DIMENSIONALITY REDUCTION

DATA/MSML 603: Principles of Machine Learning

Last Time

- Clustering
  - k-means clustering & k-medoids clustering
  - Hierarchical clustering
    - Agglomerative vs. divisive
- Gaussian mixture model
  - Probabilistic model
  - Parameters can be estimated using Expectation-Maximization (EM) algorithm

Data Visualization & Dimensionality Reduction

How Can We Visualize High Dimensional Data? (1/2)

- E.g., 53 blood and urine tests for 65 patients
- Difficult to see the correlations between the features in high-dimensional data

How Can We Visualize High Dimensional Data? (2/2)

Instances

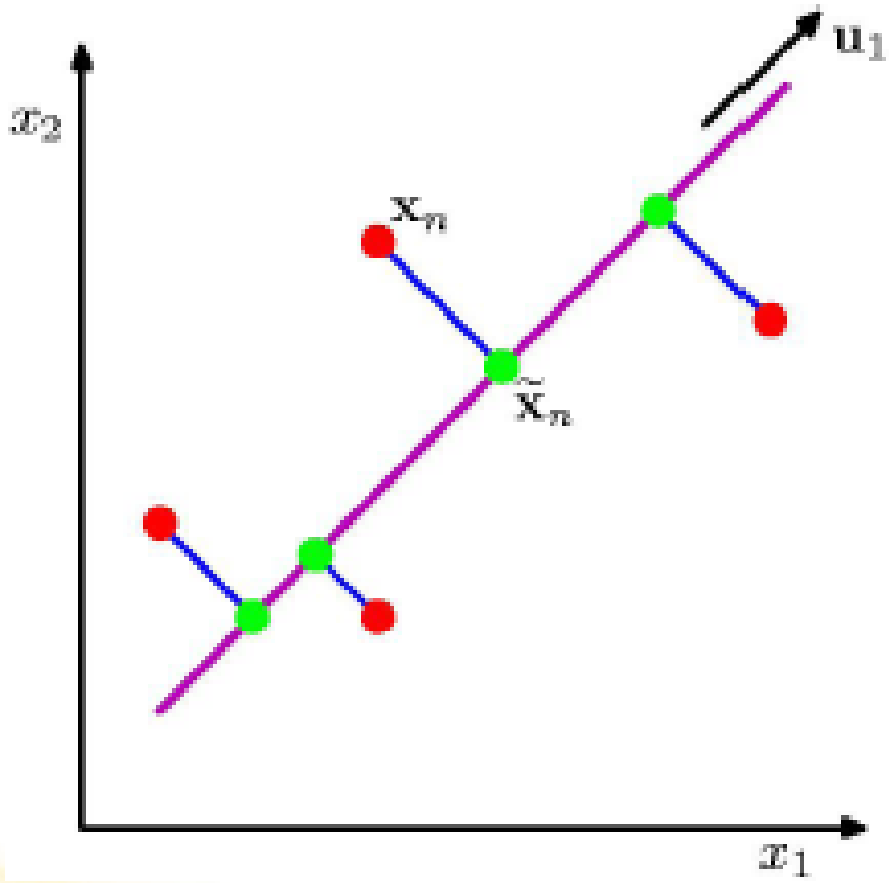
	H-WBC	H-RBC	H-Hgb	H-Hct	H-MCV	H-MCH	H-MCHC
A1	8.0000	4.8200	14.1000	41.0000	85.0000	29.0000	34.0000
A2	7.3000	5.0200	14.7000	43.0000	86.0000	29.0000	34.0000
A3	4.3000	4.4800	14.1000	41.0000	91.0000	32.0000	35.0000
A4	7.5000	4.4700	14.9000	45.0000	101.0000	33.0000	33.0000
A5	7.3000	5.5200	15.4000	46.0000	84.0000	28.0000	33.0000
A6	6.9000	4.8600	16.0000	47.0000	97.0000	33.0000	34.0000
A7	7.8000	4.6800	14.7000	43.0000	92.0000	31.0000	34.0000
A8	8.6000	4.8200	15.8000	42.0000	88.0000	33.0000	37.0000
A9	5.1000	4.7100	14.0000	43.0000	92.0000	30.0000	32.0000

Features

Data Visualization

- Is there a representation better than the raw features?
  - Is it really necessary to show all the 53 dimensions?
  - … what if there are strong correlations between the features?
- Could we find the smallest subspace of the 53-dimensional space that keeps the most information about the original data?
- One solution: **Principal component analysis (Lecture 4)**
  - **Preserve the information in the directions in which there is most variance (captured by eigenvectors)**

Principle Component Analysis

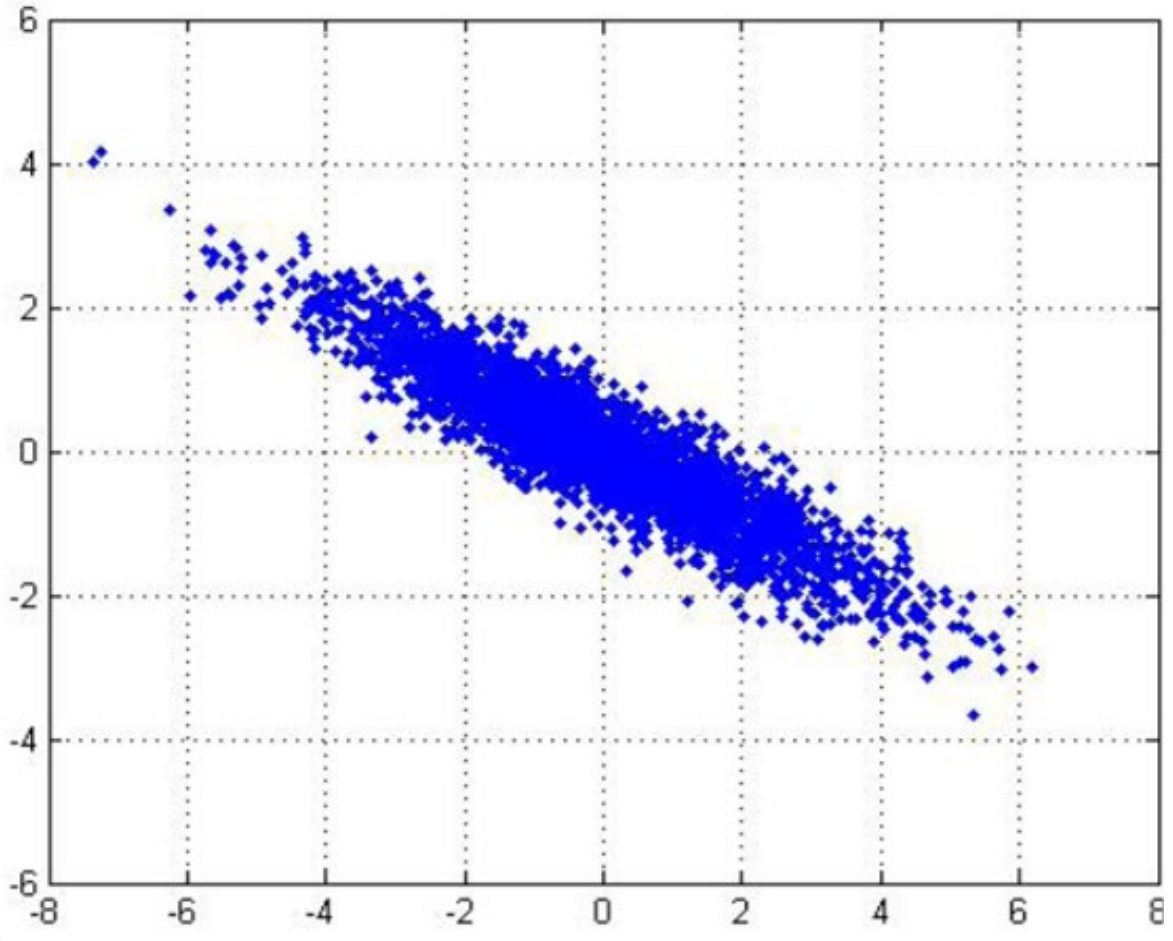


- Orthogonal projection of data onto lower-dimensional linear space that …
  - Maximizes variance of the projected data (purple line)
  - Minimizes mean squared distance between data points and projections

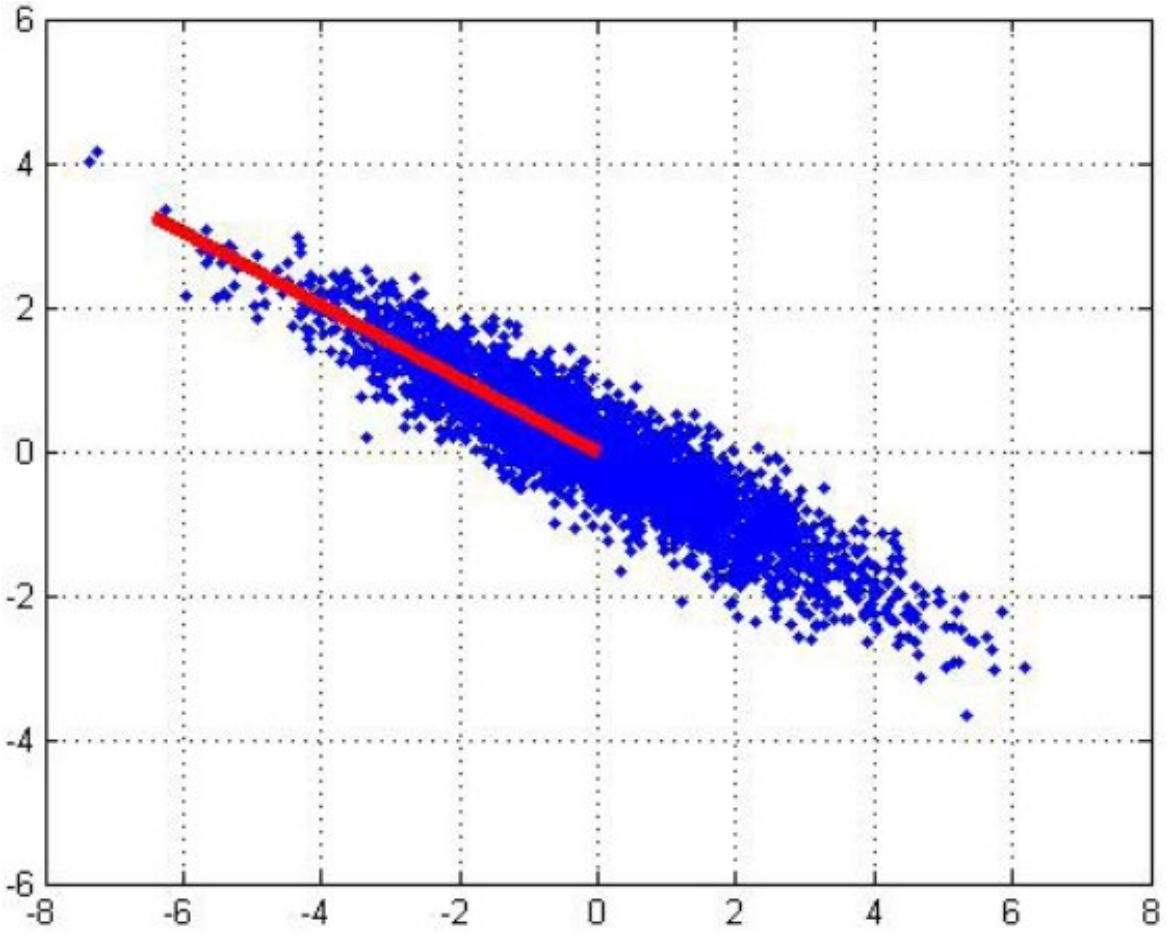
The Principal Components

- Vectors originating from the center of mass
- First principal component points in the **direction of the largest variance**
- Each subsequent principal component:
  - is orthogonal to the previous ones, and
  - points in the directions of the largest variance of the **residual subspace**

