Introduction to Machine Learning

Lecture 7 Representation and Clustering - Nonlinear Dimensionality Reduction

Hongteng Xu

中国人民大學 RENMIN UNIVERSITY OF CHINA

高瓴人工智能学院 Gaoling School of Artificial Intelligence

Outline

Review

- **Curse of dimensionality**
- **Principal component analysis:** principle, implementation, and statistical ML viewpoint
- **Other linear dimensionality reduction method:** RPCA, NMF, Subspace clustering, compressive sensing

Outline

Review

- **Curse of dimensionality**
- **Principal component analysis:** principle, implementation, and statistical ML viewpoint
- **Other linear dimensionality reduction method:** RPCA, NMF, Subspace clustering, compressive sensing

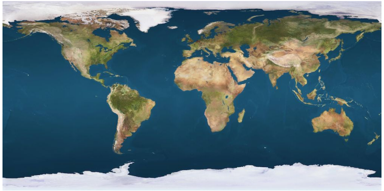
Today

- Manifold learning (MDS, ISOMAP, LLE, Diffusion Map, ...)
- Large-scale manifold learning (t-SNE)
- Kernel method (Kernel PCA)
- Autoencoders

Linear and Nonlinear Dimensionality Reduction

3D data

Linear map to 2D space



Nonlinear map to 2D space.

Recall The Desired Properties of Dimensionality Reduction

- **Minimizing reconstruction error**

$$\exists g : \mathcal{Z} \mapsto \mathcal{X},\ \boldsymbol{x} \approx g(f(\boldsymbol{x})). \tag{1}$$

- (Equivalently,) **Maximizing mutual information** (or minimizing information loss)

Recall The Desired Properties of Dimensionality Reduction

- **Minimizing reconstruction error**

$$\exists g : \mathcal{Z} \mapsto \mathcal{X},\ \boldsymbol{x} \approx g(f(\boldsymbol{x})). \tag{1}$$

- (Equivalently,) **Maximizing mutual information** (or minimizing information loss)
- **(Nearly) Isometry:**

$$d_{\mathcal{Z}}(f(\boldsymbol{x}), f(\boldsymbol{x}')) \approx d_{\mathcal{X}}(\boldsymbol{x}, \boldsymbol{x}'). \tag{2}$$

The Properties of Various Linear DR Methods

Method	(R)PCA	NMF	Compressive Sensing

The Properties of Various Linear DR Methods

Method	(R)PCA	NMF	Compressive Sensing
Reconstruction power	Yes	Yes	Conditionally, Yes

The Properties of Various Linear DR Methods

Method	(R)PCA	NMF	Compressive Sensing
Reconstruction power	Yes	Yes	Conditionally, Yes
(Nearly) Isometry	No	No	Conditionally, Yes

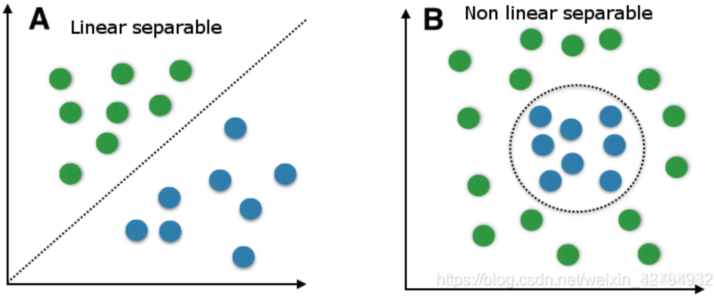
The Properties of Various Linear DR Methods

Method	(R)PCA	NMF	Compressive Sensing
Reconstruction power	Yes	Yes	Conditionally, Yes
(Nearly) Isometry	No	No	Conditionally, Yes

- The conditions of compressive sensing is based on the “sparse representation” assumption of data and the restricted isometric property (RIP) of random projection.

A Typical Case Breaking Isometry Seriously

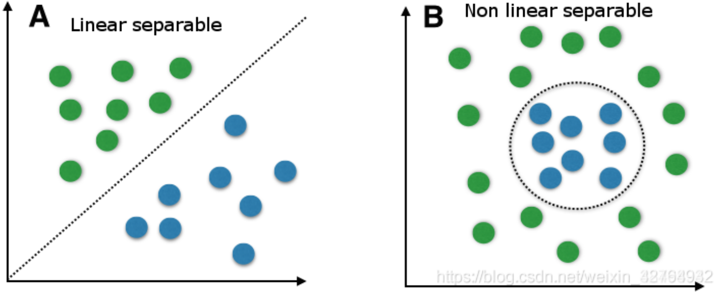
Linear inseparable data



- Classic linear DR method does not work well for linear inseparable data in general.

A Typical Case Breaking Isometry Seriously

Linear inseparable data



- Classic linear DR method does not work well for linear inseparable data in general.
- **Q: Can we obtain better isometry? If yes, how to do it?**

Typical Nonlinear Dimensionality Reduction Methods

(Classic) Manifold Learning (1995 - 2010)

- Multi-dimensional Scaling (MDS)
- ISOMAP
- Locally Linear Embedding (LLE)
- Eigenmap (Next lecture)
- Local Tangent Space Alignment (LTSA)

Typical Nonlinear Dimensionality Reduction Methods

(Classic) Manifold Learning (1995 - 2010)

- Multi-dimensional Scaling (MDS)
- ISOMAP
- Locally Linear Embedding (LLE)
- Eigenmap (Next lecture)
- Local Tangent Space Alignment (LTSA)

Kernel Methods (also can be viewed as manifold learning)

- Diffusion Map
- Kernel PCA
- t-Distributed Stochastic Neighbor Embedding (t-SNE)

Typical Nonlinear Dimensionality Reduction Methods

(Classic) Manifold Learning (1995 - 2010)

- Multi-dimensional Scaling (MDS)
- ISOMAP
- Locally Linear Embedding (LLE)
- Eigenmap (Next lecture)
- Local Tangent Space Alignment (LTSA)

Kernel Methods (also can be viewed as manifold learning)

- Diffusion Map
- Kernel PCA
- t-Distributed Stochastic Neighbor Embedding (t-SNE)

Autoencoding (also can be viewed as manifold learning)

- Various autoencoders

(Metric) Multi-dimensional Scaling (MDS)

Motivation:

- Keep isometry as much as possible.

