UNSUPERVISED LEARNING

LECTURE : MANIFOLD LEARNING

slides due to L.Saul, V. C. Raykar, N. Verma

Topics

- PCA
- MDS

Done!

- IsoMap
- LLE
- EigenMaps

Dimensionality Reduction

- Data representation

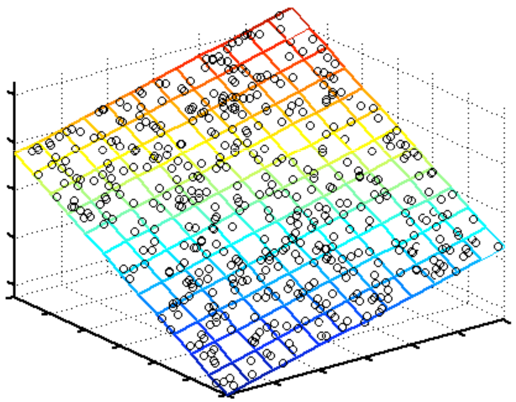
  Inputs are real-valued vectors in a high dimensional space.

- Linear structure

  Does the data live in a low dimensional subspace?

- Nonlinear structure

  Does the data live on a low dimensional submanifold?



Notations

- Inputs (**high dimensional**)

  $\boldsymbol{x_1},\boldsymbol{x_2},\ldots,\boldsymbol{x_n}$ points in R^D

- Outputs **(low dimensional)**

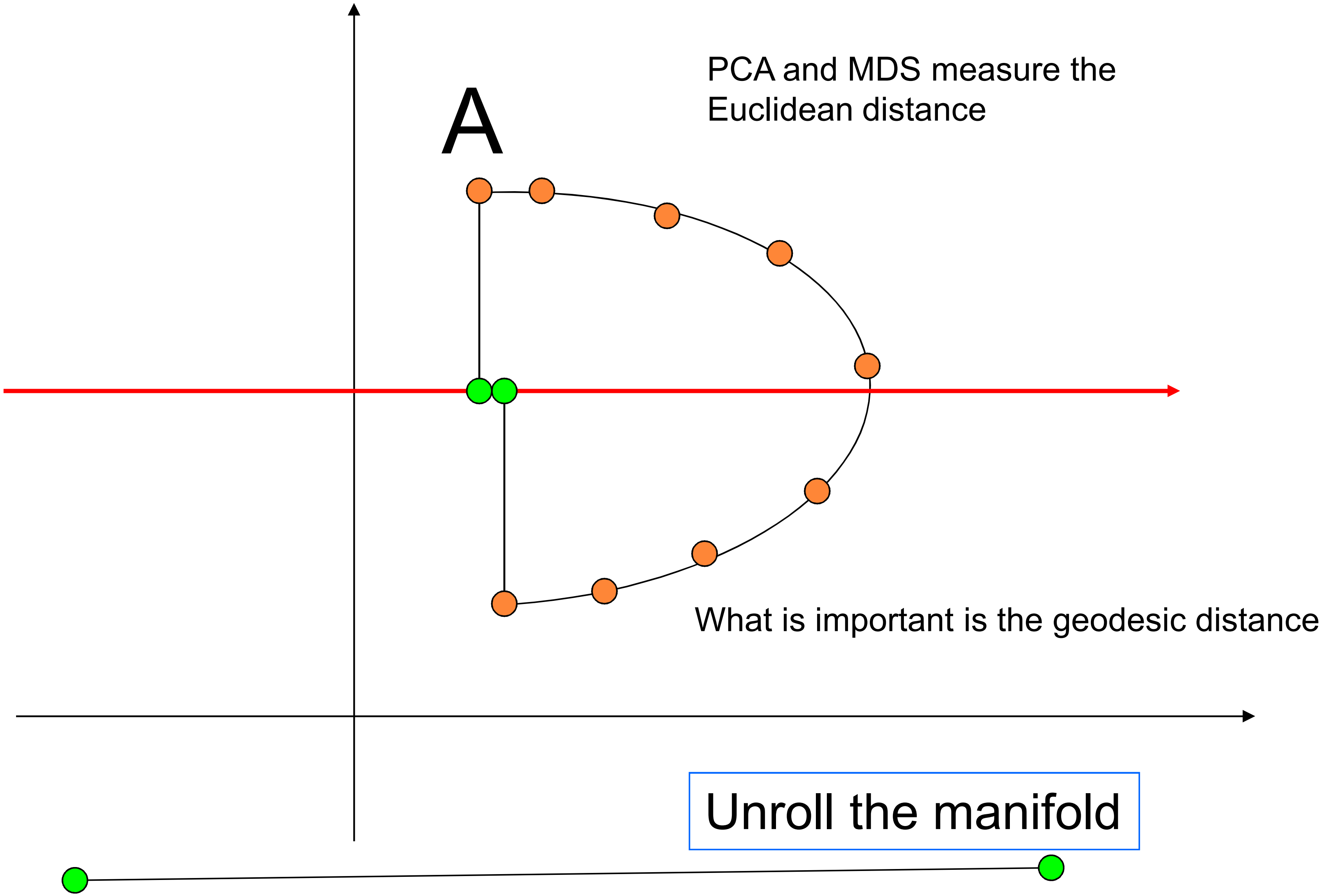
  $\boldsymbol{y_1},\boldsymbol{y_2},\ldots,\boldsymbol{y_n}$ points in R^d (d<<D)