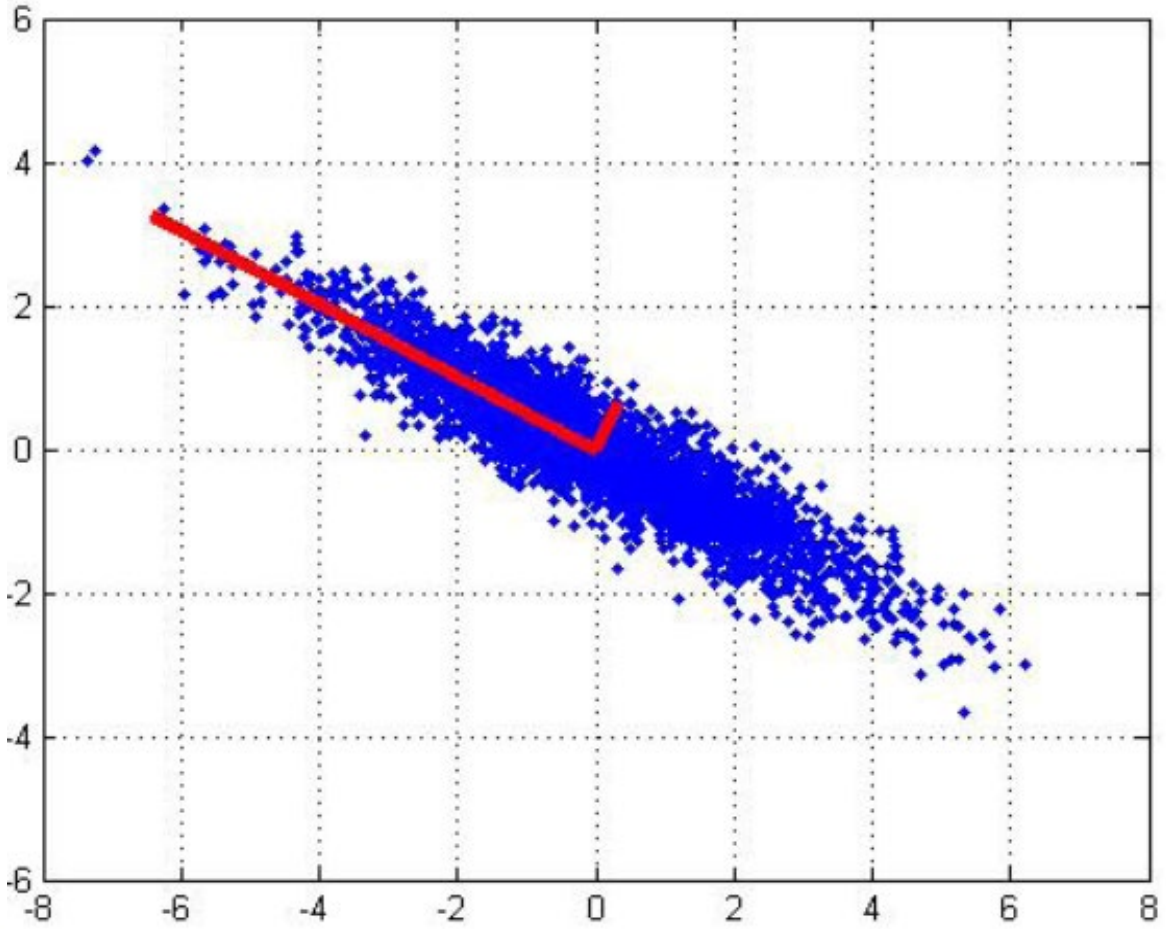
2D Gaussian Dataset



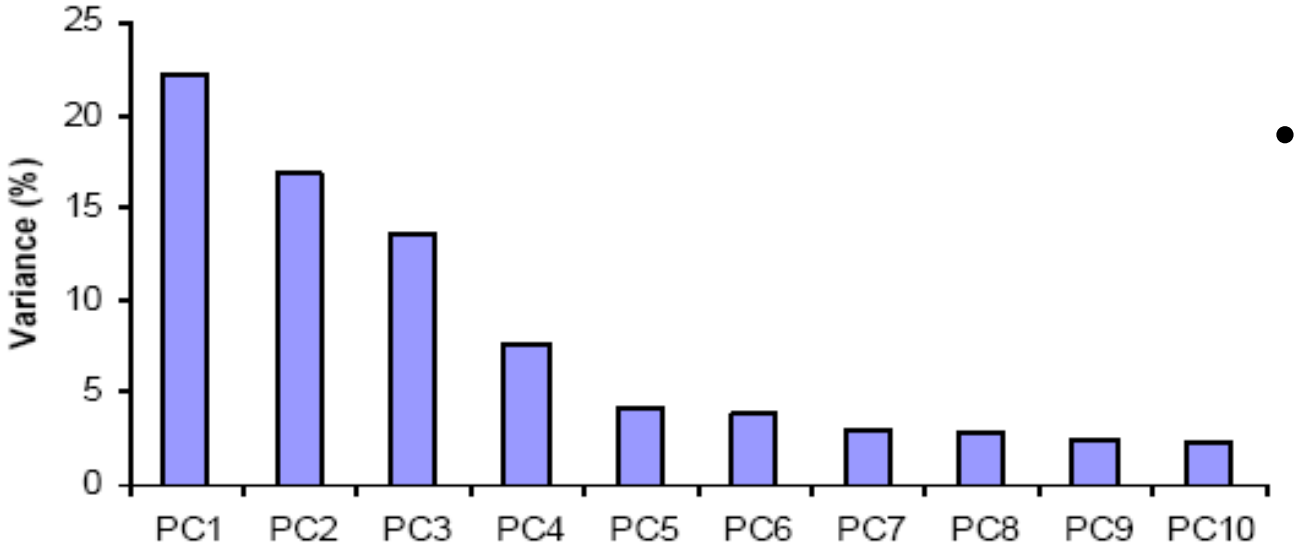
1st PCA Axis



2nd PCA Axis



Dimensionality Reduction

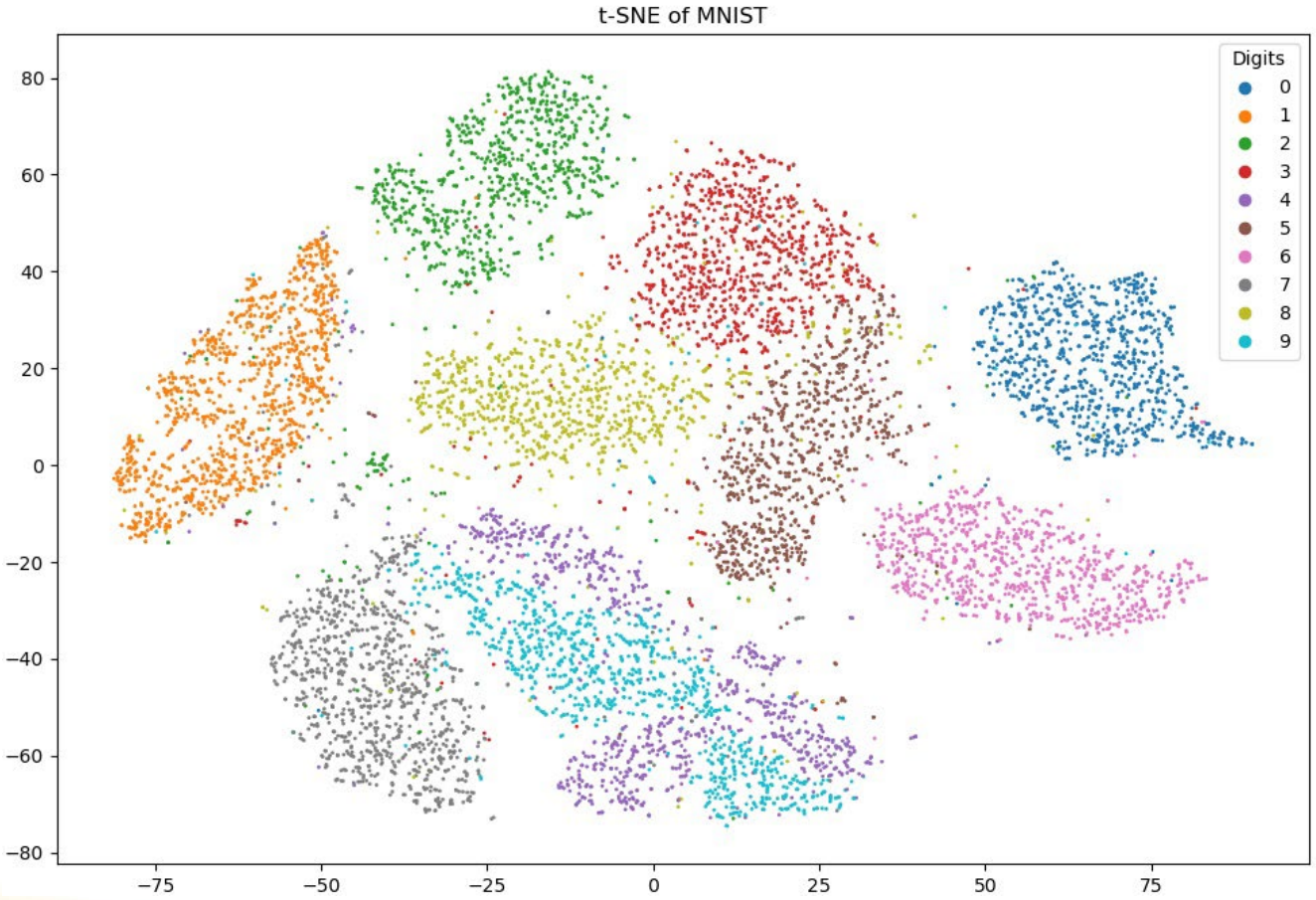


- Can ignore the components of lesser significance
- You do lose some information, but if the eigenvalues are small, you do not lose much
  - Choose only the first k eigenvectors, based on their eigenvalues
  - Final data set has only k dimensions

PCA Visualization of MNIST Digits

- Somewhat helpful, but data points are not well organized/separated among different digits
- Question: Can we do better? If so, how?

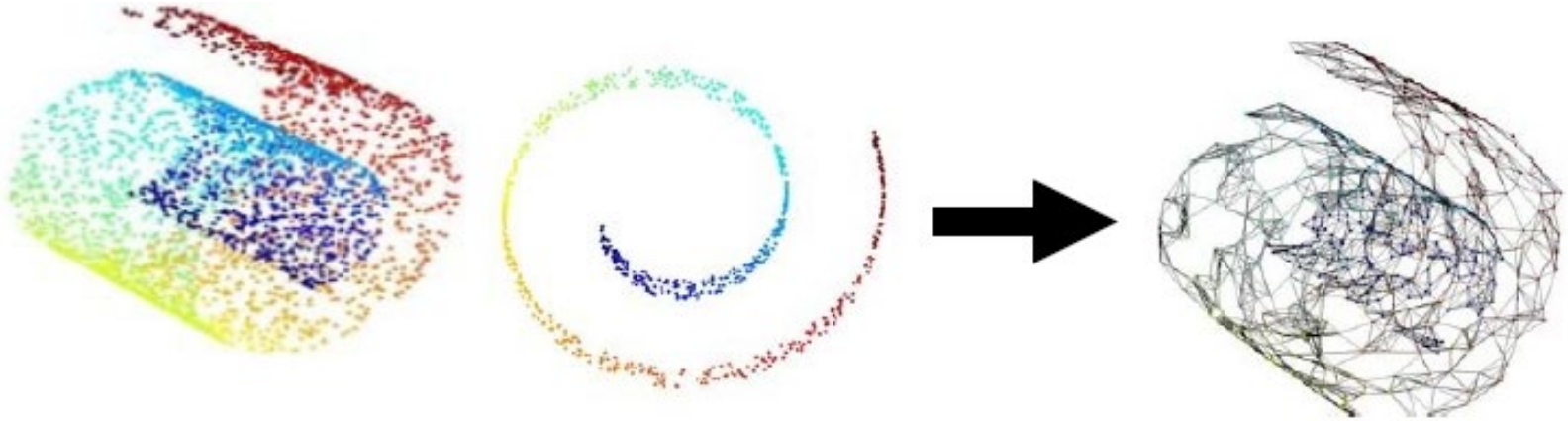
Dimensionality Reduction



Spectral Clustering

Creating a Network From a Surface

- Data points lie on surface
- Connect each point to the k closest points as measured by Euclidean distance $\mathrm{dist}(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|_2$



Creating A Network From Data (1/2)

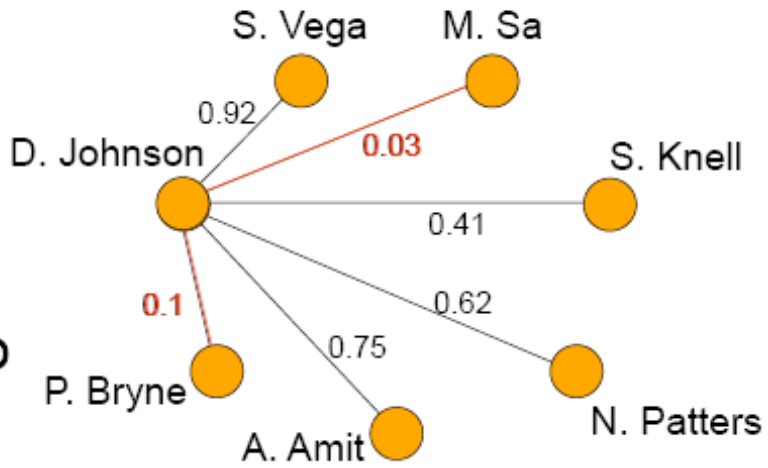
Medical Patients

Name	Age	Weight	Height	HR	SBP	DBP	SpO_2	...
D. Johnson	32	153	70	82	134	72	98%	...
S. Knell	47	169	65	130	169	93	99%	...
P. Bryne	42	128	61	102	129	77	98%	...
A. Amit	39	191	68	121	143	92	96%	...
...	...	...	...	...	...	...	...	...

