Lecture 7 Nonlinear dimensionality reduction: cMDS, ISOMAP, LLE, Laplacian Eigenmaps

Foundations of Data Science: Algorithms and Mathematical Foundations

Mihai Cucuringu
mihai.cucuringu@stats.ox.ac.uk

CDT in Mathematics of Random System
University of Oxford

21 September, 2023

Multidimensional Scaling (MDS)

Isomap

Locally Linear Embedding (LLE)

Laplacian Eigenmaps

Multidimensional Scaling (MDS)

Isomap

Locally Linear Embedding (LLE)

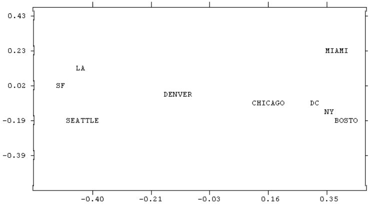
Laplacian Eigenmaps

Multidimensional Scaling (MDS)

- a means of visualizing the level of similarity of individual objects of a data set using the information contained in the **distance** matrix
- It aims to place each object in *p*-dimensional space such that the between-object distances are preserved as best as possible.

		1 BOST	2 NY	3 DC	4 MIAM	5 CHIC	6 SEAT	7 SF	8 LA	9 DENV
1	BOSTON	0	206	429	1504	963	2976	3095	2979	1949
2	NY	206	0	233	1308	802	2815	2934	2786	1771
3	DC	429	233	0	1075	671	2684	2799	2631	1616
4	MIAMI	1504	1308	1075	0	1329	3273	3053	2687	2037
5	CHICAGO	963	802	671	1329	0	2013	2142	2054	996
6	SEATTLE	2976	2815	2684	3273	2013	0	808	1131	1307
7	SF	3095	2934	2799	3053	2142	808	0	379	1235
8	LA	2979	2786	2631	2687	2054	1131	379	0	1059
9	DENVER	1949	1771	1616	2037	996	1307	1235	1059	0

(a) Input: Distance Matrix



(b) Output: 2-Dim. Embedding

Figure: Example of a 2-dimensional embedding produced by MDS given the matrix of distances among cities

Multidimensional Scaling (MDS)

- Suppose the data to be analyzed is a collection of n objects $x_i \in R^p, i = 1, \ldots, n$
- distance matrix D (size $n \times n$) containing **all** pairwise distances between the i^{th} and j^{th} object

$$D = \begin{bmatrix} D_{11} & D_{12} & D_{13} & \ldots & D_{1n} \\ D_{21} & D_{22} & D_{23} & \ldots & D_{2n} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ D_{n1} & D_{n2} & D_{n3} & \ldots & D_{nn} \end{bmatrix}, \tag{1}$$

$$D_{ij} = \| x_i - x_j \|^2, D_{ii} = 0. \tag{2}$$

- **Goal**: transform D into a cross-product matrix B, with $B_{ij} = x_j^\top x_i$ and find its eigen-decomposition
- yields an embedding of the n points into $\mathbb{R}^p$ that preserves pairwise (squared) distances
- often used for visualization if $p = \{2, 3\}$.

 Let us denote by s_i the sum of entries in row i of D

$$\begin{aligned} s_i = \sum_{j=1}^{n} D_{ij} &= \sum_{j=1}^{n} \| x_i - x_j \|^2 \\ &= \sum_{j=1}^{n} (\| x_i \|^2 + \| x_j \|^2 - 2x_i^\top x_j) \\ &= n \| x_i \|^2 + \sum_{j=1}^{n} \| x_j \|^2 - 2x_i^\top \sum_{j=1}^{n} x_j \qquad (3) \end{aligned}$$

- WLOG, assume points centered at the origin $\sum_{i=1}^{n} x_i = 0$

$$s_i = n \| x_i \|^2 + \sum_{j=1}^{n} \| x_j \|^2$$

$$s = \sum_{i=1}^{n} s_i = \sum_{i=1}^{n} \left(n \| x_i \|^2 + \sum_{j=1}^{n} \| x_j \|^2 \right) = n \sum_{i=1}^{n} \| x_i \|^2 + n \sum_{j=1}^{n} \| x_j \|^2$$

$$\implies s = 2n \sum_{i=1}^{n} \| x_i \|^2$$

Claim

$$D_{ij} - \frac{1}{n}s_i - \frac{1}{n}s_j + \frac{1}{n^2}s = -2x_i^{\mathsf{T}}x_j \tag{5}$$

Proof:

$$D_{ij} - \tfrac{1}{n}s_i - \tfrac{1}{n}s_j + \tfrac{1}{n^2}s$$

$$= \| x_i - x_j \|^2 - \frac{1}{n}\left(n \| x_i \|^2 + \sum_{j=1}^{n} \| x_j \|^2\right) - \frac{1}{n}\left(n \| x_j \|^2 + \sum_{i=1}^{n} \| x_i \|^2\right) + \frac{1}{n^2}2n\sum_{i=1}^{n} \| x_i \|^2 \tag{6}$$

$$= \| x_i - x_j \|^2 - \| x_i \|^2 - \| x_j \|^2 - \frac{2}{n}\sum_{i=1}^{n} \| x_i \|^2 + \frac{2}{n}\sum_{i=1}^{n} \| x_i \|^2 \tag{7}$$

$$= \| x_i \|^2 + \| x_j \|^2 - 2x_i^{\mathsf{T}}x_j - \| x_i \|^2 - \| x_j \|^2 \tag{8}$$

$$= -2x_i^{\mathsf{T}}x_j$$

The Gram matrix

- Consider the matrix
$$B = X^T X$$
- $X : p \times n$ of rank p (assuming $p < n$)
- $\text{rank}(B) = p, \quad p < n$

Spectral decomposition of B

$$B = U \Sigma U^T, \tag{9}$$

$$X = \Sigma^{\frac{1}{2}} U^T, \tag{10}$$

$$X^\intercal X = (U \Sigma^{\frac{1}{2}})(\Sigma^{\frac{1}{2}} U^\intercal) = U \Sigma U^T = B$$

Remark: When considering the spectrum of B, the largest eigenvalues correspond to the true intrinsic dimension of the data, while the remaining ones capture the noise.