(Metric) Multi-dimensional Scaling (MDS)

Motivation:

- Keep isometry as much as possible.

Principle:

- Given a set of data $\{\boldsymbol{x}_n\}_{n=1}^N$, we can compute a distance matrix:

$$\boldsymbol{D} = [d_{ij}] \in \mathbb{R}^{N\times N}, \quad d_{ij} = d(\boldsymbol{x}_i, \boldsymbol{x}_j). \tag{3}$$

  where $d(\cdot,\cdot)$ can be any valid metric of the sample space.

- Metric MDS aims at finding low-dimensional latent representations $\{\boldsymbol{z}_n\}_{n=1}^N$ via

$$\min_{\{\boldsymbol{z}_n\}_{n=1}^N \in \Omega} \underbrace{\left(\sum_{i\neq j} (d_{ij} - \|\boldsymbol{z}_i - \boldsymbol{z}_j\|_p)^2\right)^{1/2}}_{\mathrm{Stress}_d(\{\boldsymbol{z}_n\}_{n=1}^N)}, \tag{4}$$

  where $p = 1$ or 2 in general.

(Classic) Multi-dimensional Scaling (MDS)

Classic MDS is a special case of metric MDS:

- Use Euclidean distance $d_{ij} = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2$.

(Classic) Multi-dimensional Scaling (MDS)

Classic MDS is a special case of metric MDS:

- Use Euclidean distance $d_{ij} = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2$.
- Replace the stress loss with a **strain** loss:

$$\min \mathrm{Strain}_d(\{\boldsymbol{z}_n\}_{n=1}^N) = \left(\frac{\sum_{i,j=1}^N (k_{ij} - \boldsymbol{z}_i^T \boldsymbol{z}_j)^2}{\sum_{i,j=1}^N k_{ij}^2} \right)^{1/2},$$

(Classic) Multi-dimensional Scaling (MDS)

Classic MDS is a special case of metric MDS:

- Use Euclidean distance $d_{ij} = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2$.
- Replace the stress loss with a **strain** loss:

$$\min \operatorname{Strain}_d(\{\boldsymbol{z}_n\}_{n=1}^N) = \left(\frac{\sum_{i,j=1}^N (k_{ij} - \boldsymbol{z}_i^T \boldsymbol{z}_j)^2}{\sum_{i,j=1}^N k_{ij}^2} \right)^{1/2}, \tag{5}$$

where

$$\boldsymbol{K} = [k_{ij}] = -\frac{1}{2}\boldsymbol{C}(\boldsymbol{D} \odot \boldsymbol{D})\boldsymbol{C}, \quad \boldsymbol{C} = \boldsymbol{I}_N - \frac{1}{N}\mathbf{1}_{N\times N} \ \text{(Centering matrix)}$$

(Classic) Multi-dimensional Scaling (MDS)

Classic MDS is a special case of metric MDS:

- Use Euclidean distance $d_{ij} = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2$.
- Replace the stress loss with a **strain** loss:

$$\min \mathrm{Strain}_d(\{\boldsymbol{z}_n\}_{n=1}^N) = \left(\frac{\sum_{i,j=1}^N (k_{ij} - \boldsymbol{z}_i^T \boldsymbol{z}_j)^2}{\sum_{i,j=1}^N k_{ij}^2} \right)^{1/2}, \tag{5}$$

where

$$\boldsymbol{K} = [k_{ij}] = -\frac{1}{2}\boldsymbol{C}(\boldsymbol{D} \odot \boldsymbol{D})\boldsymbol{C}, \quad \boldsymbol{C} = \boldsymbol{I}_N - \frac{1}{N}\mathbf{1}_{N\times N} \ \text{(Centering matrix)}$$

$$\Rightarrow \boldsymbol{K} = \tilde{\boldsymbol{X}}\tilde{\boldsymbol{X}}^T = \boldsymbol{C}\boldsymbol{X}\boldsymbol{X}^T\boldsymbol{C} \ \text{(Derive It)}$$

(Classic) Multi-dimensional Scaling (MDS)

Classic MDS is a special case of metric MDS:

- Use Euclidean distance $d_{ij} = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2$.
- Replace the stress loss with a **strain** loss:

$$\min \text{Strain}_d(\{\boldsymbol{z}_n\}_{n=1}^N) = \left(\frac{\sum_{i,j=1}^N (k_{ij} - \boldsymbol{z}_i^T \boldsymbol{z}_j)^2}{\sum_{i,j=1}^N k_{ij}^2} \right)^{1/2}, \tag{5}$$

where

$$\begin{aligned} &\boldsymbol{K} = [k_{ij}] = -\frac{1}{2}\boldsymbol{C}(\boldsymbol{D} \odot \boldsymbol{D})\boldsymbol{C}, \quad \boldsymbol{C} = \boldsymbol{I}_N - \frac{1}{N}\mathbf{1}_{N\times N} \text{ (Centering matrix)} \\ &\Rightarrow \boldsymbol{K} = \tilde{\boldsymbol{X}}\tilde{\boldsymbol{X}}^T = \boldsymbol{C}\boldsymbol{X}\boldsymbol{X}^T\boldsymbol{C} \text{ (Derive It)} \end{aligned} \tag{6}$$

