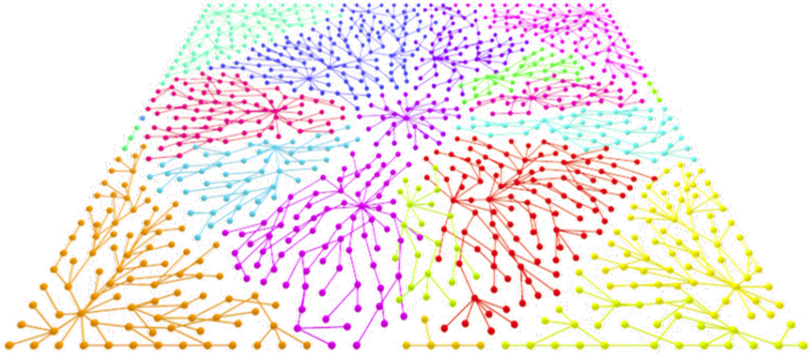
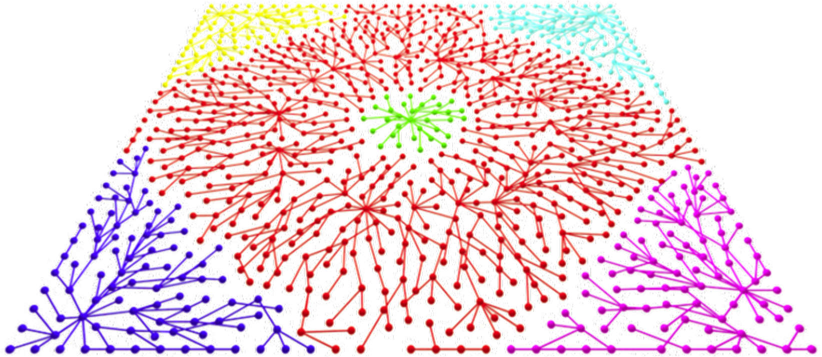
CS233, CME251: Geometric and Topological Data Analysis

Leonidas Guibas
Computer Science Department
Stanford University



Lecture 6
22 April 2020



Last Time:
Graph Methods and Spectral Approaches

Spectral Graph Theory

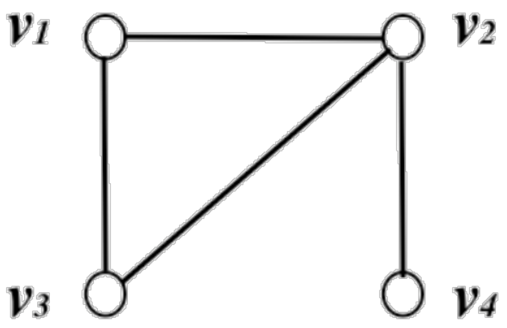
- The *spectral graph theory* studies the properties of graphs via the eigenvalues and eigenvectors of their associated graph matrices: the *adjacency matrix* and the *graph Laplacian* and its variants.
- Both matrices have been extremely well studied from an algebraic point of view.
- The Laplacian allows a natural link between discrete representations, such as graphs, and continuous representations, such as vector spaces and manifolds.
- The most important application of the Laplacian is *spectral clustering* that corresponds to a computationally tractable solution to the *graph partitionning problem*.
- Another application is *spectral matching* that solves for *graph matching*.

Adjacency Matrices

- For a graph with n vertices, the entries of the $n \times n$ adjacency matrix are defined by:

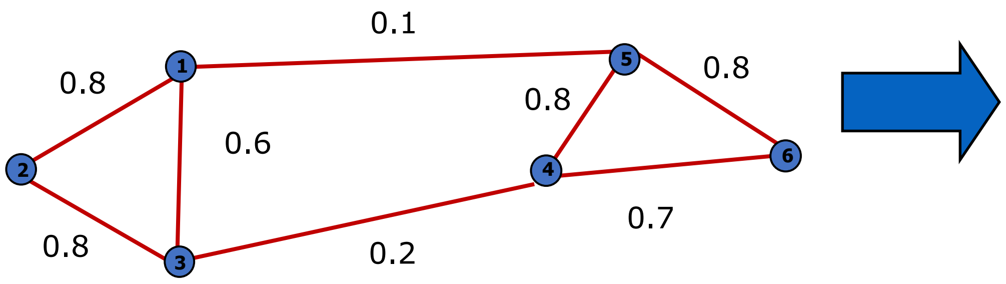
$$\mathbf{A} := \begin{cases} A_{ij} = 1 & \text{if there is an edge } e_{ij} \\ A_{ij} = 0 & \text{if there is no edge} \\ A_{ii} = 0 & \end{cases}$$

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$



Weighted Matrices

- Adjacency matrix (A)
  - n x n matrix
  - $A = [w_{ij}]$ edge weight between vertex x_i and x_j



	x_1	x_2	x_3	x_4	x_5	x_6
x_1	0	0.8	0.6	0	0.1	0
x_2	0.8	0	0.8	0	0	0
x_3	0.6	0.8	0	0.2	0	0
x_4	0	0	0.2	0	0.8	0.7
x_5	0.1	0	0	0.8	0	0.8
x_6	0	0	0	0.7	0.8	0

- Important properties:
  - Symmetric matrix
  - $\Rightarrow$ Eigenvalues are real
  - $\Rightarrow$ Eigenvectors span orthogonal basis

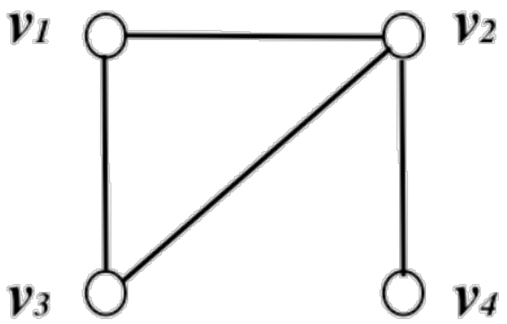
Graph (Unnormalized) Laplacian

- $\mathbf{L} = \nabla^\top \nabla$
- $(\mathbf{L}\boldsymbol{f})(v_i) = \sum_{v_j \sim v_i} (f(v_i) - f(v_j))$
- Connection between the Laplacian and the adjacency matrices:

$$\mathbf{L} = \mathbf{D} - \mathbf{A}$$

- The degree matrix: $\mathbf{D} := D_{ii} = d(v_i)$.

$$\mathbf{L} = \begin{bmatrix} 2 & -1 & -1 & 0 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 2 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix}$$



Laplacian Variants

- The normalized graph Laplacian (symmetric and semi-definite positive):

$$\mathbf{L}_n = \mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$$

- The transition matrix (allows an analogy with Markov chains):

$$\mathbf{L}_t = \mathbf{D}^{-1}\mathbf{A}$$

- The random-walk graph Laplacian:

$$\mathbf{L}_r = \mathbf{D}^{-1}\mathbf{L} = \mathbf{I} - \mathbf{L}_t$$

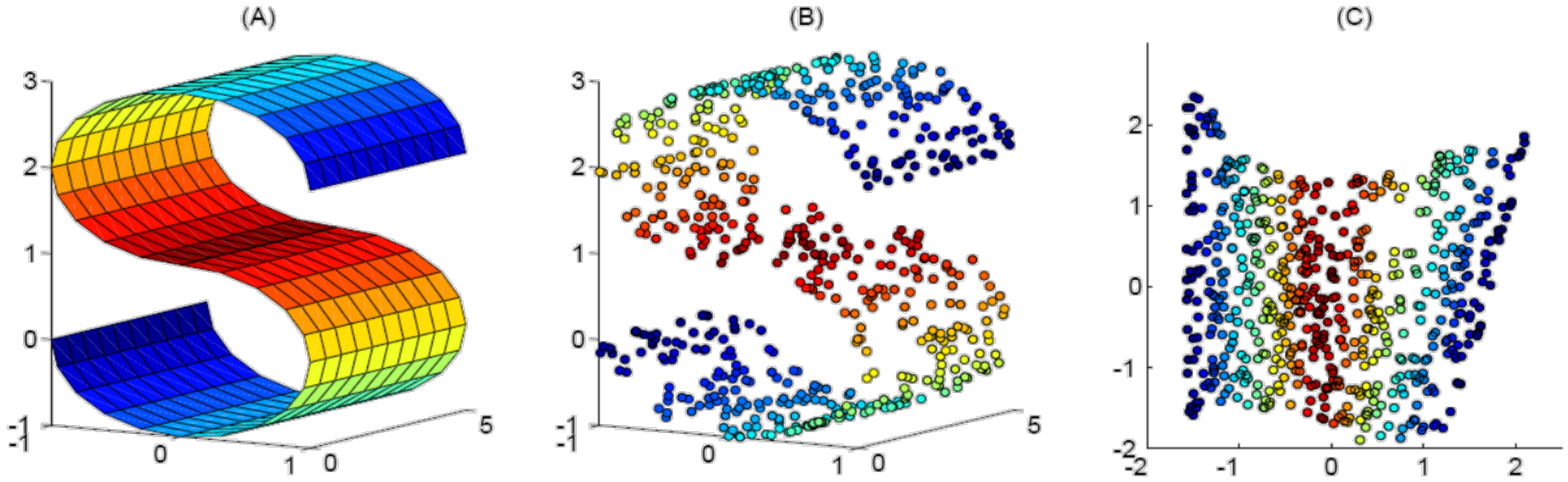
- These matrices are similar:

$$\mathbf{L}_r = \mathbf{D}^{-\frac{1}{2}}\mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}}\mathbf{D}^{\frac{1}{2}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{L}_n\mathbf{D}^{\frac{1}{2}}$$

Non-Linear Dimensionality Reduction

Non-Linear Dimensionality Reduction

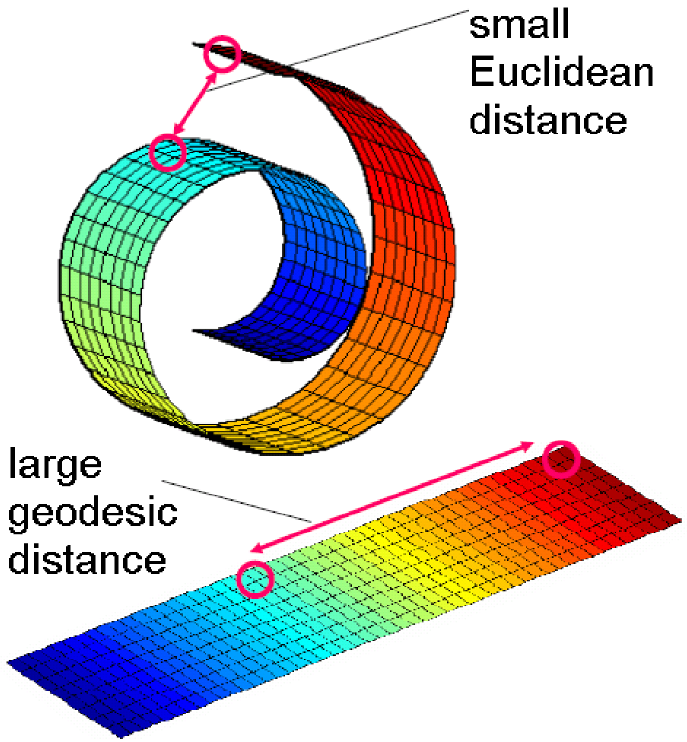
- Why do we need NLDR?
- Many data sets contain essential non-linear structures that “invisible” to PCA and MDS
- Must resort to some non-linear dimensionality reduction approaches