1.) Measure the distance between pairs

$$\text{dist}(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|_2$$

2.) Connect each patient to its k nearest neighbors



Creating A Network From Data (2/2)

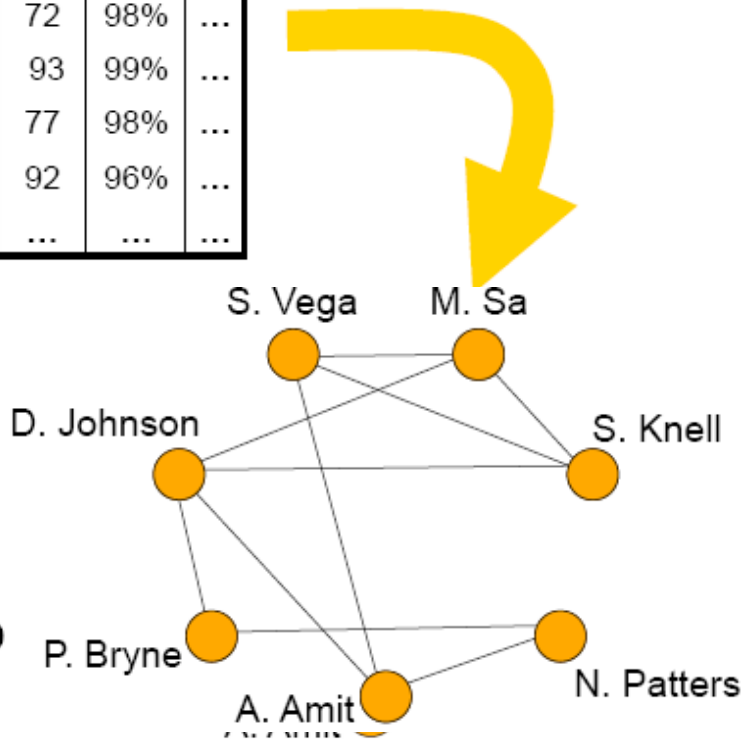
Medical Patients

Name	Age	Weight	Height	HR	SBP	DBP	SpO_2	...
D. Johnson	32	153	70	82	134	72	98%	...
S. Knell	47	169	65	130	169	93	99%	...
P. Bryne	42	128	61	102	129	77	98%	...
A. Amit	39	191	68	121	143	92	96%	...
...	...	...	...	...	...	...	...	...

1.) Measure the distance between pairs

$$\mathrm{dist}(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|_2$$

2.) Connect each patient to its k nearest neighbors

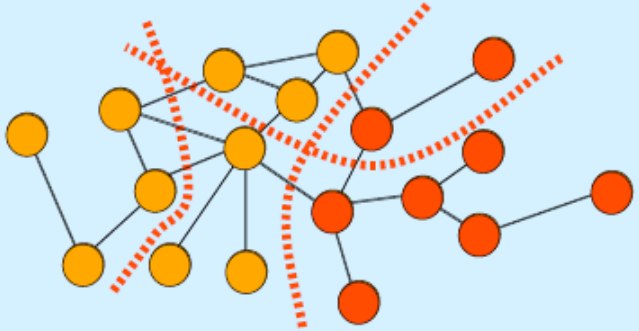


Graph constructed from data points

Graph Partitioning

- Goal: Partition the graph into multiple groups (*clusters*)

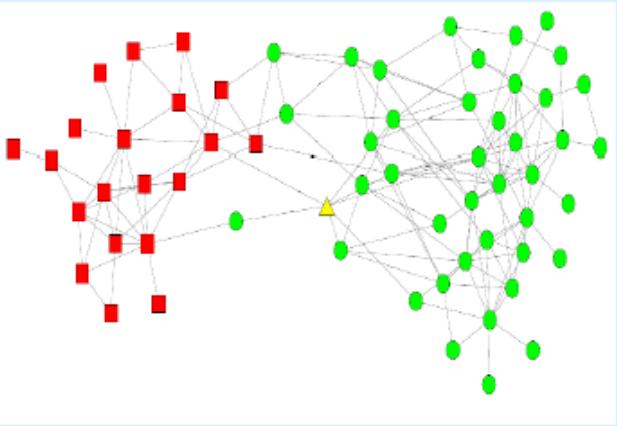
Find the two clusters



But where do we cut?

Identify the different parts of the rabbit

Social network of 62 dolphins
(Lusseau et al., 2003)



Predict how the community will split when ▲ departs

Spectral Clustering

- Algorithms that cluster points using eigenvectors of matrices derived from the data
- Obtain data representation in the low-dimensional space that can be easily clustered
- Variety of methods that use eigenvectors differently

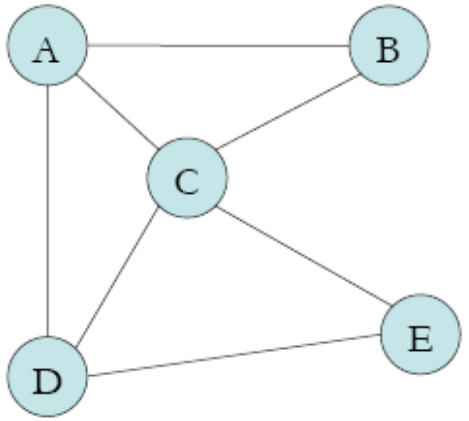
Definitions

- **NxN adjacency matrix A**
  - A(i,j) = weight on edge from i to j
  - If the graph is undirected A(i,j) = A(j,i) , i.e., A is symetric
- **NxN Transition matrix P**
  - P is row stochastic (i.e., row sums are equal to 1)
  - P(i,j) = probability of stepping on node j from node i= A(i,j) /sum(i, A(i,j))
- **NxN Laplacian matrix L**
  - L(i,j)=sum(i, A(i,j))- A(i,j)
  - Symmetric positive semi-definite for undirected graphs
  - Singular (has at least one eigenvalue of 0)

Definitions

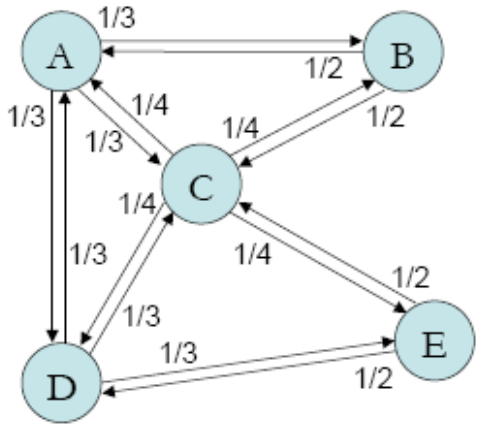
Adjacency Matrix

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



Transition Matrix

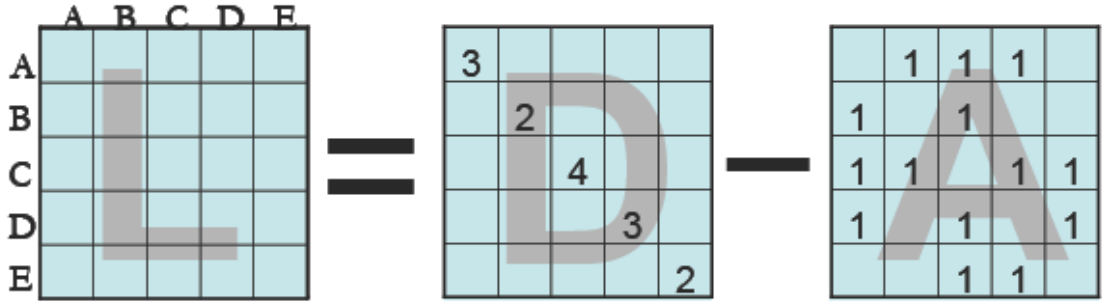
$$P = \begin{bmatrix} & 1/3 & 1/3 & 1/3 & \\ 1/2 & & 1/2 & & \\ 1/4 & 1/4 & & 1/4 & 1/4 \\ 1/3 & & 1/3 & & 1/3 \\ & & 1/2 & 1/2 & \end{bmatrix}$$

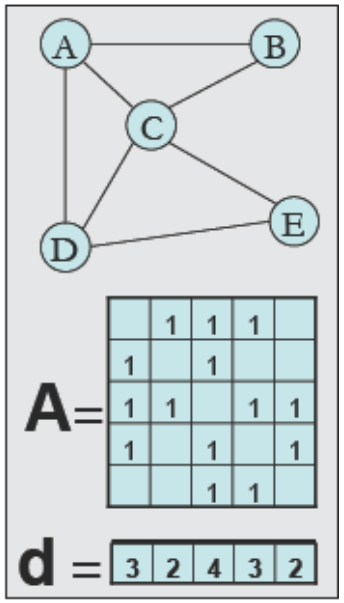


Spectral Graph Analysis

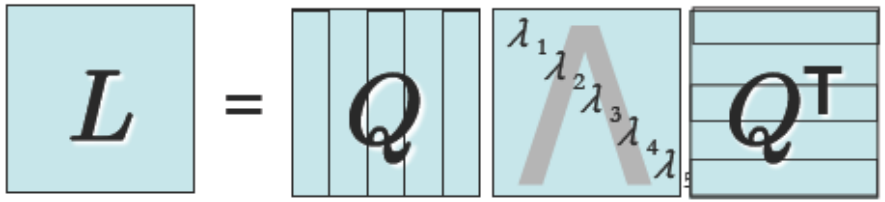
Graph Laplacian

$$L = D - A \qquad D = \text{diag}(d)$$





Take the *eigendecomposition* of L



Symmetric Matrices

- Recall that a real-valued matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ is said to be **symmetric** if $\mathbf{A} = \mathbf{A}^T$
- Useful facts

  S1. Eigenvalues of a symmetric real matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ are real

  S2. Eigenvectors of a symmetric real matrix corresponding to two distinct eigenvalues are **orthogonal**

  S3. A is nondefective and has a complete set of orthonormal eigenvectors

- Example:

$$\mathbf{B} = \begin{bmatrix} \frac{3}{2} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{3}{2} \end{bmatrix} \Rightarrow \lambda_1 = 1,\ \lambda_2 = 2,\ \mathbf{v}_1 = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix},\ \mathbf{v}_2 = \begin{bmatrix} -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$$

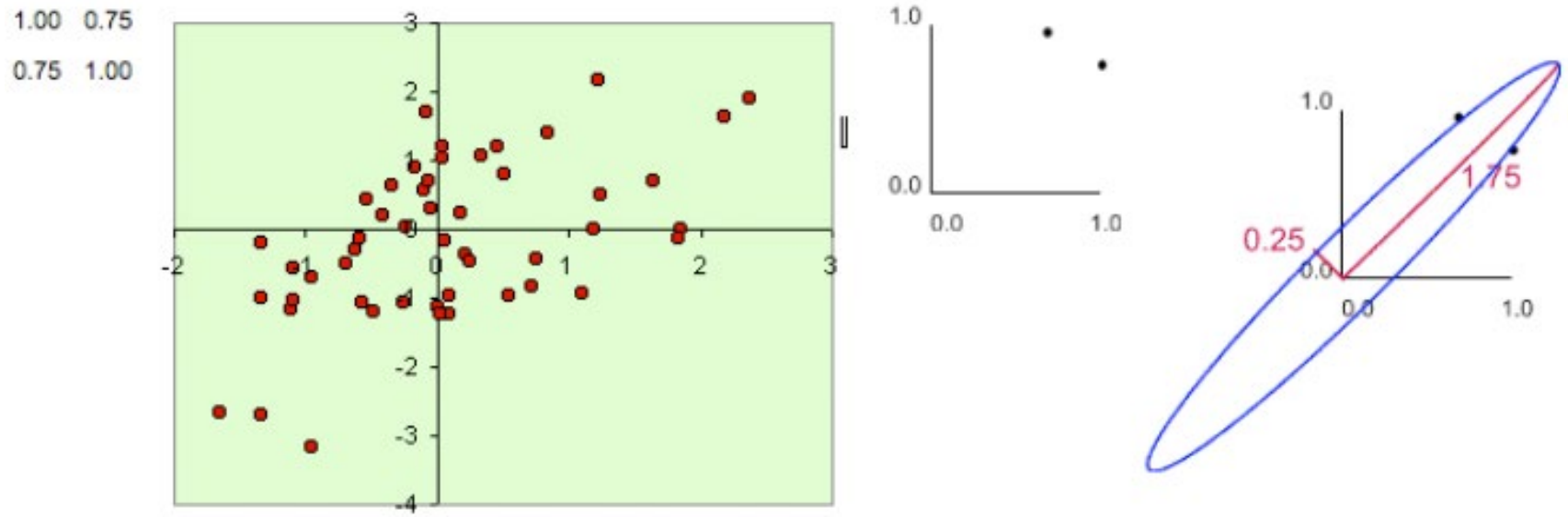
$$\mathbf{A} = \begin{bmatrix} 2 & -\sqrt{2} \\ -\sqrt{2} & 3 \end{bmatrix} \Rightarrow \lambda_1 = 1,\ \lambda_2 = 4,\ \mathbf{v}_1 = \begin{bmatrix} \frac{1}{\sqrt{3}} \\ \frac{\sqrt{2}}{\sqrt{3}} \end{bmatrix},\ \mathbf{v}_2 = \begin{bmatrix} -\frac{\sqrt{2}}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} \end{bmatrix}$$

Eigenvectors

- Intuitive interpretation: an eigenvector is a direction for a matrix
- Recall: An eigenvector of an n x n matrix A is a vector such that $Av = \lambda v$, where v is the eigenvector and λ is the corresponding eigenvalue
  - Multiplying vector v by the scalar λ effectively stretches or shrinks the vector
- An n x n symmetric matrix have n linearly independent eigenvectors