- **Solution:** If we require L-dimensional $\boldsymbol{z}$'s, we have:

$$\boldsymbol{K} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T, \quad \text{and} \quad \boldsymbol{Z} = \boldsymbol{U}_L \boldsymbol{\Lambda}_L^{\frac{1}{2}}.$$

(Classic) Multi-dimensional Scaling (MDS)

Classic MDS is a special case of metric MDS:

- Use Euclidean distance $d_{ij} = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2$.
- Replace the stress loss with a **strain** loss:

$$\min \text{Strain}_d(\{\boldsymbol{z}_n\}_{n=1}^N) = \left(\frac{\sum_{i,j=1}^N (k_{ij} - \boldsymbol{z}_i^T \boldsymbol{z}_j)^2}{\sum_{i,j=1}^N k_{ij}^2} \right)^{1/2}, \tag{5}$$

where

$$\begin{aligned} &\boldsymbol{K} = [k_{ij}] = -\frac{1}{2}\boldsymbol{C}(\boldsymbol{D} \odot \boldsymbol{D})\boldsymbol{C}, \quad \boldsymbol{C} = \boldsymbol{I}_N - \frac{1}{N}\mathbf{1}_{N\times N} \text{ (Centering matrix)} \\ \Rightarrow &\boldsymbol{K} = \tilde{\boldsymbol{X}}\tilde{\boldsymbol{X}}^T = \boldsymbol{C}\boldsymbol{X}\boldsymbol{X}^T\boldsymbol{C} \text{ (Derive It)} \end{aligned} \tag{6}$$

- **Solution:** If we require L-dimensional $\boldsymbol{z}$'s, we have:

$$\boldsymbol{K} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T, \quad \text{and} \quad \boldsymbol{Z} = \boldsymbol{U}_L\boldsymbol{\Lambda}_L^{\frac{1}{2}}. \tag{7}$$

Could you connect it with PCA or Least-Square Data Denoising?

ISOMAP

Motivation:

- Keep isometry under **geodesic distance** as much as possible.

ISOMAP

Motivation:

- Keep isometry under **geodesic distance** as much as possible.

Principle: Given a set of data $\{\boldsymbol{x}_n\}_{n=1}^{N}$

1. Determine the neighbors of each data point and construct a K-nearest neighbor (KNN) graph of the data.

ISOMAP

Motivation:

- Keep isometry under **geodesic distance** as much as possible.

Principle: Given a set of data $\{\boldsymbol{x}_n\}_{n=1}^N$

1. Determine the neighbors of each data point and construct a K-nearest neighbor (KNN) graph of the data.
2. Compute the shortest path between arbitrary two nodes and obtain an approximation of the geodesic distance matrix $\boldsymbol{D} = [d_{ij}]$
   - Dijkstra's algorithm
   - Floyd-Warshall algorithm

ISOMAP

Motivation:

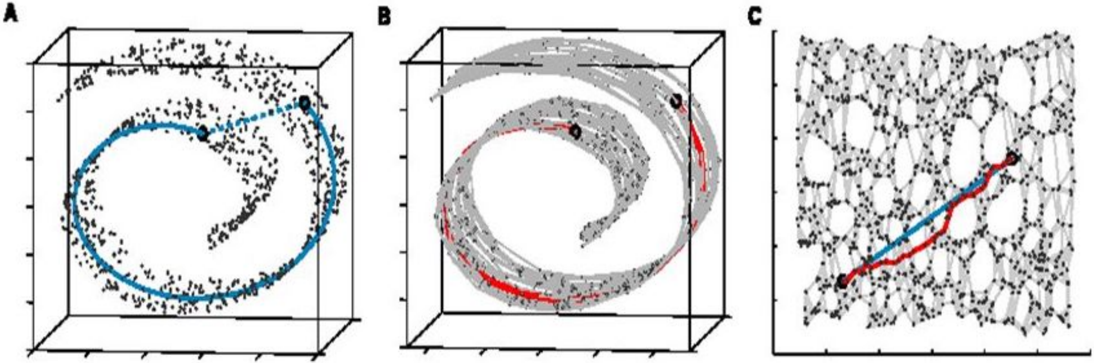
- Keep isometry under **geodesic distance** as much as possible.

Principle: Given a set of data $\{\boldsymbol{x}_n\}_{n=1}^N$

1. Determine the neighbors of each data point and construct a K-nearest neighbor (KNN) graph of the data.
2. Compute the shortest path between arbitrary two nodes and obtain an approximation of the geodesic distance matrix $\boldsymbol{D} = [d_{ij}]$
   - Dijkstra's algorithm
   - Floyd-Warshall algorithm
3. Compute low-dimensional embedding by MDS:

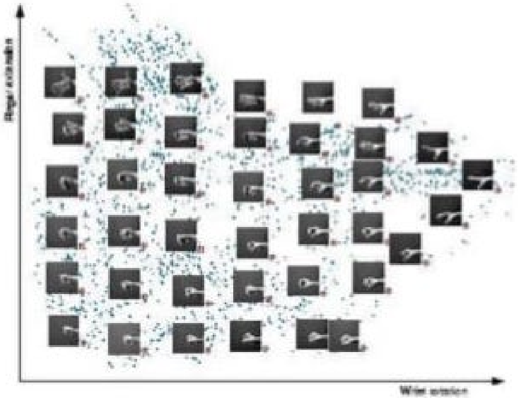
$$\boldsymbol{K} = -\frac{1}{2}\boldsymbol{C}(\boldsymbol{D} \odot \boldsymbol{D})\boldsymbol{C} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T \quad \Rightarrow \quad \boldsymbol{Z} = \boldsymbol{U}_L\boldsymbol{\Lambda}_L^{\frac{1}{2}} \tag{8}$$

ISOMAP v.s. Metric MDS



ISOMAP

Face varying in pose and illumination



Hand varying in finger extension & wrist rotation

Locally Linear Embedding (LLE)

Motivation:

- Keep isometry indirectly through inheriting local linear self-representation power.

