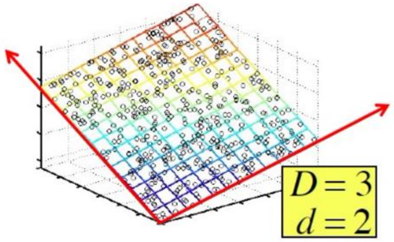
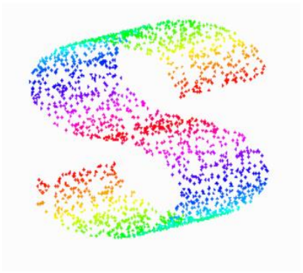
Nonlinear Dimensionality Reduction: Manifold Learning Methods (IsoMAP, LLE & LE)

Linear dimensionality reduction

Principal Component Analysis

- Finds a low-dimensional feature space (whose basis are principal components) where the variance of the projected data is maximized





Non-linear dimensionality reduction

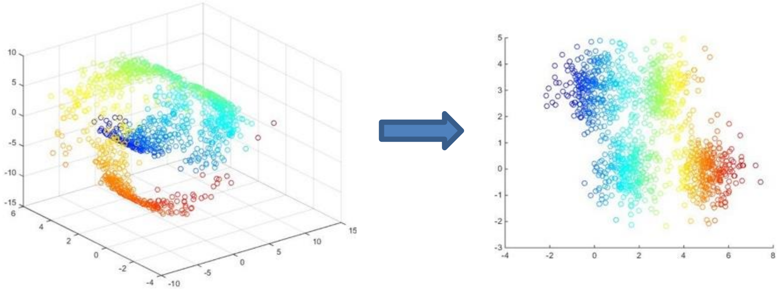
- Find a ***nonlinear*** low dimensional representation of data that reflects the ***geometric*** properties of data

- Manifold learning methods covered
  - Global view
    - Isometric mapping (Isomap) (Science'2000)
  - Local view
    - Locally Linear Embedding (LLE) (Science'2000)
    - Laplacian Eigenmaps (LE) (NeuComp 2001)

Manifold Learning

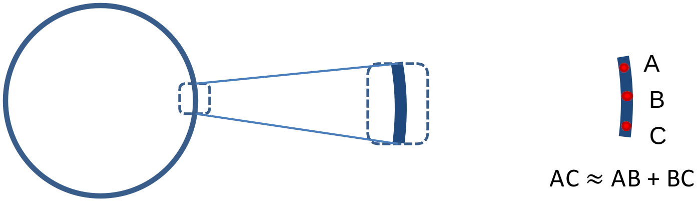
Recovering low dimensional data representation non-linearly embedded within a higher dimensional space

Key assumption: It’s simpler than it looks!



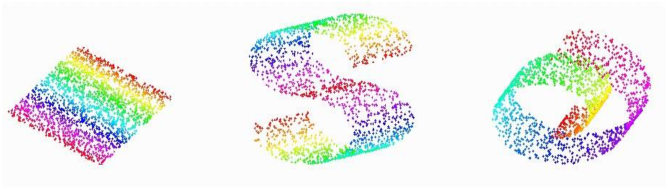
- Key assumption: High dimensional data actually resides in a lower dimensional space that is ***locally Euclidean***
- Informally, *<u>manifold</u>* is a subset of points in the high-dim space that locally looks like a low-dime space

Manifold example



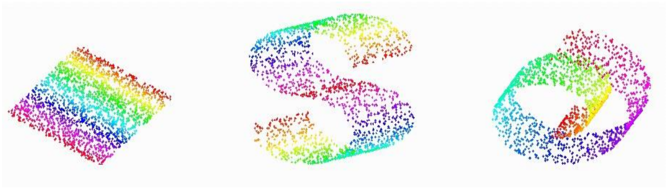
- Informally, ***manifold*** is a subset of points in the high-dim space that ***locally*** looks like a low-dim space
- Example: arc of a circle
  - consider a tiny bit of a circumference (2D) → can treat as line (1D)

More 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

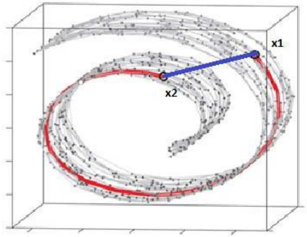
More Manifolds



- PCA: works for the 1st one, but the last two
  - Linear subspace does not explain it well
- What do we mean by “consistent locations”?
  - Preserve local relationships and structure
  - One possibility: ***preserve distances***