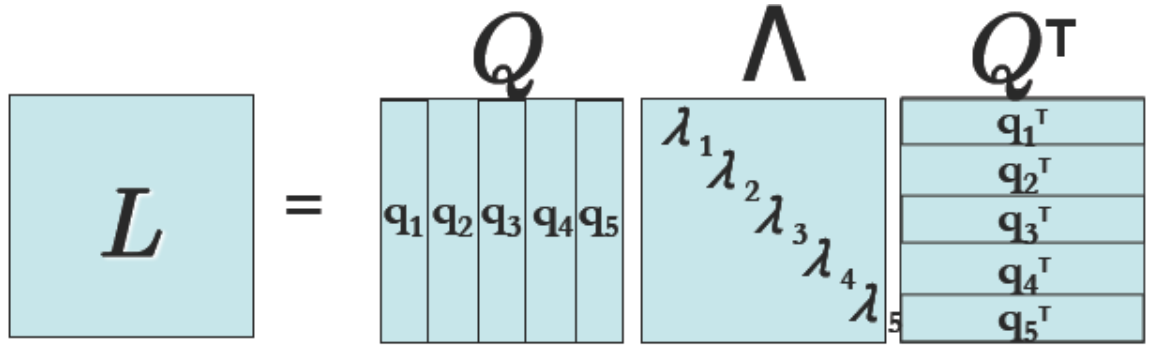
Eigenvectors Illustrated

- Consider an elliptical data cloud. The eigenvectors are then the major and minor axes of the ellipse



Spectral Graph Analysis (1/2)

Eigendecomposition



Eigenvector q_1 is constant

Eigenvalue $\lambda_1 = 0$

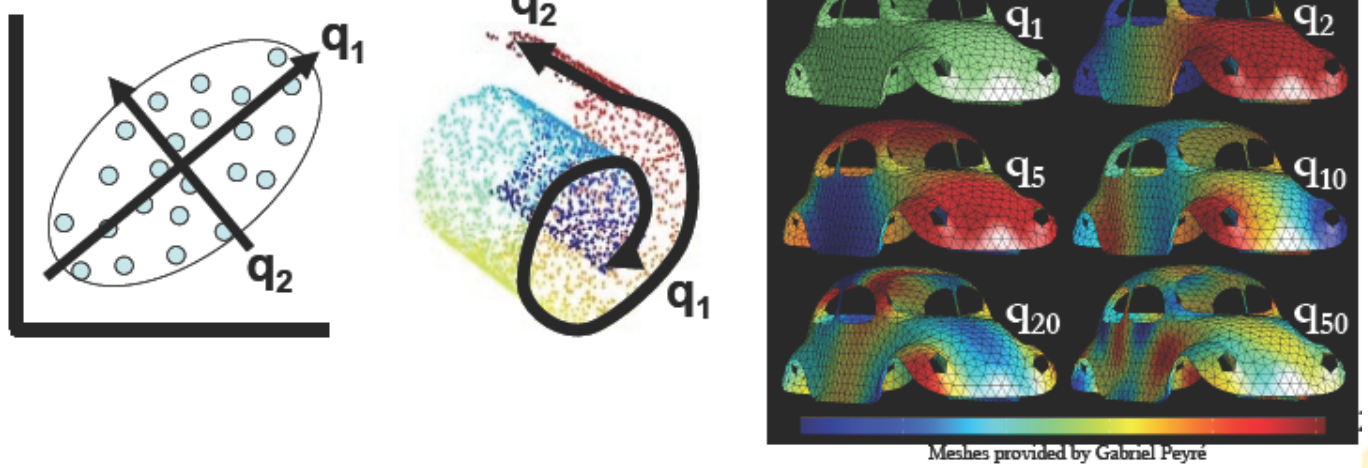
	q1	q2	q3	q4	q5
A	0.45	-0.27	-0.5	-0.65	0.22
B	0.45	-0.65	0.5	0.27	0.22
C	0.45	-0.00	0.00	0.00	-0.89
D	0.45	0.27	-0.5	0.65	0.22
E	0.45	0.65	0.5	-0.27	0.22

1	2	3	4	5
0.00	0	0	0	0
0	1.59	0	0	0
0	0	3.00	0	0
0	0	0	4.41	0
0	0	0	0	5.00

3	-1	-1	-1	
-1	2	-1		
-1	-1	4	-1	-1
-1		-1	3	-1
		-1	-1	2

Spectral Graph Analysis (2/2)

$$L = Q \Lambda Q^T$$



Method 1

- Partition using only one eigenvector at a time
- Use procedure recursively
- Example: Image segmentation
  - Use 2nd (smallest) eigenvector to define optimal cut
  - Recursively generates two clusters with each cut

Method 2

- Use k eigenvectors (k chosen by user)
- Directly compute k-way partitioning
- Experimentally has been seen to be “better”

Example

- SpectGraph.m

Spectral Clustering Algorithm

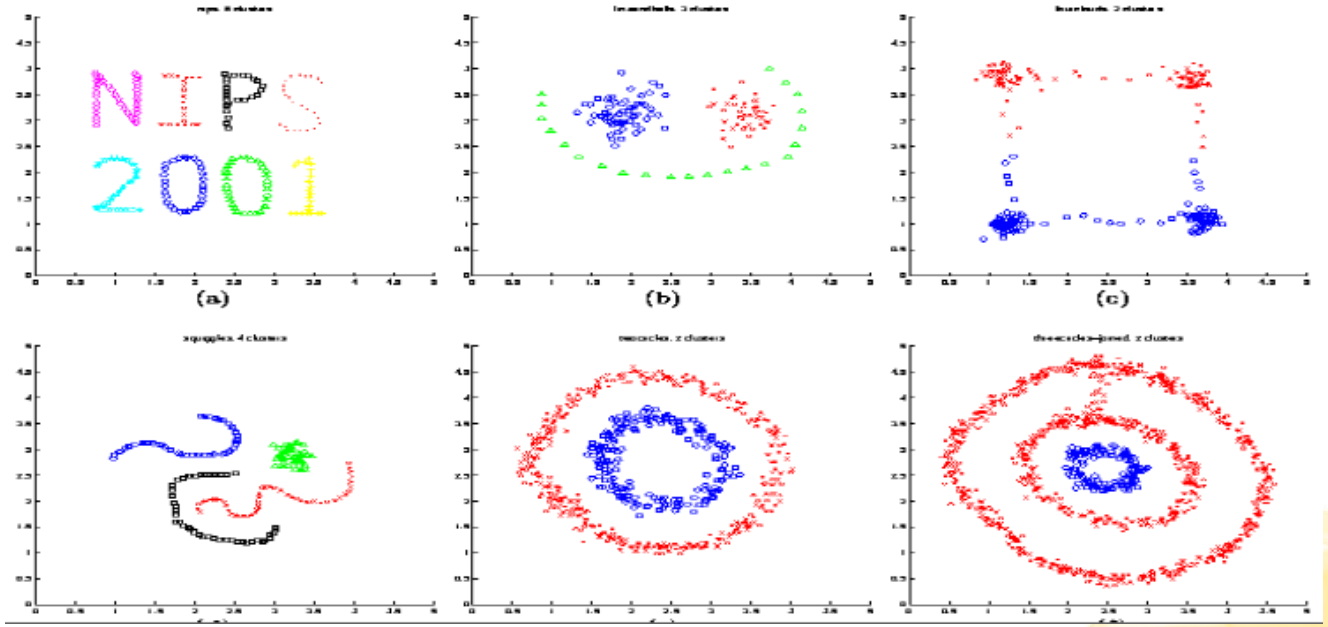
- Given a set of points $\mathrm{X}=\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$
- Form the affinity matrix

$$A_{ij} = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\sigma^2}\right) \quad \forall i \neq j \quad A_{ii} = 0$$

- Define diagonal matrix $D_{ii} = \sum_k A_{ik}$
- Form the matrix $\mathbf{L} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$
- Stack the k largest eigenvectors of $\mathbf{L}$ to form the columns of the new matrix: $\mathbf{E} = [\mathbf{e}_1 \ \mathbf{e}_2 \ \ldots \ \mathbf{e}_k]$
- Normalize each of $\mathbf{E}$'s rows to have unit length
- Cluster rows of $\mathbf{E}$ into k clusters using K-means

Why Spectral Clustering?

- Question: If we eventually use k-means clustering, why not just apply k-means clustering to the original data?
- Answer: This method allows us to cluster non-convex regions
- Example: SpectCluster.ms



Multidimensional Scaling

Multi-Dimensional Scaling

- Given pairwise dissimilarities, reconstruct a map that preserves distances
  - *Given an n-by-n distance matrix D*
  - *Find n points* $\{y_1, y_2, \ldots, yn\}$*, such that their pairwise distance resembles those of D in a lower-dimensional space*

- Objective function:

$$\sum_{i,j=1}^{n} \left(D_{ij} - D_{ij}^{Y}\right)^2$$

$D_{ij}^{X} = \|x_i - x_j\|$ $D_{ij}^{Y} = \|y_i - y_j\|$

Classical Multi-Dimensional Scaling

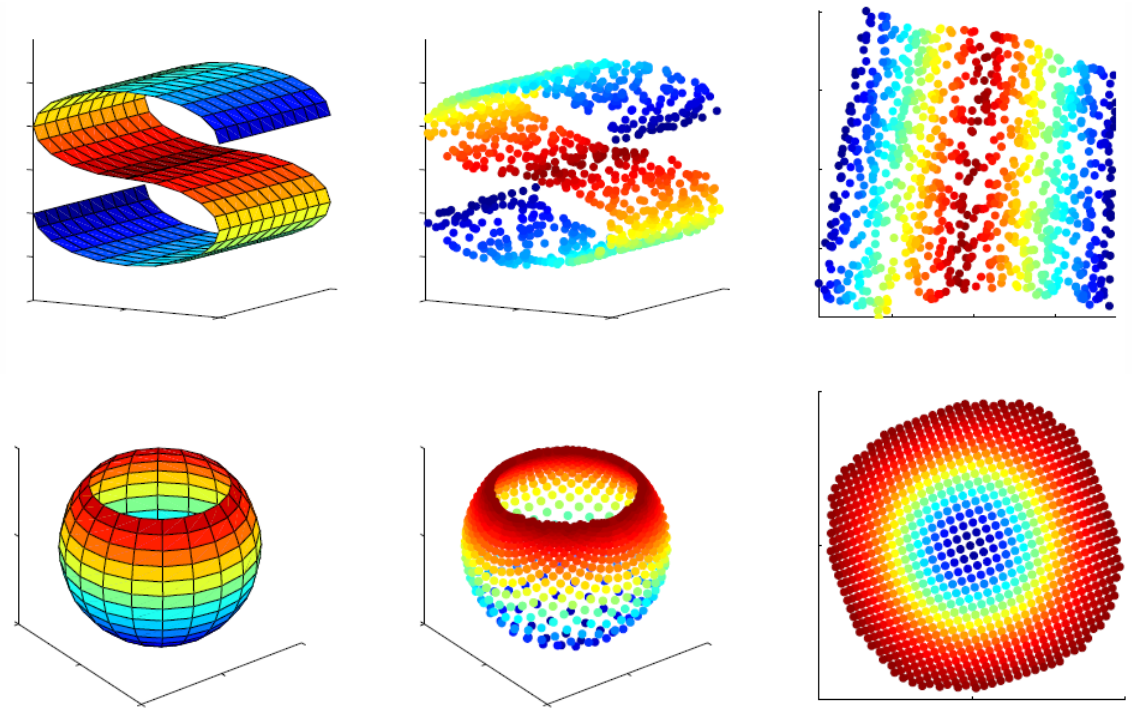
- How do we find the low-dimensional representation?
  - Suppose X is the dataset with data points as rows
  1. Compute the centered Gram matrix $B = HXX^TH,$
  2. Find the eigenvalues and eigenvectors of B: $(\lambda_i, \mathbf{u}_i), i = 1, \ldots, k$, ordered by decreasing eigenvalues
  3. Representation in k-dimensional space given by

$$Y = [\mathbf{u}_1 \ \ldots \ \mathbf{u}_k]\mathrm{diag}(\lambda_1, \ldots, \lambda_k)$$

- Example: MDS_example.m

Manifold Learning

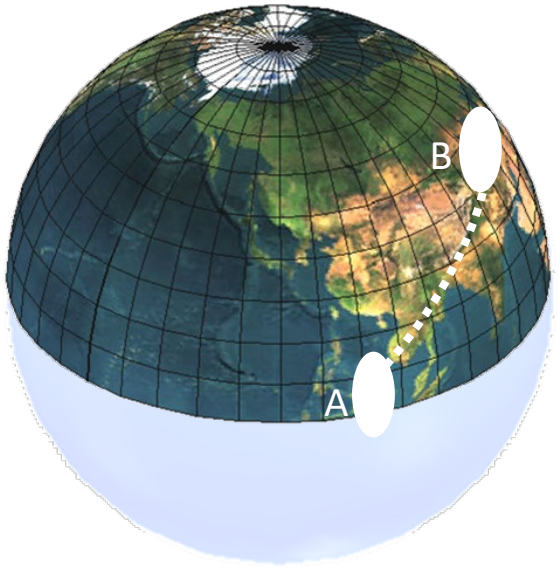
Manifold Learning



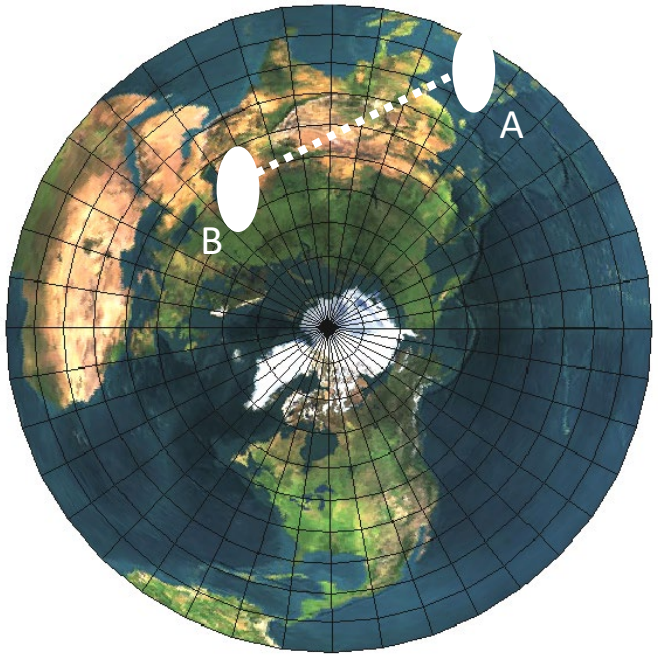
- **Manifold Learning** (or non-linear dimensionality reduction) embeds data that originally lies in a high dimensional space in a lower dimensional space, while preserving characteristic properties
  - a manifold is a topological space that locally resembles Euclidean space near each point.