- Goals

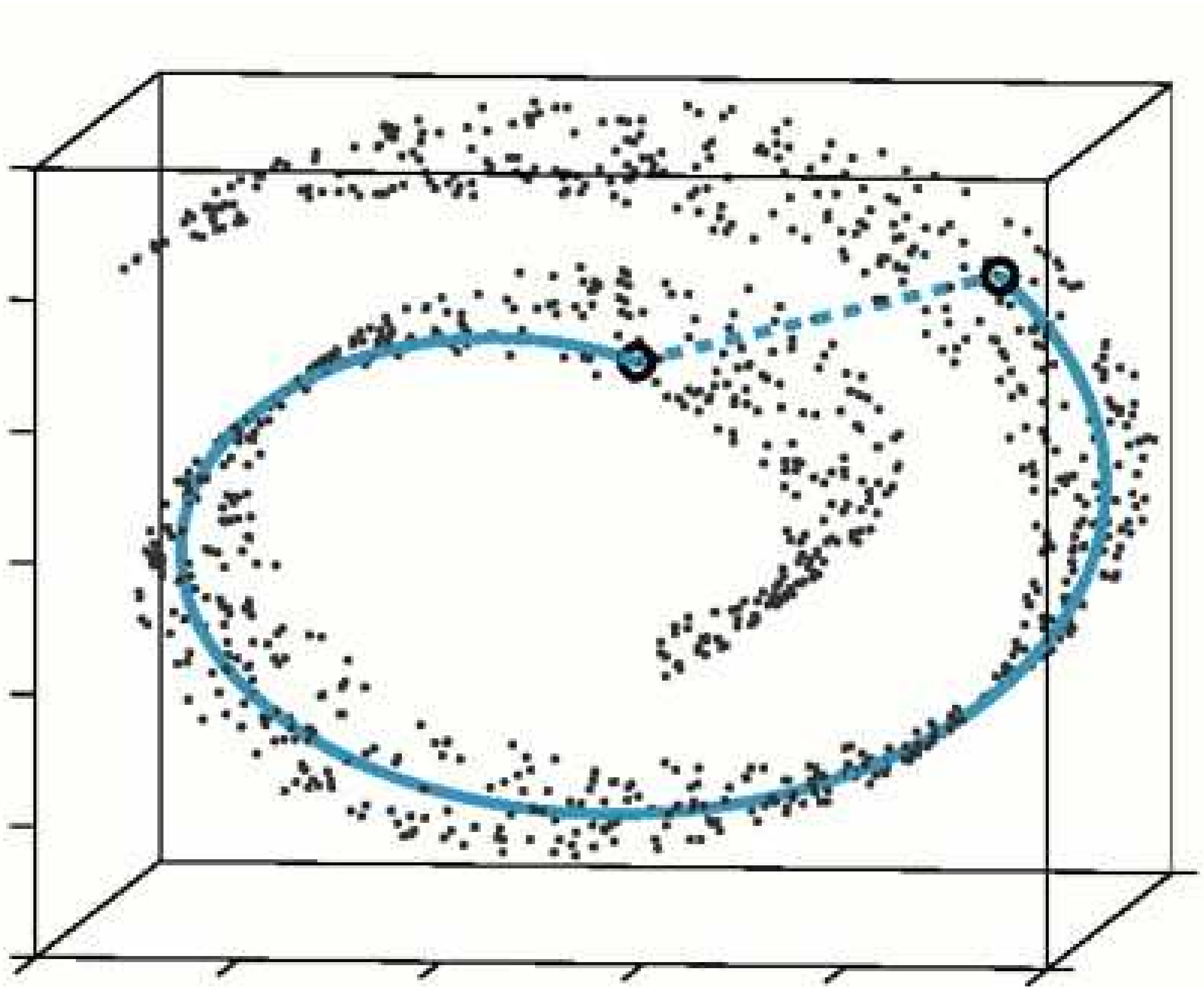
  Nearby points remain nearby.

  Distant points remain distant.

Nonlinear Manifolds



To preserve structure preserve the geodesic distance and not the euclidean distance.



Graph-Based Methods

- Tenenbaum et.al's Isomap Algorithm
  - Global approach.

    Preserves global pairwise distances.

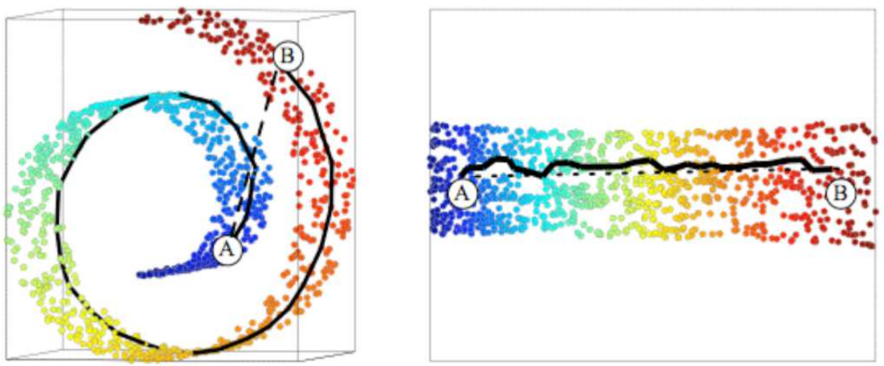
- Roweis and Saul's Locally Linear Embedding Algorithm
  - Local approach

    Nearby points should map nearby

- Belkin and Niyogi Laplacian Eigenmaps Algorithm
  - Local approach
  - minimizes approximately the same value as LLE

Isomap - Key Idea:

- Use **geodesic** instead of Euclidean distances in MDS.
  - For neighboring points Euclidean distance is a good approximation to the geodesic distance.
  - For distant points estimate the distance by a series of short hops between neighboring points. Find shortest paths in a graph with edges connecting neighboring data points.



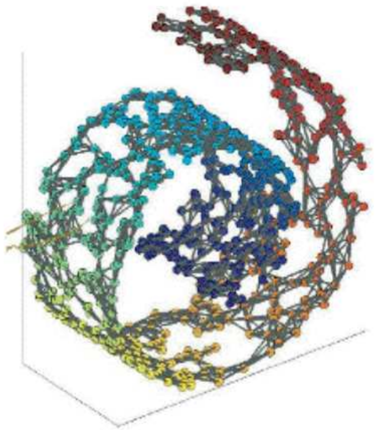
Step 1. Build adjacency graph.

- Adjacency graph

  Vertices represent inputs. Undirected edges connect neighbours.

- Neighbourhood selection

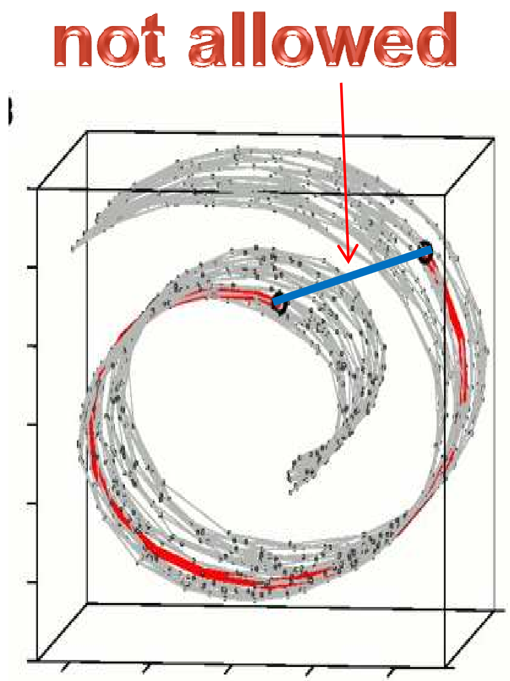
  Many options: k-nearest neighbours, inputs within radius r, prior knowledge.



Graph is discretized approximation of submanifold.

Building the graph

- Computation
  - kNN scales naively as $O(n^2 D)$
  - Faster methods exploit data structures.
- Assumptions
  1. Graph is connected.
  2. Neighbourhoods on graph reflect neighbourhoods on manifold.



Step 2. Estimate geodesics

- Dynamic programming
  - Weight edges by local distances.
  - Compute shortest paths through graph.
- Geodesic distances
  - Estimate by lengths of shortest paths: denser sampling = better estimates.
- Computation
  - Djikstra's algorithm for shortest paths $O(n^2 \log n + n^2 k)$.