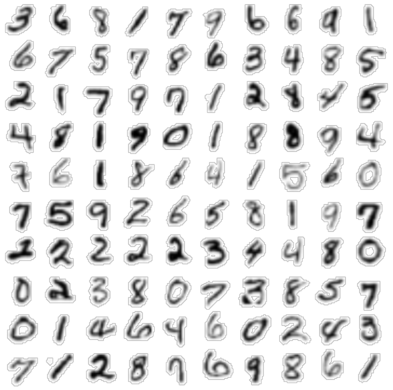
Data sampled from a non-linear manifold

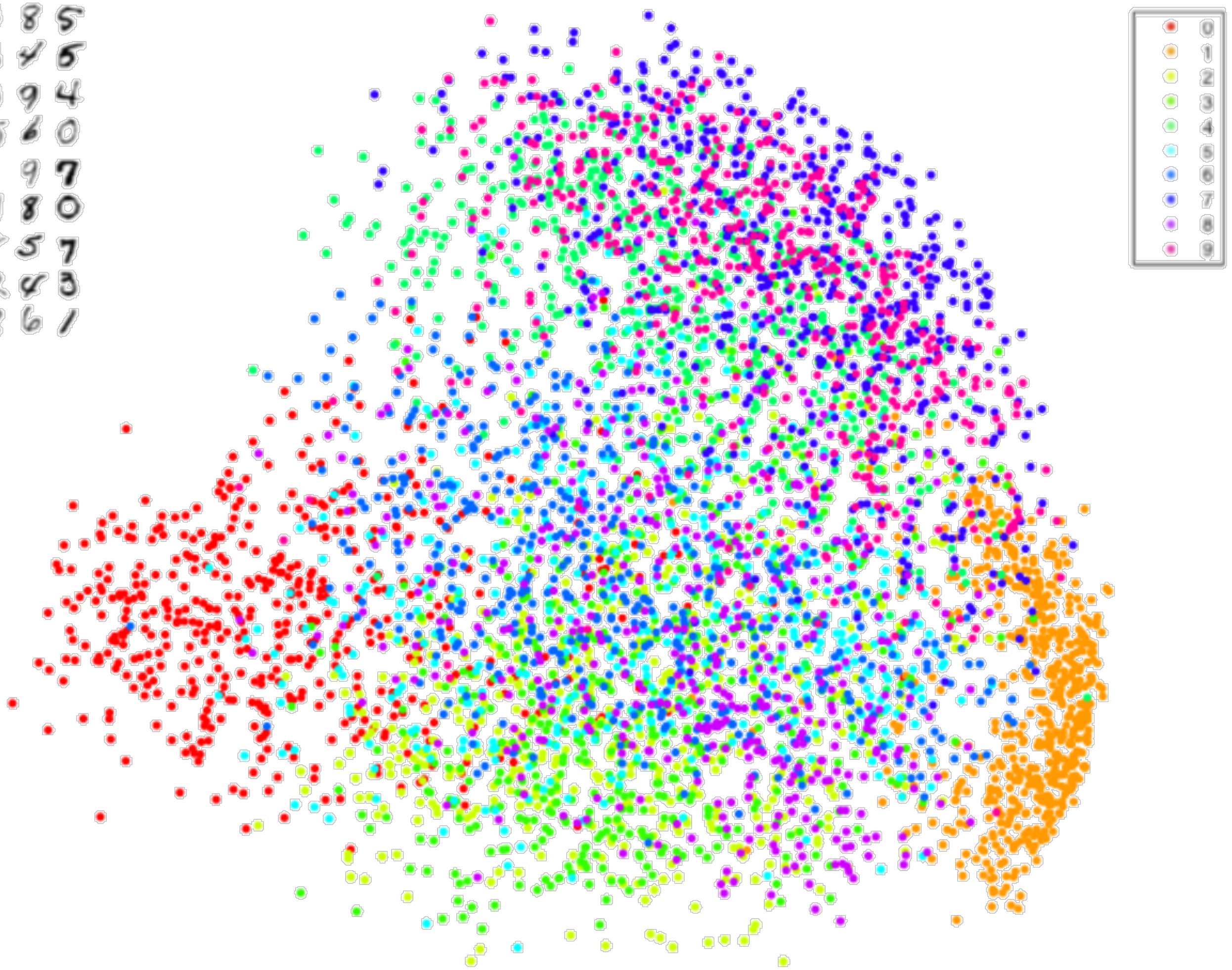
The Choice of Distance

- We can try to capture the manifold structure through the right notion of distance directly on the manifold (geodesic distance)
- That’s why PCA and MDS fail – they only see the Euclidean structures



PCA Cares About Large Distances

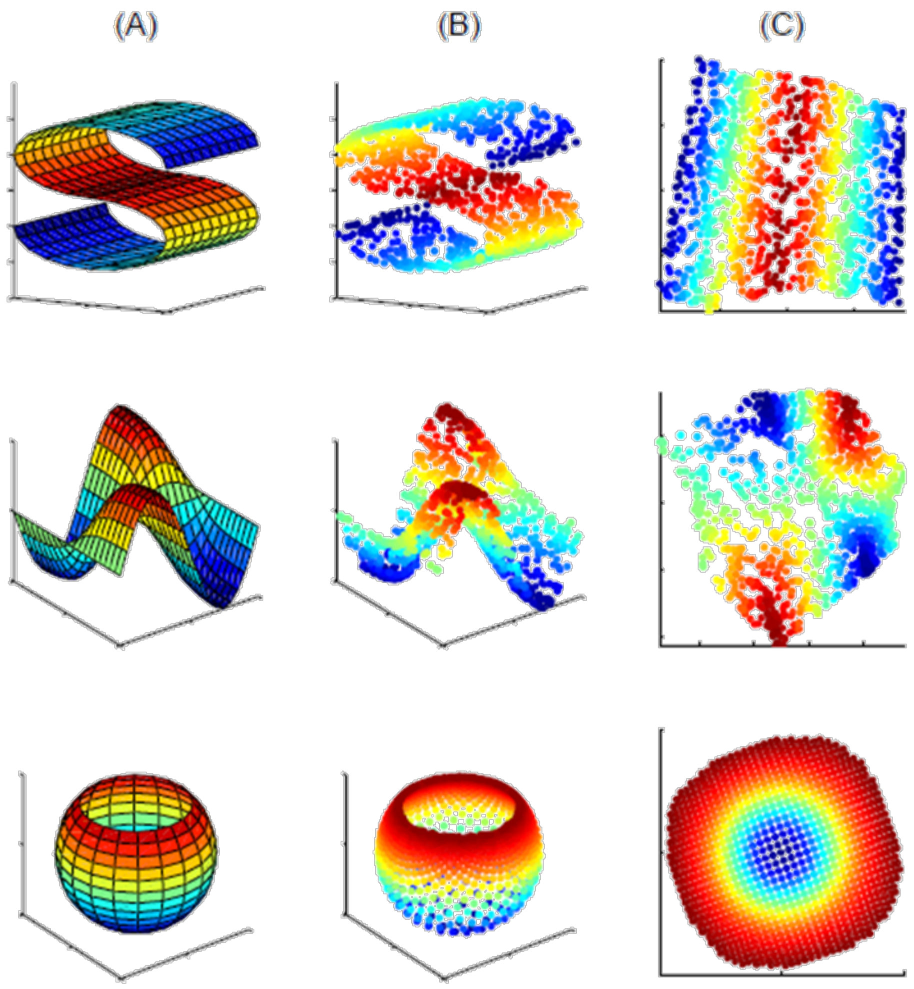




The Challenge of NLDR

- An unsupervised learning algorithm – it must discover the global internal coordinates of the manifold, without external signals that suggest how the data should be embedded in low dimensions

- Build a bridge between high- & low-dimensional spaces

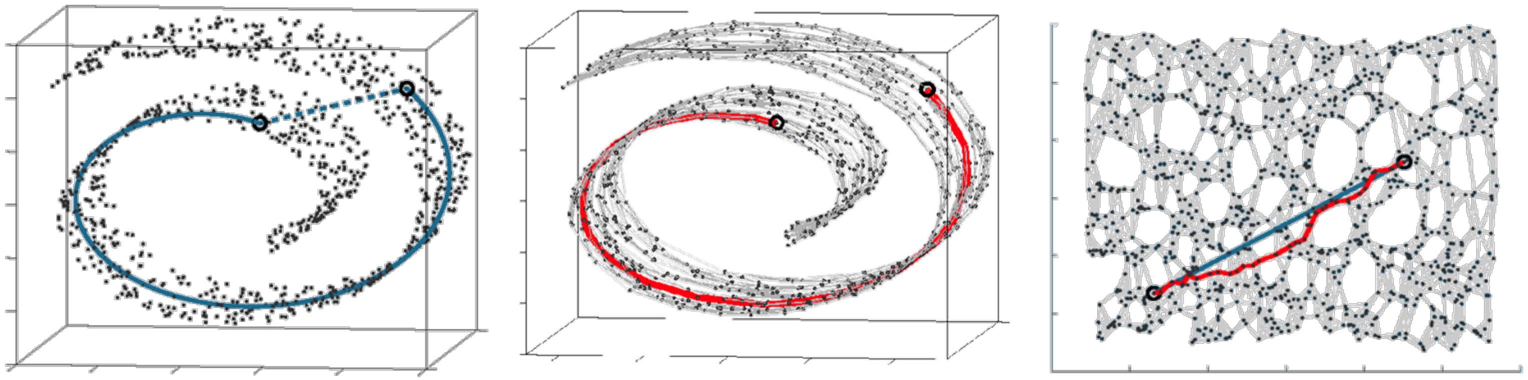


Isomap

ISOMAP (J. B. Tenenbaum, V. de Silva and J. C. Langford)

- Example of non-linear structure (Swiss roll)
  - Only the geodesic distances reflect the true low-dimensional geometry of the manifold

- ISOMAP (Isometric Feature Mapping)
  - Uses the geodesic manifold distances between all pairs
  - Preserves the intrinsic geometry of the data -- Preserves the geodesic distances
  - How to estimate geodesic distances between point pairs?



ISOMAP (Algorithm Description)

- **Step 1**
  - Form a near-neighbor graph G on the original data points, weighing the edges based on their original distances $d_X(i, j)$.

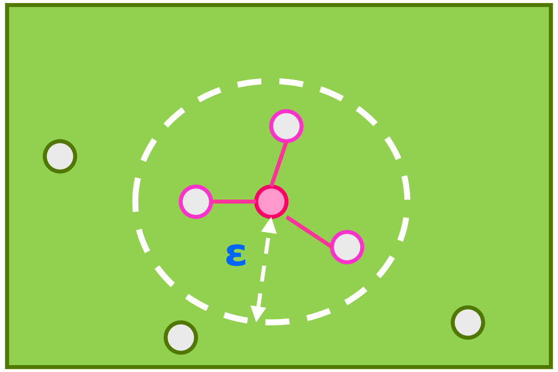
- **Step 2**
  - Estimate the geodesic distances $d_G(i, j)$ between all pairs of points on the sampled manifold by computing their shortest path distances in the graph G.

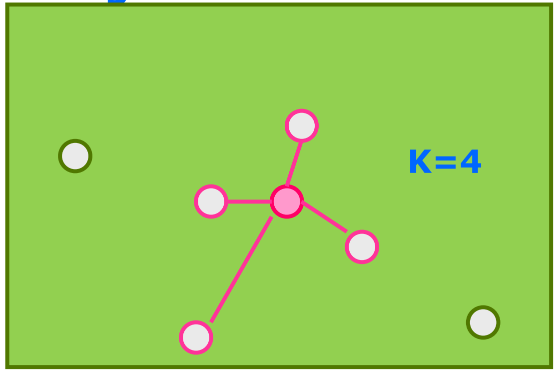
- **Step 3**
  - Construct an embedding of the data in *d*-dimensional Euclidean space *Y* that best preserves the distances (MDS).

Near Neighbor Graph

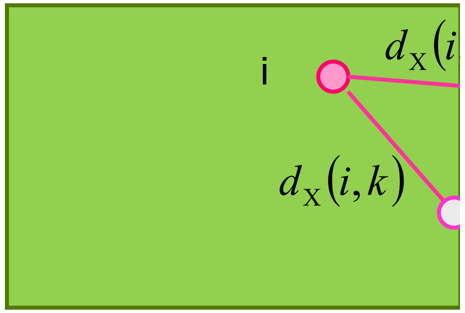
- **Step 1**
  - Determining neighboring points within a fixed radius based on the input space distance $d_X(i,j)$, or use a fixed # of neighbors

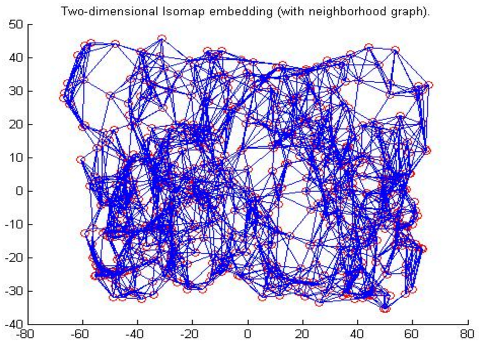
 # ε-radius **# K-nearest neighbors**





  - These neighborhood relations are represented as a weighted graph G over the data points.