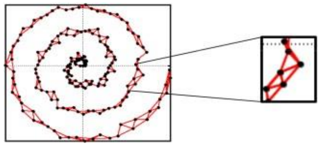
Manifolds and local distances

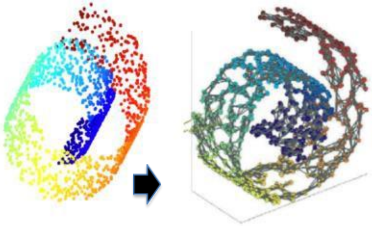
- In general, the distances induced by data (manifolds) may not be Euclidean. That is, the global distances don't respect the geometry.
- Local distances can be still approximated with Euclidean distances
- **Idea for the dimensionality reduction:**
  - Define global distances/similarity in terms of 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 Similarity/Neighborhood graph**



Similarity Graph

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



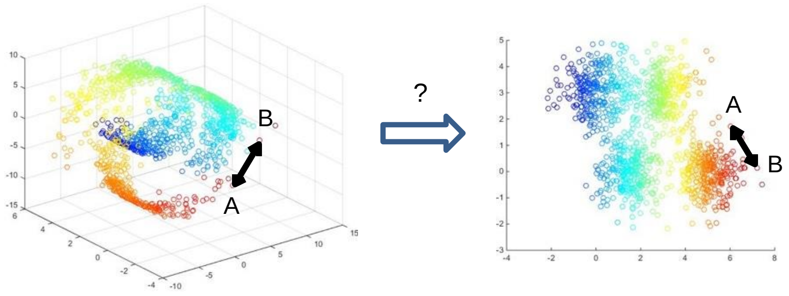


All manifold learning-based non-linear dimensionality reduction methods rely on the neighborhood graph

Geodesic Distances and Isomap

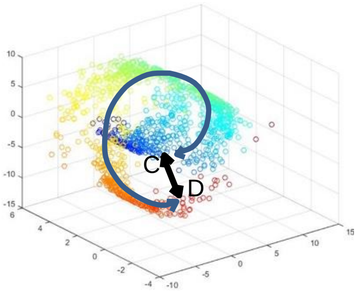
A non-linear dimensionality reduction algorithm that preserves locality information using ***geodesic distances***

General idea: Dimensionality reduction



- Find a lower dimensional representation of data that preserves pairwise distances between points
  - Multidimensional scaling (MDS)

“Global distances” VS geodesic distances



- “Global distances” cause a problem: we may not want to preserve them
- We are interested in preserving distances along the manifold

geodesic distances

Multidimensional scaling (MDS)

MDS is a classical approach that can map data points in the original high-dim space to a lower-dim space

- Idea: preserve pairwise Euclidean distance btw data
- Define similarity matrix Δ

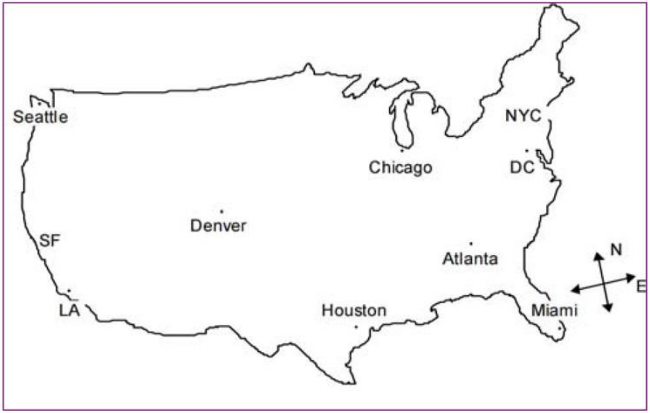
$$\Delta := \begin{pmatrix} \delta_{1,1} & \delta_{1,2} & \cdots & \delta_{1,I} \\ \delta_{2,1} & \delta_{2,2} & \cdots & \delta_{2,I} \\ \vdots & \vdots & & \vdots \\ \delta_{I,1} & \delta_{I,2} & \cdots & \delta_{I,I} \end{pmatrix}$$

- Objective: map input data x_i to z_i such that

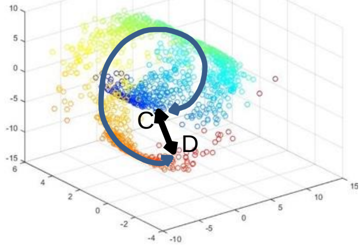
$$\min_{z_i,\ldots,z_I} \sum_{i<j} (\|z_i - z_j\| - \delta_{i,j})^2$$