Mapmaker's Problem

- How do we represent a curved surface (the Earth) on a flat/planar map?
  - Different ways of projections lead to different depictions with varying distortion
- Unclear what the best way to represent complex surfaces while minimizing distortion of distance relationships

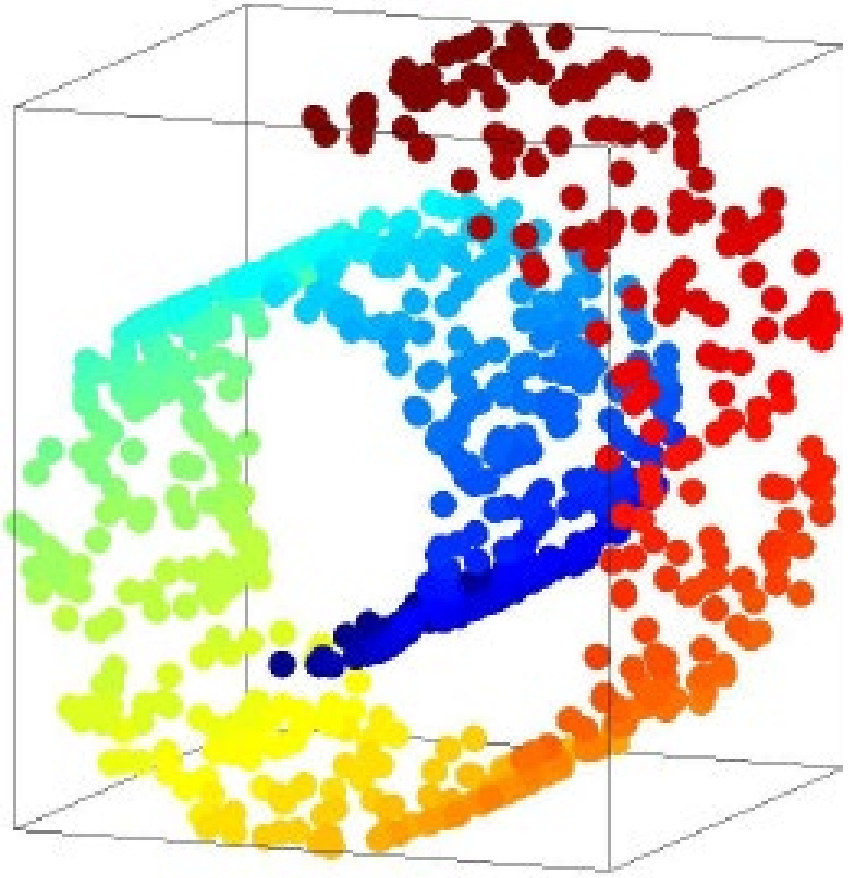


Earth (sphere)



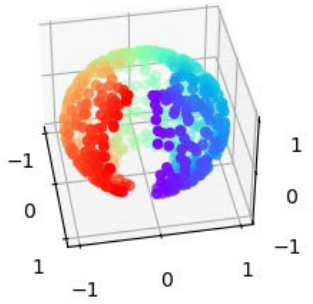
Planar map

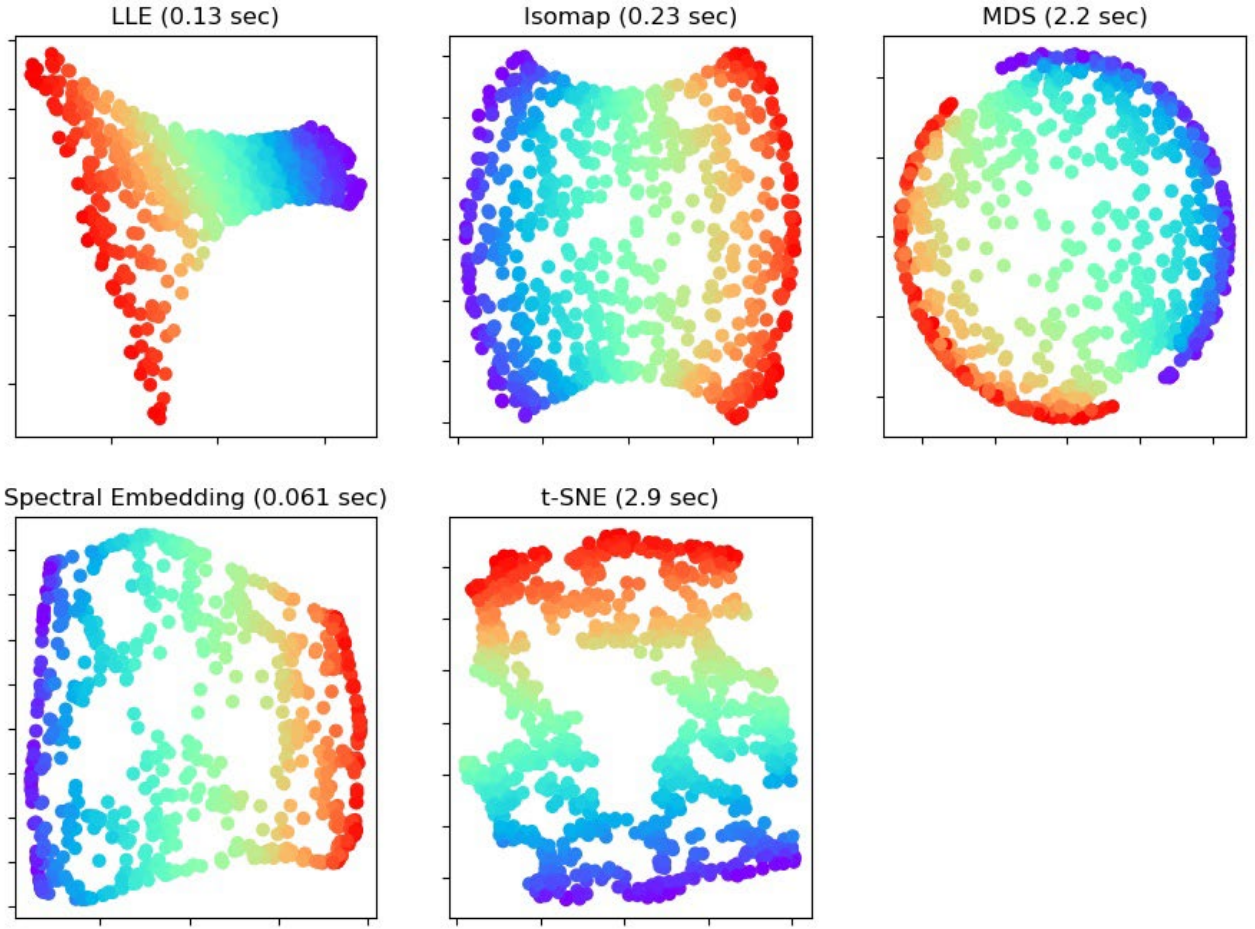
Manifold Learning



- Find a low-dimensional basis for describing high-dimensional data
- $X \rightarrow X'$ such that $\dim(X') << \dim(X)$
- Uncovers the intrinsic dimensionality (invertible)

Manifold Learning



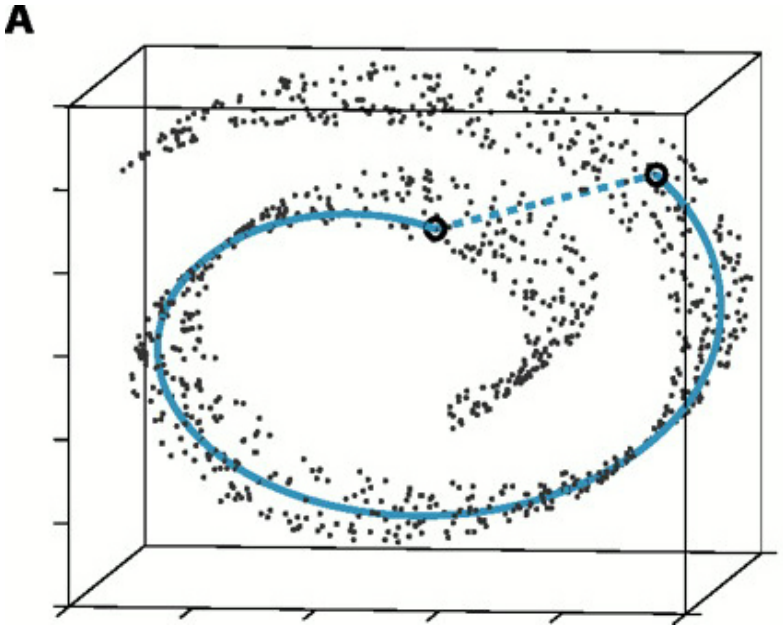


Why Manifold Learning?

1. Data compression
2. “Curse of dimensionality”
3. De-noising
4. Visualization
5. and more …

Isomap

- From Euclidian distance to manifold distance

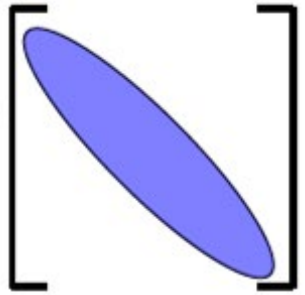


- **Neighboring** points: input-space distance
- **Faraway** points: a sequence of “short hops” between neighboring points
- **Method**: Finding shortest paths in a graph with edges connecting neighboring data points

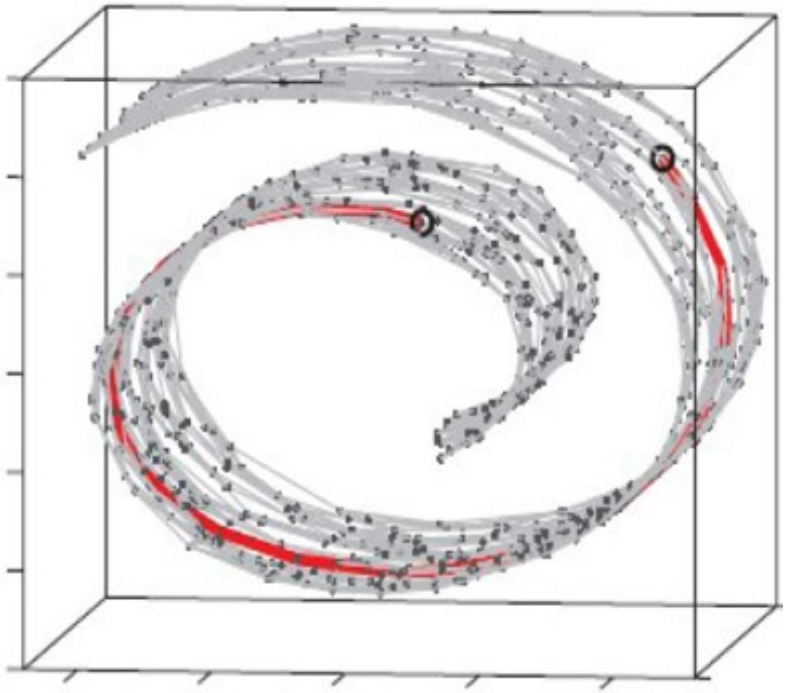
Unlike the geodesic distance, the Euclidean distance cannot reflect the geometric structure of the data points