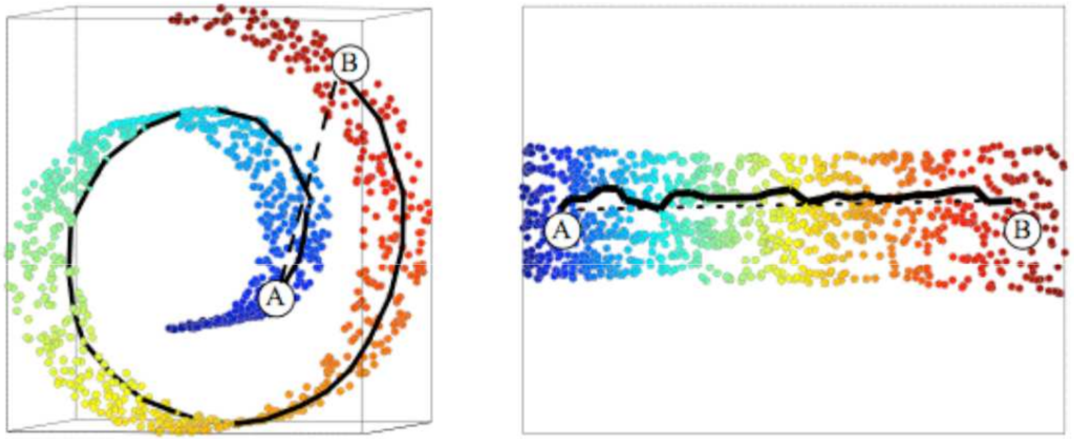
Step 3. Metric MDS

- Embedding
  - Top d eigenvectors of Gram matrix yield embedding.
- Dimensionality
  - Number of significant eigenvalues yield estimate of dimensionality.
- Computation
  - Top d eigenvectors can be computed in $O(n^2d)$.

Summary

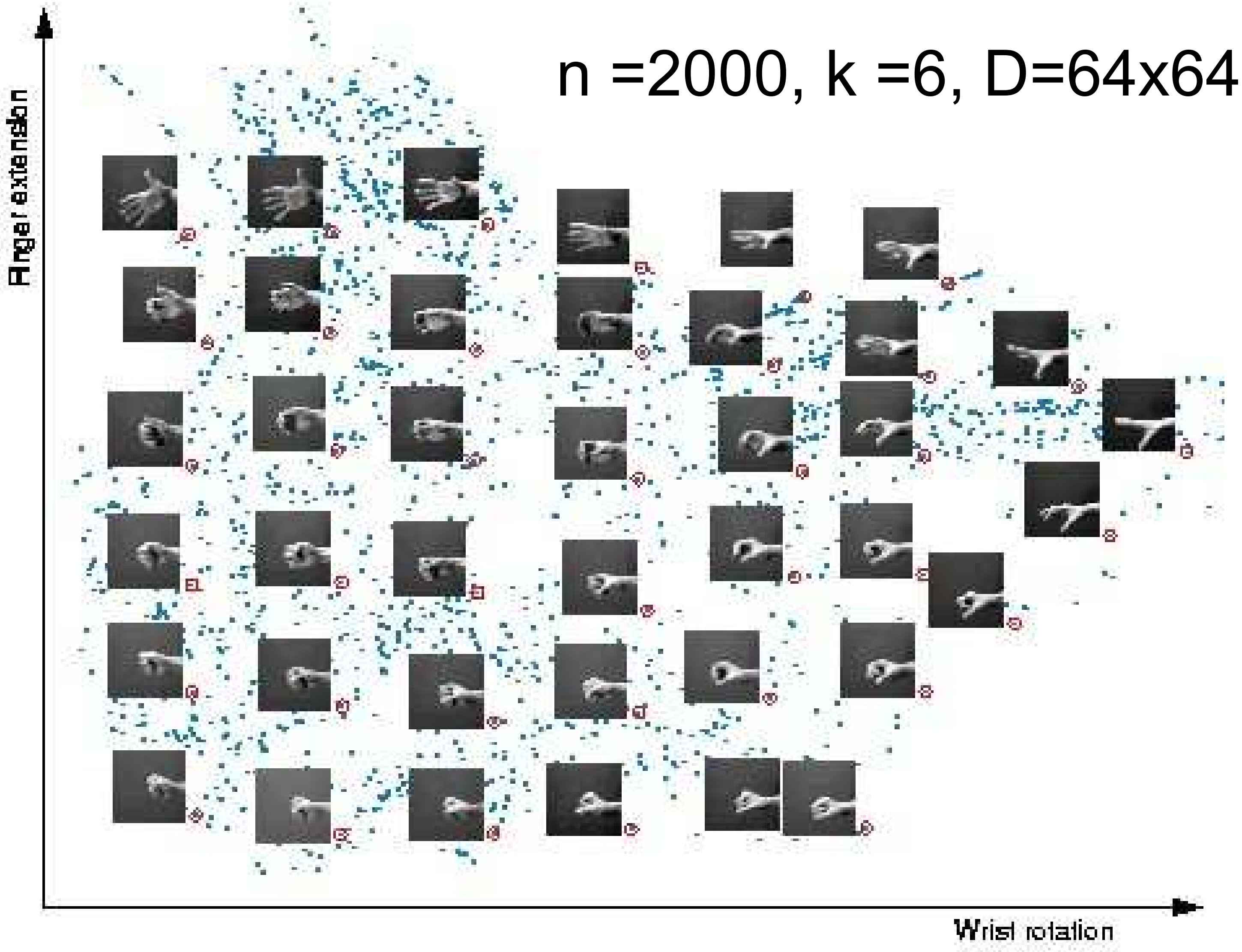
- Algorithm
  1. k nearest neighbours
  2. shortest paths through graph
  3. MDS on geodesic distances

