ADVANCED MACHINE LEARNING

Laplacian Eigenmaps and Isomap

Lecture: Prof. Aude Billard (aude.billard@epfl.ch)

Teaching Assistants:
Guillaume de Chambrier, Nadia Figueroa, Denys Lamotte

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE
Outline of the presentation

- Reminder on spectral clustering to find the number of clusters
- Detailed description of:
  - Laplacian Eigenmaps
  - Isomap
Spectral Clustering

Algorithm in the general case (S not binary)
1. Build the similarity matrix S, the diagonal matrix D and the Laplacian matrix \( L = D - S \)
2. Solve some eigenvalue decomposition problem involving S, D and L.
3. Order the eigenvalues by increasing order: \( \lambda_1 = 0 \leq \lambda_2 \leq \ldots \leq \lambda_M \)
4. Apply a threshold on the eigenvalues, such that small \( \lambda \rightarrow 0 \)
5. Determine the number of clusters by looking at the multiplicity of \( \lambda = 0 \) after step 4

This provides an indication of the number of clusters \( K \).
Example

Set of digits from 0 to 5

The 6 classes are not well separated into 6 clusters

But number of eigenvalues close to 0 is 6 → Fit the 6 cluster of the dataset (0 to 5)
Laplacian Eigenmaps step by step (1)

Building the adjacency graph

Given $M$ points $x_1, \ldots, x_M$ we construct a weighted graph with $M$ nodes (one for each point), and a set of edges connecting neighboring points.

We put an edge between nodes $i$ and $j$ if $x_i$ is among $k$ nearest neighbors of $x_j$ or if $x_j$ is among $k$ nearest neighbors of $x_i$ (the relation is symmetric).

$k$ the number of nearest neighbors for each point is the first hyper-parameter of the method.
Laplacian Eigenmaps step by step (2)

**Weighting the graph**

If there is an edge between nodes $i$ and $j$ we put as weight for the edge:

$$S_{ij} = e^{-\frac{||x_i - x_j||^2}{2\sigma^2}}$$

Else, we put $S_{ij} = 0$

We have computed the similarity matrix $S$.

$\sigma$ the width of the kernel is the second hyper-parameter of the method.
Laplacian Eigenmaps step by step (3)

Computing D and L and solve the eigenvalue decomposition problem

D is the diagonal weight matrix so:

\[ D_{ii} = \sum_j S_{ji} \]

L is the Laplacian matrix:

\[ L = D - S \]

At the end, we compute eigenvalues and eigenvectors for the generalized eigenvector problem:

\[ L y = \lambda D y \]
**Laplacian Eigenmaps**

Solve the generalized eigenvalue problem: \( Ly = \lambda Dy \iff \left( I - D^{-1}S \right) y = \lambda y \)

If \( D \) not invertible, solve: \( \min_y y^T Ly \) such that \( y^T Dy = 1 \).

Ensures minimal distorsion while preventing arbitrary scaling.

![Original dataset](image1.png)

4 classes swiss roll

![Result of Laplacian Eigenmaps](image2.png)

4 pairs of projections
Isomap step by step (1)

Building the adjacency graph

Given $M$ points $x_1, \ldots, x_M$ we look for the distance between points as the sum of weights of the shortest path in a point-graph (one node per point) to build the similarity matrix.

To build the graph, we put an edge between nodes $i$ and $j$ if $x_i$ is among $k$ nearest neighbors of $x_j$

$k$ the number of nearest neighbors for each point is the only hyper-parameter of the method.
Isomap step by step (2)
Constructing the Similarity matrix

If there is an edge between nodes $i$ and $j$ we put simply as weight for the edge the Euclidian distance between these two neighbors:

$$S_{ij} = \| x_i - x_j \|$$

Else (no edge), in order to find the distance between two points which are not in the same k-neighborhood, we simply compute the shortest path for each connected components in the weighted point-graph using the Dijkstra algorithm.

Finally, we put $S_{ij} = 0$ if $x_i$ and $x_j$ are not in a same connected component.
Advantage compare to Euclidian distance

The distance computed in the Isomap algorithm can be seen as the geodesic distance between the points. It enables to capture the underlying manifold more accurately than using Euclidian distances.

Euclidian distance

\[ d(1,6) = d(1,8) \]

Geodesic Distance (1-neighbors)

\[ d(1,6) < d(1,8) \]
Isomap step by step (3)

Applying the MDS algorithm to the S matrix

1. Center the similarity matrix: 
   \[ S'_{ij} = S_{ij} - \frac{1}{M} D_i - \frac{1}{M} D_j + \frac{1}{M} D_i D_j \]

2. Perform eigenvalue decomposition of \( S' \)

3. Consider only the eigenvectors with positive eigenvalues

4. Generate scaled projections 
   \[ y^j_i = \sqrt{\lambda_i} e^i_j \]

The geodesic distances encapsulate well the neighboring. Combined with the MDS flattening of the space, they allow to extract well the 4 classes.