Isomap

1. Build a sparse graph with nearest neighbors using a threshold on distance (or k-nearest neighbors)

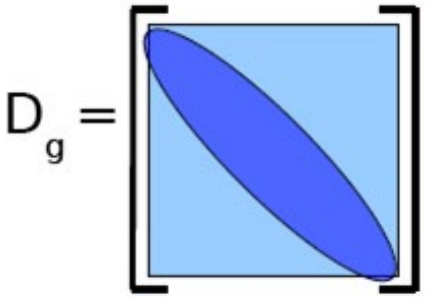
$D_g =$ 

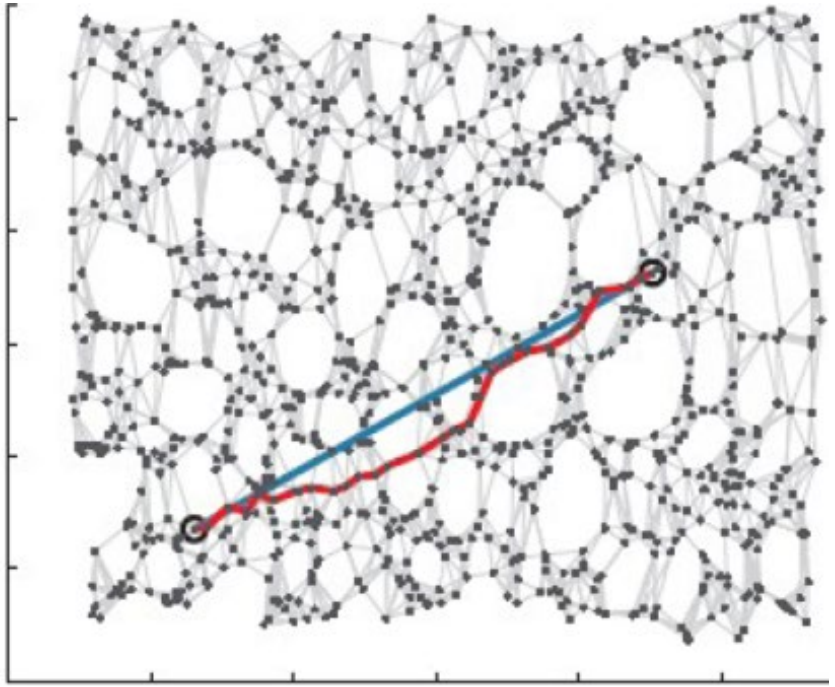
(distance matrix is sparse)



Isomap

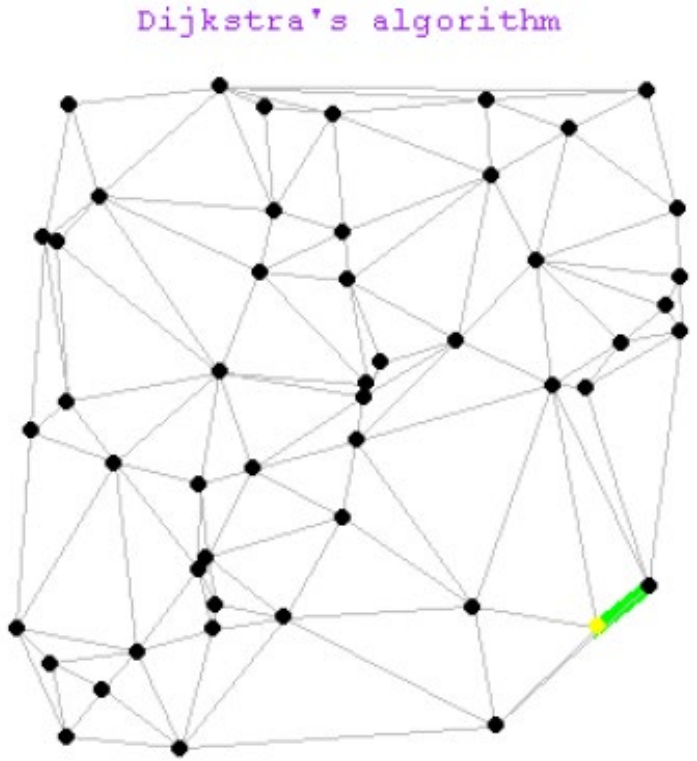
2. Infer other interpoint distances by finding shortest paths on the graph (**Dijkstra's algorithm**)





Isomap

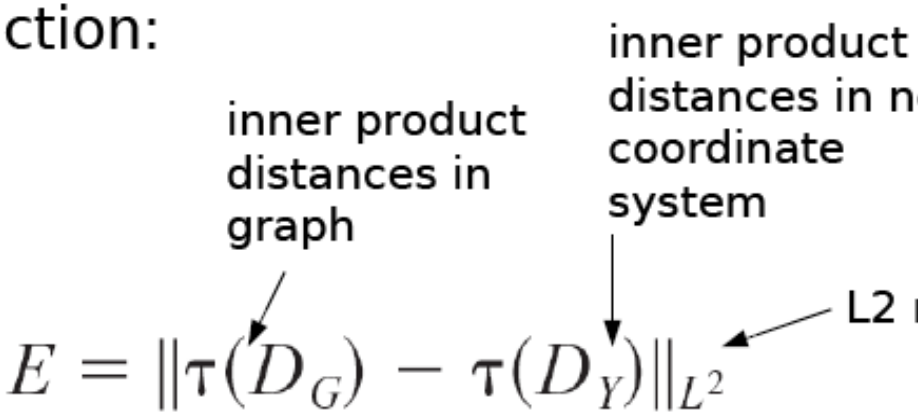
- Shortest-distance on a graph is easy to compute



Isomap

3. Build a low-dimensional embedded space to best preserve the complete distance matrix

Error function:



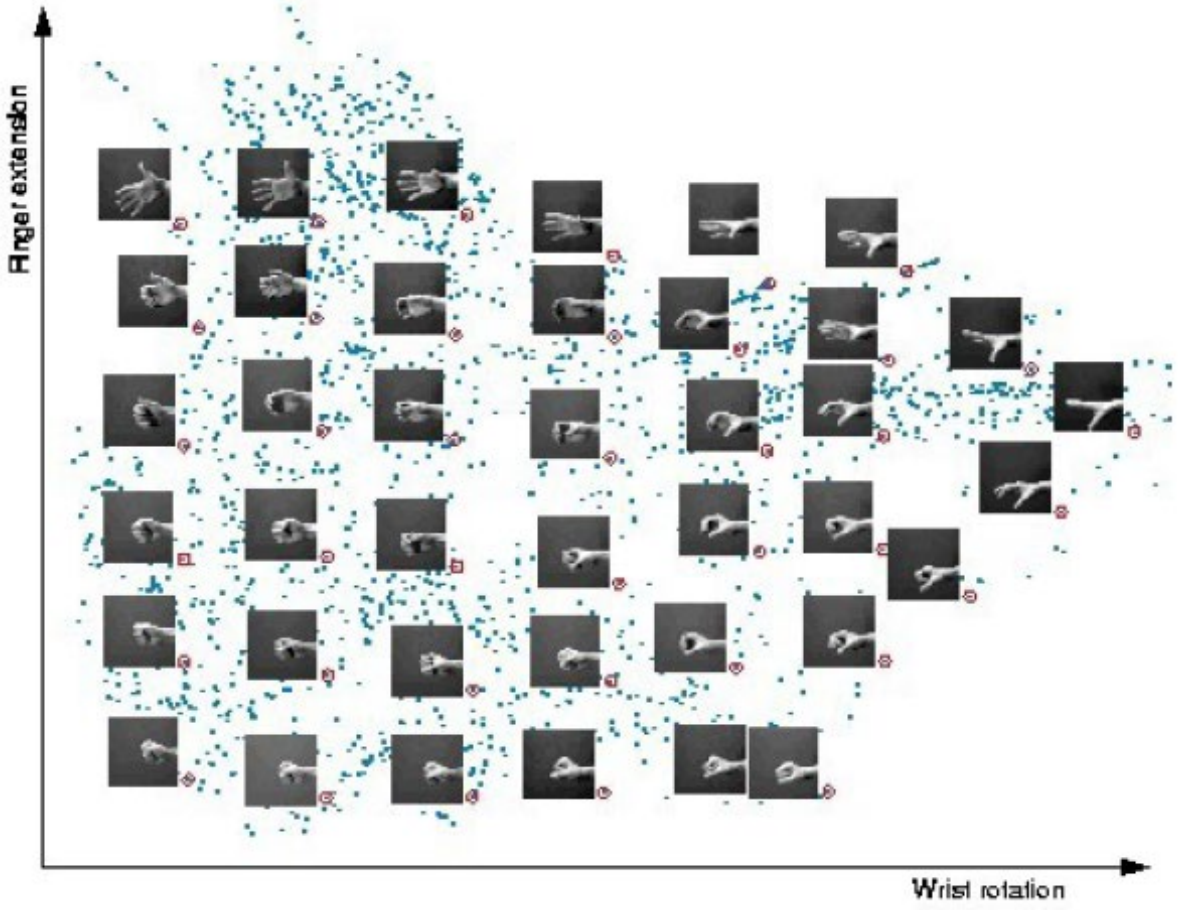
Solution – set points Y to top eigenvectors of D_g

Isomap

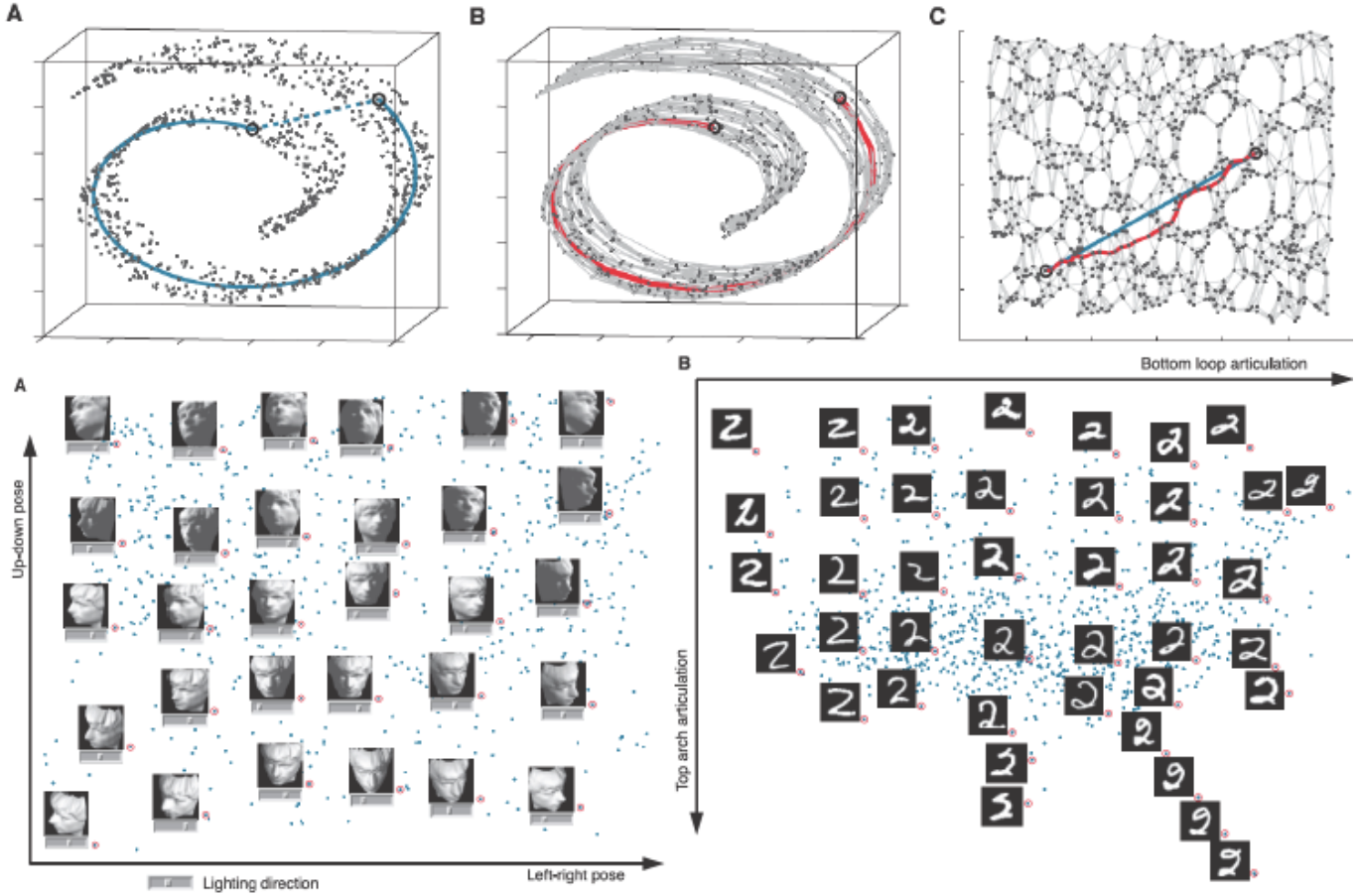
ISOMAP

- we start with many data points in high dimensional space, lying near some manifold
- For each data point i we find the points j on manifold within some Euclidean distance $d_X(i, j) \leq \epsilon$
- We construct a graph on the manifold with an edge between i and j if $d_X(i, j) \leq \epsilon$
- We find the shortest path $d_G(i, j)$ between points i and j on the graph.
- Finally, we apply classical MDS to the distances $d_G(i, j)$