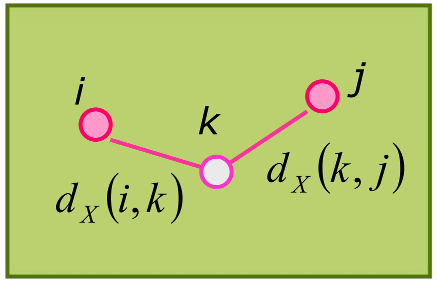


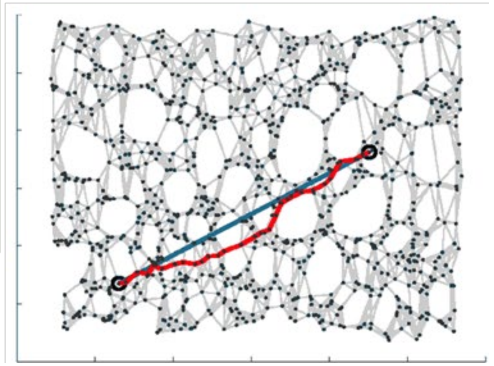
Shortest Path Computation

- **Step 2**
  - Estimating the geodesic distances $d_G(i,j)$ between all pairs of points on the manifold by computing their shortest path distances in the graph G.
  - Can be done using classic graph algorithms for the All Pairs Shortest Path (APSP) problem: Floyd/Warshall's algorithm or Dijkstra's algorithm

$$d_G(i,j) = d_X(i,j) \quad \text{neighboring } i, j$$
$$d_G(i,j) = \infty \qquad \text{othewise}$$

$$\textit{for } \mathrm{k} = 1,2,\ldots,\mathrm{N}$$
$$d_G(i,j) = \min\{\, d_X(i,j),\ d_X(i,k) + d_X(k,j) \}$$

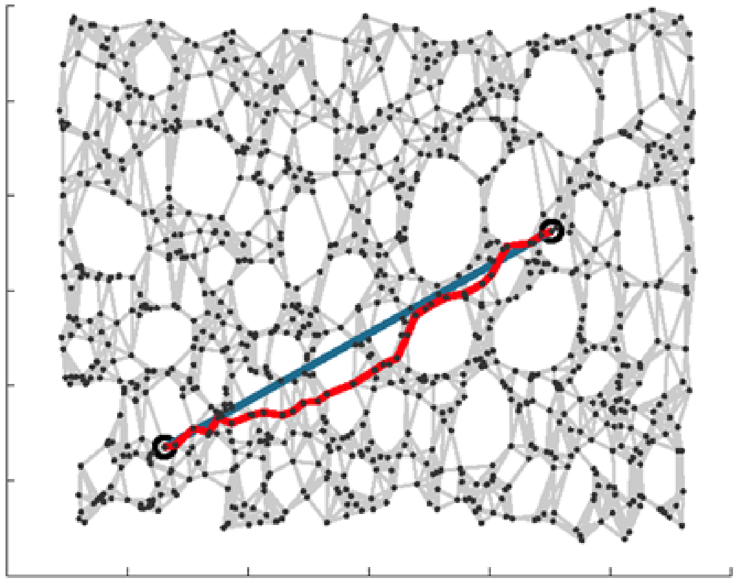




Euclidean Embedding

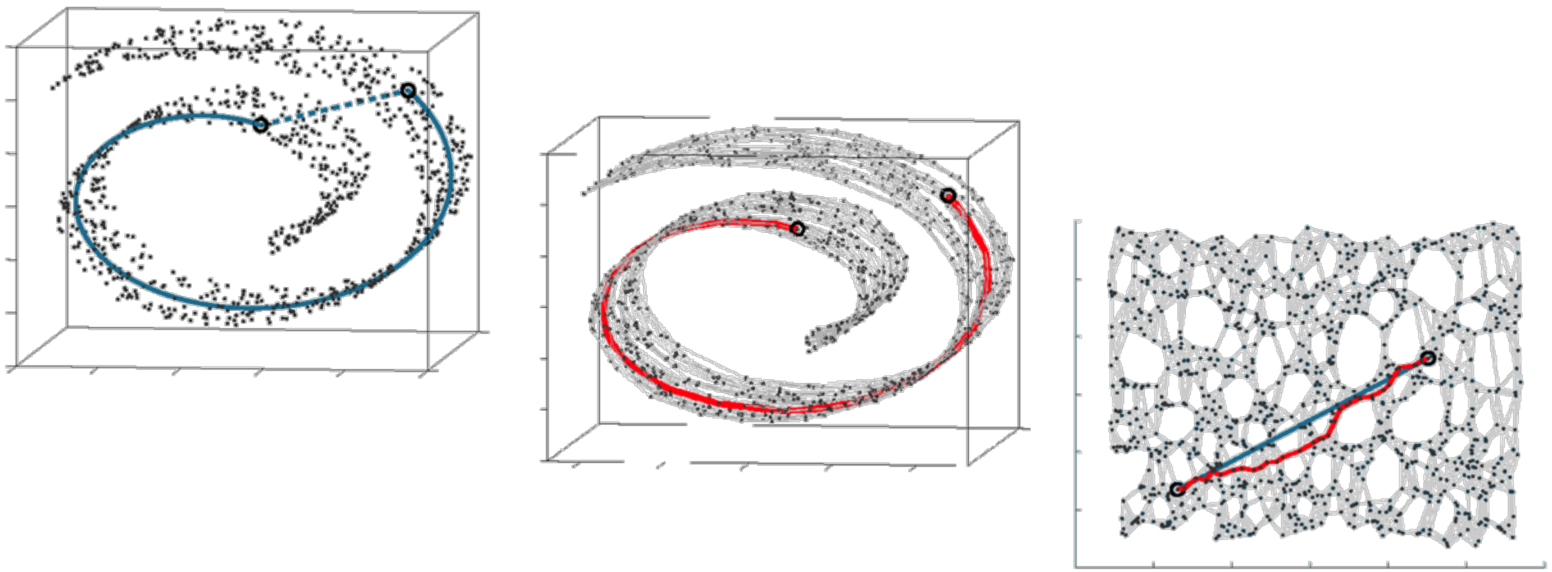
- **Step 3**
  - Constructing an embedding of the data in *d*-dimensional Euclidean space *Y* that best preserves the inter-point distances

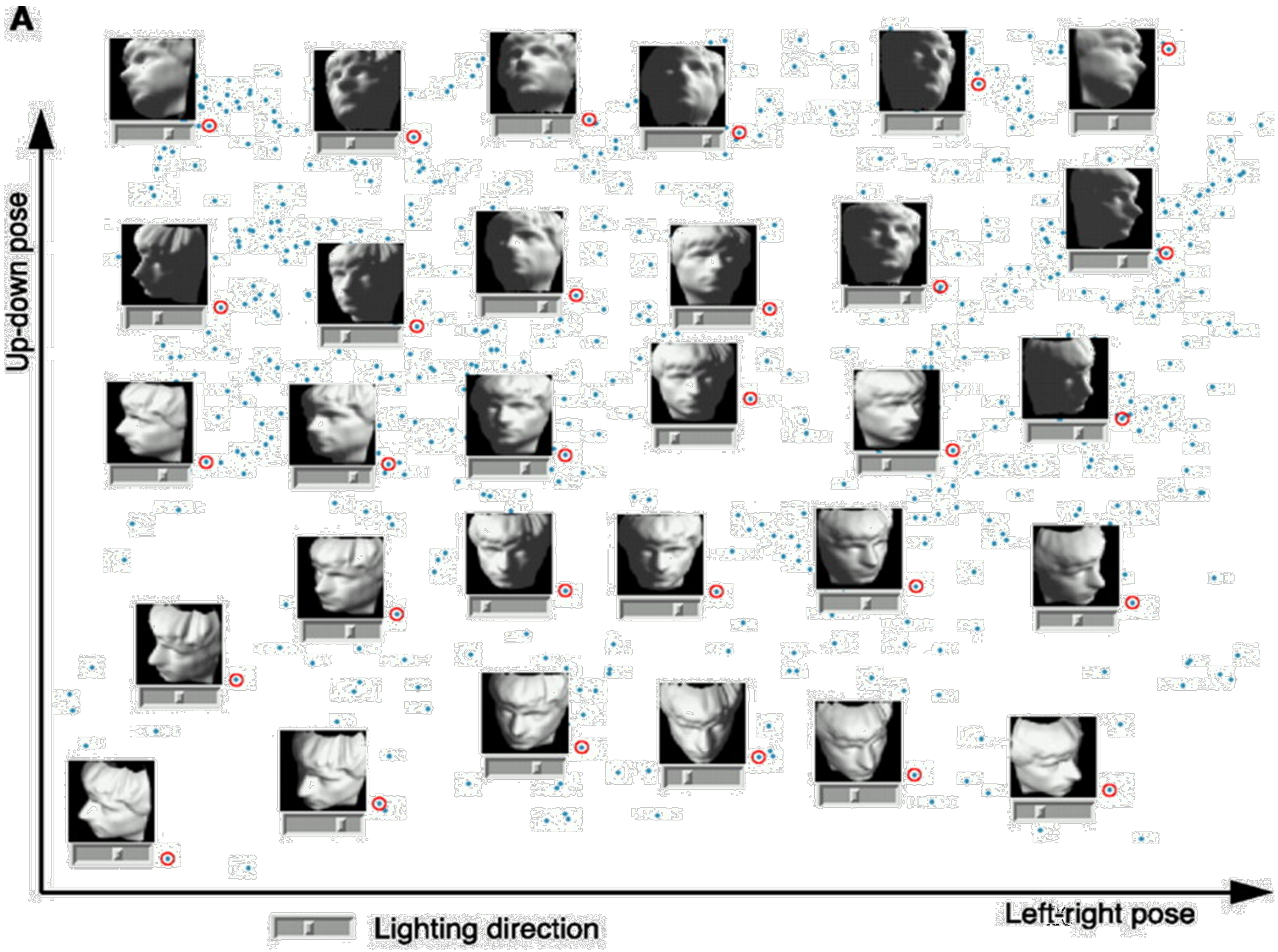
- This is of course nothing but an MDS problem



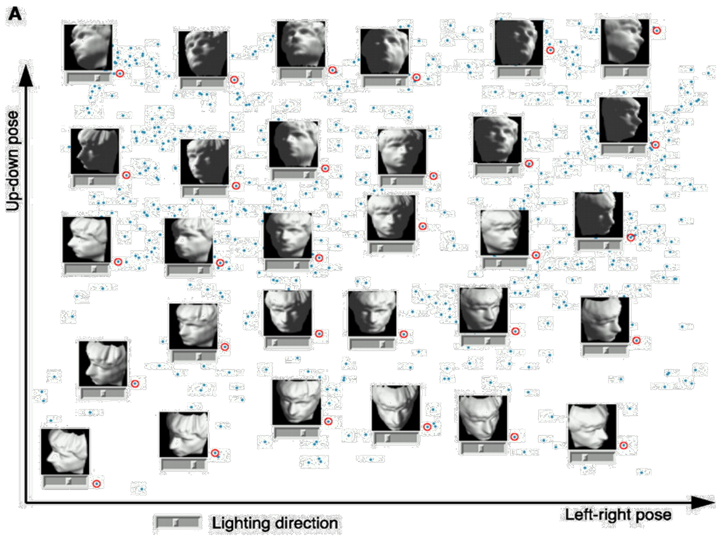
Recovery Guarantees

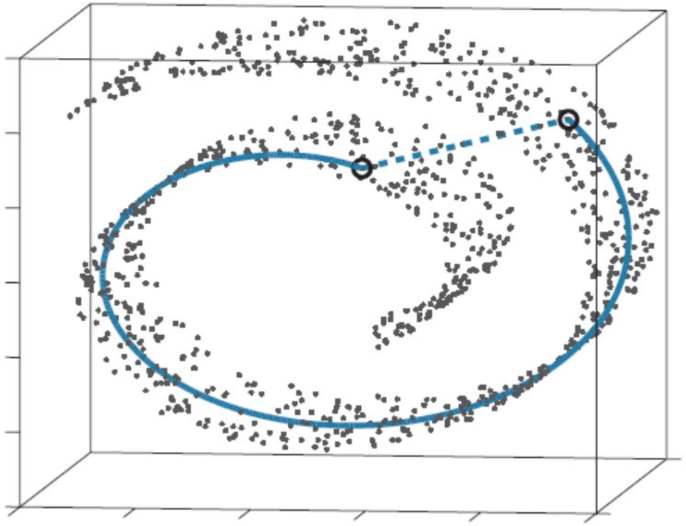
- Isomap is guaranteed asymptotically to recover the true dimensionality and geometric structure of non-linear manifolds.
- As the sample data point density increases, the graph distances provide increasingly better approximations to the intrinsic geodesic distances.