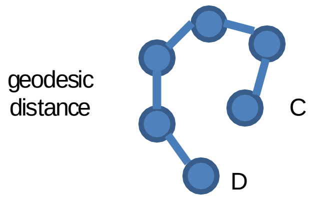
MDS Example

Given pairwise Euclidean distances (Δ matrix) between different cities in 3D, plot the cities on a 2D plane (recover location)



Similarity graph models data geometry





- Geodesic distances can be approximated **via a similarity graph**
    - in which nodes *i* represent the data point x_i
    - edge weight w_{ij} indicates the similarity of node *i* and node j
- Similarity graph captures data geometry
- Let d(i,j) be the Euclidean distance between the points i and j in the original space
- How to build similarity graph?

Similarity graph construction

ε-neighborhood graph

- If a pairwise distance d(i,j) between two data points xi and xj is d(xi, xj) $\leq$ ε, connect xi and xj
- Similarity: 1- ε

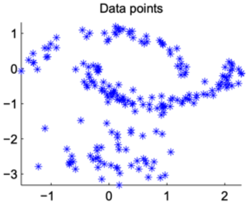
k-nearest neighbor (k-nn) graph

- If xi is one of the k nearest neighbors of xj, connect xi and xj
- Similarity (Gaussian kernel)

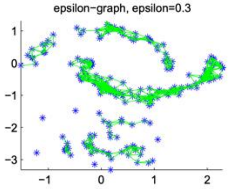
$$w_{ij} = \exp(-\| x_i - x_j \|^2 / \sigma^2)$$

Mutual k-nn graph

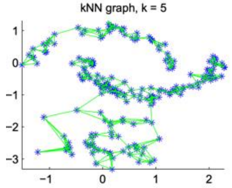
- If xi is one of the k nearest neighbors of xj
- AND xj is one of the k nearest neighbors of xi, connect them



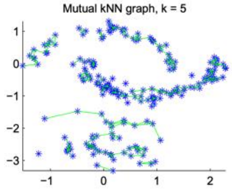
2 moons with
different densities
+ 1 Gaussian



High density region
densely connected;
Low density region
barely connected

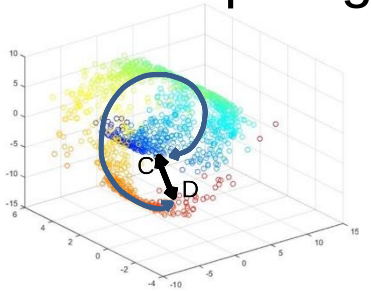


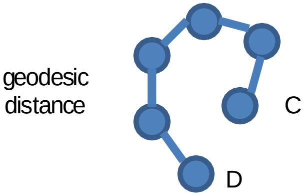
Connect data points
“on different scales”



Connect data points
within regions of
constant density

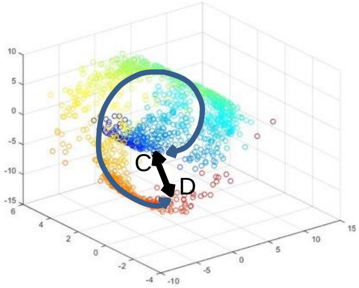
Computing geodesic distances

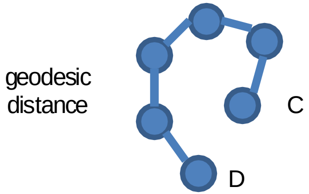




- Given the similarity graph, compute shortest paths between each pair of points
  - E.g., using the Floyd-Warshall algorithm
- Set geodesic distance between nodes i and j to the length (sum of weights) of the shortest path between them
- Define a new similarity matrix Δ based on geodesic distances

Isomap: summary

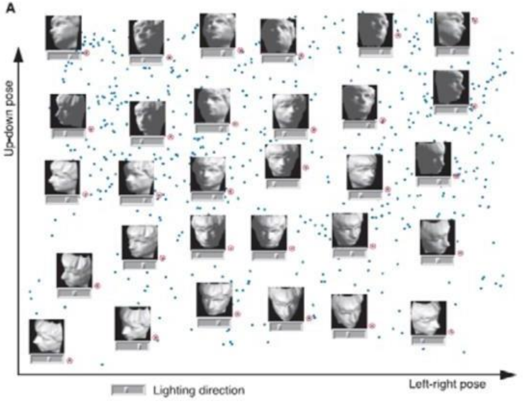




1. Construct the similarity graph G
2. Compute shortest paths for all node pairs
3. Geodesic distances are the lengths of the shortest paths
4. Construct similarity matrix using geodesic distances
5. Apply MDS on the similarity matrix