Swiss Roll

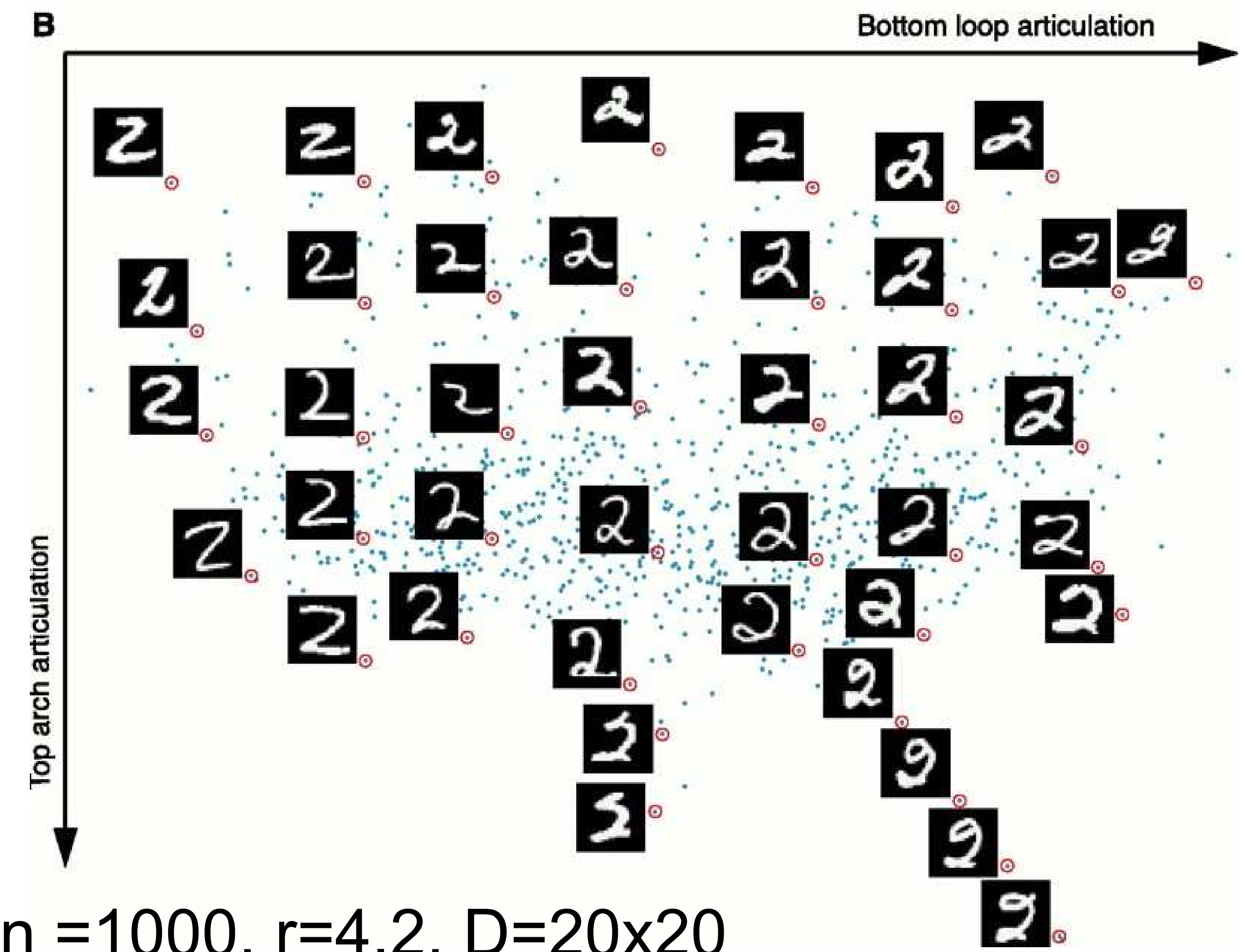


n (points) =1024
k (neighbors) =12

Isomap: Two-dimensional embedding of hand images (from Josh. Tenenbaum, Vin de Silva, John Langford 2000)



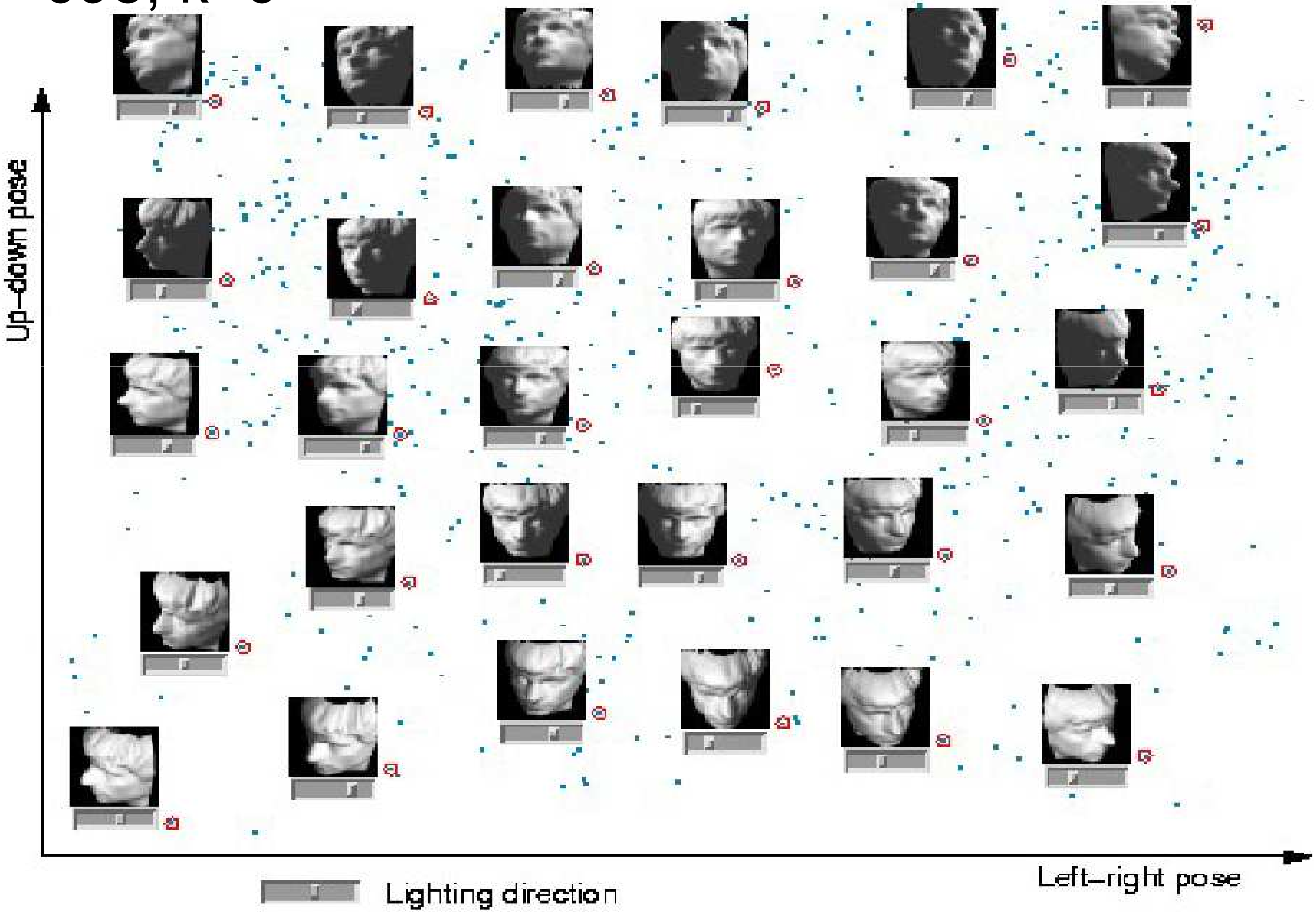
Isomap: two-dimensional embedding of hand-written '2' (from Josh. Tenenbaum, Vin de Silva, John Langford 2000)



n =1000, r=4.2, D=20x20

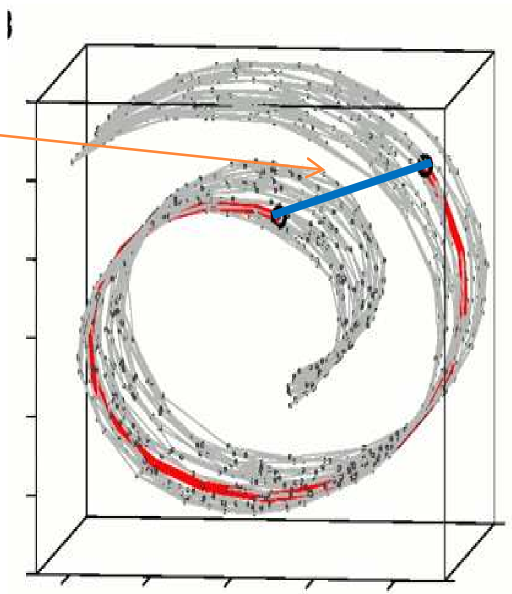
Isomap: three-dimensional embedding of faces (from Josh. Tenenbaum, Vin de Silva, John Langford 2000)

n =698, k=6



Properties of Isomap

- Strengths :
  - Preserves the global data structure
  - Performs global optimization
  - Non-parametric (Only heuristic is neighbourhood size)
- Weaknesses :
  - Sensitive to “shortcuts”
  - Very slow