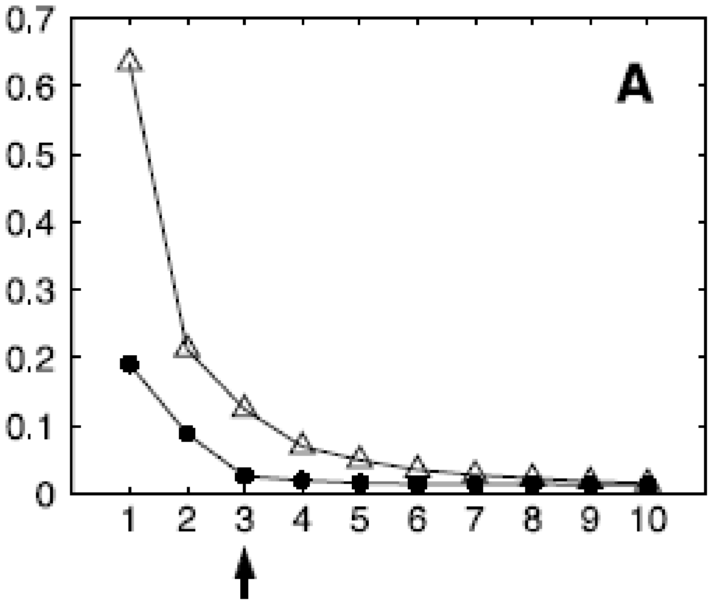


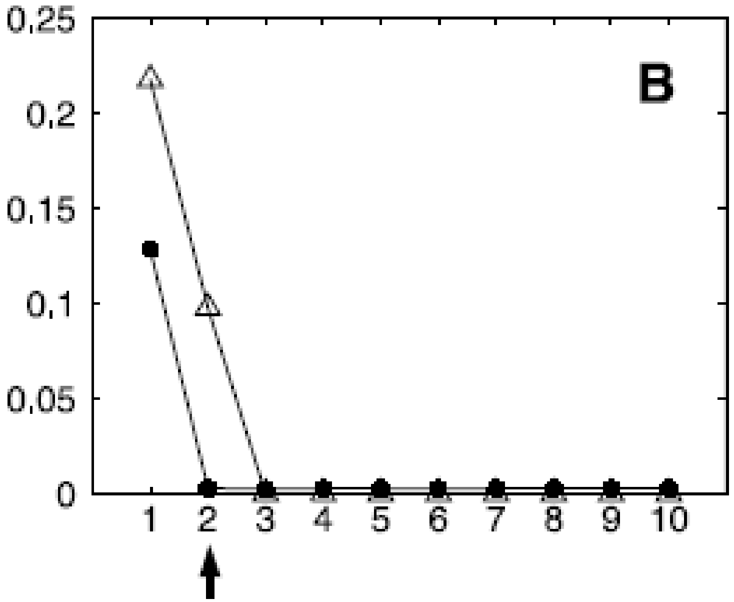

A
Up-down pose
Left-right pose
Lighting direction

ISOMAP Examples









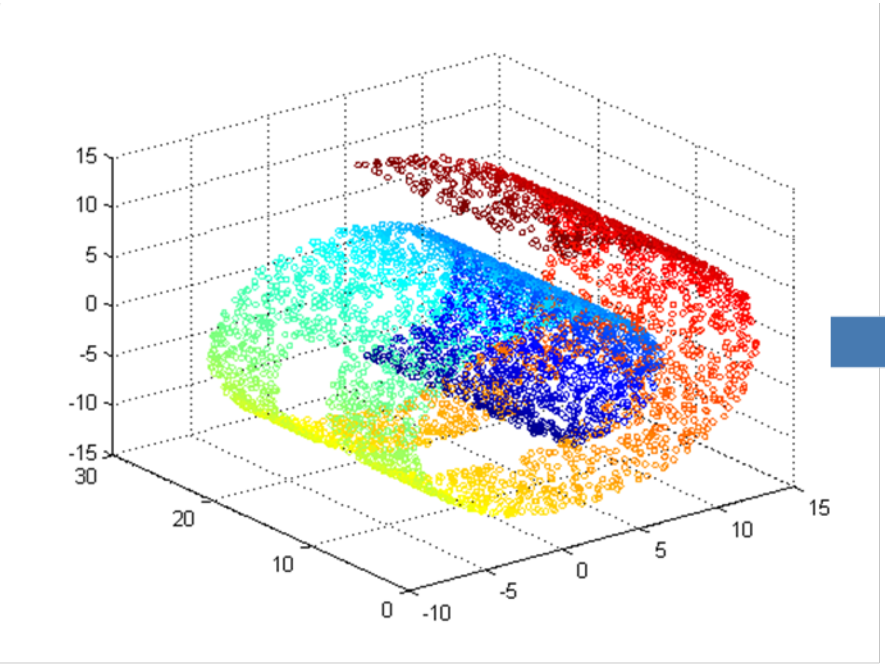
MDS : open triangles

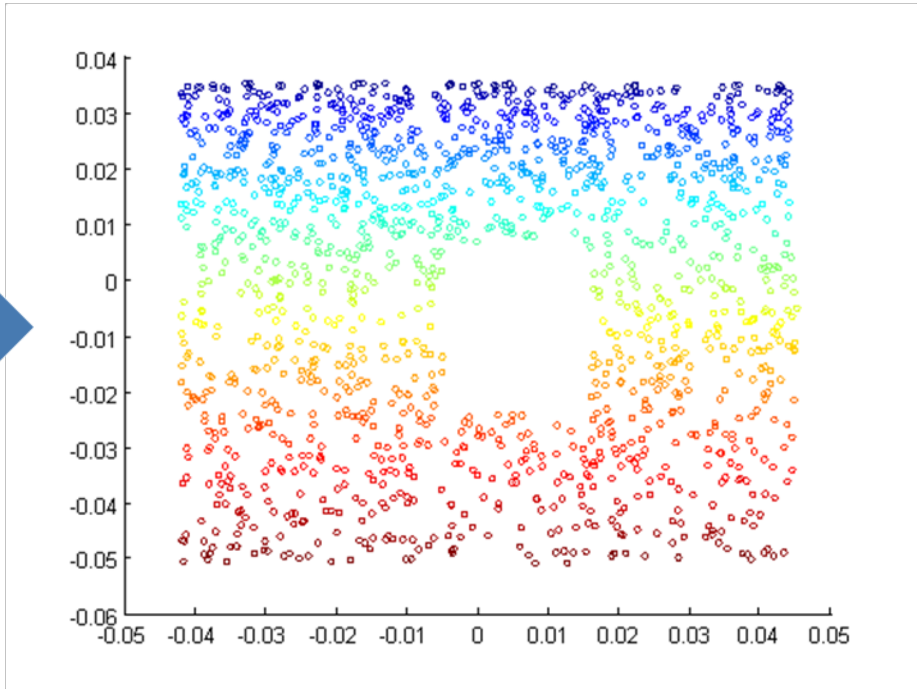
Isomap : filled circles

Laplacian Eigenmaps

Laplacian Eigenmaps (M. Belkin, P. Niyogi)

- Start same as Isomap, but use a spectral embedding in lieu of MDS





Hole distorts long geodesic distances, but affects less diffusion distances

Locally Linear Embeddings

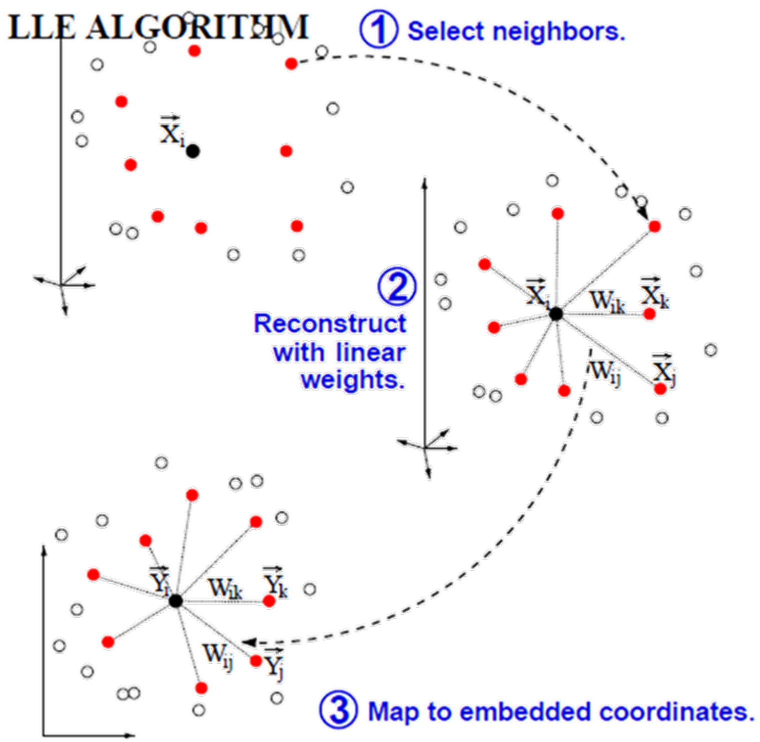
Locally Linear Embeddings (LLE) (S. T. Roweis and L. K. Saul)

- Define neighborhood relations between points (build NN graph)
  - *k* nearest neighbors
  - ε-balls
- Find weights that reconstruct each data point from its neighbors:

$$\min_{\sum_j w_{ij}=1} \left\| \mathbf{x_i} - \sum_{j \in N(i)} w_{ij} \mathbf{x_j} \right\|^2$$

- Find low-dimensional coordinates so that the same weights hold: $\mathbf{x_1}', \ldots, \mathbf{x_n}' \in R^d$

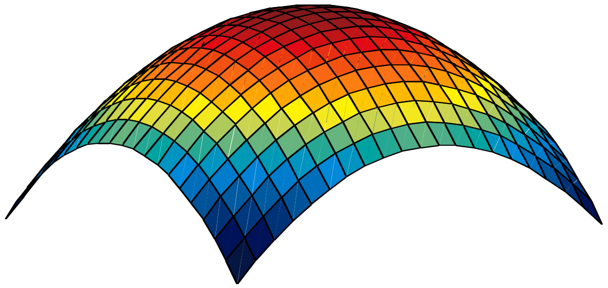
$$\min_{\mathbf{x_1}', \ldots, \mathbf{x_n}'} \sum_i \left\| \mathbf{x_i}' - \sum_{j \in N(i)} w_{ij} \mathbf{x_j}' \right\|^2$$



$x_i' = Y_i$

From Local to Global

- The weights w_{ij} capture the local shape
  - Invariant to translation, rotation and scale of the neighborhood
  - If the neighborhood lies on a manifold, the *local* mapping from the global coordinates (R^D) to the surface coordinates (R^d) is almost linear
  - Thus, the weights w_{ij} should hold also for manifold (R^d) coordinate system!