Locally Linear Embedding (LLE)

Motivation:

- Keep isometry indirectly through inheriting local linear self-representation power.

Principle:

- **Local linear self-representation:** Given a sample $\boldsymbol{x}$ and its K neighbors $\boldsymbol{X} = [\boldsymbol{x}_1, ..., \boldsymbol{x}_K]$, where $d(\boldsymbol{x}, \boldsymbol{x}_k) \leq \tau$ for $k = 1, ..., K$.

$$\exists \boldsymbol{w} \in \mathbb{R}^K, \text{ such that } \boldsymbol{x} \approx \boldsymbol{X}\boldsymbol{w}.$$

Locally Linear Embedding (LLE)

Motivation:

- Keep isometry indirectly through inheriting local linear self-representation power.

Principle:

- **Local linear self-representation:** Given a sample $\boldsymbol{x}$ and its K neighbors $\boldsymbol{X} = [\boldsymbol{x}_1, ..., \boldsymbol{x}_K]$, where $d(\boldsymbol{x}, \boldsymbol{x}_k) \leq \tau$ for $k = 1, ..., K$.

$$\exists \boldsymbol{w} \in \mathbb{R}^K, \text{ such that } \boldsymbol{x} \approx \boldsymbol{X}\boldsymbol{w}. \tag{9}$$

- **LLE (Locally Linear Embedding):** Given $\boldsymbol{X} = [\boldsymbol{x}_1, ..., \boldsymbol{x}_N]$, find low-dimensional representations $\boldsymbol{Z} = [\boldsymbol{z}_1, ..., \boldsymbol{z}_N]$ that inherit the local linear self-representation relations.

Locally Linear Embedding (LLE)

Solution:

1. Compute the linear coefficients

$$\min_{\boldsymbol{W}} \sum_{i=1}^{N} \|\boldsymbol{x}_i - \boldsymbol{X}_i \boldsymbol{w}_i\|_2^2, \;\; s.t. \; \sum_{k=1}^{K} w_{ik_i} = 1, \text{ and } \{k_i\}_{k=1}^{K} = \text{neighbors' index} \quad (10)$$

where $\boldsymbol{W} = [w_{ij}] \in \mathbb{R}^{N \times N}$, and $\forall \boldsymbol{i} \in \{1, ..., N\}$

$$w_{ij} = \begin{cases} w_{ik_i} & j \in \{k_i\}_{k=1}^{K} \\ 0 & \text{Otherwise.} \end{cases}$$

Locally Linear Embedding (LLE)

Solution:

1. Compute the linear coefficients

$$\min_{\boldsymbol{W}} \sum_{i=1}^{N} \|\boldsymbol{x}_i - \boldsymbol{X}_i \boldsymbol{w}_i\|_2^2, \;\; s.t. \; \sum_{k=1}^{K} w_{ik_i} = 1, \text{ and } \{k_i\}_{k=1}^{K} = \text{neighbors' index} \quad (10)$$

where $\boldsymbol{W} = [w_{ij}] \in \mathbb{R}^{N \times N}$, and $\forall \boldsymbol{i} \in \{1, ..., N\}$

$$w_{ij} = \begin{cases} w_{ik_i} & j \in \{k_i\}_{k=1}^{K} \\ 0 & \text{Otherwise.} \end{cases} \tag{11}$$

2. Compute the embeddings:

$$\min_{\boldsymbol{Z}} \sum_{i=1}^{N} \|\boldsymbol{z}_i - \boldsymbol{Z}_i \boldsymbol{w}_i\|_2^2 = \min_{\boldsymbol{Z}} \operatorname{tr}(\boldsymbol{Z\Phi Z}^T), \quad s.t. \; \boldsymbol{ZZ}^T = \boldsymbol{I}_L, \boldsymbol{Z}\boldsymbol{1}_L = \boldsymbol{0}. \tag{12}$$

where $\boldsymbol{\Phi} = [\phi_{ij}]$ and $\phi_{ij} = \delta_{ij} - w_{ij} - w_{ji} + \boldsymbol{w}_i^T \boldsymbol{w}_j$.

Locally Linear Embedding (LLE)

Solution:

1. Compute the linear coefficients

$$\min_{\boldsymbol{W}} \sum_{i=1}^{N} \|\boldsymbol{x}_i - \boldsymbol{X}_i \boldsymbol{w}_i\|_2^2, \;\; s.t. \sum_{k=1}^{K} w_{ik_i} = 1, \text{ and } \{k_i\}_{k=1}^{K} = \text{neighbors' index} \tag{10}$$

where $\boldsymbol{W} = [w_{ij}] \in \mathbb{R}^{N \times N}$, and $\forall \boldsymbol{i} \in \{1, ..., N\}$

$$w_{ij} = \begin{cases} w_{ik_i} & j \in \{k_i\}_{k=1}^{K} \\ 0 & \text{Otherwise.} \end{cases} \tag{11}$$

