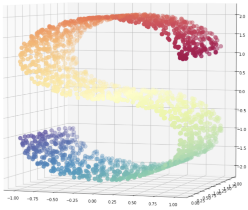
Mathematical techniques in data science

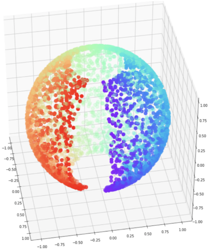
Lecture 34: Manifold learning

Admins

- There will be no Homework 5
- Project presentations:
  - Wed 05/11
  - Fri 05/13
  - Mon 05/16
- Project report due: Thu 05/19

Manifold learning





- high-dimensional data often has a low-rank structure
- question: how can we discover low dimensional structures in data?

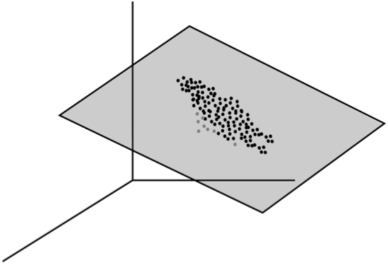
Some definitions

- Metric space: a space on which one can compute the distance between any two points
- Manifold: every point has a neighborhood that is homeomorphic to an open subset of an Euclidean space
- a manifold is locally Euclidean while globally its structure is more complex
- The dimension of a manifold is equal to the dimension of this Euclidean space

Topics

- Linear methods
  - *Principal component analysis*
  - **Multi-dimensional scaling (MDS)**
- Non linear methods
  - **Isomap**
  - Spectral embedding
  - Locally linear embedding (LLE)
  - t-distributed Stochastic Neighbor Embedding (t-SNE)

Principal component analysis



Problem: How can we discover low dimensional structures in data?

- Principal components analysis: construct projections of the data that capture most of the *variability* in the data.
- Provides a low-rank approximation to the data.
- Can lead to a significant dimensionality reduction.

Multidimensional scaling

Multidimensional scaling (MDS)

- is a means of visualizing the level of similarity of individuals of a dataset
- seeks a low-dimensional representation of the data that respects the distances in the original high-dimensional space
- the goal of an MDS analysis is to find a spatial configuration of objects when all that is known is some measure of their general (dis)similarity

Problem settings

- The data to be analyzed is a collection of n objects on which a distance function is defined: d_{ij} is the distance between objects i and object j
- Given d_{ij}, MDS want to finds vector $z_1, z_2, \ldots, z_n \in \mathbb{R}^d$ such that

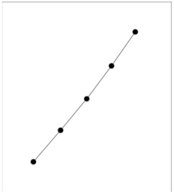
$$d_{ij} \approx \|z_i - z_j\|$$

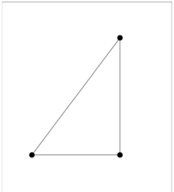
- MDS is formulated as an optimization problem

$$\min_{x_1,\ldots,x_n} \sum_{i<j} (d_{ij} - \|x_i - x_j\|)^2$$

Problem settings

$$D = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 \\ 1 & 0 & 1 & 2 & 3 \\ 2 & 1 & 0 & 1 & 2 \\ 3 & 2 & 1 & 0 & 1 \\ 4 & 3 & 2 & 1 & 0 \end{bmatrix} \qquad D = \begin{bmatrix} 0 & 3 & 4 \\ 3 & 0 & 5 \\ 4 & 5 & 0 \end{bmatrix}$$





MDS is formulated as an optimization problem

$$\min_{x_1,\dots,x_n} \sum_{i<j} \left(d_{ij} - \|x_i - x_j\|\right)^2$$

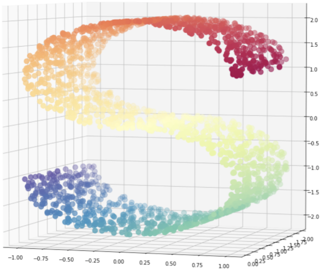
MDS

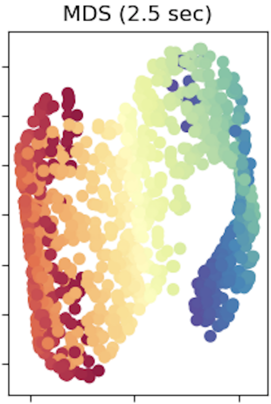
- MDS is formulated as an optimization problem