$$\min_{\sum_j w_{ij}=1} \left\| \mathbf{x_i} - \sum_{j \in N(i)} w_{ij} \mathbf{x_j} \right\|^2$$

$$\min_{\mathbf{x_1}',\ldots,\mathbf{x_n}'} \sum_i \left\| \mathbf{x_i}' - \sum_{j \in N(i)} w_{ij} \mathbf{x_j}' \right\|^2$$

Solving the Minimizations

- Linear least squares (using Lagrange multipliers)

$$\min_{\sum_j w_{ij}=1} \left\| \mathbf{x_i} - \sum_{j \in N(i)} w_{ij} \mathbf{x_j} \right\|^2$$

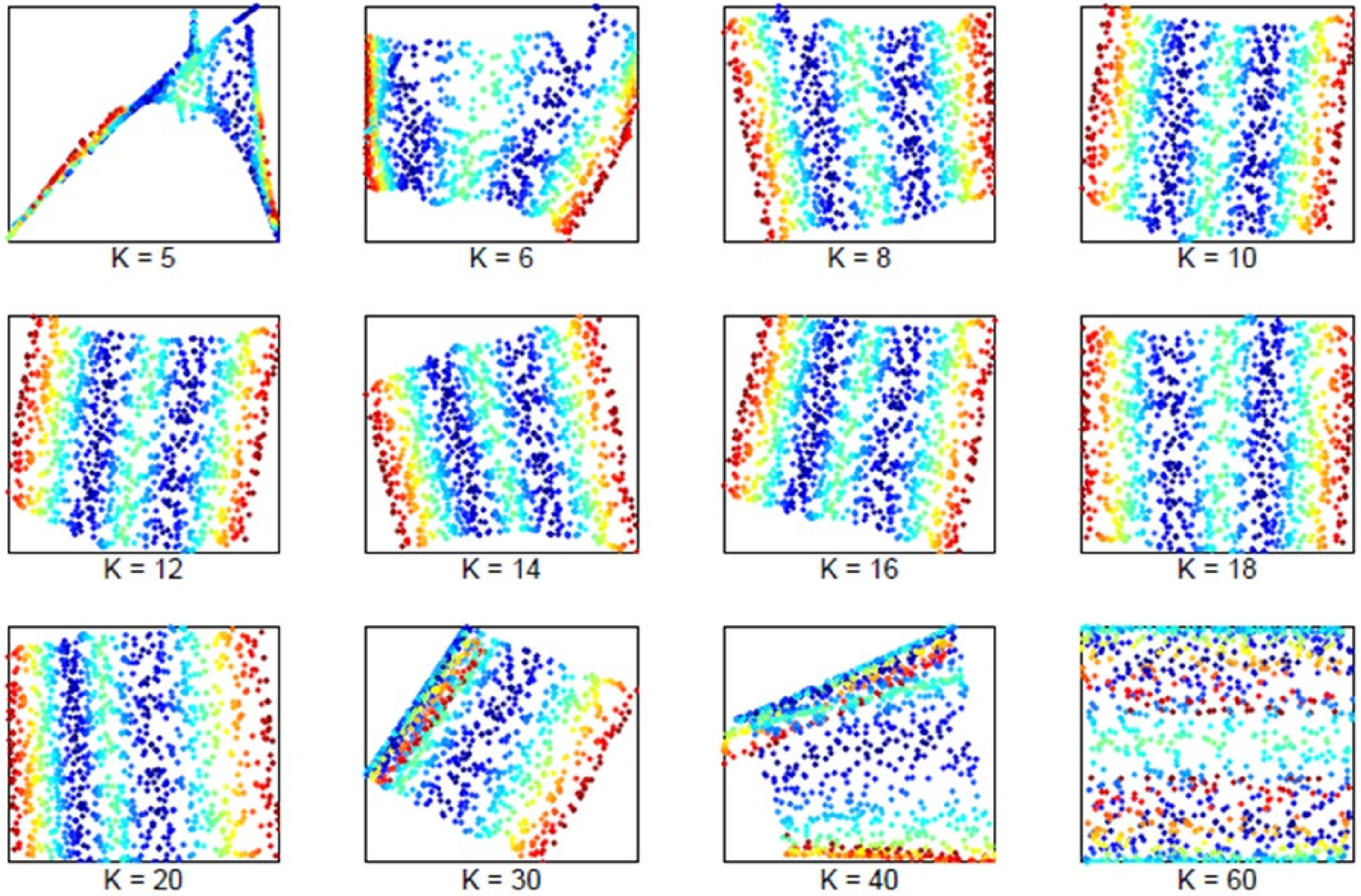
- To find $\mathbf{x_1}', \ldots, \mathbf{x_n}' \in R^d$ that minimize,

$$\min_{\mathbf{x_1}', \ldots, \mathbf{x_n}'} \sum_i \left\| \mathbf{x_i}' - \sum_{j \in N(i)} w_{ij} \mathbf{x_j}' \right\|^2$$

a sparse eigenvalue problem is solved. Additional constraints are added for conditioning:

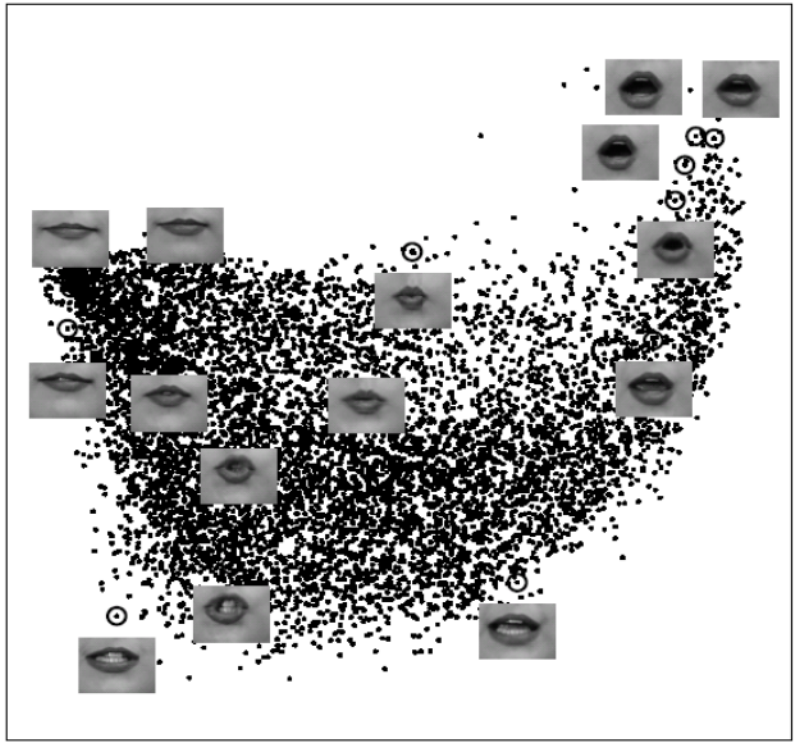
$$\sum_i \mathbf{x_i}' = 0, \qquad \frac{1}{n} \sum_i \mathbf{x_i}' \mathbf{x_i}'^T = I$$