Claim

$$B = -\frac{1}{2}HDH \tag{11}$$

where H is the scaling matrix

$$H = I - \frac{1}{n}ee^{\mathsf{T}}$$

with $e = [1, \ldots, 1]^T$.

- Denote

$$S_{n\times 1} = [s_1, \ldots, s_n]^T$$

where (recall) $s_i = \sum_{j=1}^{n} D_{ij}$. Note the following hold true:

$$De = S \qquad \text{and} \qquad e^{\mathsf{T}}D = S^T \tag{12}$$

$$\begin{aligned} B &= -\frac{1}{2}HDH & (13)\\ &= -\frac{1}{2}(I - \frac{1}{n}ee^T)D(I - \frac{1}{n}ee^T) & (14)\\ &= -\frac{1}{2}(I - \frac{1}{n}ee^T)(D - \frac{1}{n}Dee^T) & (15)\\ &= -\frac{1}{2}(D - \frac{1}{n}Se^T - \frac{1}{n}ee^TD + \frac{1}{n^2}ee^TSe^T) & (16) \end{aligned}$$

We previously showed that $B_{ij} = -\frac{1}{2}(D_{ij} - \frac{1}{n}s_i - \frac{1}{n}s_j + \frac{1}{n^2}s) = x_i^T x_j$.

Final remarks on MDS

- classical MDS assumes Euclidean distances
- MDS can be generalized to incorporate additional nonnegative weights W_{ij} on each distance (useful when some distances are missing, or most distances are noisy, but some are known)
- The optimization involves minimizing an energy known in the literature as *stress*

$$\text{Stress}_D(x_1, \ldots, x_n) = \left(\sum_{1 \le i < j \le n} (d_{ij} - ||x_i - x_j||)^2 \right)^{1/2} \tag{17}$$

- one approach (De Leeuw) to minimize stress is to iteratively minimize a (simple convex) *majorizing* function of two variables
- for a generic function f, with input variable X, we say that $g(X, Y)$ majorizes $f(X)$ if $g(X, Y) \ge f(X)$ & $g(X, X) = f(X)$
- non-metric MDS (monotonic relationship btw. the item-item dissimilarities and the Euclidean distances btw. items)
- *ordinal embedding*: find an embedding of n points $\{\vec{x}_i\}_{i=1}^n$ in $\mathbb{R}^d$ s.t.

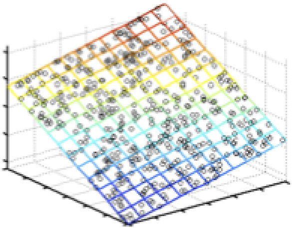
$$\forall (i, j, k, l) \in \mathcal{C}, \quad \|\vec{x}_i - \vec{x}_j\|_2 < \|\vec{x}_k - \vec{x}_l\|_2, \tag{18}$$

where $\mathcal{C}$ denotes the set of ordinal constraints.

Dimensionality Reduction

Data representation

- Inputs are real-valued vectors in a high-dimensional space
- Linear structure: data lives in a low-dimensional subspace
- Nonlinear structure: data lives on a low-dimensional submanifold



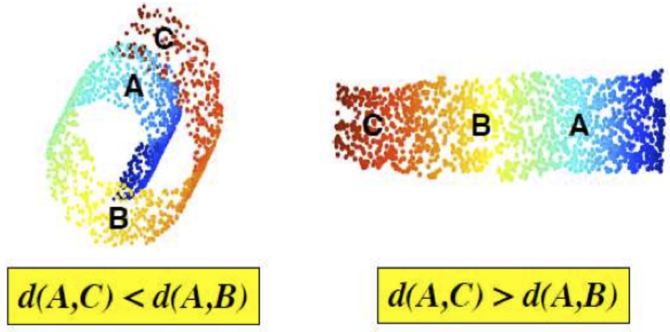
Dimensionality Reduction

- Inputs (high dimensional) $x_1, x_2, \ldots, x_n$ points in $\mathbb{R}^D$
- Outputs (low dimensional) $y_1, y_2, \ldots, y_n$ points in $\mathbb{R}^d (d << D)$

- Goals:
  - Nearby points remain nearby.
  - Distant points remain distant.