Spectral Methods

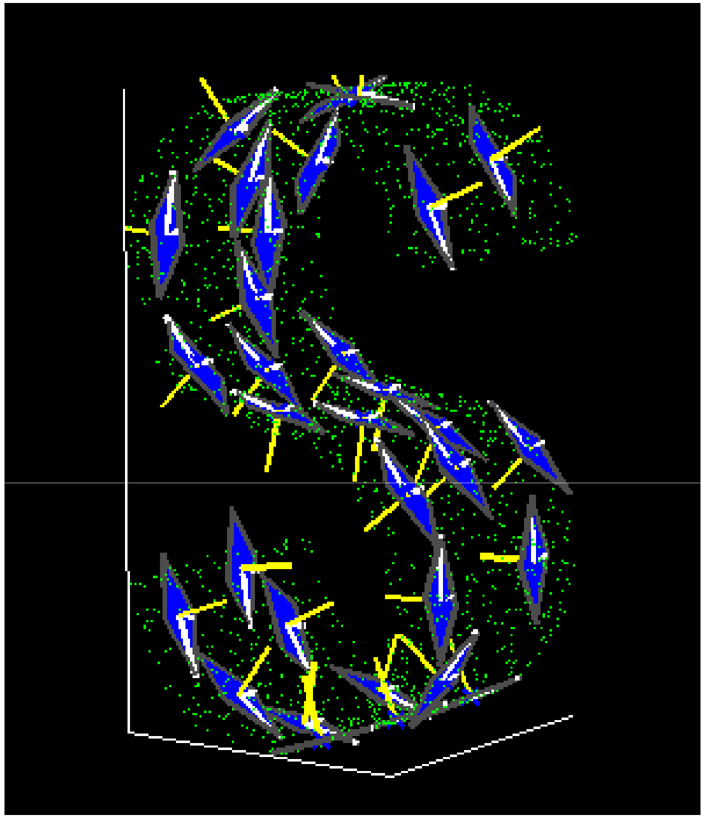
- Common framework
  1. Derive sparse graph from *kNN.*
  2. Derive matrix from graph weights.
  3. Derive embedding from eigenvectors.

- Varied solutions

  Algorithms differ in step 2. Types of optimization: shortest paths, least squares fits, semidefinite programming.

Locally Linear Embedding (LLE)

- Assume that data lies on a manifold: each sample and its neighbors lie on approximately linear subspace
- Idea:
  1. Approximate data by a set of linear patches
  2. Glue these patches together on a low dimensional subspace s.t. neighborhood relationships between patches are preserved.

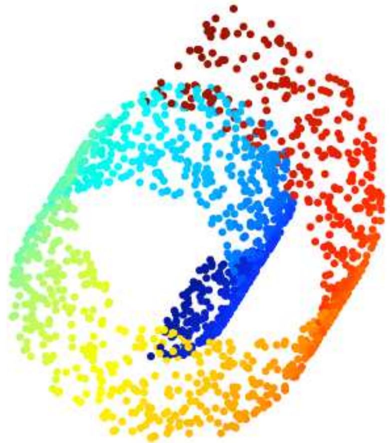


Algorithm: http://cs.nyu.edu/~roweis/lle/algorithm.html

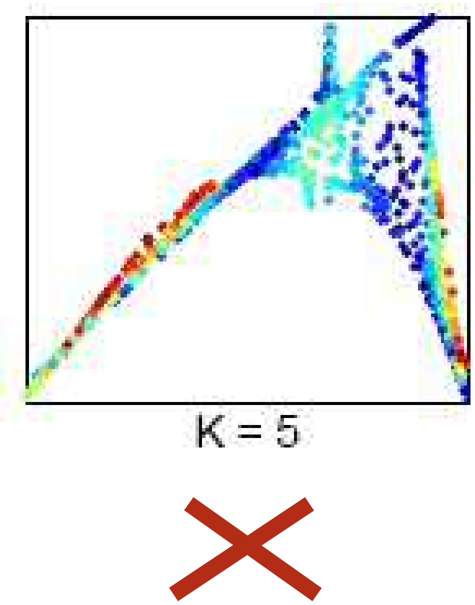
LLE at glance

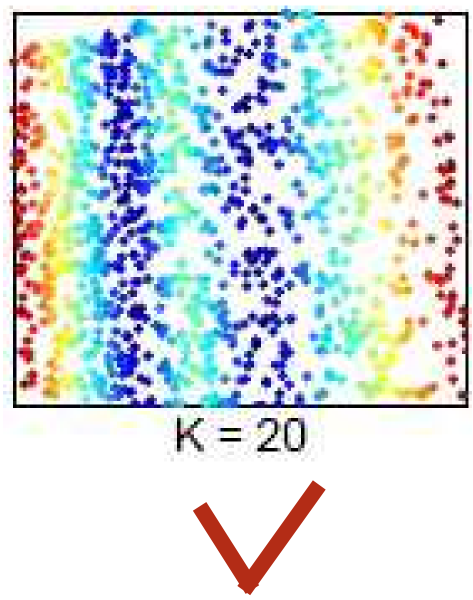
- Steps
  1. Nearest neighbour search.
  2. Least squares fits.
  3. Sparse eigenvalue problem.
- Properties
  - Obtains highly nonlinear embeddings.
  - Not prone to local minima.
  - Sparse graphs yield sparse problems.