Effect of Neighborhood Size on LLE

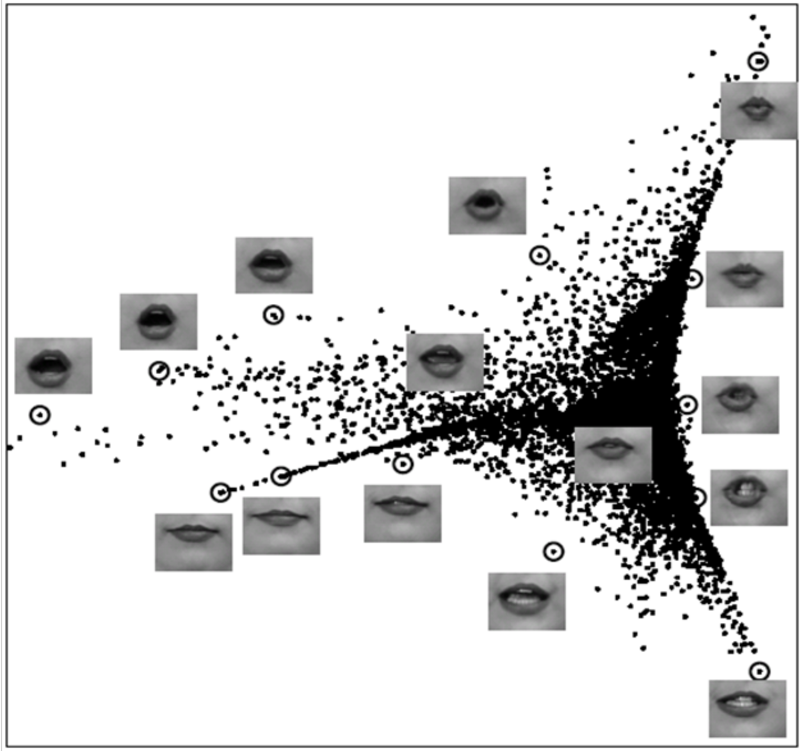


LLE Experiments

- Lips

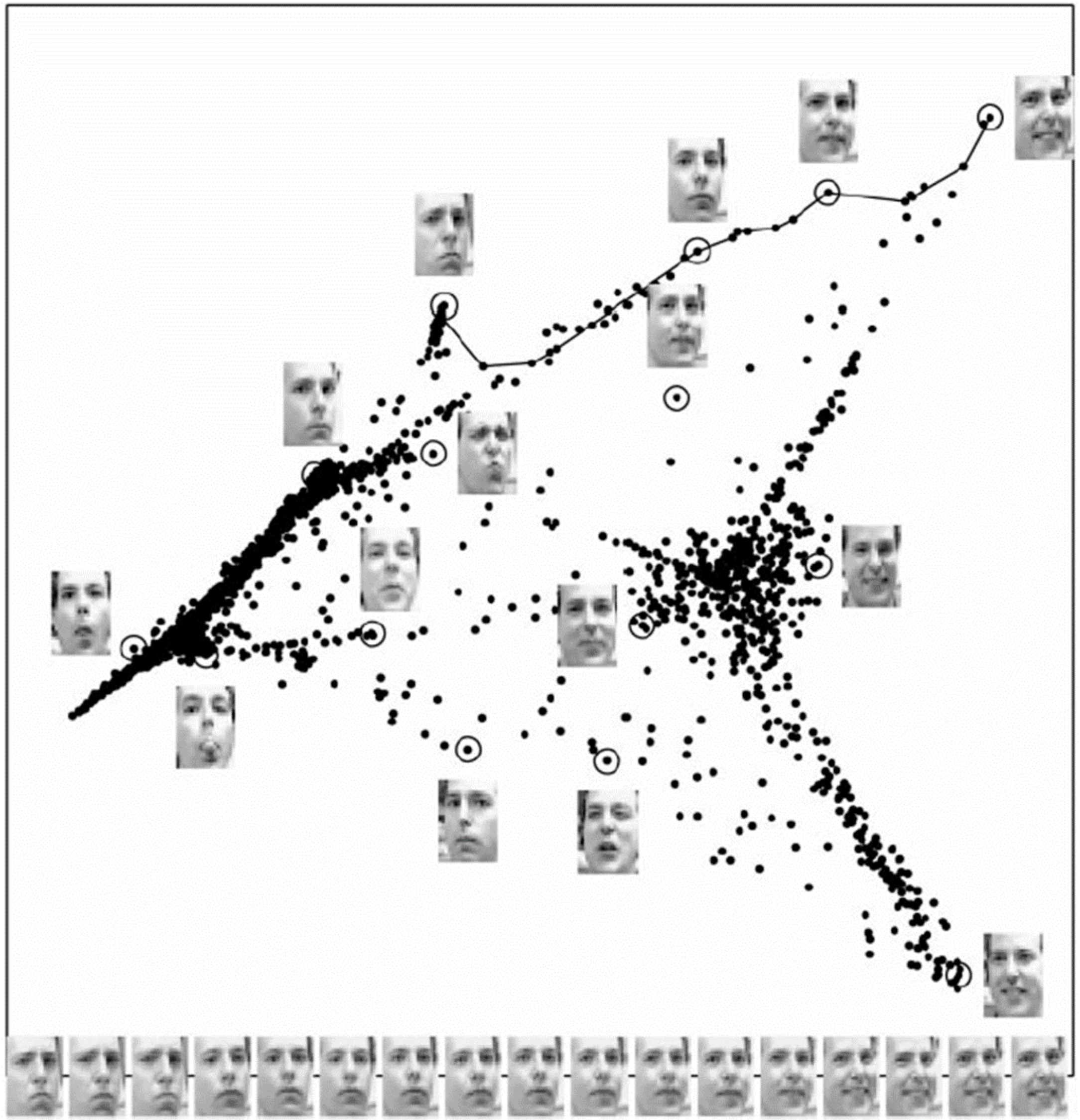


PCA

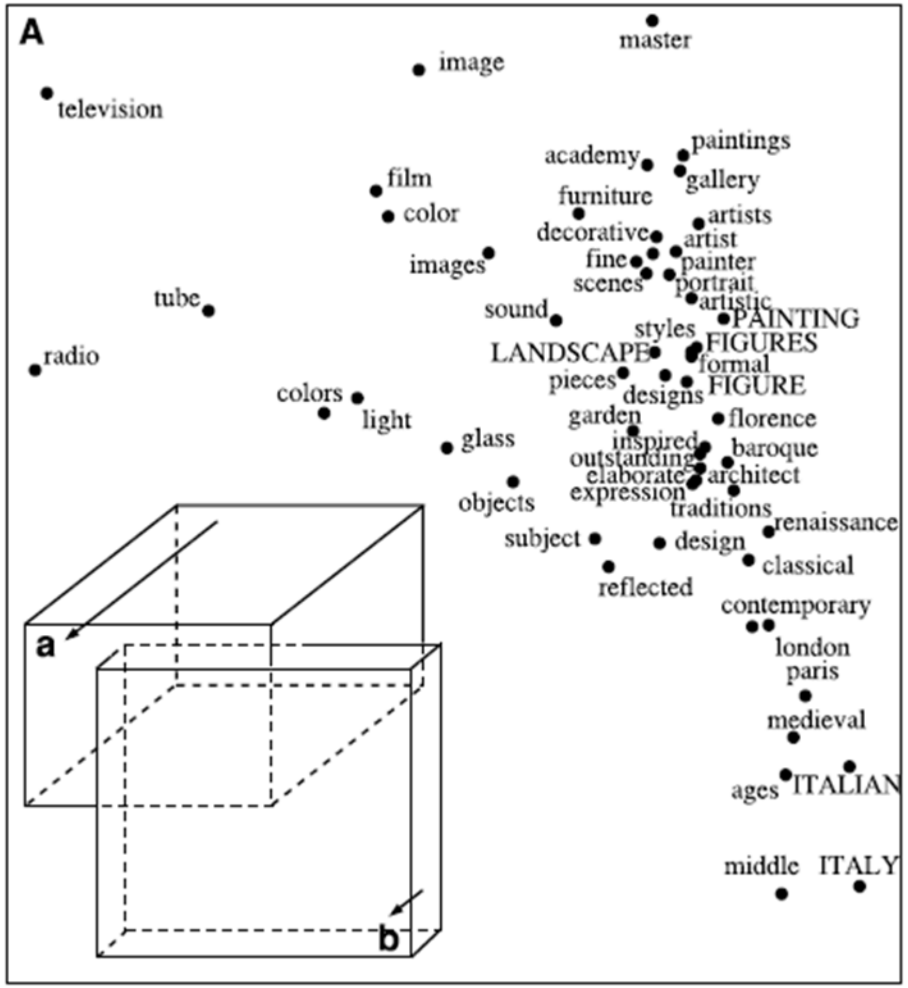


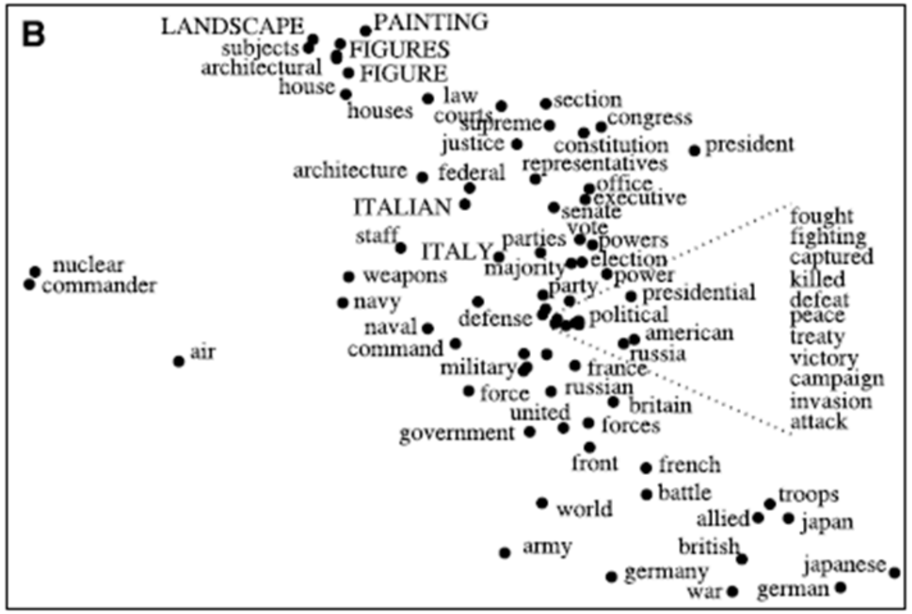
LLE

LLE Experiments: Faces



LLE Experiments: Words





Based on word counts in encyclopedia articles.

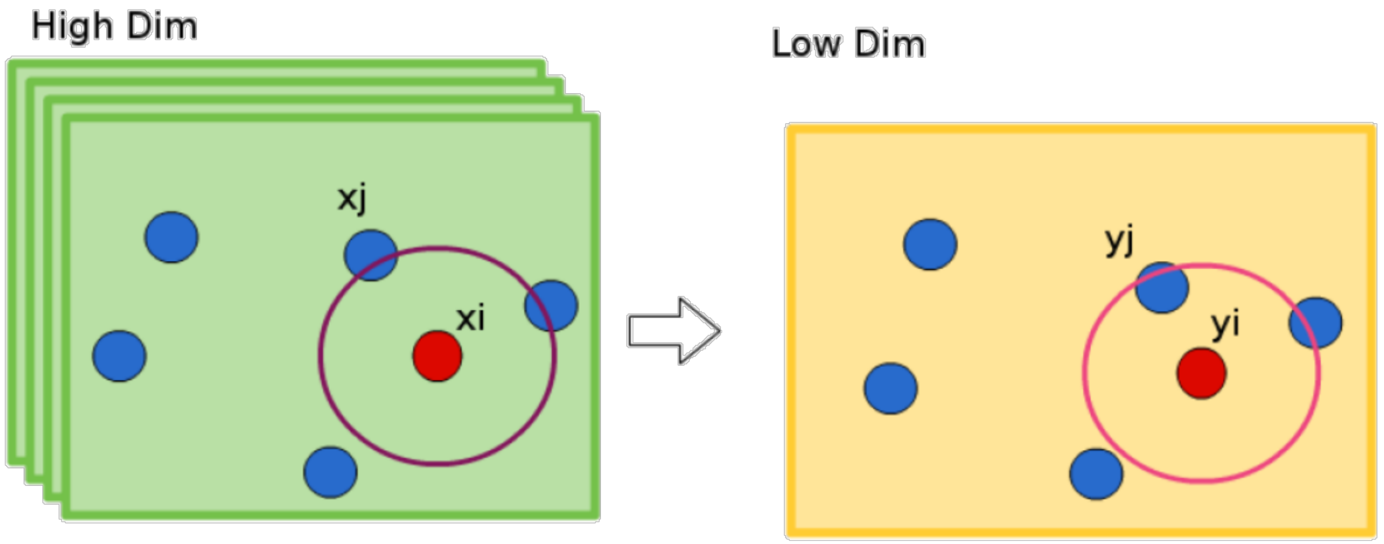
Note how LLE collocates words with similar semantic contexts in this continuous space.

t-SNE

Laurens van der Maaten and Geoffrey Hinton, JMLR 2008

t-Distributed Stochastic Neighbor Embedding

Measure pairwise similarities between high-dimensional and low-dimensonal objects



$$p_{j|i} = \frac{exp(-||x_i - x_j||^2/2\sigma_i^2)}{\sum_{k \neq i} exp(-||x_i - x_k||^2/2\sigma_i^2)}$$

Stochastic Neighbor Embedding

Converting the high-dimensional Euclidean distances into conditional probabilities that represent similarities

- Similarity of datapoints in High Dimension

$$p_{j|i} = \frac{exp(-||x_i - x_j||^2/2\sigma_i^2)}{\sum_{k \neq i} exp(-||x_i - x_k||^2/2\sigma_i^2)}$$

- Similarity of datapoints in Low Dimension

$$q_{j|i} = \frac{exp(-||y_i - y_j||^2)}{\sum_{k \neq i} exp(-||y_i - y_k||^2)}$$

- Cost function

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

Minimize the cost function using gradient descent

Symmetric SNE

- Minimize a single KL divergence between a joint probability distribution

$$C = KL(P||Q) = \sum_i \sum_{j \neq i} p_{ij} log \frac{p_{ij}}{q_{ij}}$$

- The obvious way to redefine the pairwise similarities is

$$p_{ij} = \frac{exp(-||x_i - x_j||^2/2\sigma^2)}{\sum_{k \neq l} exp(-||x_l - x_k||^2/2\sigma^2)}$$

$$q_{ij} = \frac{exp(-||y_i - y_j||^2)}{\sum_{k \neq l} exp(-||y_l - y_k||^2)}$$

However, in practice we symmetrize (or average) the conditionals

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}$$

Set the bandwidth σ_i such that the conditional has a fixed perplexity (effective number of neighbors) $Perp(P_i) = 2^{H(P_i)}$, typical value is about 5 to 50

t-Distribution

Use heavier tail distribution than Gaussian in low-dim space, we choose

$$q_{ij} \propto (1 + ||y_i - y_j||^2)^{-1}$$

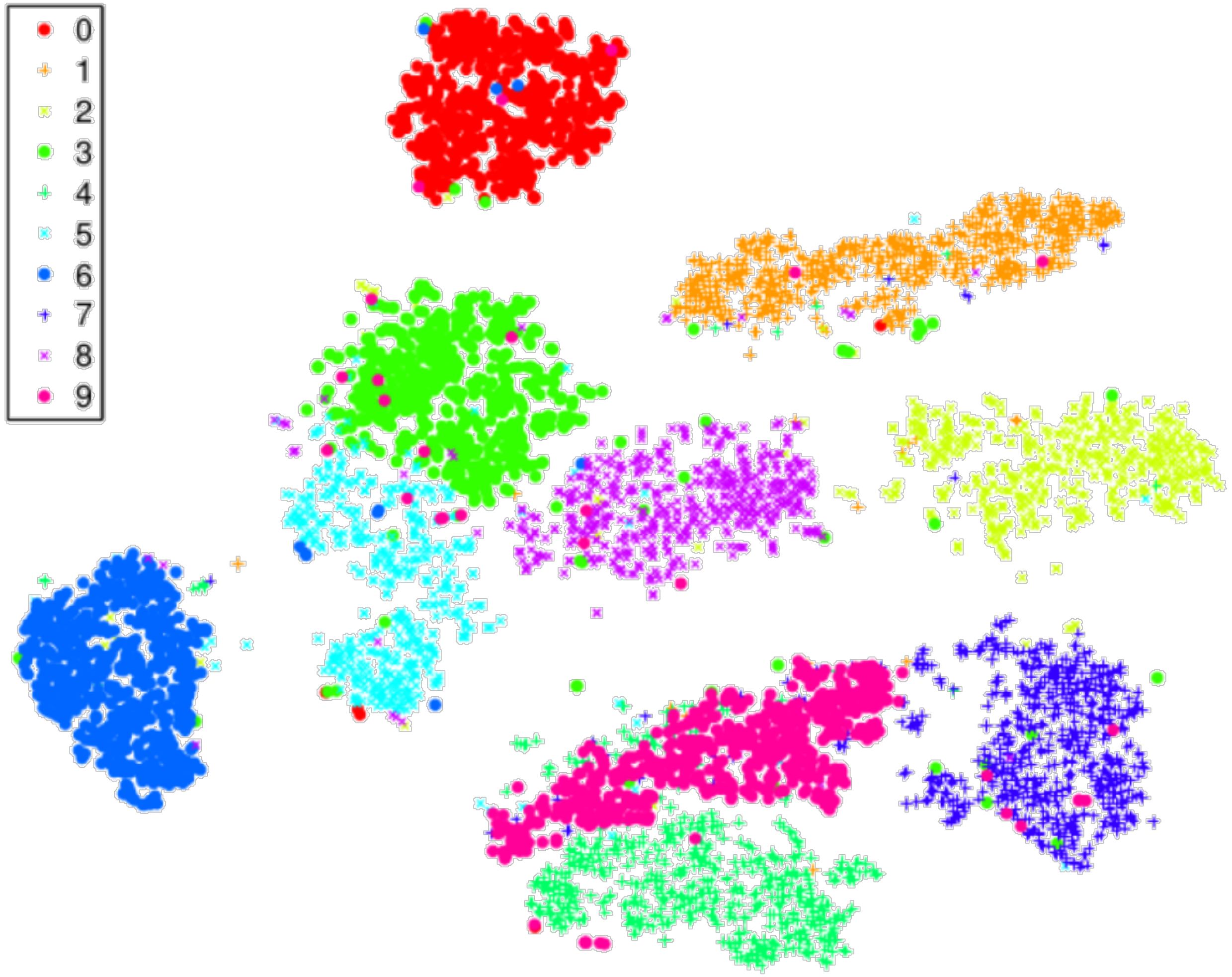
- Similarity of datapoints in High Dimension

