \lambda_2 = 3160.986, \quad e_1 = \begin{pmatrix} -0.637 \\ 0.187 \\ -0.253 \\ 0.704 \end{pmatrix}, \quad e_2 = \begin{pmatrix} -0.586 \\ 0.214 \\ 0.706 \\ -0.334 \end{pmatrix}.$$

$$\mathbf{X} = \begin{bmatrix} -0.637 & -0.586 \\ 0.187 & 0.214 \\ -0.253 & 0.706 \\ 0.704 & -0.334 \end{bmatrix} \begin{bmatrix} \sqrt{9724.168} & 0 \\ 0 & \sqrt{3160.986} \end{bmatrix} = \begin{bmatrix} -62.831 & -32.97448 \\ 18.403 & 12.02697 \\ -24.960 & 39.71091 \\ 69.388 & -18.76340 \end{bmatrix}.$$

Isometric Feature Mapping

Isometric feature mapping (Isomap) is one of representative algorithms of manifold learning.

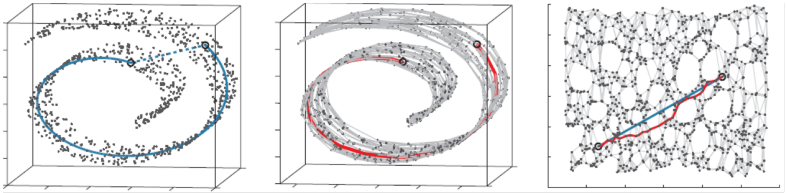
It uses manifold learning to obtain a global geometric framework from local metric information to solve the problem of nonlinear dimensionality reduction.

It does not use Euclidean distance when calculating the distance between data points on the high-dimensional manifold, but uses geodesic distances in differential geometry.

Isometric mapping, as the name suggests, is to map the geodesic distances between the original data on a nonlinear manifolds isometrically to the distances on the two-dimensional linear manifold.

Isometric Feature Mapping

Isomap is explained using a *Swiss roll* dataset to illustrate how it uses geodesic distances for nonlinear dimensionality reduction.



https://www.science.org/doi/10.1126/science.290.5500.2319

(a) For two points on a manifold, their Euclidean distance is not equal to geodesic distance.
(b) The neighborhood graph allows an approximation (red segments) to true geodesic path.
(c) 2-dimensional embedding recovered by Isomap best preserves the shortest path distances.

Stochastic Neighbor Embedding

Stochastic neighbor embedding (SNE) uses Gaussian distribution for both high-dimensional input space and low-dimensional output space.

Let $D_{\text{high}} = \{\boldsymbol{x}\} \subseteq \mathbb{R}^h$ and $D_{\text{low}} = \{\boldsymbol{y}\} \subseteq \mathbb{R}^l$ denote input and output dataset.

- In high-dimensional space, compute the asymmetric conditional probability $p_{j|i}$ as the similarity between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$.
- In low-dimensional space, calculate the similar conditional probability $q_{j|i}$ between $\boldsymbol{y}_i$ and $\boldsymbol{y}_j$ corresponding to $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$.

$$p_{j|i} = \frac{\exp\left(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-\|\boldsymbol{x}_i - \boldsymbol{x}_k\|^2 / 2\sigma_i^2\right)}, \qquad q_{j|i} = \frac{\exp\left(-\|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2\right)}{\sum_{k \neq i} \exp\left(-\|\boldsymbol{y}_i - \boldsymbol{y}_k\|^2\right)}.$$

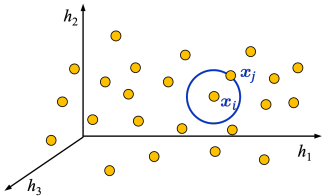
Stochastic Neighbor Embedding

- To maintain the same similarity between the data pairs in high- and low-dimensional spaces, let the cost (loss) function:

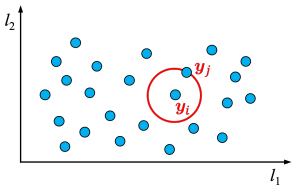
$$C = \sum_i D_{KL}\left(P_i || Q_i\right) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}.$$

- Minimize the cost (loss) function by gradient descent.

$$\frac{\partial C}{\partial \boldsymbol{y}_i} = 2 \sum_j \left(p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j}\right)\left(\boldsymbol{y}_i - \boldsymbol{y}_j\right).$$



(a)



(b)

t-Distributed Stochastic Neighbor Embedding

The t-distributed stochastic neighbor embedding (t-SNE) is an improved version of SNE.

It has made the following important improvements:

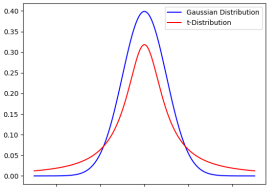
- In high-dimensional space, use symmetric conditional probability p_{ij} based on Gaussian distribution.
- In low-dimensional space, use t-distribution, i.e., joint probability distribution q_{ij}, in response to the "crowding problem".

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2m}, \qquad q_{ij} = \frac{\left(1 + \|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2\right)^{-1}}{\sum_{k \neq l} \left(1 + \|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2\right)^{-1}}.$$

t-Distributed Stochastic Neighbor Embedding

The "crowding problem":

- There is no enough room to accommodate all neighbors in Gaussian distribution.
- The t-distribution vs. Gaussian distribution is illustrated in the right figure.



The cost (loss) function and minimization for t-SNE:

$$C = \sum_i D_{KL}\left(P_i||Q_i\right) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}.$$

$$\frac{\partial C}{\partial \boldsymbol{y}_i} = 4 \sum_j \left(p_{ij} - q_{ij}\right)(\boldsymbol{y}_i - \boldsymbol{y}_j)\left(1 + \|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2\right)^{-1}.$$

SNE vs. t-SNE

SNE	t-SNE
$p_{j\|i} = \frac{\exp\left(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-\|\boldsymbol{x}_i - \boldsymbol{x}_k\|^2 / 2\sigma_i^2\right)}$	$p_{ij} = \frac{p_{j\|i} + p_{i\|j}}{2m}$
$q_{j\|i} = \frac{\exp\left(-\|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2\right)}{\sum_{k \neq i} \exp(-\|\boldsymbol{y}_i - \boldsymbol{y}_k\|^2)}$	$q_{ij} = \frac{(1+\|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2)^{-1}}{\sum_{k \neq l} (1+\|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2)^{-1}}$
$C = \sum_i D_{KL}(P_i \|\| Q_i) = \sum_i \sum_j p_{j\|i} \log \frac{p_{j\|i}}{q_{j\|i}}$	$C = D_{KL}(P \|\| Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}$
$\frac{\partial C}{\partial \boldsymbol{y}_i} = 2 \sum_j (p_{j\|i} - q_{j\|i} + p_{i\|j} - q_{i\|j})(\boldsymbol{y}_i - \boldsymbol{y}_j)$	$\frac{\partial C}{\partial \boldsymbol{y}_i} = 4 \sum_j (p_{ij} - q_{ij})(\boldsymbol{y}_i - \boldsymbol{y}_j)(1 + \|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2)^{-1}$

Thank You