$$\min_{x_1,\ldots,x_n} \sum_{i<j} \left(d_{ij} - \|x_i - x_j\|\right)^2$$

- the idea is simple, but is easily generalizable

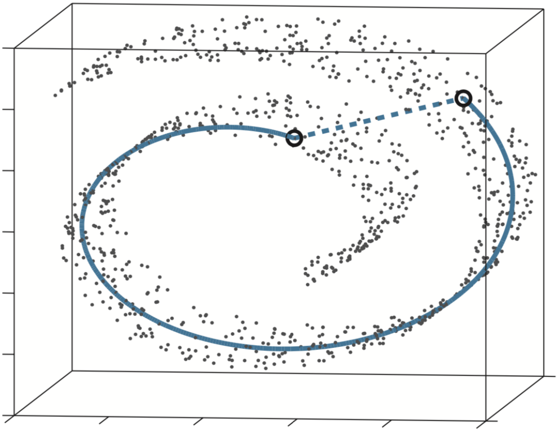
MDS


2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00


MDS (2.5 sec)

Isometric feature mapping (Isomap)

Distance on a manifold



Isomap

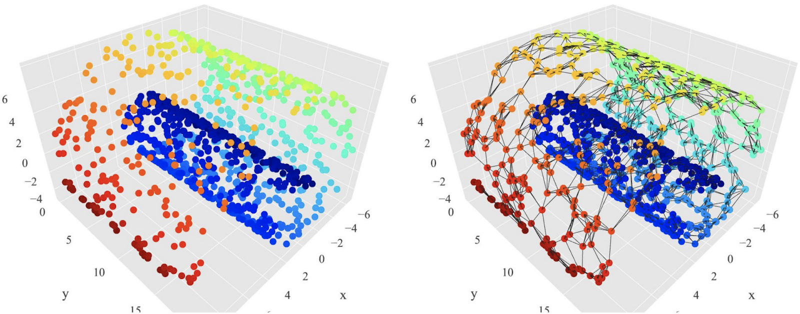
Isomap differs from MDS in one vital way - the construction of the distance matrix.

- In MDS, the distance between two points is just the euclidean distance
- In Isomap, the distances between points are the weight of the shortest path in a point-graph

Isomap: neighbor graph

- For each point, determine either
  - K nearest neighbors
  - all points in a fixed radius
- Construct a neighborhood graph.
  - each point is connected to other if it is a K nearest neighbor.
  - edge length equal to Euclidean distance between the points

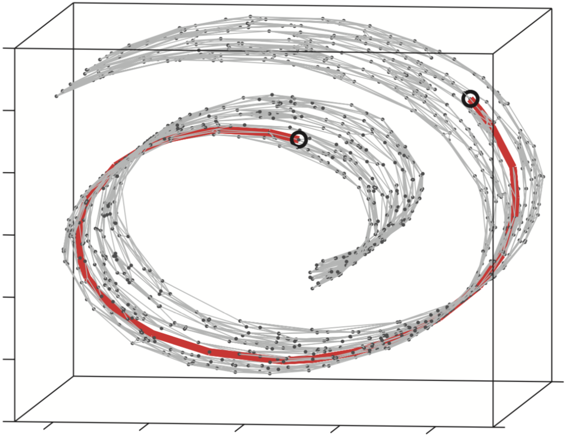
Neighbor graph



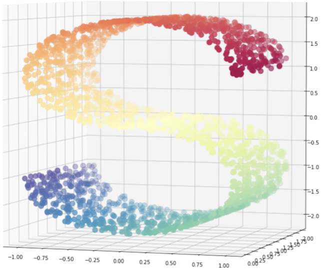
Isomap: compute intrinsic distance

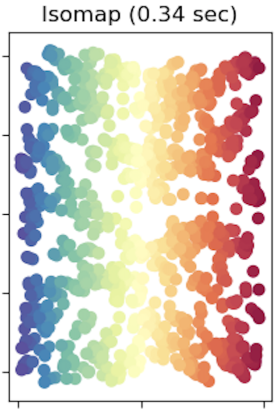
- Compute shortest path between two nodes
  - Dijkstra's algorithm
  - Floyd–Warshall algorithm
- Compute lower-dimensional embedding using MDS
- The graph distance is non-Euclidean, so when embedded back into Euclidean space, some distortion occur

Intrinsic distance



Isomap


2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00


Isomap (0.34 sec)

Locally linear embedding

Locally linear embedding

- A manifold is locally Euclidean while globally its structure is more complex
- Locally, the relation between data points in a neighborhood is linear/affine
- Idea: try to preserve this linear structure