$$p_{ij} = \frac{exp(-||x_i - x_j||^2/2\sigma^2)}{\sum_{k \neq l} exp(-||x_l - x_k||^2/2\sigma^2)}$$

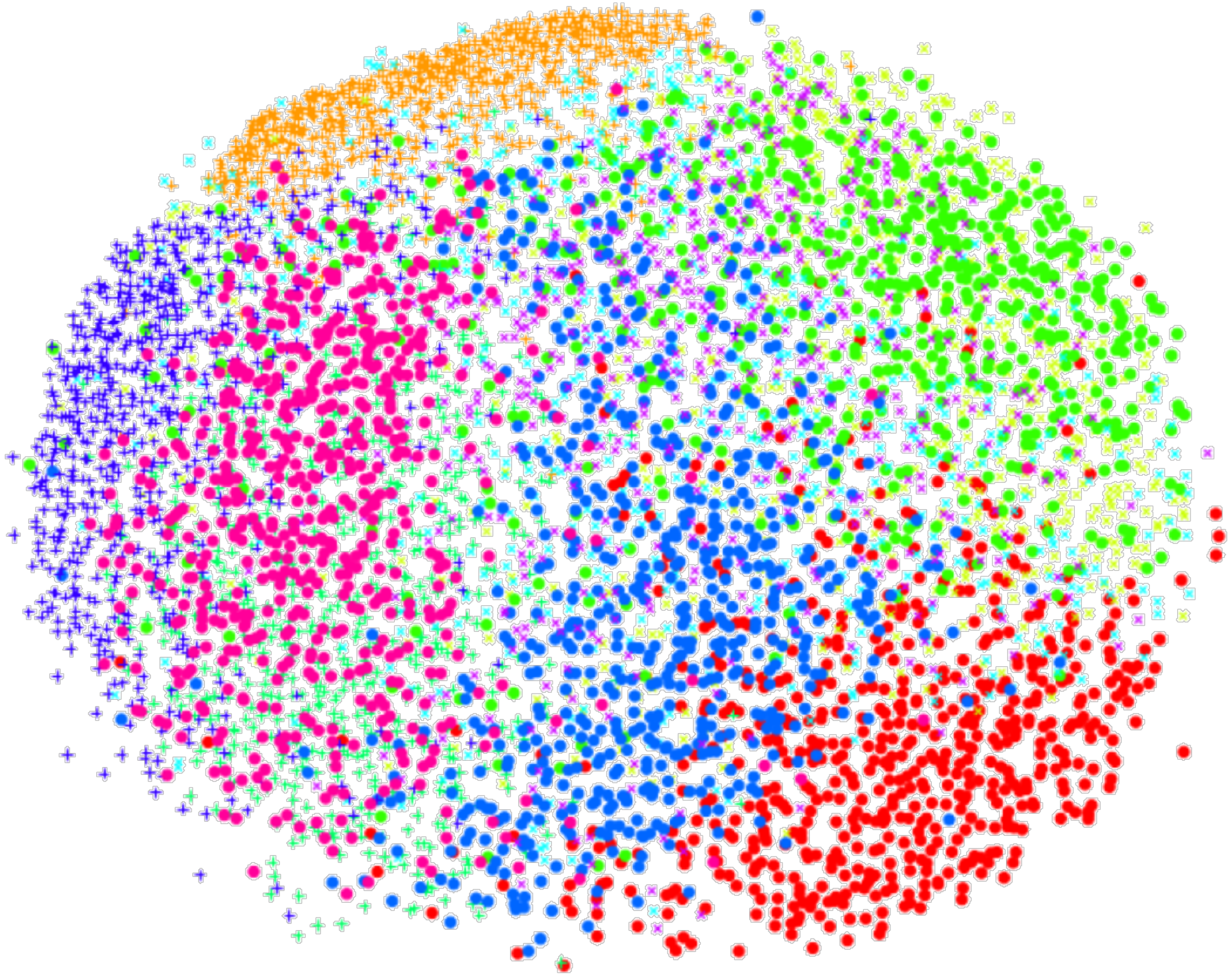
- Similarity of datapoints in Low Dimension

$$q_{ij} = \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_{k \neq l}(1 + ||y_k - y_l||^2)^{-1}}$$

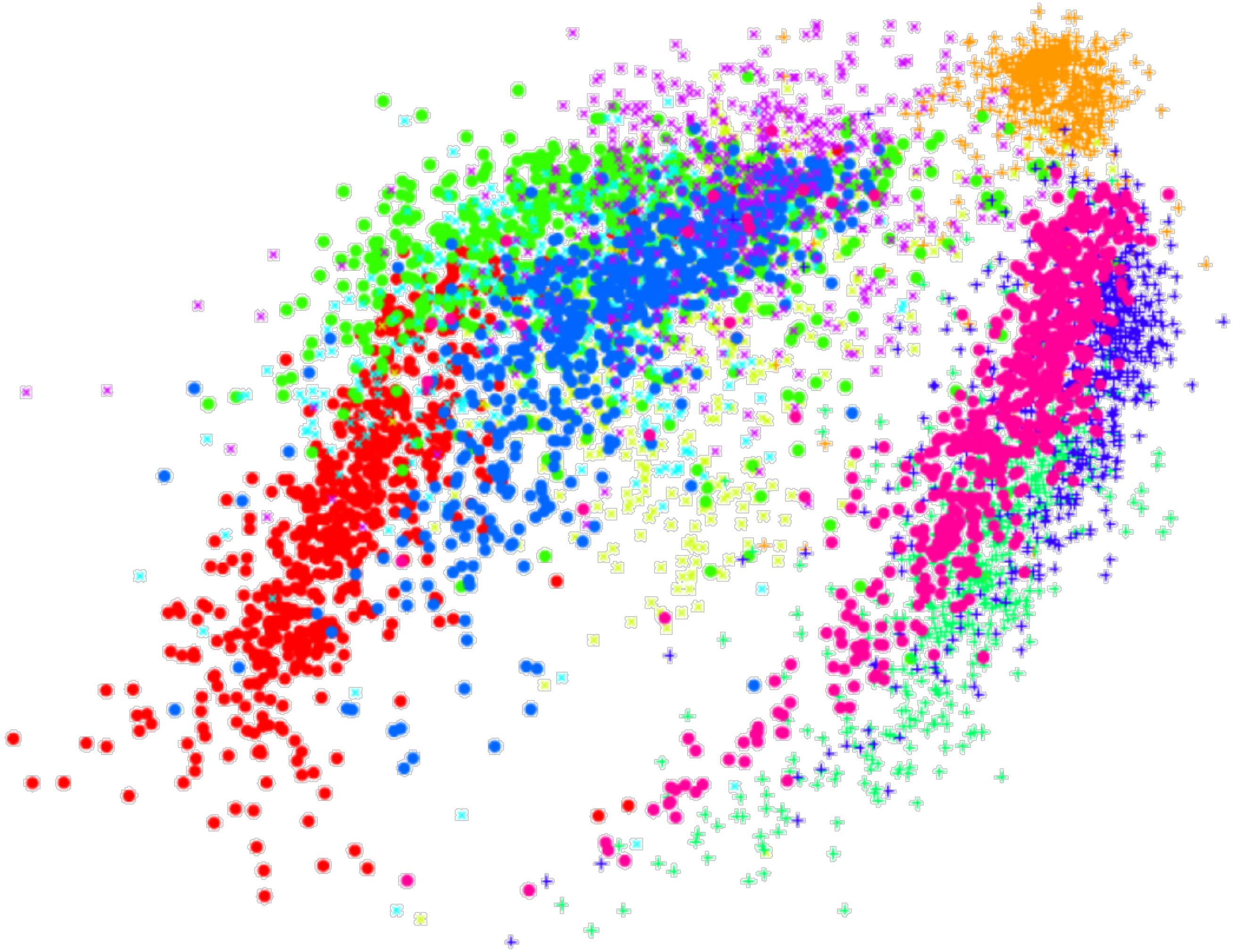
MNIST t-SNE



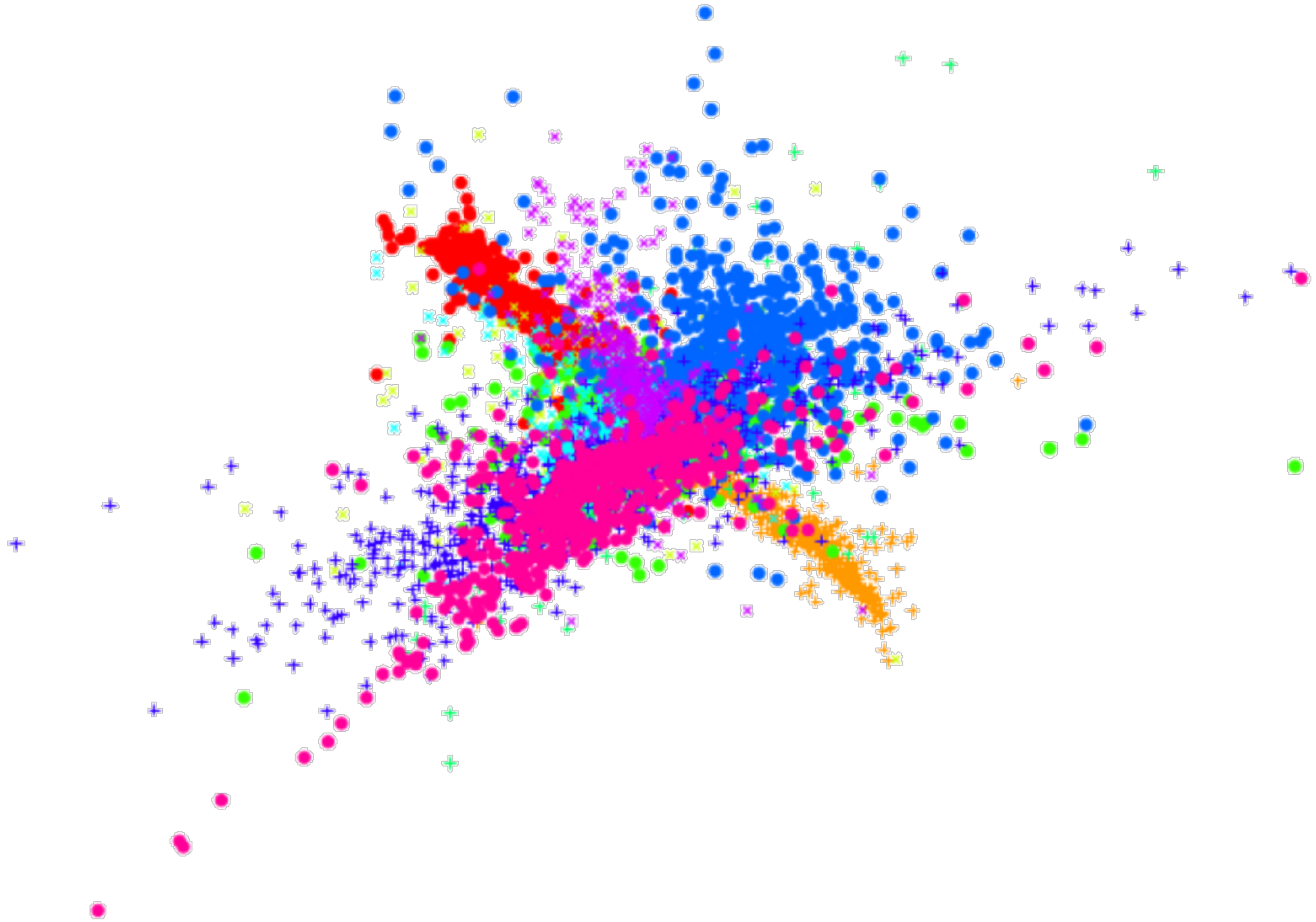
MNIST MDS w. Sammon Mapping



MNIST Isomap



MNIST LLE



Source Codes

- t-SNE (Matlab, CUDA, Binary, Python, Torch, Julia, R and JavaScript)
- Parametric t-SNE (Matlab)
- Barnes-Hut-SNE (with C++, Matlab, Python, Torch, and R wrappers)