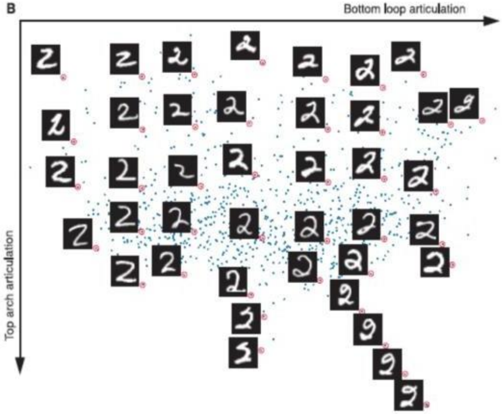
Isomaps: Example

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



Isomap: Example

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



Isomap

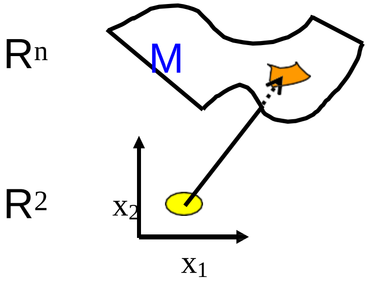
- **Global view**
- **Advantages:**
  - Non-linear dimensionality reduction
  - Non-iterative polynomial time algorithm
  - Guarantee of globally optimality:
    - For intrinsically Euclidean manifolds, a guarantee of asymptotic convergence to the true structure
    - The ability to discover manifolds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

Local Reconstruction and LLE

A non-linear dimensionality reduction algorithm that preserves locality information based on ***local reconstruction***

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?



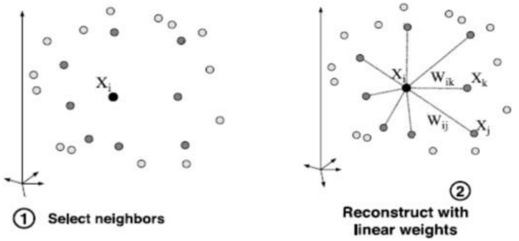
LLE: Intuition

- 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

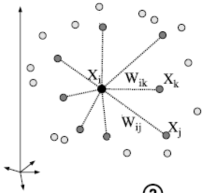
- **Step 1:** Select neighbors for each data instance x_i
- **Step 2:** Each data instance is written as a convex combination of its neighbors. Weights of the convex combination 'reconstruct' each point from its neighbors.



LLE Algorithm

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

$$\min \left\| X_i - \sum_{i=1}^{k} W_{ij} X_j \right\|^2$$



② Reconstruct with linear weights

Note: Assign weights under two constraints:

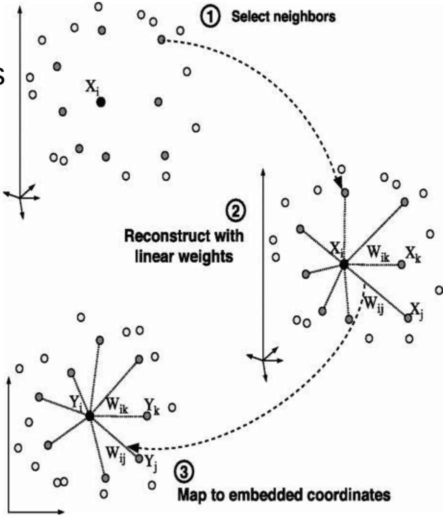
- $W_{ij} = 0$ if X_j does not belong to set of neighbors of X_i
- The rows of the weight matrix sum to one i.e. $\sum_j W_{ij} = 1$
- Closed-form solution
  - **Constrained Least Squares Problem**

LLE Algorithm

- **Step 3:** Map R^n to a low-dimensional embedding R^k
- Each high-dim observation X_i is mapped to a low-dim vector Y_i
- Choose Y_i to minimize the embedding cost function

$$min_Y \sum_i \left\| Y_i - \sum_j W_{ij} Y_j \right\|^2$$

The cost function can be minimized by solving a **sparse NxN eigenvalue problem** (appendix B in the paper).



