Non-linear dimensionality reduction and kernels: eigenmaps, isomaps, locally linear embeddings

Presented by: Hanzhong (Victor) Zheng

Review of Dimensionality Reduction

- Dimensionality reduction can be done through

  1. **feature selection**: only keeps the most relevant variables from the original dataset.

  2. **dimensionality reduction**: finds the smaller set of new variables, containing basically the same information as the original variables.

- Dimensionality reduction can also be categorized into:
  - linear dimensionality reduction (e.g. PCA, SVD)
  - non-linear dimensionality reduction (e.g. autoencoders, kernel PCA and others).

Non-linear dimensionality reduction

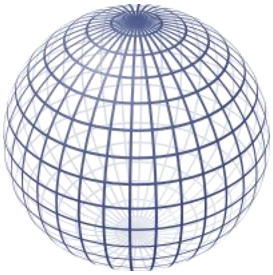
- **This lecture:** find a nonlinear low dimensional representation of data that reflects the data topology
- **Methods covered:**
  - Isomaps
  - Locally Embedding Space (LLE)
  - Eigenmaps

Topology

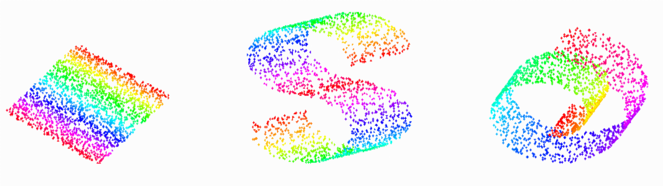
- Topology: “*the study of qualitative properties of certain objects that are invariant under a certain kind of transformation, especially those properties that are invariant under a certain kind of invertible transformation*”
- “A topologist is one who doesn’t know the difference between a doughnut and a coffee cup” –John L. Kelley (In General Topology 1995, 88 footnote)

Manifolds

- Definition: A manifold is a topological space that locally resembles Euclidean space **near each point**.



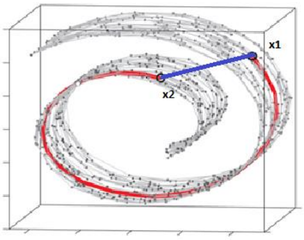
Manifolds



- Three examples of manifolds
- All three are two-dimensional data embedded in 3D
  - Linear, “S”-shape, “Swiss roll”
- For all three examples, we would like to recover:
  - Their two-dimensional representation
  - “Consistent” coordinates of the data in the 2D

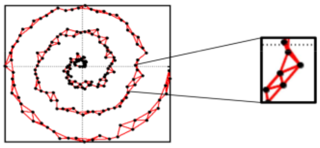
Manifolds and local distances

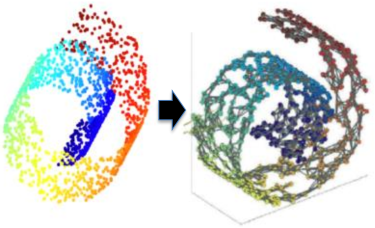
- In general, the distances induced by data (manifolds) may not be Euclidean. That is the global distances don't respect the geometry.
- Local distances can be still approximated with Euclidean distances
- **Idea for the dimensionality reduction:**
  - Define global distances/similarity in terms of local distances/similarity
  - Use these to define a low-dimensional embedding (low-dimensional representation) of the data
- **The idea can be implemented with the help of the Neighborhood graph**



Neighborhood Graph

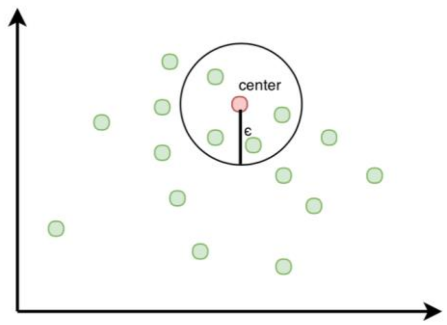
- **A neighborhood graph**
  - Vertices = data points
  - Edges and their weights reflect local similarity or local distances
  - Only points close to each other (neighbors) are connected