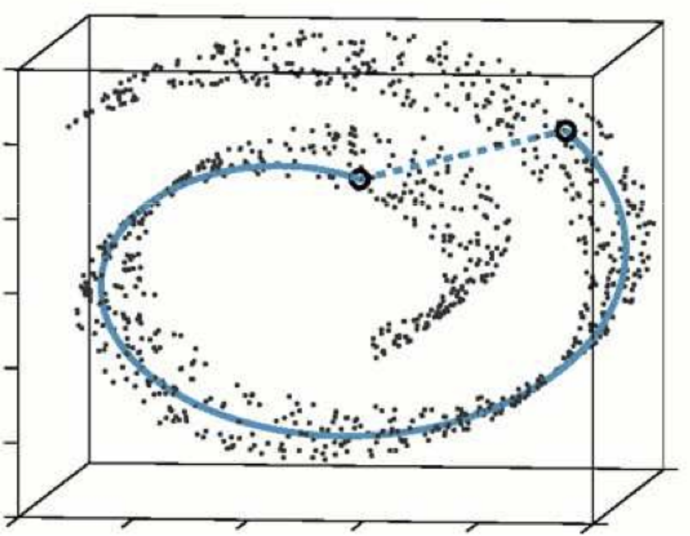
Non-metric MDS for manifolds?

- The (rank) ordering of Euclidean distances is NOT preserved in ”manifold learning”
- Euclidean distance can be misleading (*dumbbell* cloud of points)



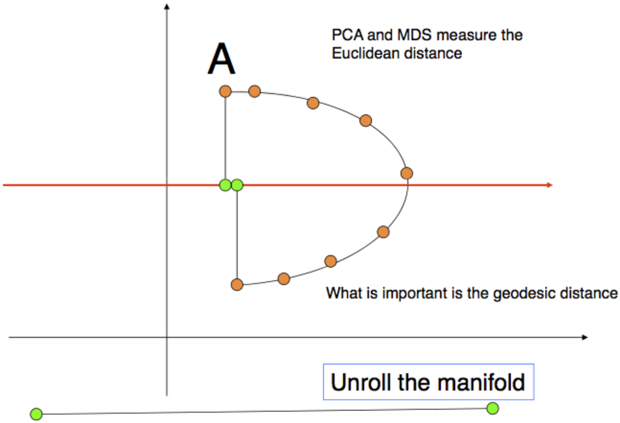
Preserving structure

- To preserve structure preserve the geodesic distance and not the Euclidean distance!

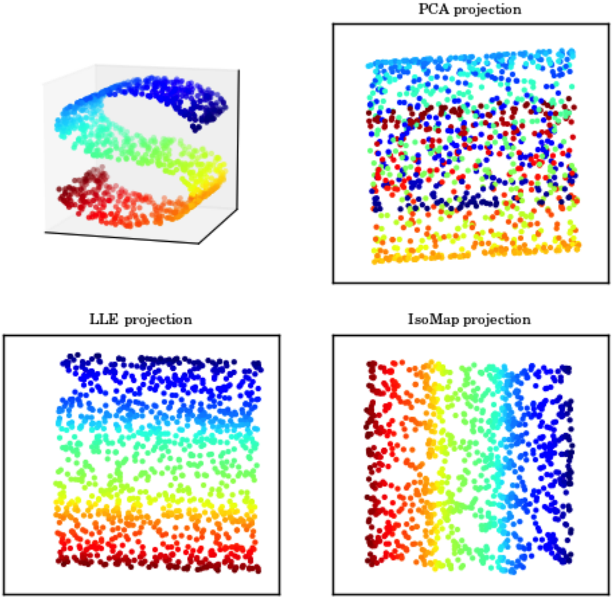


Nonlinear manifolds

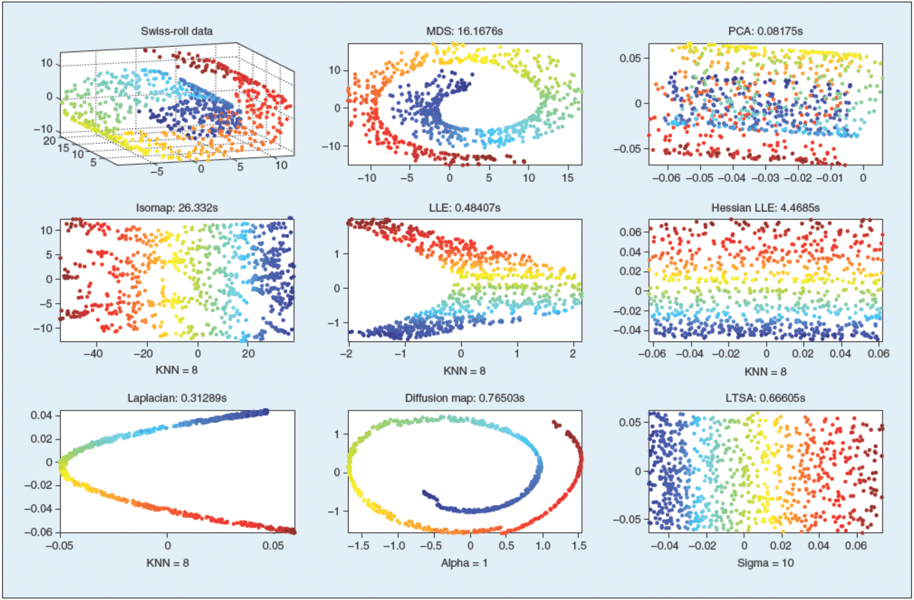
- PCA and MDS measure the Euclidean distance
- what matters most is the geodesic distance (shortest path distance)