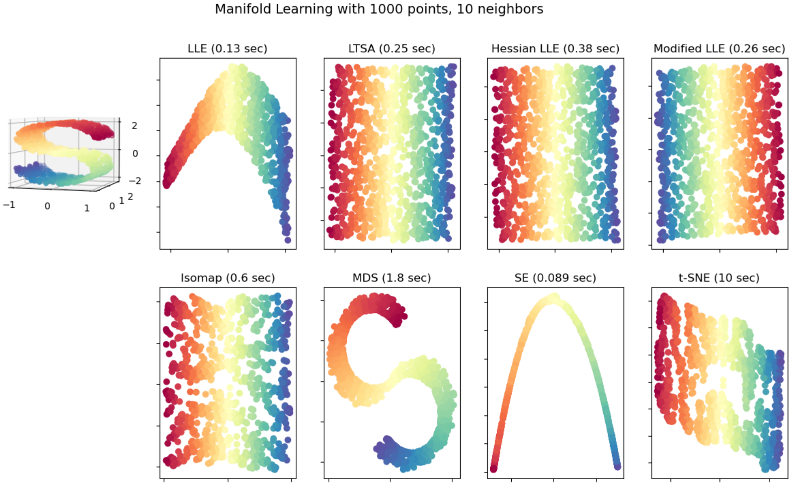
2. Compute the embeddings:

$$\min_{\boldsymbol{Z}} \sum_{i=1}^{N} \|\boldsymbol{z}_i - \boldsymbol{Z}_i \boldsymbol{w}_i\|_2^2 = \min_{\boldsymbol{Z}} \operatorname{tr}(\boldsymbol{Z}\boldsymbol{\Phi}\boldsymbol{Z}^T), \quad s.t. \, \boldsymbol{Z}\boldsymbol{Z}^T = \boldsymbol{I}_L, \boldsymbol{Z}\boldsymbol{1}_L = \boldsymbol{0}. \tag{12}$$

where $\boldsymbol{\Phi} = [\phi_{ij}]$ and $\phi_{ij} = \delta_{ij} - w_{ij} - w_{ji} + \boldsymbol{w}_i^T \boldsymbol{w}_j$.

3. $\boldsymbol{\Phi} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T$, $\lambda_1 \geq ... \geq \lambda_N$, and $\boldsymbol{Z} = \boldsymbol{U}_{N-L+1:N}$.

Classic Manifold Learning Methods



Most of them require construct a KNN graph first.

Nonlinear DR Methods: Kernel PCA

Motivation:

- Extend PCA to linear inseparable (equivalently, nonlinear separable) situations.

Nonlinear DR Methods: Kernel PCA

Motivation:

- Extend PCA to linear inseparable (equivalently, nonlinear separable) situations.
- Metric MDS and ISOMAP can be viewed as special cases of kernel PCA.

Nonlinear DR Methods: Kernel PCA

Motivation:

- Extend PCA to linear inseparable (equivalently, nonlinear separable) situations.
- Metric MDS and ISOMAP can be viewed as special cases of kernel PCA.

(Linear) PCA:

- Consider the covariance matrix of zero-mean samples: $\boldsymbol{C} = \boldsymbol{X}^T\boldsymbol{X} \in \mathbb{R}^{D\times D}$.
- Apply Eigenvalue Decomposition $\boldsymbol{C} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^T$
- Consider the projection along top-L principal components: $\boldsymbol{X}\boldsymbol{V}_L$.

Nonlinear DR Methods: Kernel PCA

Motivation:

- Extend PCA to linear inseparable (equivalently, nonlinear separable) situations.
- Metric MDS and ISOMAP can be viewed as special cases of kernel PCA.

(Linear) PCA:

- Consider the covariance matrix of zero-mean samples: $\boldsymbol{C} = \boldsymbol{X}^T\boldsymbol{X} \in \mathbb{R}^{D\times D}$.
- Apply Eigenvalue Decomposition $\boldsymbol{C} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^T$
- Consider the projection along top-L principal components: $\boldsymbol{X}\boldsymbol{V}_L$.

SVD-based Implementation:

- Apply SVD $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$.
- Consider the projection on top-L principal components: $\boldsymbol{X}\boldsymbol{V}_L = \boldsymbol{U}_L\boldsymbol{\Sigma}_L$.

Kernel PCA

SVD-based Implementation:

- Apply SVD $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$
- Consider the projection on top-L principal components: $\boldsymbol{X}\boldsymbol{V}_L = \boldsymbol{U}_L\boldsymbol{\Sigma}_L$.

Kernel PCA

SVD-based Implementation:

- Apply SVD $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$
- Consider the projection on top-L principal components: $\boldsymbol{X}\boldsymbol{V}_L = \boldsymbol{U}_L\boldsymbol{\Sigma}_L$.

Revisit the SVD-based Implementation from a viewpoint of linear kernel

- The Gram matrix of linear kernel $\boldsymbol{K} = \boldsymbol{X}\boldsymbol{X}^T \in \mathbb{R}^{N\times N}$.

Kernel PCA

SVD-based Implementation:

- Apply SVD $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$
- Consider the projection on top-L principal components: $\boldsymbol{X}\boldsymbol{V}_L = \boldsymbol{U}_L\boldsymbol{\Sigma}_L$.

Revisit the SVD-based Implementation from a viewpoint of linear kernel

- The Gram matrix of linear kernel $\boldsymbol{K} = \boldsymbol{X}\boldsymbol{X}^T \in \mathbb{R}^{N\times N}$.
- The Eigenvalue decomposition: $\boldsymbol{K} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T$

Kernel PCA

SVD-based Implementation:

- Apply SVD $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$
- Consider the projection on top-L principal components: $\boldsymbol{X}\boldsymbol{V}_L = \boldsymbol{U}_L\boldsymbol{\Sigma}_L$.