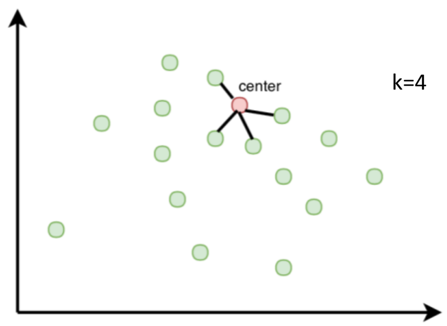
Neighborhood Graph

Approach 1: select and connect all points within the ε neighborhood



Neighborhood Graph construction

Approach 2: pick and connect k nearest neighbor points



Neighborhood Graph

- **Distance-based neighborhood graph:**
  - Edge weight: Euclidean distance between two data points – *d* (x_i,x_j)
- **Similarity-based neighborhood graph:**
  - Edge weight:
    - Simple: W_{ij} = 1 if connected, 0 otherwise
    - Kernel: $W_{i,j} = exp\left(-\frac{\|x_i - x_j\|^2}{t}\right)$ if connected, 0 otherwise

Non-linear dimensionality reduction

- **Three methods to define lower dimensional embedding of data instances:**
  - Isomaps
  - Locally Embedding Space (LLE)
  - Eigenmaps
- All of these rely on the neighborhood graph connecting only data instances close to each other

- First let us introduce Multi-dimensional scaling (MDS) method …

Multidimensional Scaling (MDS)

- MDS is a classical approach that can map the original high dimensional space to a lower dimensional space. It attempts to preserve the pairwise distance among the data points.
- It is used when we want to visualize high dimensional data say in 2 or 3D

Multidimensional Scaling (MDS)

Idea: MDS maps points to a low dimensional space (say of dimension k) such that the Euclidean distances between the points in this new space approximate the original distance matrix.

$$\Delta := \begin{pmatrix} \delta_{1,1} & \delta_{1,2} & \dots & \delta_{1,I} \\ \delta_{2,1} & \delta_{2,2} & \dots & \delta_{2,I} \\ \vdots & \vdots & & \vdots \\ \delta_{I,1} & \delta_{I,2} & \dots & \delta_{I,I} \end{pmatrix}$$

Map input points x_i to z_i such that $\|z_i - z_j\| \text{»} d_{ij}$

- Classical MDS: the norm $\|\cdot\|$ is the Euclidean distance
- Objective function:

$$\min_{z_i,\dots,z_I} \sum_{i<j} (\|z_i - z_j\| - \delta_{i,j})^2$$

Isomap

Algorithm

- **Step 1:** Construct the neighborhood graph G with edge weights corresponding to local distances
- **Step 2:** compute the shortest distance between all pairs of points:
  - The shortest distance can be calculated via Floyd or Dijkstra algorithm.

$$\Delta := \begin{pmatrix} \delta_{1,1} & \delta_{1,2} & \dots & \delta_{1,I} \\ \delta_{2,1} & \delta_{2,2} & \dots & \delta_{2,I} \\ \vdots & \vdots & & \vdots \\ \delta_{I,1} & \delta_{I,2} & \dots & \delta_{I,I} \end{pmatrix}$$

- **Step 3:** Construct the k-dimensional embedding
  - Use classical MDS to find a k-dimensional embedding

$$\min_{z_i,\dots,z_I} \sum_{i<j} (\|z_i - z_j\| - \delta_{i,j})^2$$

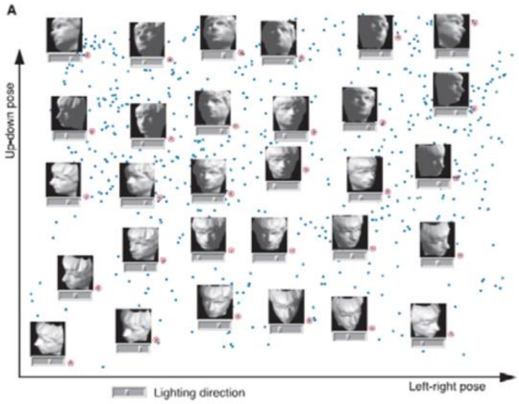
All $z_1, z_2, \dots z_I$ are k-dimensional

Isomap

- **Advantages:**
  - Non-linear dimensionality reduction
  - Non-iterative polynomial time algorithm
  - Guarantee of globally optimality:
    - For intrinsically Euclidean manifolds, a guarantee of asymptotic convergence to the true structure
    - The ability to discover manifolds of arbitrary dimensionality
- **Disadvantage:**
  - Sensitive to noise
  - Few free parameters