LLE: Eigenvalue Problem

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

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

$$\Phi(Y) = \sum_i \left\| Y_i - \sum_j W_{ij} Y_j \right\|^2 = \sum_i \left\| Y_i - [Y_1, Y_2, \ldots, Y_n] W_i^T \right\|^2$$

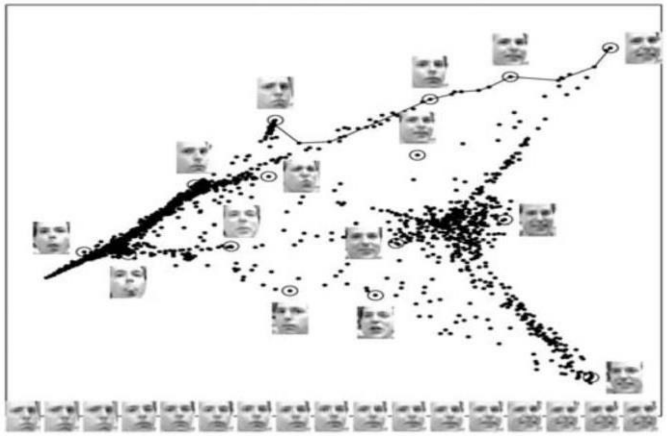
$$= \left\| [Y_1, Y_2, \ldots, Y_n] - [Y_1, Y_2, \ldots, Y_n][W_1^T, W_2^T, \ldots, W_n^T] \right\|_F^2$$

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

$$= trace(YMY^T)$$

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

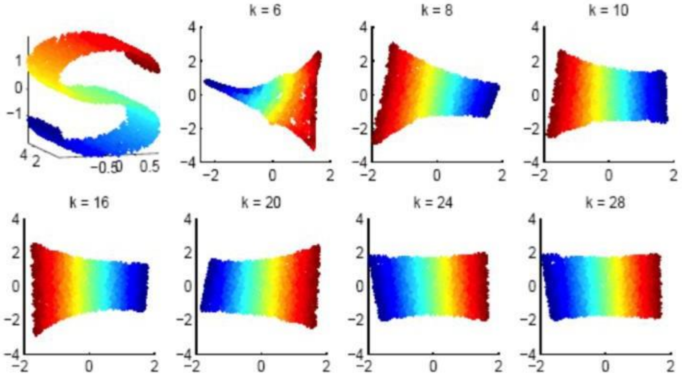


FIG. 5. *S-curve (top left) and computed 2D coordinates by LLE with various neighborhood size k.*

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

Graph Spectral Theory & Laplacian Eigenmaps (LE)

A graph spectral theory approach to non-linear dimensionality reduction algorithm

Laplacian Eigenmaps

- Problem: Given a set $(x_1, x_2, \ldots, x_n)$ of n points in R^d, find a set of points $(y_1, y_2, \ldots, y_n)$ 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

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

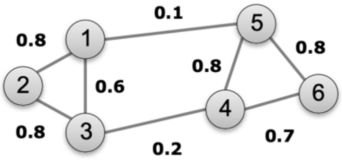
- **Step 2:** Construct Graph-Laplacian matrix
  - Construct diagonal weight matrix D from the weight matrix

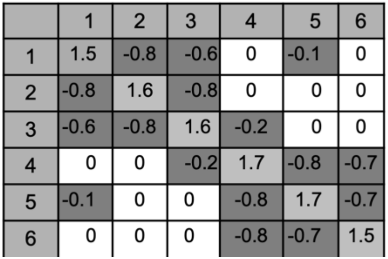
$$D_{ii} = \sum_j W_{ji}$$

  - Construct Laplacian matrix L=D-W
  - Laplacian is a symmetric, positive semi-definite matrix

Graph Laplacian matrix (L): Weighted graph

- ***N* x *N*** diagonal matrix
- **L=D - W**





	1	2	3	4	5	6
1	1.5	-0.8	-0.6	0	-0.1	0
2	-0.8	1.6	-0.8	0	0	0
3	-0.6	-0.8	1.6	-0.2	0	0
4	0	0	-0.2	1.7	-0.8	-0.7
5	-0.1	0	0	-0.8	1.7	-0.7
6	0	0	0	-0.8	-0.7	1.5

Properties of Graph Laplacian

- L is symmetric, i.e., $(L^T = L)$
  - Eigenvectors are real-valued and orthogonal