Revisit the SVD-based Implementation from a viewpoint of linear kernel

- The Gram matrix of linear kernel $\boldsymbol{K} = \boldsymbol{X}\boldsymbol{X}^T \in \mathbb{R}^{N\times N}$.
- The Eigenvalue decomposition: $\boldsymbol{K} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T$
- The PCA corresponds to the scaling of the top-L eigenvectors of $\boldsymbol{K}$: $\boldsymbol{U}_L\boldsymbol{\Lambda}_L^{\frac{1}{2}}$.

Kernel PCA

SVD-based Implementation:

- Apply SVD $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$
- Consider the projection on top-L principal components: $\boldsymbol{X}\boldsymbol{V}_L = \boldsymbol{U}_L\boldsymbol{\Sigma}_L$.

Revisit the SVD-based Implementation from a viewpoint of linear kernel

- The Gram matrix of linear kernel $\boldsymbol{K} = \boldsymbol{X}\boldsymbol{X}^T \in \mathbb{R}^{N\times N}$.
- The Eigenvalue decomposition: $\boldsymbol{K} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T$
- The PCA corresponds to the scaling of the top-L eigenvectors of $\boldsymbol{K}$: $\boldsymbol{U}_L\boldsymbol{\Lambda}_L^{\frac{1}{2}}$.

Replacing the linear kernel to arbitrary kernel, you obtain the Kernel PCA:)

- $K(\boldsymbol{x},\boldsymbol{y}) = (\boldsymbol{x}^T\boldsymbol{y}+1)^2$, $\exp(-\|\boldsymbol{x}-\boldsymbol{y}\|^2/h)$, ...

Kernel PCA

Principle:

- N D-dimensional samples can be linearly separated when $D \geq N$.

Kernel PCA

Principle:

- N D-dimensional samples can be linearly separated when $D \geq N$.
- Recall that a kernel function equals to the inner product of samples in an implicit, maybe infinite-dimensional, feature space (Lecture 5).

$$K(\boldsymbol{x}, \boldsymbol{y}) = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{y}) \rangle_{\mathcal{F}} = \phi(\boldsymbol{x})^T \phi(\boldsymbol{y})$$

Kernel PCA

Principle:

- N D-dimensional samples can be linearly separated when $D \geq N$.
- Recall that a kernel function equals to the inner product of samples in an implicit, maybe infinite-dimensional, feature space (Lecture 5).

$$K(\boldsymbol{x}, \boldsymbol{y}) = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{y}) \rangle_{\mathcal{F}} = \phi(\boldsymbol{x})^T \phi(\boldsymbol{y}) \tag{13}$$

- Because we are never working directly in the (implicit) feature space.
- The kernel PCA does not compute the principal components directly, but the projections of data onto the top-L components.

Kernel PCA

- Denote $\boldsymbol{\Phi}(\boldsymbol{X}) = [\phi(\boldsymbol{x}_1), ..., \phi(\boldsymbol{x}_N)]^T \in \mathbb{R}^{N \times \dim(F)}$

Kernel PCA

- Denote $\boldsymbol{\Phi}(\boldsymbol{X}) = [\phi(\boldsymbol{x}_1), ..., \phi(\boldsymbol{x}_N)]^T \in \mathbb{R}^{N \times \dim(F)}$
- An "functional" SVD: $\boldsymbol{\Phi}(\boldsymbol{X}) = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}(\boldsymbol{X})$, where $\boldsymbol{V}(\boldsymbol{X}) \in \mathbb{R}^{N \times \dim(F)}$ are principal components of $\mathcal{F}$.

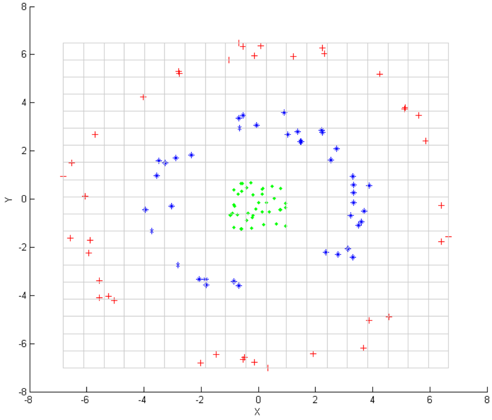
Kernel PCA

- Denote $\mathbf{\Phi}(\boldsymbol{X}) = [\phi(\boldsymbol{x}_1), ..., \phi(\boldsymbol{x}_N)]^T \in \mathbb{R}^{N\times \dim(F)}$
- An "functional" SVD: $\mathbf{\Phi}(\boldsymbol{X}) = \boldsymbol{U}\mathbf{\Sigma}\boldsymbol{V}(\boldsymbol{X})$, where $\boldsymbol{V}(\boldsymbol{X}) \in \mathbb{R}^{N\times \dim(F)}$ are principal components of $\mathcal{F}$.
- **Instead of applying the functional SVD, consider the eigenvalue decomposition of $\boldsymbol{K} = \mathbf{\Phi}(\boldsymbol{X})\mathbf{\Phi}^T(\boldsymbol{X}) = \boldsymbol{U}\mathbf{\Sigma}^2\boldsymbol{U}^T = \boldsymbol{U}\mathbf{\Lambda}\boldsymbol{U}^T$**