Isomap Results: Hands



Isomap

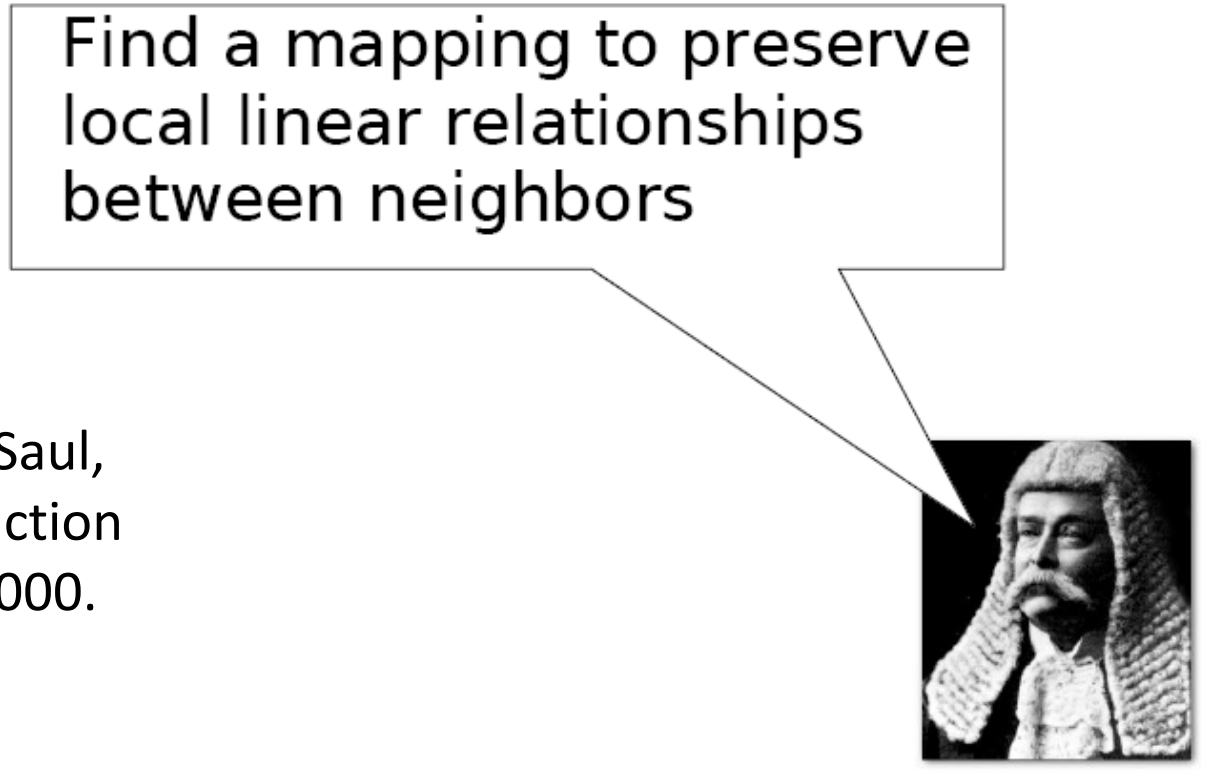


Isomap: Pros and Cons

- preserves global structure
- few free parameters
- sensitive to noise, noise edges
- computationally expensive (dense matrix eigen-reduction)

Locally Linear Embedding

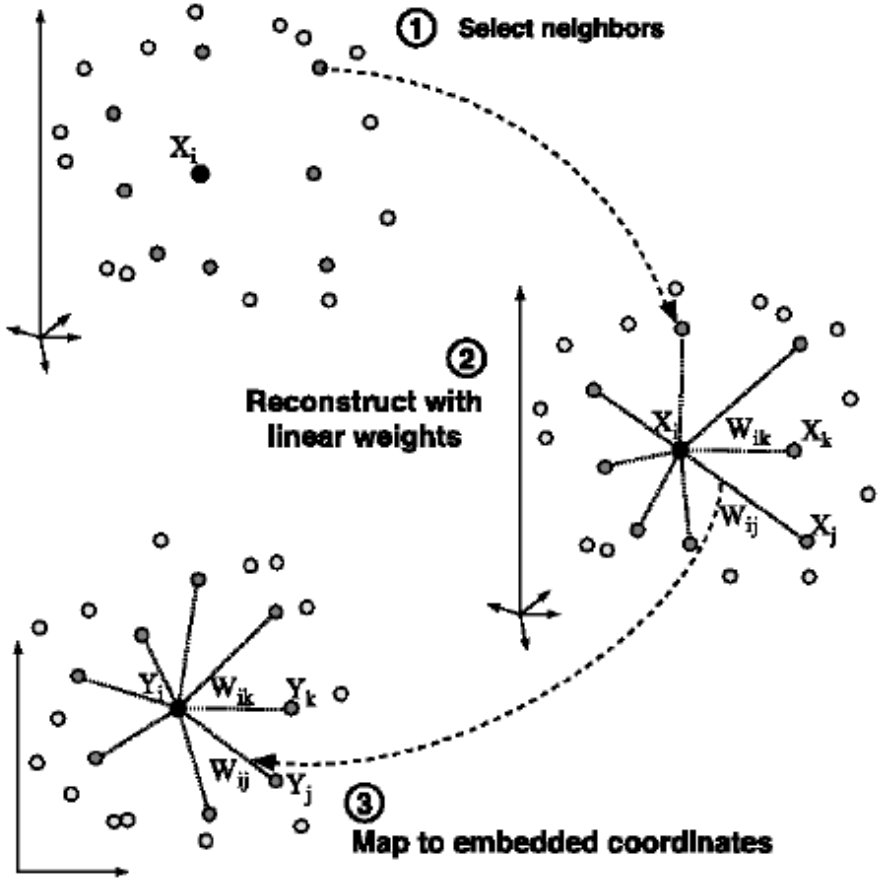
Locally Linear Embedding



- Sam T. Roweis and Lawrence K. Saul, “Nonlinear Dimensionality Reduction by Locally Linear Embedding”, 2000.

Locally Linear Embedding

1. For each data point, select K nearest neighbors using some distance metric
2. Reconstruct each data point as a weighted sum of neighbors
3. Map to embedded coordinates of low-dimensional space using the weights from step 2



Locally Linear Embedding: Why Neighbors?

1. LLE assumes that each point and its neighbors form a locally linear patch on some low dimensional manifold
   - LLE tries find low dimensional embedding that preserves the “local” structure in dataset (through weights in Step 2)
   - Assumes that data points lie on a smooth manifold of low dimension

Locally Linear Embedding: Two Key Steps

2. Find weight matrix W of linear coefficients:

$$\mathcal{E}(W) = \sum_i \left| \vec{X}_i - \sum_j W_{ij} \vec{X}_j \right|^2$$

- Weights must sum to one for each data point: $\sum_j W_{ij} = 1$

- Weight for non-neighbor set to zero
- Weights summarize the contributions of neighbors to data point
- LLE tries to preserves the weights when reducing dimensionality of data

Locally Linear Embedding: Two Key Steps

3. Find projected vectors Y to minimize reconstruction error:

$$\Phi(Y) = \sum_i \left| \vec{Y}_i - \sum_j W_{ij} \vec{Y}_j \right|^2$$

- Must solve for whole dataset simultaneously
- Preserve the weights calculated in high-dimensional space as much as possible
- Constraints imposed to avoid degenerate solution

$$\frac{1}{n} \sum_{i=1}^{n} \vec{Y}_i \vec{Y}_i^T = \mathbf{I}$$

outer product

Locally Linear Embedding

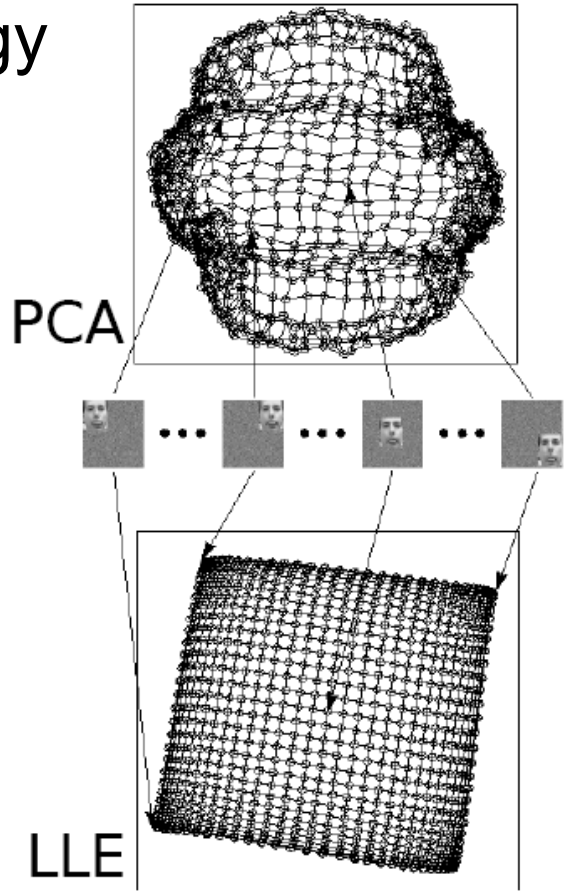
- Solution given by the eigenvectors of

$$(\mathbf{I} - \mathbf{W})^T(\mathbf{I} - \mathbf{W})$$

  - Eigenvector corresponding to the smallest eigenvalue is constant
  - Eigenvectors corresponding to the next p smallest eigenvalues used as the coordinates of data points (where p is the dimension of the lower dimensional space)

Locally Linear Embedding: Result

- Preserves local topology



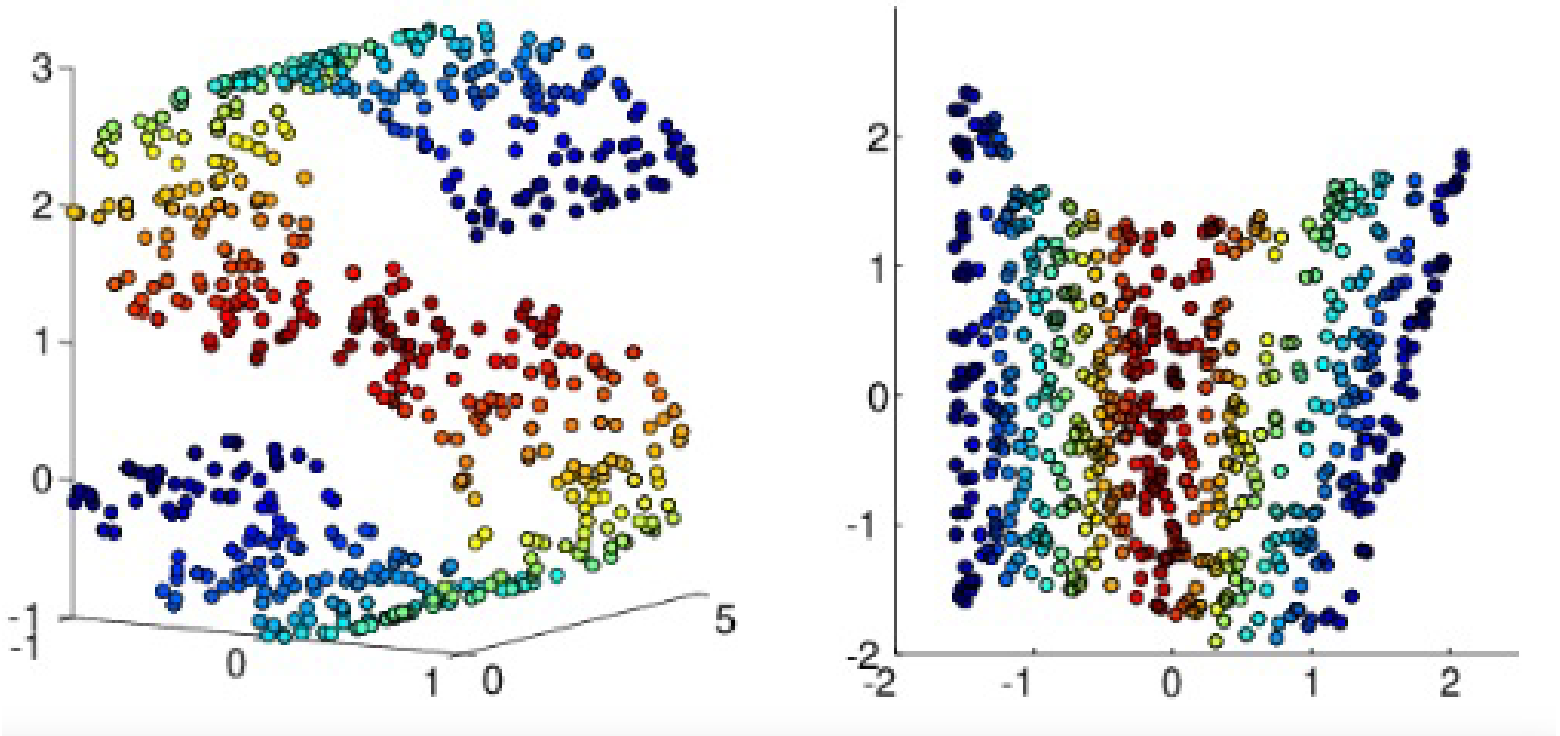
Locally Linear Embedding: Pros

- Effective for manifold learning when linear models (e.g., PCA) fail
- No local minima, one free parameter (K)
- Incremental and fast
- Simple linear algebra operations (considers eigenvectors corresponding to smallest eigenvalues of some matrix constructed from weights)
- Preserves local geometry of data