Preserving structure



Lots of methods



Graph-Based Methods

- Isomap Algorithm
  - Global approach: Preserves global pairwise distances
  - Joshua B Tenenbaum, Vin de Silva, John C Langford, *A global geometric framework for nonlinear dimensionality reduction*
  - Science (2000)
  - 16,377 (2023) citations
- Locally Linear Embedding (LLE) Algorithm
  - Local approach: Nearby points should map nearby
  - Roweis, Sam T., and Lawrence K. Saul. *"Nonlinear dimensionality reduction by locally linear embedding"*, Science (2000)
  - 18,000 (2023) citations
- Laplacian Eigenmaps Algorithm
  - Local approach: minimizes approx. the same value as LLE
  - Belkin, Mikhail, and Partha Niyogi. *Laplacian eigenmaps for dimensionality reduction and data representation*, Neural computation (2003)
  - 9,300 (2023) citations

Multidimensional Scaling (MDS)

Isomap

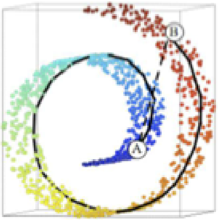
Locally Linear Embedding (LLE)

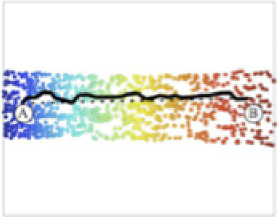
Laplacian Eigenmaps

Isomap - Key Idea

Use geodesic instead of Euclidean distances in MDS:

- For neighboring points: the 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





Assumptions

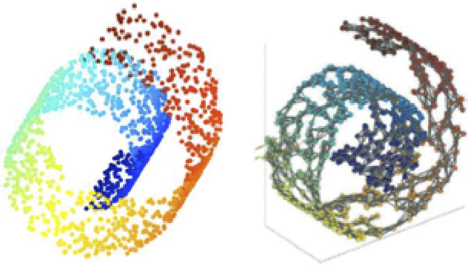
- Graph is connected.
- Neighborhoods on the graph reflect neighborhoods on the manifold (no *shortcuts* connect different portions of swiss roll.)

Isomap: Step 1 - Building adjacency graph

Neighbourhood selection - many options:

- k-nearest neighbours
- inputs within radius r
- prior knowledge.

Graph is discretized approximation of submanifold:



Computation (in $\mathbb{R}^d$)

- kNN scales naively as $O(n^2 d)$
- fast methods exploit data structures: $O(nd + kn)$, $O(ndk)$
- approximate nearest neighbor $O\left(\frac{1}{\epsilon^d} \log n\right)$

Isomap: 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 $O(n^2 \log n + n^2 m)$

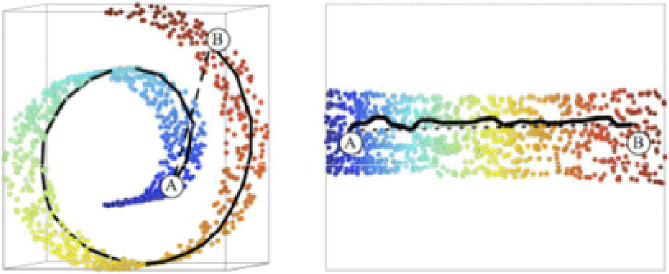
Isomap: 3 - Classical/Metric MDS

- Embedding
  - Top d eigenvectors of Gram matrix yield the desired embedding (recall the previous material on cMDS)
- Dimensionality
  - Number of significant eigenvalues yield estimate of dimensionality (look for a large spectral gap)
- Computation
  - Top d eigenvectors can be computed in $O(n^2 d)$

Summary of the ISOMAP Algorithm:

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

Swiss Roll

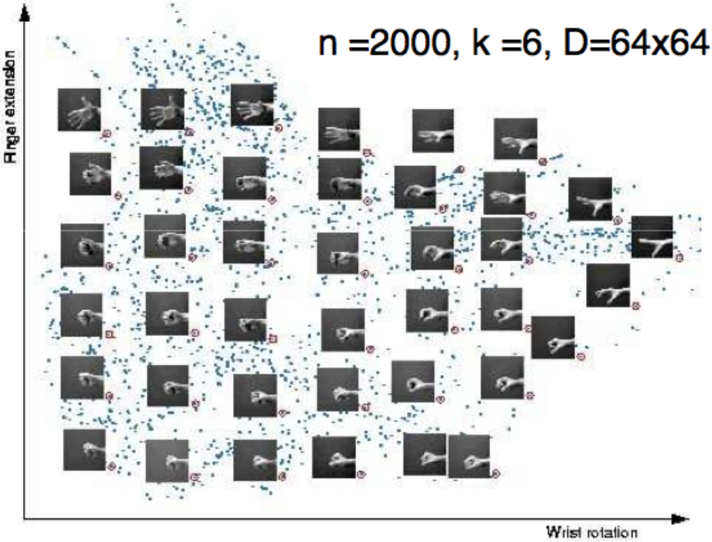


n (points) =1024

k (neighbors) =12

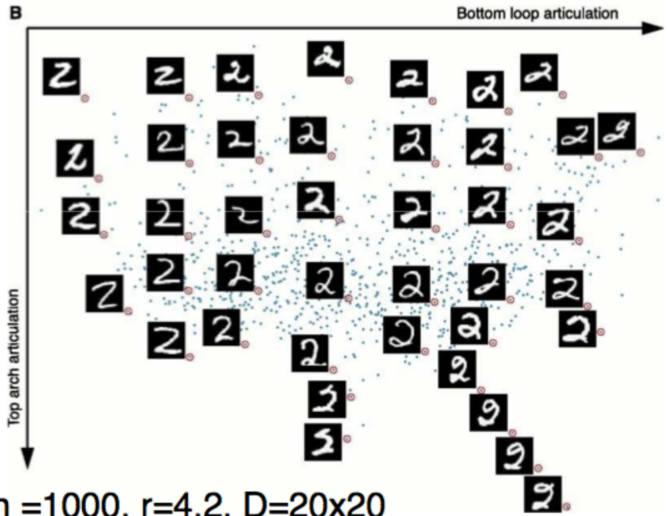
Hands

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



The digit 2

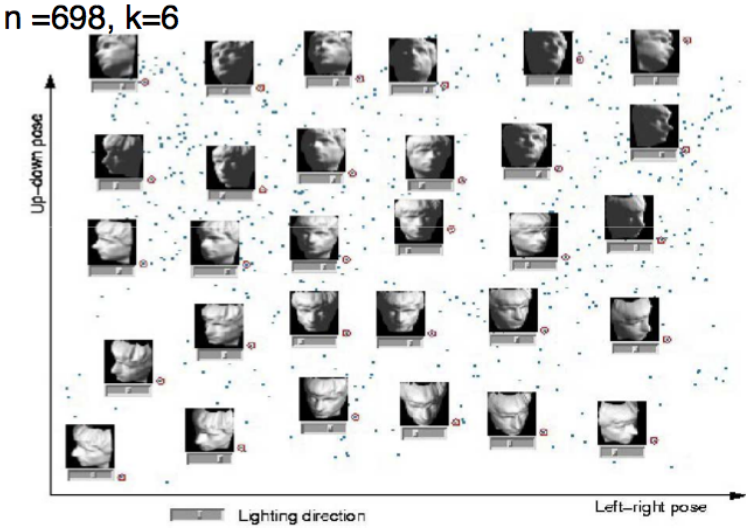
Isomap: two-dimensional embedding of hand-written ‘2’ (from Josh. Tenenbaum, Vin de Silva, John Langford 2000)



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

Faces

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



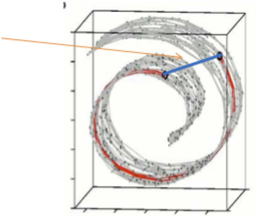
Properties of Isomap

Strengths

- preserves the global data structure
- performs global optimization
- non-parametric (the only parameter is the neighbourhood size)
- provable convergence guarantees
  - given that x_i is sampled sufficiently dense, ISOMAP will approximate closely the original distance as measured in manifold M
  - approx. geodesic distance in M by short Euclidean distance hops

Weaknesses

- very slow: need to compute pairwise shortest path between all sample pairs (i, j): Global + Non-sparse + Cubic complexity $O(n^3)$
- sensitive to "shortcuts"



ISOMAP- Theoretical considerations

-Convergence proof rests upon the idea that one can approximate the geodesic distance in M by short Euclidean distance hops.

-Consider the following quantities for a pair of points $x, y \in M$

- $d_M(x, y) = \inf_\gamma\{\text{length}(\gamma)\}$
  where γ varies over the set of smooth arcs connecting x to y in M
- $d_G(x, y) = \min_P(||x_0 - x_1|| + \ldots + ||x_{p-1} - x_p||)$
  where P varies over all paths along the edges of G starting from the source node $x = x_0$ and ending at $y = x_p$
- $d_S(x, y) = \min_P(d_M(x_0, x_1) + \ldots + d_M(x_{p-1}, x_p))$
- one can show that $d_M \approx d_S$ and $d_S \approx d_G$, leading to the desired result that $d_G \approx d_M$

Main result in [Bernstein, de Silva, Langford, and Tenenbaum 2000]: (under a long list of assumptions), the following is valid for all $x, y \in M$

$$(1 - \lambda_1)d_M(x, y) \leq d_G(x, y) \leq (1 + \lambda_2)d_M(x, y), \tag{19}$$

where λ_1, λ_2 relate to the minimum radius of curvature of M, and to a certain δ-sampling condition for every point on M.

Multidimensional Scaling (MDS)

Isomap

Locally Linear Embedding (LLE)

Laplacian Eigenmaps

Locally Linear Embedding (LLE)

Assumption:

- data lies on a manifold: each sample and its neighbors lie on an approximately linear subspace

Approach:

1. Approximate the data cloud by a set of linear patches
2. Glue these patches together on a low-dimensional subspace in such a way that the neighborhood relationships between patches are preserved.

Properties:

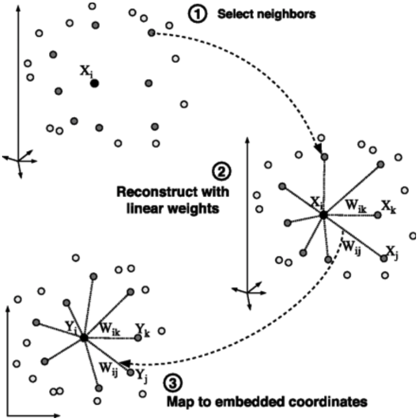
1. can obtain highly nonlinear embeddings
2. not prone to get stuck at local minima
3. sparse graphs lead to sparse problems, hence scalable

`https://cs.nyu.edu/~roweis/lle/algorithm.html`

• Roweis, Sam T., and Lawrence K. Saul. *Nonlinear dimensionality reduction by locally linear embedding*, Science (2000): 2323-2326.