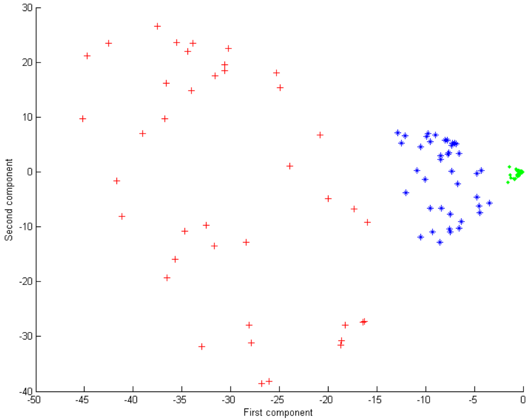
Kernel PCA

- Denote $\mathbf{\Phi}(\boldsymbol{X}) = [\phi(\boldsymbol{x}_1), ..., \phi(\boldsymbol{x}_N)]^T \in \mathbb{R}^{N\times \dim(F)}$
- An "functional" SVD: $\mathbf{\Phi}(\boldsymbol{X}) = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}(\boldsymbol{X})$, where $\boldsymbol{V}(\boldsymbol{X}) \in \mathbb{R}^{N\times \dim(F)}$ are principal components of $\mathcal{F}$.
- **Instead of applying the functional SVD, consider the eigenvalue decomposition of $\boldsymbol{K} = \mathbf{\Phi}(\boldsymbol{X})\mathbf{\Phi}^T(\boldsymbol{X}) = \boldsymbol{U}\boldsymbol{\Sigma}^2\boldsymbol{U}^T = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T$**
- Kernel PCA = Linear PCA in the feature space associated with the kernel.

Kernel PCA



8
6
4
2
0
-2
-4
-6
-8
Y
-8
-6
-4
-2
0
2
4
6
8
X



30
20
10
0
-10
-20
-30
-40
Second component
-50
-45
-40
-35
-30
-25
-20
-15
-10
-5
0
First component

Revisit MDS and ISOMAP as Kernel PCA Methods

- MDS (Linear Kernel):

$$\boldsymbol{K} = -\frac{1}{2}\boldsymbol{C}(\boldsymbol{D} \odot \boldsymbol{D})\boldsymbol{C} = \boldsymbol{C}\boldsymbol{X}\boldsymbol{X}^T\boldsymbol{C} = \tilde{\boldsymbol{X}}\tilde{\boldsymbol{X}}^T$$

Revisit MDS and ISOMAP as Kernel PCA Methods

- MDS (Linear Kernel):

$$\boldsymbol{K} = -\frac{1}{2}\boldsymbol{C}(\boldsymbol{D} \odot \boldsymbol{D})\boldsymbol{C} = \boldsymbol{C}\boldsymbol{X}\boldsymbol{X}^T\boldsymbol{C} = \tilde{\boldsymbol{X}}\tilde{\boldsymbol{X}}^T \tag{14}$$

- ISOMAP (A positive semi-definite kernel based on doubly centered geodesic distance, a.k.a., Mercer kernel):

$$\boldsymbol{K} = -\frac{1}{2}\boldsymbol{C}(\boldsymbol{D}_{geo} \odot \boldsymbol{D}_{geo})\boldsymbol{C} \tag{15}$$

t-Stochastic Neighborhood Embedding

Motivation:

- Large-scale dimensionality reduction.
- Visualization and clustering of high-dimensional data.

Principle:

- Given $\{\boldsymbol{x}_n\}_{n=1}^N$, define a probability p_{ij} that is proportional to the similarity between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$:

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N} \quad \text{and} \quad p_{ii} = 0 \tag{16}$$

$$p_{j|i} = \frac{\exp(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 / (2\sigma_i^2))}{\sum_{k \neq i} \exp(-\|\boldsymbol{x}_i - \boldsymbol{x}_k\|^2 / (2\sigma_i^2))}. \tag{17}$$

Nonlinear DR Methods: t-Stochastic Neighborhood Embedding

- t-SNE aims to learn $\{\boldsymbol{z}_n \in \mathbb{R}^K\}_{n=1}^N$ ($K = 2, 3$ typically) by
  1. Define a similarity q_{ij} between $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$ as

$$q_{ij} = \frac{(1 + \|\boldsymbol{z}_i - \boldsymbol{z}_j\|^2)^{-1}}{\sum_k \sum_{k \neq l} (1 + \|\boldsymbol{z}_k - \boldsymbol{z}_l\|^2)^{-1}} \quad \text{and} \quad q_{ii} = 0 \tag{18}$$

where $\{q_{ij}\}$ yield a t-distribution.

Nonlinear DR Methods: t-Stochastic Neighborhood Embedding

- t-SNE aims to learn $\{\boldsymbol{z}_n \in \mathbb{R}^K\}_{n=1}^N$ ($K = 2, 3$ typically) by

  1. Define a similarity q_{ij} between $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$ as

  $$q_{ij} = \frac{(1 + \|\boldsymbol{z}_i - \boldsymbol{z}_j\|^2)^{-1}}{\sum_k \sum_{k \neq l} (1 + \|\boldsymbol{z}_k - \boldsymbol{z}_l\|^2)^{-1}} \quad \text{and} \quad q_{ii} = 0 \tag{18}$$

     where $\{q_{ij}\}$ yield a t-distribution.

  2. Learn the $\boldsymbol{z}$'s via

  $$\min_{\boldsymbol{Z}} \mathrm{KL}(P \| Q) := \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}. \tag{19}$$

     SGD is applied to solve this problem.

Nonlinear DR Methods: t-Stochastic Neighborhood Embedding

- t-SNE aims to learn $\{\boldsymbol{z}_n \in \mathbb{R}^K\}_{n=1}^N$ ($K = 2, 3$ typically) by
  1. Define a similarity q_{ij} between $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$ as

$$q_{ij} = \frac{(1 + \|\boldsymbol{z}_i - \boldsymbol{z}_j\|^2)^{-1}}{\sum_k \sum_{k \neq l} (1 + \|\boldsymbol{z}_k - \boldsymbol{z}_l\|^2)^{-1}} \quad \text{and} \quad q_{ii} = 0 \tag{18}$$

     where $\{q_{ij}\}$ yield a t-distribution.
  2. Learn the $\boldsymbol{z}$'s via

$$\min_{\boldsymbol{Z}} \mathrm{KL}(P \| Q) := \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}. \tag{19}$$

     SGD is applied to solve this problem.