Locally Linear Embedding: Cons

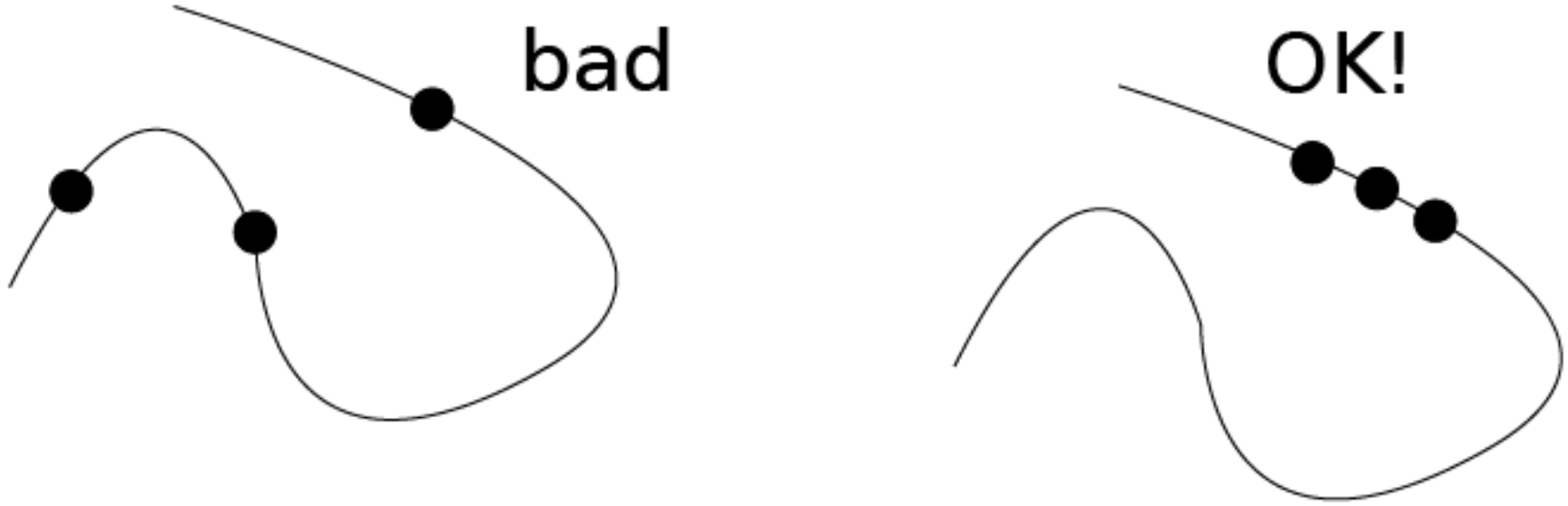
- Can distort global structure (because it only considers “local” properties)
- Computationally heavy with large dataset (needs to find K nearest neighbors for every data point and solve an eigenvalue problem)
- Sensitive to noise and choice of K

Locally Linear Embedding in Action



No Free Lunch

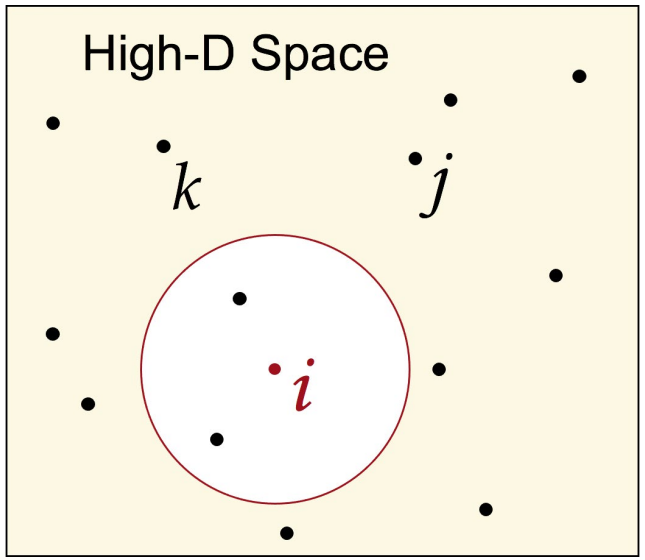
- The “curvier” your manifold, the denser your data must be



t-SNE

Stochastic Neighborhood Embedding

- A probabilistic version of local MDS
  - More important to get local distances right than non-local ones
  - Has a probabilistic way of deciding if a pairwise distance is “local”

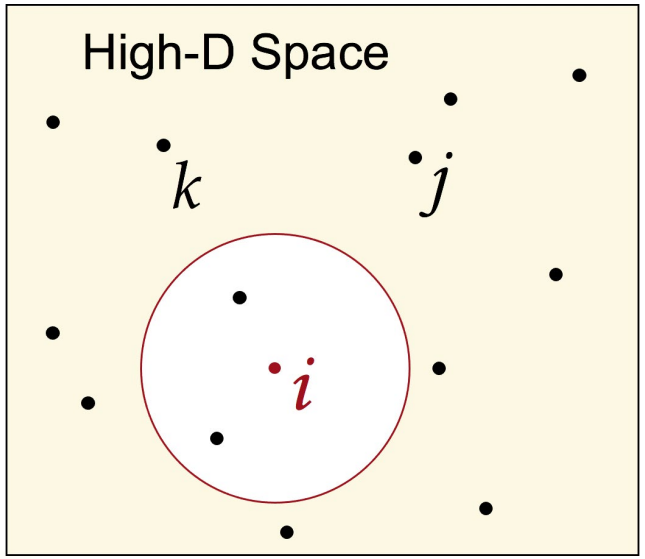


probability of picking j
given that you start at i

$$p_{j|i} = \frac{e^{-d(\mathbf{x}_i,\mathbf{x}_j)}}{\sum_{k\neq i} e^{-d(\mathbf{x}_i,\mathbf{x}_k)}}$$

Stochastic Neighborhood Embedding

- Give each data point a location in the low-dimensional space
  - Evaluate this representation by seeing how well the low-dimensional probabilities model the high-dimensional ones



probability of picking j
given that you start at i

$$q_{j|i} = \frac{e^{-d_{ij}^2}}{\sum_k e^{-d_{ik}^2}}$$

The Cost Function

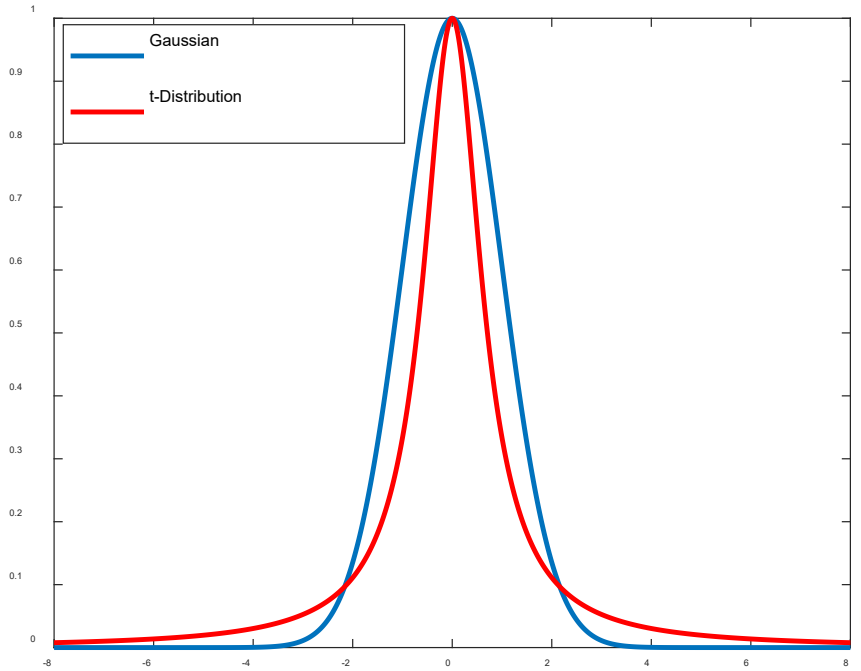
- Preserve the **probabilities of points being neighbors**

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

- For points where $p_{j|i}$ is large and $q_{j|i}$ is small we lose a lot
  - Nearby points in high-dimension really want to be nearby in low-dimension
- For points where $q_{j|i}$ is large and $p_{j|i}$ is small we lose a little because we waste some of the probability mass in the Q_i distribution.
  - Widely separated points in high-dimension have a mild preference for being widely separated in low-dimension

T-SNE

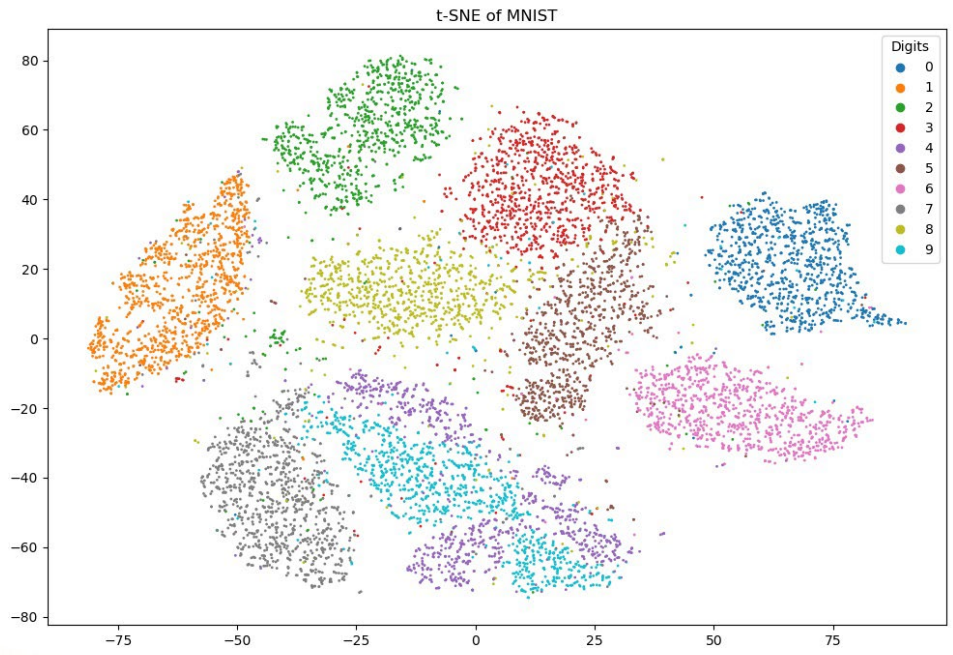
- By using a Gaussian to compute $p_{j|i}$ and a heavy-tailed student's t to compute $q_{j|i}$
  - a heavy-tailed Student-t distribution (with one-degree of freedom, which is the same as a Cauchy distribution) is used to measure similarities between low-dimensional points in order to allow dissimilar objects to be modeled far apart in the map

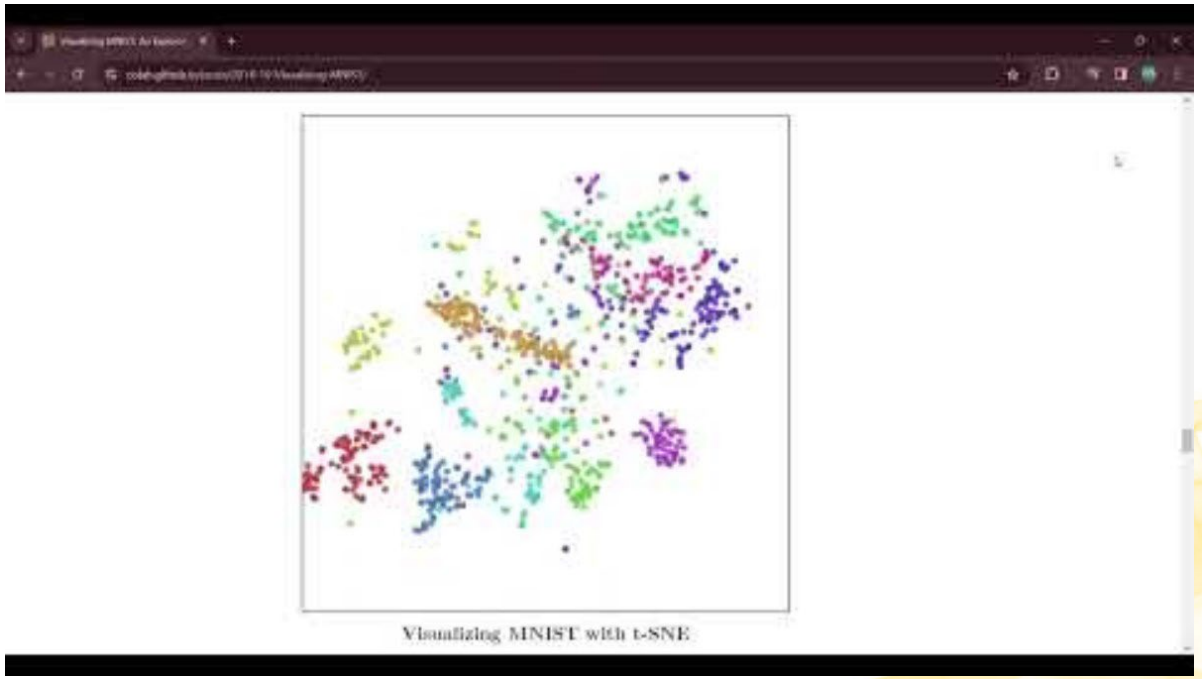


$$p_{j|i} = \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma_i^2)}{\sum_{k\neq i}\exp(-\|\mathbf{x}_i - \mathbf{x}_k\|^2/2\sigma_i^2)} \qquad q_{ij} = \frac{(1+\|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}}{\sum_{k\neq i}(1+\|\mathbf{y}_i - \mathbf{y}_k\|^2)^{-1}}$$

T-SNE

Visualization of 5,000 digits from the MNIST dataset produced by t-SNE.





t-SNE

- Pros
  - Handles nonlinear data
  - Preserves local structure in dataset
- Cons
  - Computationally complex – needs to compute pairwise conditional probabilities for each data point
  - Requires hyperparameter tuning