Google Scholar: 14957(2020); 16416 (2021); 17136 (2022); 18,000 (2023)

LLE: Main Steps



Steps

1. Nearest neighbour search.
2. Solve for reconstruction weights W & Least-squares fits.
3. Compute embedding coordinates Y using weights W

Step 1: nearest neighbour search

For each node X_i, $i = 1, \ldots, n$

- compute the distance from X_i to every other point X_j
- find the K smallest distances
- assign the corresponding points to be neighbours of X_j

More efficient computationally:

- use standard algorithms for k-nearest neighbor (k-nn) search
- or even settle for approximation algorithms, that compute k-nearest neighbors
- *Randomized approximate nearest neighbors algorithm*, Peter Wilcox Jones, Andrei Osipov, and Vladimir Rokhlin, PNAS 2011 `https://www.pnas.org/content/108/38/15679.full`
- *nearest neighbor search* is an established area in theoretical computer science.

Step 2: computing the reconstruction weights W

- characterize local geometry of each neighbourhood by weights W_{ij}
- compute weights by reconstructing each input (linearly) from neighbours (assume neighbours lie on locally linear patches of a low-dimensional manifold)

Minimize reconstruction error

- write each point as a linear combination of its neighbors
- weights chosen to minimize the reconstruction error

$$\min_{W} \sum_{i=1} \left(X_i - \sum_j W_{ij} X_j \right)^2 \tag{20}$$

- set $W_{ij} = 0$, if X_j is not a neighbor of X_i
- weights must sum to one: $\sum_{ij} W_{ij} = 1$ (invariance to translation)
- optimal weights W_{ij} obey an important **symmetry**: for any particular data point, they are invariant to rotations, rescalings, and translations of that data point and its neighbors
- weights characterize intrinsic geometric properties of each neighborhood, as opposed to properties that depend on a particular frame of reference.

Step 3: computing the LLE Embedding

- aim to find points $y_i \in \mathbb{R}^d, i = 1, \ldots, n$ to minimize

$$\sum_{i=1}^{n} \left\| y_i - \sum_{j=1}^{n} w_{ij} y_j \right\|^2 \tag{21}$$

- subject to

$$\sum_{i=1}^{n} y_i y_i^T = I_{d \times d} \tag{22}$$

$$\sum_{i=1}^{n} y_i = 0_{d \times 1} \tag{23}$$

- condition (22) means that the points are uncorrelated
- condition (23) centers outputs on origin
- (22) + (23) impose unit covariance matrix
- this eliminates the trivial solution $y_i = 0, i = 1, \ldots, n$
- explicitly, if we denote by $y_i(k)$ the k^{th} entry of y_i, we get

$$\left(\sum_{i=1}^{n} y_i y_i^T \right)_{kj} = \sum_{i=1}^{n} (y_i y_i^T)_{kj} = \sum_{i=1}^{n} y_i(k) y_i(j) = (Y^T Y)_{kj} \tag{24}$$

Step 3: computing the LLE Embedding

- (from prev slide), if we denote by $y_i(k)$ the k^{th} entry of y_i, we get

$$\left(\sum_{i=1}^{n} y_i y_i^T\right)_{kj} = \sum_{i=1}^{n} (y_i y_i^T)_{kj} = \sum_{i=1}^{n} y_i(k) y_i(j) = (Y^T Y)_{kj}$$

- where $k, j = 1, \ldots, d$
- Y is an $n \times d$ matrix given by

$$Y = \begin{pmatrix} - & y_1^T & - \\ - & y_2^T & - \\ & \vdots & \\ - & y_n^T & - \end{pmatrix} \tag{25}$$

- think of $Y^T Y$ as a scaled version of the covariance matrix for the vectors y_i.

Step 3: computing the LLE Embedding

- to find the embedding $y_i \in \mathbb{R}^d$ we seek, recall we aim to minimize

$$\sum_{i=1}^{n} \left\| y_i - \sum_{j=1}^{n} w_{ij} y_j \right\|^2 \tag{26}$$

$$= \sum_{i=1}^{n} \left(y_i - \sum_{k=1}^{n} w_{ik} y_k \right)^T \left(y_i - \sum_{l=1}^{n} w_{il} y_l \right)$$

$$= \sum_{i=1}^{n} y_i^T y_i - \sum_{i=1}^{n} y_i^T \sum_{l=1}^{n} w_{il} y_l - \sum_{i=1}^{n} \sum_{k=1}^{n} w_{ik} y_k^T y_i + $$

$$+ \sum_{i=1}^{n} \sum_{k=1}^{n} w_{ik} y_k^T \sum_{l=1}^{n} w_{il} y_l \tag{27}$$

$$= \sum_{i,j=1}^{n} \delta_{ij} y_i^T y_j - \sum_{i,j=1}^{n} w_{ij} y_i^T y_j - \sum_{i,j=1}^{n} w_{ji} y_i^T y_j + $$

$$+ \sum_{k,l=1}^{n} \left(\sum_{i=1}^{n} w_{ik} w_{il} \right) y_k^T y_l \tag{28}$$

Step 3: computing the LLE Embedding

$$= \sum_{i,j=1}^{n} \delta_{ij} y_i^T y_j - \sum_{i,j=1}^{n} w_{ij} y_i^T y_j - \sum_{i,j=1}^{n} w_{ji} y_i^T y_j + \sum_{i,j=1}^{n} \left(\sum_{k=1}^{n} w_{ki} w_{kj} \right) y_i^T y_j$$

$$= \sum_{i,j=1}^{n} \left(\delta_{ij} - w_{ij} - w_{ji} + \sum_{k=1}^{n} w_{ki} w_{kj} \right) y_i^T y_j \quad (29)$$

$$= \sum_{i,j=1}^{n} M_{ij} y_i^T y_j \quad (30)$$

where

$$M_{ij} = \delta_{ij} - w_{ij} - w_{ji} + \sum_{k=1}^{n} w_{ki} w_{kj} \quad (31)$$