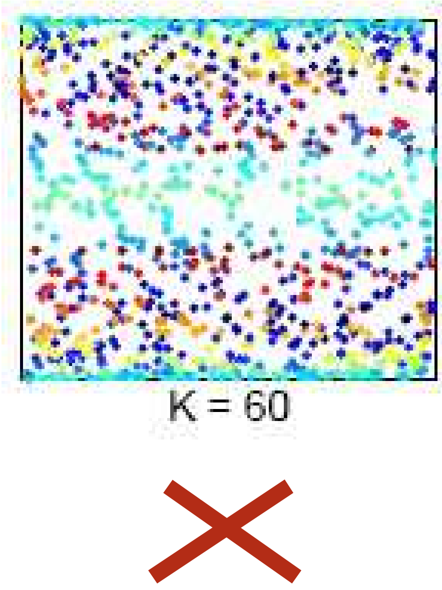
Step 1. Nearest neighbours search



Effect of Neighbourhood Size

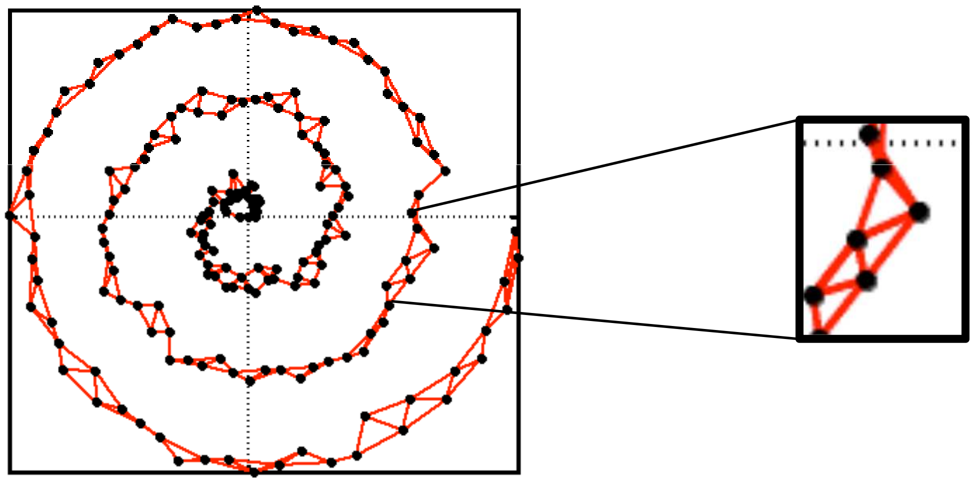






Step 2. Compute weights

- Characterize local geometry of each neighbourhood by weights Wij.



- Compute weights by reconstructing each input (linearly) from neighbours.

Linear reconstructions

- Local linearity
  - Assume neighbours lie on locally linear patches of a low dimensional manifold.
- Minimize reconstruction error
  - Each point can be written as a linear combination of its neighbors.
  - The weights chosen to minimize the reconstruction error:

$$\min_{W} \sum_{i} \left| x_i - \sum_{j} W_{ij} x_j \right|^2$$

Least squares fits (Computing W_{ij})

- Local reconstructions
  - Choose weights to minimize: $\Phi(W) = \sum_i \left| x_i - \sum_j W_{ij} x_j \right|^2$
- Constraints
  - Set $W_{ij} = 0$ if x_j is not a neighbor of x_i
  - Weights must sum to one: $\sum_j W_{ij} = 1$

invariance to translation

- Local invariance
  - Optimal weights W_{ij} are invariant to rotation, translation, and scaling.

Step 3. Finding the Embedding

- Low dimensional representation

  Map inputs to outputs: $x_i \in R^D \to y_i \in R^d$

- Minimize reconstruction errors

  Optimize outputs for fixed weights:

$$\Psi(y) = \sum_i \left| y_i - \sum_j W_{ij} y_j \right|^2$$

- Constraints:
  - Center outputs on origin $\sum_i y_i = 0$
  - Impose unit covariance matrix $\frac{1}{N} \sum_i y_i y_i = I_d$

Minimization

- Quadratic form:

$$\Psi(y) = \sum_{ij} M_{ij}\left(y_i \cdot y_j\right)$$

$$M_{ij} = \delta_{ij} - W_{ij} - W_{ji} + \sum_k W_{ki} W_{kj},$$

$$\delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

It can be shown that

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

Sparse eigenvalue problem

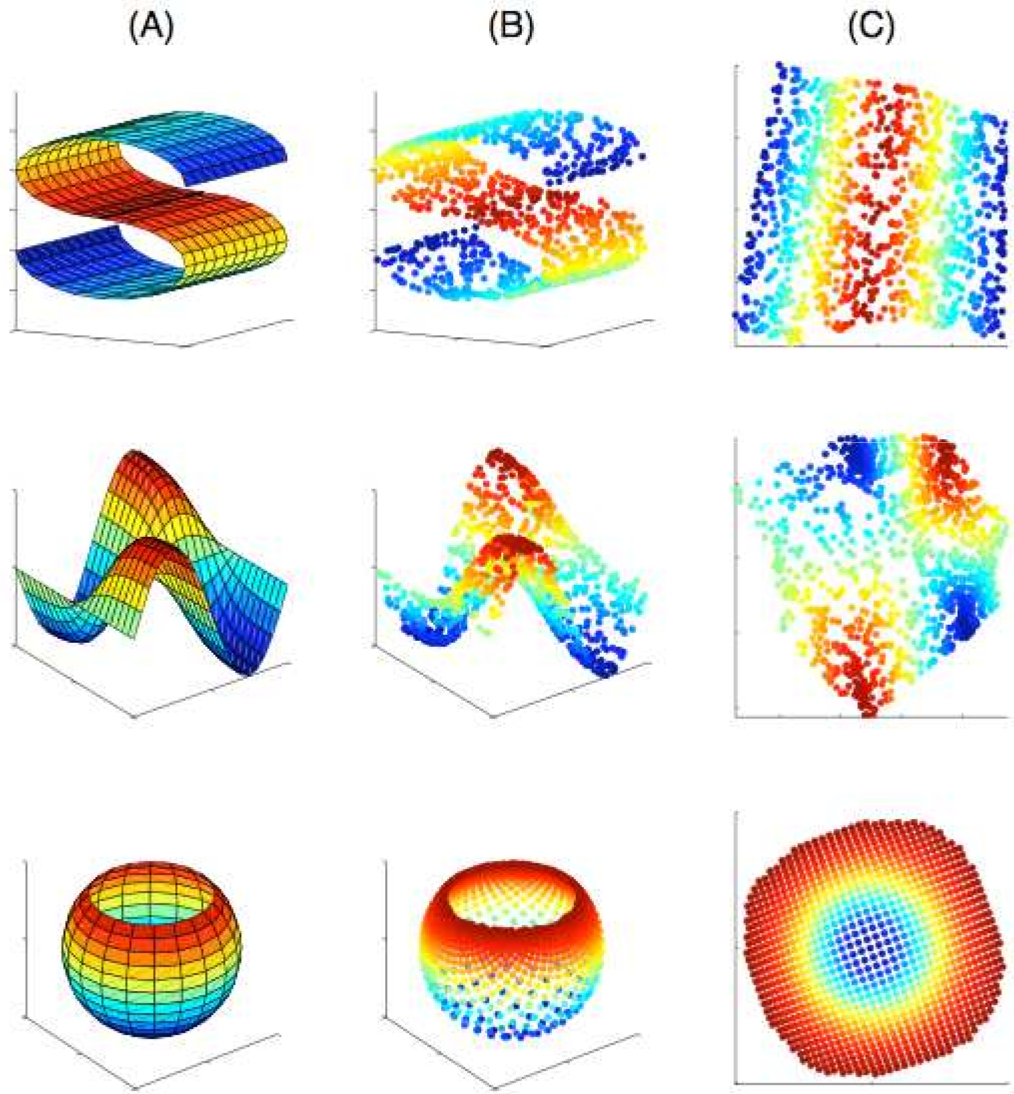
- Optimal embedding

  given by bottom d+1 eigenvectors, corresponding to the d+1 smallest eigenvalues (Rayleigh-Ritz theorem).