Rationality:

- Similar $\boldsymbol{x}$'s are modeled by nearby $\boldsymbol{z}$'s and dissimilar $\boldsymbol{x}$'s are modeled by distant $\boldsymbol{z}$'s with high probability.
- Under special settings, approximate a simple form of spectral clustering.

Nonlinear DR Methods: Autoencoders

Drawbacks of the above methods:

- From inductive paradigm to transductive paradigm
- High complexity and inscalable inference
- No reconstruction power

Nonlinear DR Methods: Autoencoders

Drawbacks of the above methods:

- From inductive paradigm to transductive paradigm
- High complexity and inscalable inference
- No reconstruction power

Motivation:

- Achieve (nearly) isometry and reconstruction power jointly.
- Achieve inductive inference
- Reduce complexity of other nonlinear method

Nonlinear DR Methods: Autoencoders

Principle:

- Revisit PCA from a viewpoint of autoencoding.

$$\boldsymbol{X}^* = \arg\,\min_X \|\boldsymbol{X}_{noisy} - \boldsymbol{X}\|_F^2, \quad s.t.\ \mathrm{rank}(\boldsymbol{X}) \le L$$

Nonlinear DR Methods: Autoencoders

Principle:

- Revisit PCA from a viewpoint of autoencoding.

$$
\begin{aligned}
\boldsymbol{X}^* &= \arg\min_X \|\boldsymbol{X}_{noisy} - \boldsymbol{X}\|_F^2, \quad s.t.\ \mathrm{rank}(\boldsymbol{X}) \leq L \\
&= \boldsymbol{U}_L \boldsymbol{\Sigma}_L \boldsymbol{V}_L^T
\end{aligned}
$$

Nonlinear DR Methods: Autoencoders

Principle:

- Revisit PCA from a viewpoint of autoencoding.

$$
\begin{aligned}
\boldsymbol{X}^* &= \arg\min_{X} \|\boldsymbol{X}_{noisy} - \boldsymbol{X}\|_F^2, \quad s.t.\ \mathrm{rank}(\boldsymbol{X}) \leq L \\
&= \boldsymbol{U}_L \boldsymbol{\Sigma}_L \boldsymbol{V}_L^T \\
&\Rightarrow \text{Encoder: } \boldsymbol{X}_{noisy} \boldsymbol{V}_L = \boldsymbol{U}_L \boldsymbol{\Sigma}_L \\
&\quad\ \ \text{Decoder: } \boldsymbol{X}^* = \boldsymbol{X}_{noisy} \boldsymbol{V}_L \boldsymbol{V}_L^T
\end{aligned} \tag{20}
$$

- $\boldsymbol{V}_K$ and $\boldsymbol{V}_K^T$ work as the encoder and the decoder, respectively.

Nonlinear DR Methods: Autoencoders

- In general, a typical autocoder consists of

$$\text{Encoder: } f : \mathcal{X} \mapsto \mathcal{Z}, \quad \text{Decoder: } g : \mathcal{Z} \mapsto \mathcal{X}$$

Nonlinear DR Methods: Autoencoders

- In general, a typical autocoder consists of

$$\text{Encoder: } f: \mathcal{X} \mapsto \mathcal{Z}, \quad \text{Decoder: } g: \mathcal{Z} \mapsto \mathcal{X} \tag{21}$$

- Given a set of data $\{\boldsymbol{x}_n\}_{n=1}^N$,

$$\min_{f,g} \underbrace{\sum_{n=1}^{N} \text{loss}(\boldsymbol{x}_n, g(f(\boldsymbol{x}_n)))}_{\text{loss}(\boldsymbol{X}, g(f(\boldsymbol{X})))} + \text{reg}(q_{\boldsymbol{Z}|\boldsymbol{X}}, p_{\boldsymbol{Z}}) \tag{22}$$

  where $p_{\boldsymbol{Z}}$ is a predefined prior distribution of latent codes, while $q_{\boldsymbol{Z}|\boldsymbol{X}}$ is the posterior distribution of the latent codes given the corresponding data, which is determined by the encoder f.

Nonlinear DR Methods: Autoencoders

- In general, a typical autocoder consists of

$$\text{Encoder: } f : \mathcal{X} \mapsto \mathcal{Z}, \quad \text{Decoder: } g : \mathcal{Z} \mapsto \mathcal{X} \tag{21}$$

- Given a set of data $\{\boldsymbol{x}_n\}_{n=1}^N$,

$$\min_{f,g} \underbrace{\sum_{n=1}^{N} \text{loss}(\boldsymbol{x}_n, g(f(\boldsymbol{x}_n)))}_{\text{loss}(\boldsymbol{X}, g(f(\boldsymbol{X})))} + \text{reg}(q_{\boldsymbol{Z}|\boldsymbol{X}}, p_{\boldsymbol{Z}}) \tag{22}$$

  where $p_{\boldsymbol{Z}}$ is a predefined prior distribution of latent codes, while $q_{\boldsymbol{Z}|\boldsymbol{X}}$ is the posterior distribution of the latent codes given the corresponding data, which is determined by the encoder f.

- More details will be given in the following lectures.

In Summary

- Introduce classic manifold learning methods
- Introduce the most widely-used manifold learning method, t-SNE
- Introduce the kernelization of PCA.
- Introduce autoencoders (briefly)

Next...

- Data representation, clustering, and unsupervised learning
- Kmeans and spectral clustering
- Evaluation of clustering method.