- L is a positive semi-definite (PSD) matrix
  - For all real-valued vectors x: $x^T L x \geq 0$

$$x^T Lx = x^T (D - W)x = x^T Dx - x^T Wx$$

$$= \sum_i d_i x_i^2 - \sum_{i,j} w_{ij} x_i x_j \qquad (\text{where } d_i = \textstyle\sum_j w_{ij})$$

$$= \frac{1}{2}\left(\sum_{i,j} w_{ij} x_i^2 - 2 \sum_{i,j} w_{ij} x_i x_j + \sum_{i,j} w_{ij} x_i^2 \right)$$

$$= \frac{1}{2} \sum_{i,j=1}^{N} w_{ij} (x_i - x_j)^2 \geq 0$$

  - All eigenvalues of L are ≥0

Laplacian Eigenmaps Algorithm

- **Step 3:** Finding of the lower dimensional embedding
  - Find $Y = (y_1, y_2, ..., y_n)^T$ that preserves local similarities
  - **Objective:** minimize

$$\sum_{ij} (y_i - y_j)^2 W_{ij} = Y^T L Y$$

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

$$L\mathbf{v} = \lambda D\mathbf{v}$$

Euclidean Embedding Space

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

$$
\begin{aligned}
&Lv_0 = \lambda_0 D v_0 \\
&Lv_1 = \lambda_1 D v_1 \\
&\ldots \\
&Lv_{n-1} = \lambda_{n-1} D v_{n-1} \\
&Lv_n = \lambda_n D v_n \\
&0 = \lambda_0 \leq \lambda_1 \leq \ldots \leq \lambda_{n-1}
\end{aligned}
$$

Laplacian Eigenmaps (LE)

- Leave out the eigenvector v_0 w.r.t. eigenvalue $\lambda_0 = 0$ and use next k eigenvectors $(v_1, v_2, ..., v_k)$ for embedding in k-dim Euclidean space.

- Each data point x can be mapped into

$$x_i \rightarrow (v_1(i), v_2(i), \ldots, v_k(i))$$

$v_j(i)$ is the coordinate of point x_i at jth dimension at k-dimensional space

Locally Linear Embedding (LLE) vs. Laplacian Eigenmap (LE)

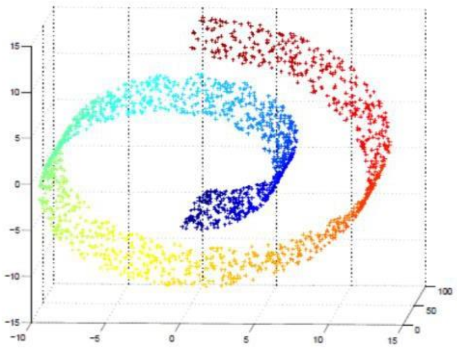
- LE is connected with Laplacian Eigenmap

- LLE minimizes $Y^T (I - W)^T (I - W) Y$ w.r.t. 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 graph Laplacian L^2

Laplacian Eigenmap Example

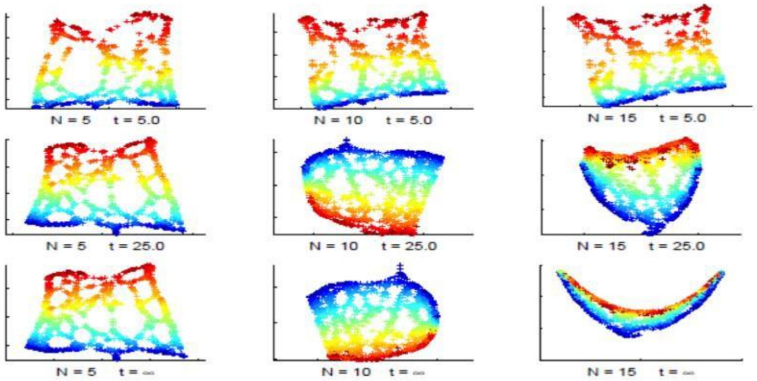
- Swiss roll



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

Summary

- Manifold learning-based *non-linear* dim reduction
    - Isomap: Global view
    - LLE and LE: Local view
- Understand low-dim data embedded in high-dim space

- Linear dim reduction technique (PCA, SVD, NMF) fails for this type of data

- All three preserve local geometry (inter-point relations)

References and acknowledgments

- Roweis, S.T.& Saul, L. K.(2000), ‘Nonlinear dimensionality reduction by locally linear embedding’, Science
- Tenenbaum, J.B., de Silva, V.& Langford, J.C.(2000), ‘A global geometric framework for nonlinear dimensionality reduction’, Science
- Belkin, Mikhail, and Partha Niyogi. (2003) "Laplacian eigenmaps for dimensionality reduction and data representation." Neural computation
- 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
- https://trevorcohn.github.io/comp90051-2017/slides/16_manifold_learning.pdf