Locally linear embedding

1. For each data point x_i in p dimensions, we find its K-nearest neighbors $\mathcal{N}(i)$ in Euclidean distance.

2. We approximate each point by an affine mixture of the points in its neighborhood:

$$\min_{W_{ik}} ||x_i - \sum_{k \in \mathcal{N}(i)} w_{ik} x_k||^2 \tag{14.102}$$

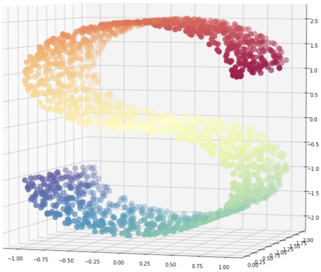
   over weights w_{ik} satisfying $w_{ik} = 0,\ k \notin \mathcal{N}(i)$, $\sum_{k=1}^{N} w_{ik} = 1$. w_{ik} is the contribution of point k to the reconstruction of point i. Note that for a hope of a unique solution, we must have $K < p$.

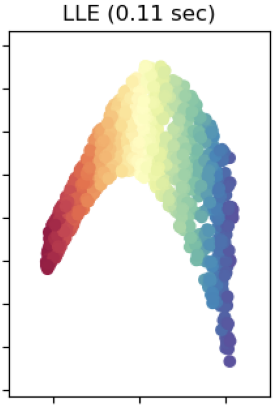
3. Finally, we find points y_i in a space of dimension $d < p$ to minimize

$$\sum_{i=1}^{N} ||y_i - \sum_{k=1}^{N} w_{ik} y_k||^2 \tag{14.103}$$

   with w_{ik} fixed.

LLE


2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00


LLE (0.11 sec)

t-distributed stochastic neighbor embedding

t-SNE

- All methods proposed so far are great, and they work well if $\mathcal{M}$ is a manifold of low-dimension (2 dimension)
- Sometimes, even if the dimension of $\mathcal{M}$ is high, we still want to embed it to $\mathbb{R}^2$ for learning

t-SNE

- There are many problems with embedding high-dimensional manifold to low-dimensional space
- Structural differences
  - in ten dimensions, it is possible to have 11 data points that are mutually equidistant
  - there is no way to model this faithfully in a two-dimensional map
- Crowding problem:
  - the volume of a sphere centered on datapoint i scales as r^m, where r is the radius and m the dimensionality of the sphere
  - the area of the two-dimensional map that is available to accommodate moderately distant data points will not be nearly large enough compared with the area available to accommodate nearby data points

Stochastic neighbor embedding

- converting the high-dimensional Euclidean distances between data points into conditional probabilities that represent similarities
- The similarity of datapoint x_j to datapoint x_i is the conditional probability, $p_{j|i}$, that x_i would pick x_j as its neighbor if neighbors were picked in proportion to their probability density under a Gaussian centered at x_i

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

Stochastic neighbor embedding

- Assume that the data points are mapped to $y_1, y_2, \ldots, y_n$ in low-dimension
- we construct a similar quantity for a y

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

- Goal: Minimize the difference between the two probabilities

$$\min_y \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

t-SNE

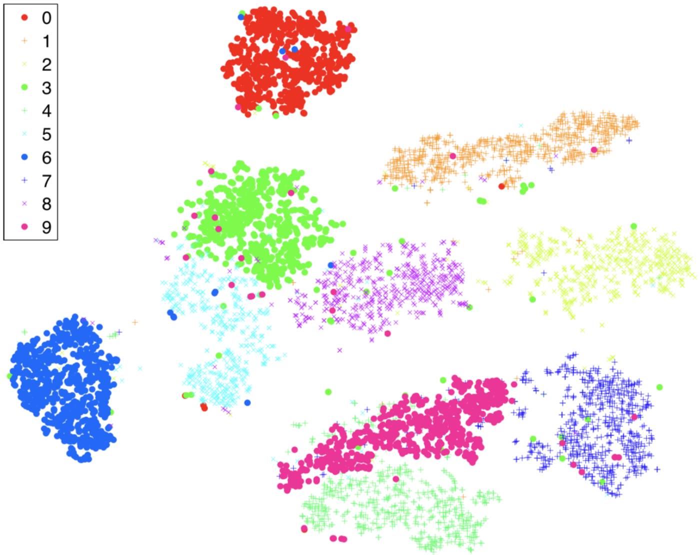
- employ a Student t-distribution with one degree of freedom (which is the same as a Cauchy distribution) as the heavy-tailed distribution in the low-dimensional map

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

- Goal: Minimize the difference between the two probabilities

$$\min_y \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

Visualization of MNIST by t-SNE



Visualization of MNIST by Isomap