- M is an $n \times n$ symmetric matrix $M = (I - W)^T (I - W)$
- M is non-negative (all its eigenvalues are non-negative)
- denoting by $\mathbf{1}_n$ the all-ones vector of length n, and observing that the rows of W sum to 1, yields

$$M\mathbf{1} = (I - W)^T (I - W)\mathbf{1} = (I - W)^T (\mathbf{1} - \mathbf{1}) = 0 \quad (32)$$

- $\mathbf{1}$ is an eigenvector with corresponding eigenvalue $\lambda = 0$.

Denote by Y_k the k^{th} column of Y defined in (25)

$$Y_k = \begin{pmatrix} y_1(k) \\ \vdots \\ y_n(k) \end{pmatrix} \tag{33}$$

Note that

$$\sum_{k=1}^{d} Y_k^T M Y_k = \sum_{k=1}^{d} \left(\sum_{j=1}^{n} Y_k(j) \ \ (MY_k)(j) \right) \tag{34}$$

$$= \sum_{k=1}^{d} \left(\sum_{j=1}^{n} Y_k(j) \sum_{i=1}^{n} M_{ji} Y_k(i) \right) \tag{35}$$

$$= \sum_{i,j=1}^{n} M_{ij} \left(\sum_{k=1}^{d} Y_k(j) Y_k(i) \right) \tag{36}$$

$$= \sum_{i,j=1}^{n} M_{ij} \left(\sum_{k=1}^{d} y_j(k) y_i(k) \right) = \sum_{i,j=1}^{n} M_{ij} y_i^T y_j \tag{37}$$

which is where we left off in (30) for the original objective function.

Computing the Locally Linear Embedding (LLE) (cont.)

Altogether, the initial objective function

$$\sum_{i=1}^{n} \left\| y_i - \sum_{j=1}^{n} w_{ij} y_j \right\|^2 \tag{38}$$

becomes

$$\begin{array}{cl} \min\limits_{Y_1, \ldots, Y_d} & \sum_{k=1}^{d} Y_k^T M Y_k \\ \text{s.t.} & Y^T Y = I_d, \\ & Y_k \mathbf{1}_n = 0, \forall k = 1, \ldots, d \end{array} \tag{39}$$

Also note that

$$\sum_{i,j=1}^{n} M_{ij} y_i^T y_j = \text{trace}(Y^T M Y) \tag{40}$$

To minimize the obj in (39) subject to the constraints, we next consider the Lagrangian.

Computing the Locally Linear Embedding (LLE) (cont.)

- To minimize the obj in (39) subject to the constraints, we next consider the Lagrangian

$$L(Y_1, \ldots, Y_d,\ \phi_1, \ldots, \phi_d) = \sum_{k=1}^{d} Y_k^T M Y_k - \sum_{k=1}^{d} \phi_k (Y_k^T Y_k - 1) \quad (41)$$

- This is actually a *relaxation* since we discarded
  - all the off-diagonal constraints from $Y^T Y = I_d$
  - and also the centering constraints $Y_k \mathbf{1}_n = 0$
- It will turn out that the solution of the relaxed problem will satisfy all the required constraints in (39)
- consider the partials

$$\frac{\partial L}{\partial Y_k} = (M + M^T) Y_k - 2\phi_k Y_k \quad (42)$$

Computing the Locally Linear Embedding (LLE) (cont.)

- taking partials

$$\frac{\partial L}{\partial Y_k} = (M + M^T)Y_k - 2\phi_k Y_k \tag{43}$$

- leads to

$$MY_k = \phi_k Y_k \tag{44}$$

  making Y_k an eigenvector of M

- since M is symmetric, the condition $Y^T Y = I_d$ holds
- also, the all-ones vector $\mathbf{1}_n$ is an eigenvector, and thus all other eigenvectors satisfy $Y_k \mathbf{1}_n = 0$
- L is thus a sum of d eigenvalues, which is minimized by choosing Y_k to be the d eigenvectors corresponding to the d smallest eigenvalues, ignoring the first trivial eigenvector $\mathbf{1}_n$.

Multidimensional Scaling (MDS)

Isomap

Locally Linear Embedding (LLE)

Laplacian Eigenmaps

Laplacian Eigenmaps

Belkin, Mikhail, and Partha Niyogi. *Laplacian eigenmaps and spectral techniques for embedding and clustering*" Advances in neural information processing systems. 2002

Google Scholar citations: 5858 (2023); 5458 (2022); 5062 (2021); 4435 (2020);

Laplacian Eigenmaps

- Input: a set $\{x_1, \ldots, x_n\}$ of n points, $x_i \in \mathbb{R}^D$
- Output: find a set $\{y_1 \ldots, y_n\}, y_i \in \mathbb{R}^D$ such that y_i represents x_i as best as possible.
- Assumption: $x_i \in \mathcal{M}$, where $\mathcal{M}$ is a manifold embedded in $\mathbb{R}^D$.