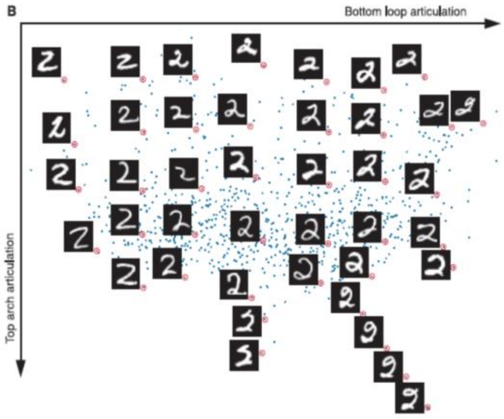
Isomaps: Example

- Dimensionality Reductior for visual perception
  - 64 x 64 image
  - 698 raw images
  - Isomap (k = 6)



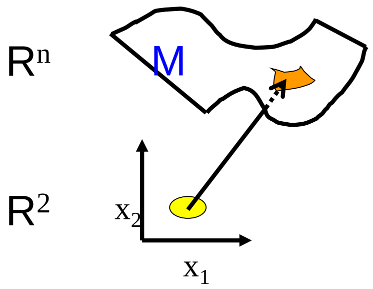
Isomap: Example

- Handwritten ‘2’
  - 1000 handwritten 2s
  - Isomap (ϵ = 4.2)



Locally Linear Embedding (LLE)

- Manifold Characteristics/Key Assumption
  - We expect each data point and its neighbors to lie on or close to a locally linear patch
  - But, how to combine all local patches together?

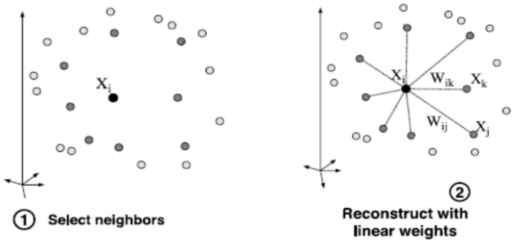


LLE: Intuition

- Assume that manifold is approximately “linear” when viewed locally (in a small neighborhood)
- A good projection should preserve this local geometric property as much as possible

LLE algorithm

- **Step 1:** Select neighbors for each data instance x_i
- **Step 2:** Each data instance is written as a convex combination of its neighbors. Weights of the convex combination 'reconstruct' each point from its neighbors.



LLE Algorithm

- **Step 2:** The weights chosen aim to minimize the reconstruction error.

$$\min_{W} \left\| X_i - \sum_{i=1}^{K} W_{ij} X_j \right\|^2$$

Note: Assign weights under two constraints:

– $W_{ij} = 0$ if X_j does not belong to set of neighbors of X_i

– The rows of the weight matrix sum to one i.e.

$$\sum_{j} W_{ij} = 1$$

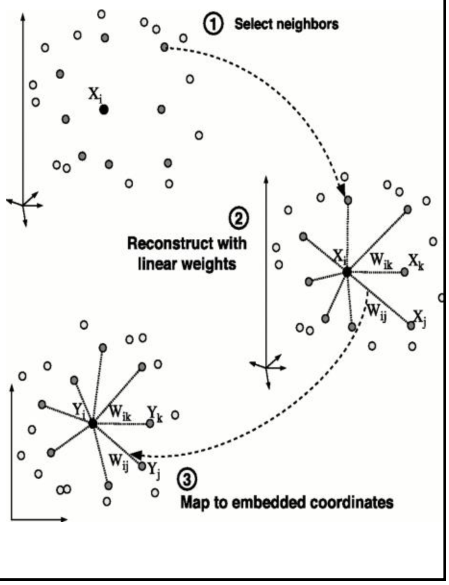
LLE Algorithm

- **Step 3:** Map R^n to a low-dimensional embedding R^k

$$\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}^{R^n} \rightarrow \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}^{R^k}$$

The cost function can be minimized by solving a sparse NxN eigenvalue problem:

$$\min_y \underset{i=1}{\text{å}} \left\| Y_i - \underset{j}{\text{å}} W_{ij} Y_j \right\|^2$$



LLE: Eigenvalue Problem

The following is a more direct and simpler derivation for Y:

Define $M = (I - W)^T (I - W)$

$$\text{F}(Y) = \underset{i}{\text{å}} \left\| Y_i - \underset{j}{\text{å}} W_{ij} Y_j \right\|^2 = \underset{i}{\text{å}} \left\| Y_i - [Y_1, Y_2, \ldots, Y_n] W_i^T \right\|^2$$