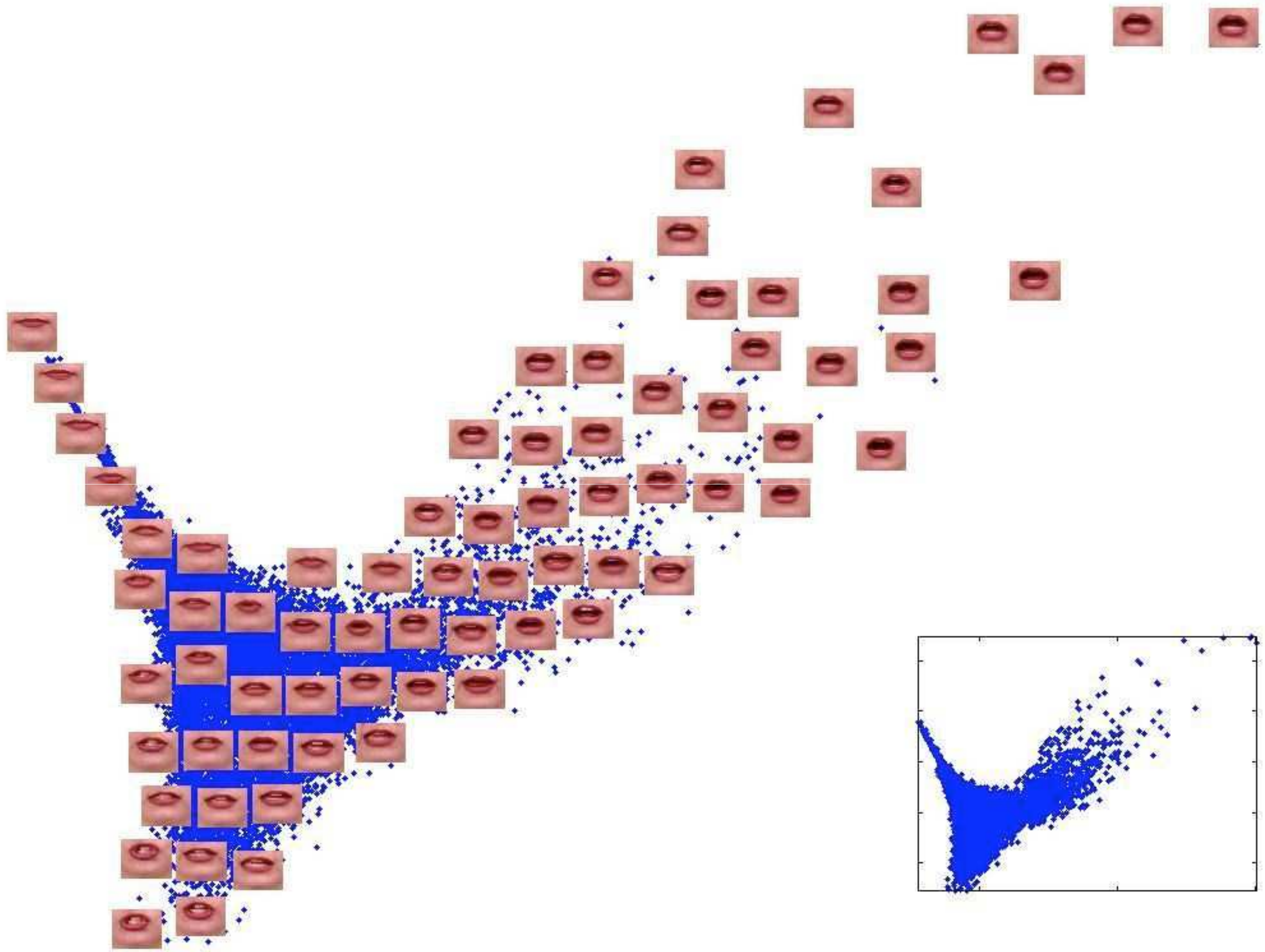
- Solution
  - Discard bottom eigenvector [1 1 … 1] (with eigenvalue zero).
  - Other eigenvectors satisfy constraints.

Surfaces

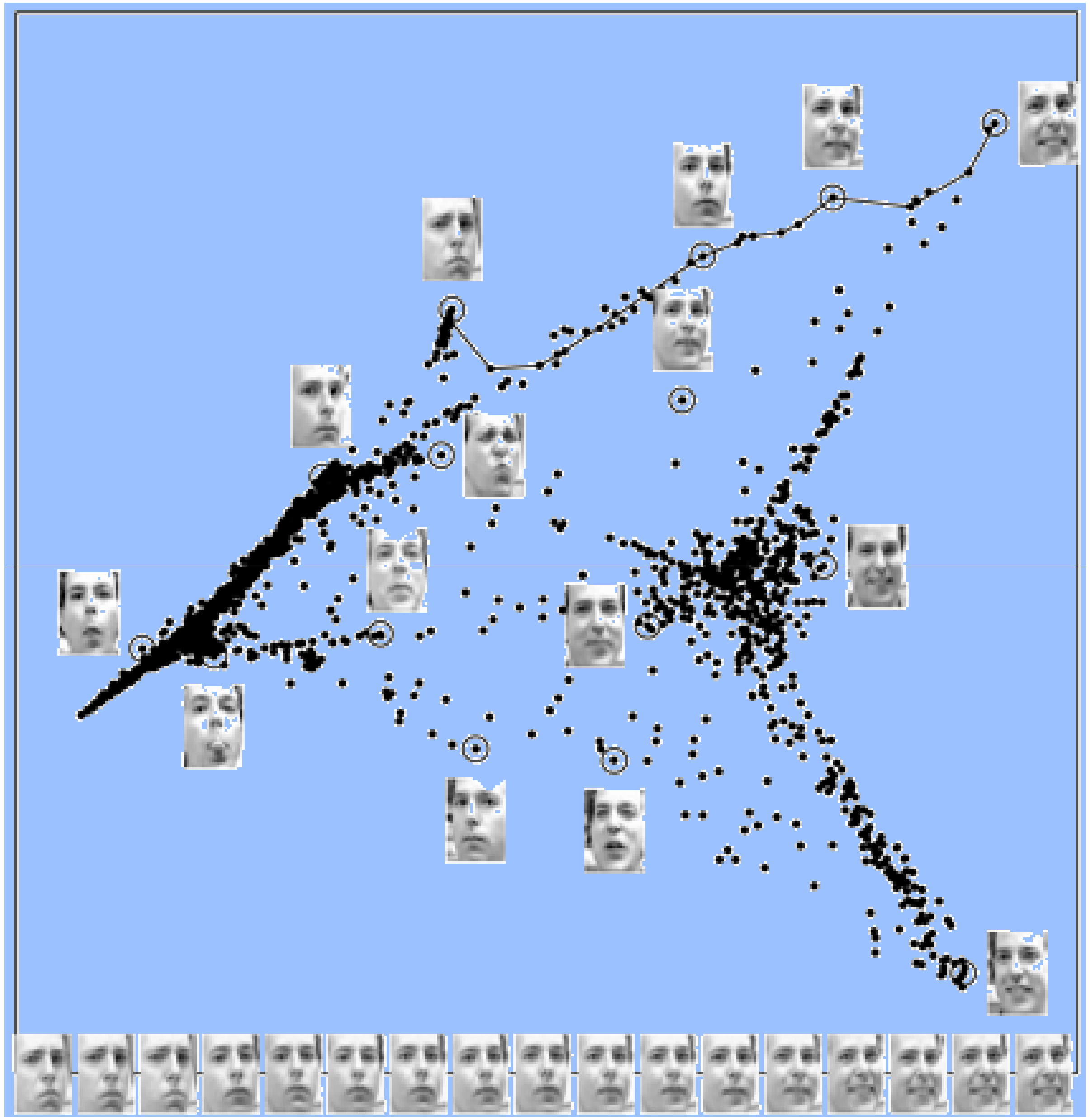
N=1000
inputs
k=8
nearest
neighbors



Lips
N=15960
images
K=24
neighbors
D=65664
pixels
d=2
(shown)