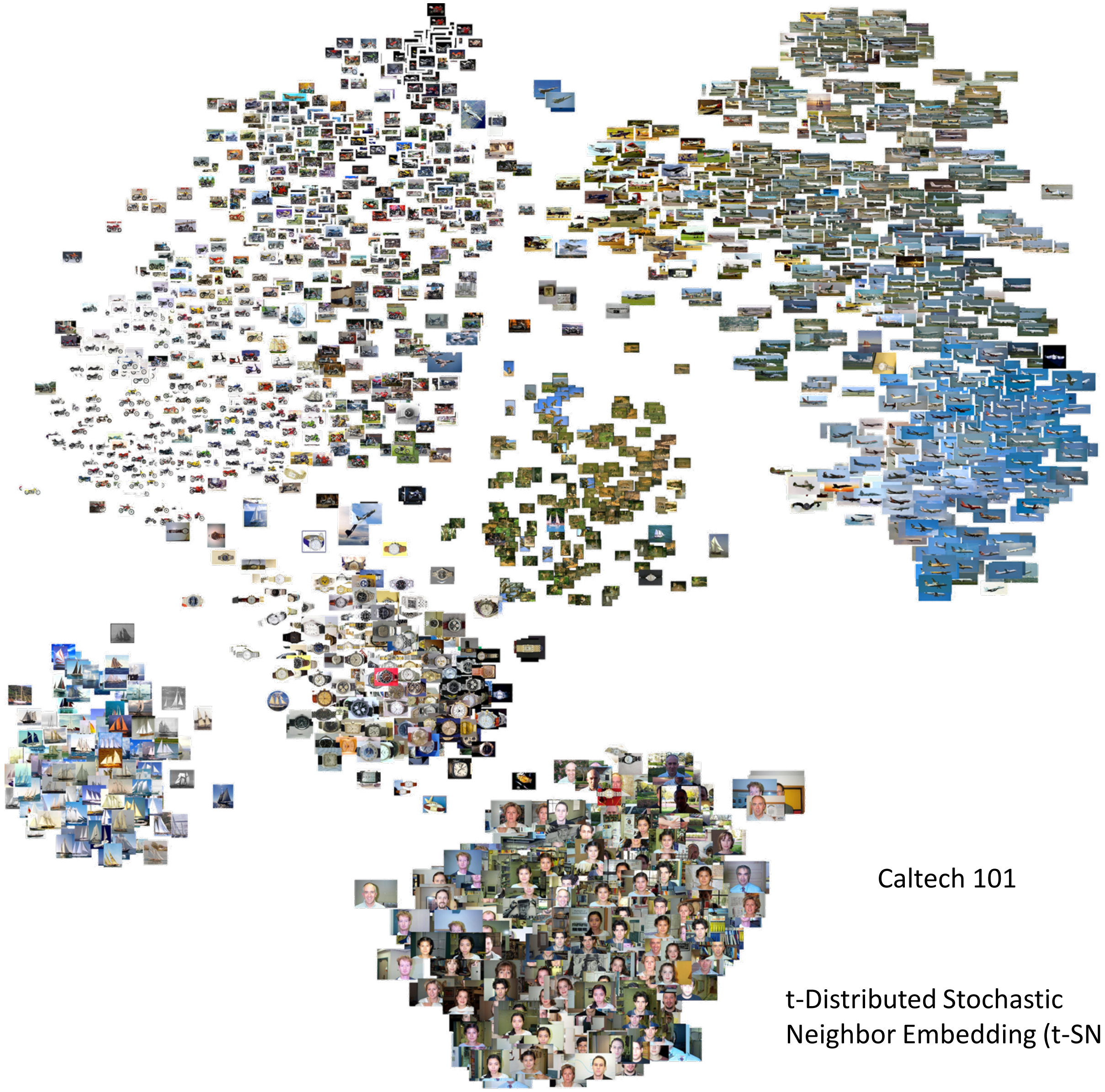
Many More Methods

Many NLDR Methods

Contents [hide]

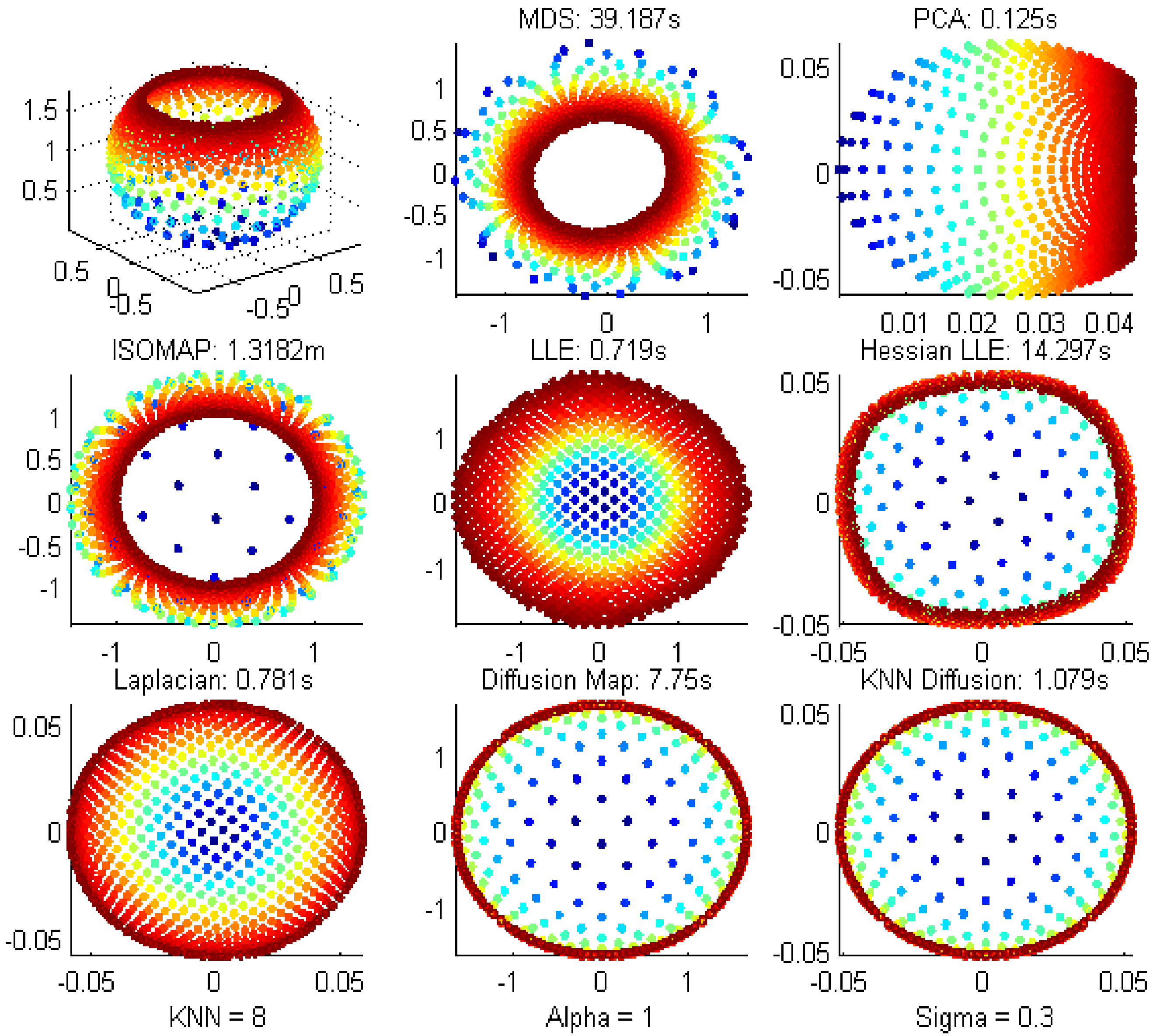
1 Related Linear Decomposition Methods
2 Applications of NLDR
3 Manifold learning algorithms
  3.1 Sammon's mapping
  3.2 Self-organizing map
  3.3 Principal curves and manifolds
  3.4 Autoencoders
  3.5 Gaussian process latent variable models
  3.6 Curvilinear component analysis
  3.7 Curvilinear distance analysis
  3.8 Diffeomorphic dimensionality reduction
  3.9 Kernel principal component analysis
  3.10 Isomap
  3.11 Locally-linear embedding
  3.12 Laplacian eigenmaps
  3.13 Manifold alignment
  3.14 Diffusion maps
  3.15 Hessian Locally-Linear Embedding (Hessian LLE)
  3.16 Modified Locally-Linear Embedding (MLLE)
  3.17 Relational perspective map
  3.18 Local tangent space alignment
  3.19 Local multidimensional scaling
  3.20 Maximum variance unfolding
  3.21 Nonlinear PCA
  3.22 Data-driven high-dimensional scaling
  3.23 Manifold sculpting
  3.24 t-distributed stochastic neighbor embedding
  3.25 RankVisu
  3.26 Topologically constrained isometric embedding
4 Methods based on proximity matrices
5 See also
6 References
7 External links

From Wikipedia



Caltech 101

t-Distributed Stochastic
Neighbor Embedding (t-SNE)


MDS: 39.187s
PCA: 0.125s
ISOMAP: 1.3182m
LLE: 0.719s
Hessian LLE: 14.297s
Laplacian: 0.781s
Diffusion Map: 7.75s
KNN Diffusion: 1.079s
KNN = 8
Alpha = 1
Sigma = 0.3

	MDS	PCA	ISOMAP	LLE	Laplacian	Diffusion Map	KNN Diffusion	Hessian
Speed	Very slow	Extremely fast	Extremely slow	Fast	Fast	Fast	Fast	Slow
Infers geometry?	NO	NO	YES	YES	YES	MAYBE	MAYBE	YES
Handles non-convex?	NO	NO	NO	MAYBE	MAYBE	MAYBE	MAYBE	YES
Handles non-uniform sampling?	YES	YES	YES	YES	NO	YES	YES	MAYBE
Handles curvature?	NO	NO	YES	MAYBE	YES	YES	YES	YES
Handles corners?	NO	NO	YES	YES	YES	YES	YES	NO
Clusters?	YES	YES	YES	YES	NO	YES	YES	NO
Handles noise?	YES	YES	MAYBE	NO	YES	YES	YES	YES
Handles sparsity?	YES	YES	YES	YES	YES	NO	NO	NO may crash
Sensitive to parameters?	NO	NO	YES	YES	YES	VERY	VERY	YES

The End