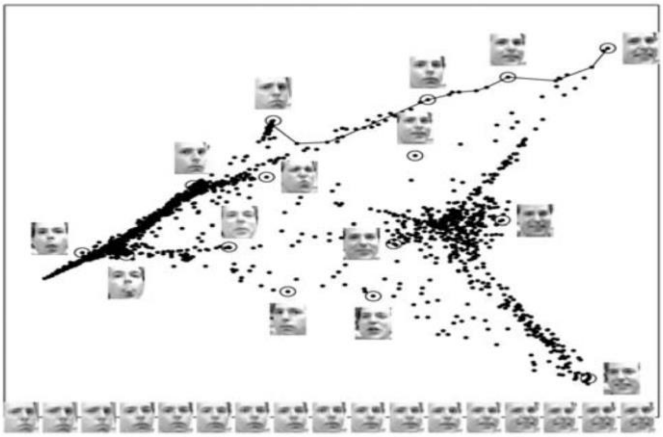
$$= \left\| [Y_1, Y_2, \ldots, Y_n] - [Y_1, Y_2, \ldots, Y_n][W_1^T, W_2^T, \ldots, W_n^T] \right\|_F^2$$

$$= \left\| Y - YW^T \right\|_F^2 = \left\| Y(I - W^T) \right\|_F^2 = trace(Y(I - W)^T (I - W) Y^T)$$

$$= trace(YMY^T)$$

where $Y = [Y_1, Y_2, \ldots, Y_n]$

LLE Example



Images of faces mapped into the embedding space described by the first two coordinates of LLE. Representative faces are shown next to circled points. The bottom images correspond to points along the top-right path (linked by solid line) illustrating one particular mode of variability in pose and expression.

LLE: effect of k

- Require dense data points on the manifold for good estimation

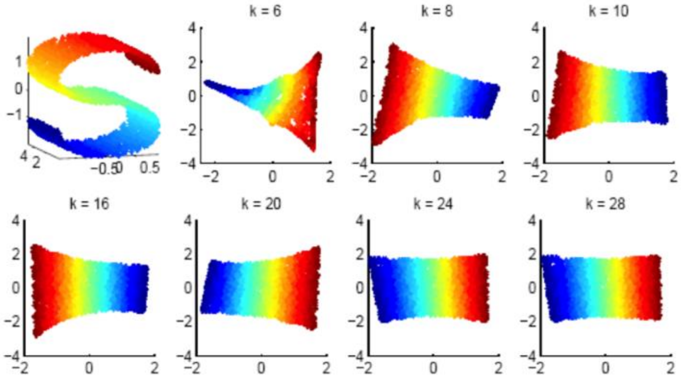


FIG. 5. *S-curve (top left) and computed 2D coordinates by LLE with various neighborhood size k.*

Limitations of LLE

- require dense data points on the manifold for good estimation
- A good neighborhood seems essential to their success
  - How to choose k?
  - Too few neighbors
    - Results in rank deficient tangent space and lead to over-fitting
  - Too many neighbors
    - Tangent space will not match local geometry well

Laplacian Eigenmaps

- Problem: Given a set (x_1, x_2, …, x_n) of n points in R^d, find a set of points (y_1, y_2,…,y_n) in R^k (k << d) such that y_i represents x_i.
- Steps
  - Build the adjacency graph
  - Choose the weights for edges in the graph
  - Eigen-decomposition of the graph Laplacian
  - Form the low-dimensional embedding

Laplacian Eigenmaps Algorithm

- **Step 1:** Construct the neighborhood graph with weights modeling local similarities:
  - Simple: 1 if connected; 0 otherwise
  - Kernels: Gaussian or Heat kernels

$$W_{ij} = e^{-\frac{\|x_i - x_j\|^2}{t}}$$

- **Step 2:** Construct Graph-Laplacian matrix
  - Construct diagonal weight matrix D from the weight matrix

    $$D_{ii} = \sum_j W_{ji}$$
  - Construct Laplacian matrix L = D-W
  - Laplacian is a symmetric, positive semi-definite matrix

Laplacian Eigenmaps Algorithm

- **Step 3:** Finding of the lower dimensional embedding
  - Find $Y = (y_1, y_2, \ldots, y_n)^T$ that preserves local similarities
  - Note: each y is low-dimensional

Objective: minimize

$$\sum_{ij} (y_i - y_j)^2 W_{ij}$$

- The above objective function can be rewritten as:

$$\frac{1}{2}\sum_{i,j} (y_i - y_j)^2 W_{ij} = Y^T L Y$$

- **Solution:** Compute eigenvalues and eigenvectors of the generalized eigenvector problem