Algorithm:

- Construct a graph $G = (V, E)$, undirected and symmetric, where $V = x_1, \ldots, x_n$, and $(x_i, x_j) \in E$ if x_i is "close" to x_j, where "close" means, for example, that
  - x_i and x_j are at most ϵ distance apart
  - or that x_i is within the K nearest neighbors of x_j (or vice versa for symmetry).
- Choose weights for each edge; use a Gaussian Kernel

$$W_{ij} = \begin{cases} e^{-||x_i - x_j||^2/t} & \text{if } (x_i, x_j) \in E \\ 0 & \text{otherwise.} \end{cases} \quad (45)$$

- alternatively, we can use a "parameter-free" approach: $W_{ij} = 1$ if $(x_i, x_j) \in E$, and $W_{ij} = 0$ otherwise.

Laplacian Eigenmaps

- Define the diagonal matrix D of row sums of W

$$D_{ii} = \sum_j W_{ij} \tag{46}$$

- Build graph Laplacian $L = D - W$ (symmetric, positive definite)
- Find eigenvectors of the generalized eigenvector problem

$$Lf = \lambda Df \tag{47}$$

- let $f_0, \ldots, f_d$ be the solutions to the eigenvalue problem ordered such that $0 = \lambda_0 \leq \ldots \leq \lambda_d$, that is

$$\begin{aligned} L\, f_0 &= \lambda D\, f_0 \\ L\, f_1 &= \lambda D\, f_1 \\ &\vdots \\ L\, f_d &= \lambda D\, f_d \end{aligned} \tag{48}$$

- drop $f_0 = [1, \ldots, 1]^T$ and define the embedding of x_i into $\mathbb{R}^d$

$$x_i \mapsto (f_0(i), \ldots, f_d(i)) \tag{49}$$

Laplacian Eigenmaps - Analysis

- considering the mapping of the original points x_i to the line.
- nearby points in $\mathbb{R}^D$ (corresponding to connected points in the graph G) should be mapped to nearby points on the line.
- denote this map by $(y_1, \ldots, y_n)$, with $y_i \in \mathbb{R}$.
- a reasonable criterion for such a map is

$$\min_{y_1,\ldots,y_n} \sum_{i,j} (y_i - y_j)^2 W_{ij}. \tag{50}$$

- if x_i and x_j are close, then W_{ij} is large, thus there is heavy penalty if those are mapped apart.

Laplacian Eigenmaps - derivation ($\mathbb{R}$)

$$\begin{aligned}\sum_{i,j}(y_i - y_j)^2 W_{ij} &= \sum_{i,j}(y_i^2 - 2y_iy_j + y_j^2)W_{ij}\\ &= \sum_i y_i^2 \sum_j W_{ij} - 2\sum_{i,j} y_iy_jW_{ij} + \sum_j y_j^2 \sum_i W_{ij}\\ &= \sum_i y_i^2 D_{ii} - 2\sum_{i,j} y_iy_jW_{ij} + \sum_j y_j^2 D_{jj}\\ &= \sum_{i,j} y_iy_jD_{ij} + \sum_{i,j} y_iy_jD_{ij} - 2\sum_{i,j} y_iy_jW_{ij}\\ &= 2\sum_{i,j} y_iy_j(D - W)_{ij} = 2y^TLy\end{aligned}$$

- which also shows that L is positive semidefinite
- the minimization problem is $\operatorname{argmin}\ y^TLy$
- to remove an arbitrary scaling from the embedding & eliminate the trivial solution, add the constraint $y^TDy = 1$

$$\operatorname*{argmin}_{y^TDy=1}\ y^TLy \tag{51}$$

Laplacian Eigenmaps

- D_{ii} construed as a measure of the importance of vertex i
- the normalization $D^{-1}L$ (equivalent to the normalization above), has a probabilistic interpretation
- consider the Lagrangian
$$H = y^T L y + \lambda(y^T D y - 1) \tag{52}$$
- differentiate and equate to zero
$$(L + L^T)y = \lambda(D + D^T)y, \tag{53}$$
- which amounts to
$$Ly = \lambda D y \tag{54}$$
- thus, y is an eigenvector; enforce $y^T D y = 1$ by dividing y by the scaling factor $(\sum_i y_i^2 D_{ii})^{1/2}$
- for y an eigenvector, the objective function evaluates to $y^T L y = \lambda y^T D y = \lambda$ due to the constraint
- optimal solution: eigenvector of the smallest eigenvalue
- $\mathbf{1}_n$ is an eigenvector with eigenvalue 0 (due to the row stochastic normalization); discard this solution
- use eigenvector corresponding to the next smallest eigenvalue

Laplacian Eigenmaps - the d-dimensional case

- so far we looked at the case of embedding into 1-D
- in the more general case, we are looking for a d-dimensional embedding, that is, a matrix

$$Y = (y_1, \ldots, y_d), \quad y_i \in \mathbb{R}^n \tag{55}$$

- the i^{th} row is the embedding of the i point in $\mathbb{R}^d$. Each column is of length n and gives one of the coordinates.
- in this setting, we aim to minimize

$$\sum_{i,j} ||y^{(i)} - y^{(j)}||^2 W_{ij} = \ldots = 2\,\mathrm{trace}(Y^T L Y) \tag{56}$$

where $y^{(i)}$ is the i-th row of Y, that is the d-dimensional representation of x_i
- the minimization problem becomes

$$\min_{Y^T D Y = I} trace(Y^T L Y) \tag{57}$$

- as before, the solution is given by the eigenvectors of L corresponding to the lowest eigenvalues, after discarding the constant eigenvector $\mathbf{1}_n$ corresponding to eigenvalue $\lambda = 0$