Pose and expression
N=1965 images
k=12 nearest neighbors
D=560 pixels
d=2 (shown)



Properties of LLE

- Strengths:
  - Fast
  - No local minima
  - Non-iterative
  - Non-parametric (only heuristic is neighbourhood size).
- Weaknesses:
  - Sensitive to “shortcuts”
  - No estimate of dimensionality

LLE versus Isomap

- Many similarities
  - Graph-based, spectral method
  - No local minima
- Essential differences
  - Does not estimate dimensionality ☹
  - No theoretical guarantees ☹
  - Constructs sparse vs. dense matrix ☺
  - Preserves weights vs. distances
  - Much faster ☺

Laplacian Eigenmaps

- Map nearby inputs to nearby outputs, where nearness is encoded by graph.

- Summary of the Algorithm
  1. Identify k-nearest neighbours (as in LLE)
  2. Assign weights to neighbours
  3. Sparse eigenvalue problem

Step 2. Construct the graph

- Vertices represent inputs.
- Undirected edges connect neighbours.
- Assign weights to neighbours:
  - Simple: $W_{ij} = 1$

    or
  - Heat kernel $W_{ij} = \exp\left(-\beta \left\| x_i - x_j \right\|^2\right)$

Step 3. Graph Laplacian

- Compute outputs by minimizing:

$$\Psi(y) = \sum_{ij} W_{ij} \left\| y_i - y_j \right\|^2$$ under appropriate constraints

$$\Psi(y) = \sum_{ij} W_{ij} \left(y_i^2 + y_j^2 - 2 y_i y_j \right)$$ W_{ij} is symmetric

$$= \sum_i y_i^2 D_{ii} + \sum_j y_j^2 D_{jj} - 2 \sum_{ij} y_i y_j W_{ij} = 2 y^t L y$$

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

Graph Laplacian $L = D - W$

Step 3. Generalized eigenvalue problem

- Minimize $y^t L y$
  constrained by $y^t D y = 1$

- Optimal embedding: $(Le = \lambda De)$
  given by bottom d+1 eigenvectors (corresponding to the d+1 smallest eigenvalues).

- Solution:
  Discard bottom eigenvector [1 1 … 1] (with eigenvalue zero). Other eigenvectors satisfy constraints.

Analysis on Manifolds

- Consider Riemannian manifold $\Omega \in \Re^D$
  - a real differentiable manifold in which tangent space is equipped with dot product.
- Laplace Beltrami operator
  - Ω has a 'natural' operator Δ on differentiable functions.
  - Δ is a second order differential operator defined as a "divergence of the gradient"

$$\Delta = \sum_i \frac{\partial^2}{\partial x_i^2}$$

Spectral desomposition of Δ

- Assume $\mathscr{L}^2(\Omega)$ is space of all square integrable functions on Ω
- Δ is a self-adjoint positive semi-definate operator and its eigenfunctions form the basis.
- Thus all f in $\mathscr{L}^2(\Omega)$ can be written as

$$f(x) = \sum_i \alpha_i e_i(x)$$

(provided Ω is compact)

Smoothness functional

- Defined as

$$S(f) = \int_{\Omega} |\nabla f|^2 d\omega = \int f \Delta f \ d\omega = \langle \Delta f, f \rangle_{L^2(\Omega)}$$

  - value close to zero implies *f* being smooth.
- Since

$$S(e_i) = \langle \Delta e_i, e_i \rangle = \lambda_i$$

we have

$$S(f) = \langle \Delta f, f \rangle = \left\langle \sum_i \alpha_i \Delta e_i, \sum_i \alpha_i e_i \right\rangle = \sum_i \lambda_i \alpha_i$$

choosing the lowest p eigenfunctions provides a maximally smooth approximation to the manifold.

Spectral graph theory

- Weighted graph is discretized representation of manifold.
- Laplacian measures smoothness of functions over manifold and graph.

Manifold: $$\int_{\Omega} |\nabla f|^2 d\omega = \int f \Delta f \; d\omega$$

Graph: $$\sum_{ij} W_{ij} \left(f_i - f_j\right)^2 = f^t L f$$

Interpreting Laplacian Eigenmaps

- Eigenvectors

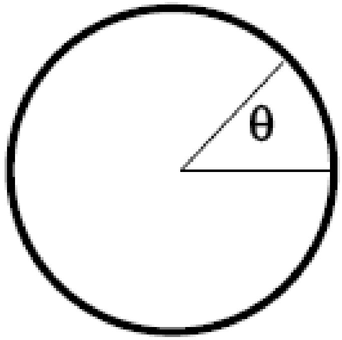
  functions from nodes to $\mathbb{R}$ in a way that "close by" points are assigned "close by" values.

- Eigenvalues

  measure how close are the values of neighbouring points – smoothness.

Example: S1 (the circle)

- Continuous
  - Eigenfunctions of Laplacian are basis for periodic functions on circle, ordered by smoothness.
  - Eigenvalues measure smoothness.

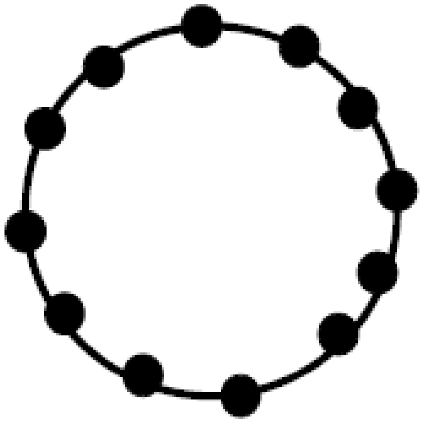


$$-\frac{\partial^2 f_m}{\partial^2 \theta} = \lambda_m f_m(\theta)$$

$$f_m(\theta) = \begin{Bmatrix} \sin(m\theta) \\ \cos(m\theta) \end{Bmatrix} \text{ with } \lambda_{\mathrm{m}} = m^2$$

Example: S1 (the circle)

- Discrete (n equally spaced points)
  - Eigenvectors of graph Laplacian are discrete sines and cosines.
  - Eigenvalues measure smoothness.



Graph embedding from Laplacian eigenmaps:

$$\boxed{\vec{y}_k = \left(\cos\left(2\pi k/n\right), \sin\left(2\pi k/n\right)\right)}$$

Laplacian vs LLE

- More similar than different
  - Graph-based, spectral method
  - Sparse eigenvalue problem
  - Similar results in practice
- Essential differences
  - Preserves locality vs local linearity
  - Uses graph Laplacian