Generalized Eigenvector Problem

- Suppose we have the n points such that (x_1, x_2, …, x_n) in R^d
- Construct the weight matrix between each point

$$W = \begin{pmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & w_{22} & \cdots & w_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n1} & w_{n1} & \cdots & w_{nn} \end{pmatrix}$$

- Based on equation $|W - / I| = 0$, we can compute the Eigenvalues $/ = (/_0, /_1, \ldots, /_n)$

Euclidean Embedding Space

Once we have all the Eigenvalues, then we solve equation $Lv = / Dv$ to compute the Eigenvector v

$$Lv = / Dv$$

$$Lv - / Dv = 0$$

$$(L - / D)v = 0$$

Since $(L - / D)$has the dimension $n * n$

Then Eigenvector v has the dimension $n * 1$

Euclidean Embedding Space

- Let $v_0, v_1, \ldots, v_{n-1}$ be the eigenvector solutions to the equation $Lv = / Dv$, ordered according to their eigenvalues,

$$\begin{aligned}
&Lv_0 = /_0 Dv_0 \\
&Lv_1 = /_1 Dv_1 \\
&\ldots \\
&Lv_{n-1} = /_{n-1} Dv_{n-1} \\
&Lv_n = /_n Dv_n \\
&0 = /_0 \text{ £ } /_1 \text{ £ } \ldots \text{ £ } /_{n-1}
\end{aligned}$$

Euclidean Embedding Space

- Then we leave out the eigenvector v_0 corresponding to eigenvalue $/_0$ and use the next k eigenvectors $(v_1, v_2, \ldots, v_k)$ for embedding in k-dimensional Euclidean space.
- The k-dimensional embedding space is denoted as $Y = (y_1, y_2, \ldots, y_k)^T$

Euclidean Embedding Space

Once we find all Eigenvectors, then, we can project the data points to lower dimension R^k embedding space

$$X=(x_1,x_2,\dots,x_n)^{R^d}\to Y=\begin{array}{c}\begin{array}{cccc} y_1 & y_2 & \dots & y_k \end{array}\\ \begin{pmatrix} v_1(1) & v_2(1) & \dots & v_k(1) \\ v_1(2) & v_2(2) & \dots & v_k(2) \\ \vdots & \vdots & \ddots & \vdots \\ v_1(n) & v_2(n) & \dots & v_k(n) \end{pmatrix}^T\end{array}$$

We reduce the dimension from n * d to n * k

$y_1=(v_1(1),v_1(2),\dots,v_1(n))$ represents the coordinates of all n data points at 1st dimension in the embedding space.

Optimal embedding space

- Objective function:

$$\min\frac{1}{2}\sum_{i,j}(y_i - y_j)^2 W_{ij} = \underset{\substack{y^T Dy=1\\ y^T D1=0}}{\operatorname{argmin}}\, y^T Ly$$

- The constraint $y^T Dy=1$ removes arbitrary scaling factor
- The constraint $y^T D1=0$ removes a translation invariance in y

Laplacian Eigenmaps: Embedding Space

After the optimization, each data point can be maped into the optimal k-dimensional embedding space $Y = (y_1, y_2, \ldots, y_k)^T$

$$x_i^{R^d} \rightarrow (v_1(i), v_2(i), \ldots, v_k(i))$$

$v_j(i)$ is the coordinate of point x_i at jth dimension at k-dimensional space, where $1 £ j £ k$

Laplacian Eigenmaps Example

- Suppose we want to project $X = \{x_1, x_2, \ldots, x_n\}^{R^m}$ into 2 dimensional space
- Let $\{/_0, /_1, /_2 \ldots, /_n\}$ to be the calculated eigenvalues with increasing order $0 = /_0 £ /_1 £ \ldots £ /_n$
- Let $v = \{v_1, v_2, \ldots, v_n\}$ be the eigenvector solutions of equation $Lv = / Dv$
- Take $\{v_1, v_2\}$ for embedding in 2-dimensional space

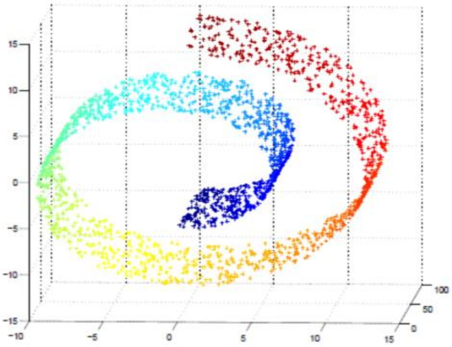
$$\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}^{R^m} \rightarrow \begin{pmatrix} v_1(1) & v_2(1) \\ v_1(2) & v_2(2) \\ \vdots & \vdots \\ v_1(n) & v_2(n) \end{pmatrix}^{R^2}$$

Locally Linear Embedding and Laplacian Eigenmap

- LLE is connected with Laplacian Eigenmap
- LLE minimizes $Y^T (I - W)^T (I - W) y$ to reduce eigenvectors of $(I - W)^T (I - W)$
- Actually, finding eigenvectors of $(I - W)^T (I - W)$ can be re-interpreted as finding eigenvectors of iterated Laplacian L^2.

Laplacian Eigenmap Example

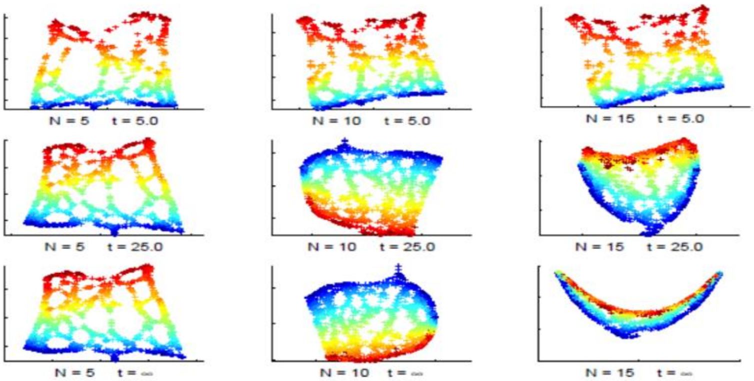
- Swiss roll



2000 random data points on the manifold

Laplacian Eigenmap Example

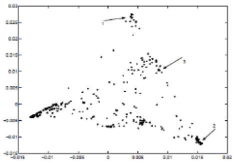
- 2D embedding of the swiss roll

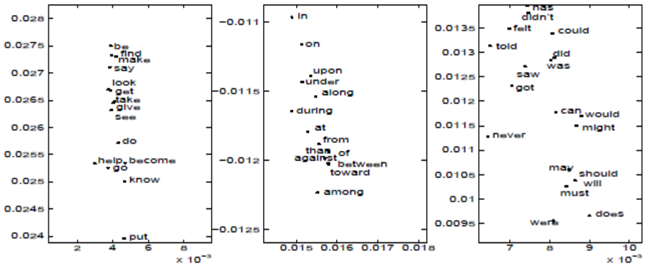


Free parameters, N and t. N = Number of neighbors, t = Heat kernel parameter

Laplacian Eigenmap Example

- 300 most frequent words from Brown corpus
- Each word is represented by a 600 dimensional vector
- Laplacian Eigenmap with N = 14, t = inf





Framgents labeled by arrows, from left to right. The first is exclusively infinites of verbs, the second contains prepositions and the third mostly modal and auxiliariy verbs

Summary

- Isomap, LLE and Laplacian Eigenmap: non-linear dimensionality reduction technique
- Useful for learning manifolds, understanding low dimensional data embedded in high dimensional space.
- Linear dimensionality reduction technique (PCA, SVD) fails for this type of data.
- All three can preserve local geometry (inter-point relationships)

References and acknowledgments

- Roweis, S. T. & Saul, L. K. (2000), 'Locally linear embedding', Science 290, 2323–2326.
- Tenenbaum, J. B., de Silva, V. & Langford, J. C. (2000), 'A global geometric framework for nonlinear dimensionality reduction', Science 290, 2319–2323.
- http://www.cs.nyu.edu/~roweis/lle/
- http://isomap.stanford.edu/
- http://en.wikipedia.org/wiki/Nonlinear_dimensionality_reduction
- http://web.mit.edu/6.454/www/www_fall_2003/ihler/slides.pdf
- http://cseweb.ucsd.edu/~saul/teaching/cse291s07/laplacian.pdf
- www.public.asu.edu/~jye02/CLASSES/Spring.../Lec19-Isomap.ppt
- www.public.asu.edu/~jye02/CLASSES/Spring.../Lec20-LLE.ppt
- www.public.asu.edu/~jye02/CLASSES/.../Lec21-LaplacianEig.ppt