Lecture 16. Manifold Learning

COMP90051 Statistical Machine Learning

Semester 2, 2017
Lecturer: Andrey Kan

THE UNIVERSITY OF MELBOURNE

Swiss roll image: Evan-Amos, Wikimedia Commons, CC0

This lecture

- Introduction to manifold learning
  - Motivation
  - Focus on data transformation
- Unfolding the manifold
  - Geodesic distances
  - Isomap algorithm
- Spectral clustering
  - Laplacian eigenmaps
  - Spectral clustering pipeline

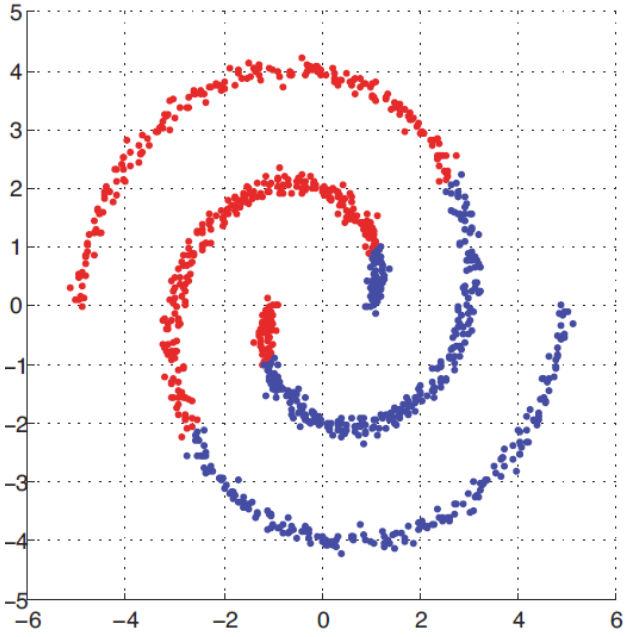
Manifold Learning

Recovering low dimensional data representation non-linearly embedded within a higher dimensional space

The limitation of k-means and GMM

- K-means algorithm can find spherical clusters
- GMM can find elliptical clusters
- These algorithms will struggle in cases like this

K-means clustering



desired result

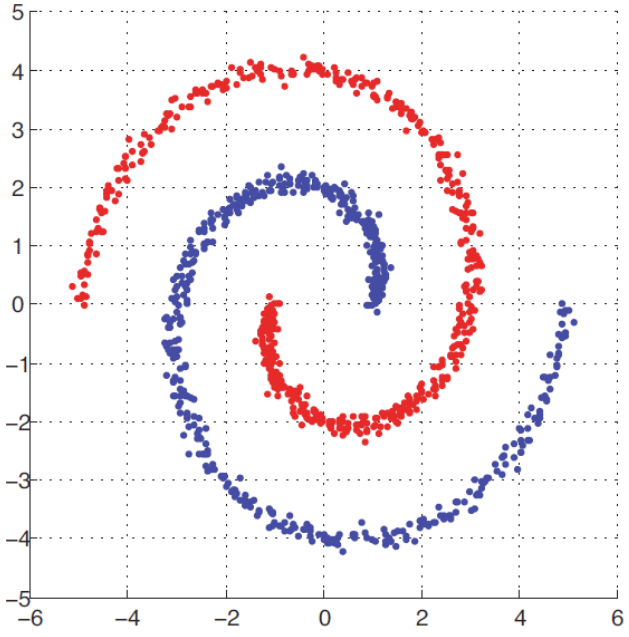


Figure from Murphy

Focusing on data geometry

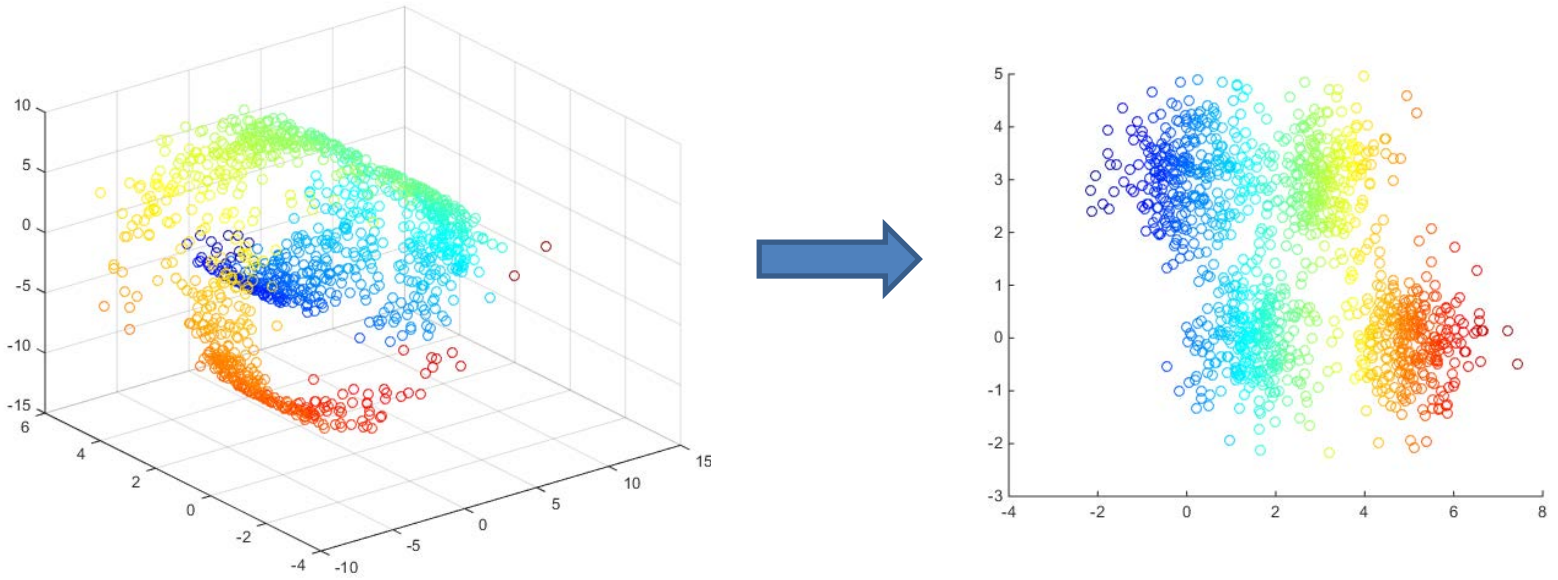
- We are not dismissing the k-means algorithm yet, but we are going to put it aside for a moment
- One approach to address the problem in the previous slide would be to introduce improvements to algorithms such as k-means
- Instead, let's focus on geometry of the data and see if we can *transform the data* to make it amenable for simpler algorithms
  - Recall "transform the data vs modify the model" discussion in supervised learning

Non-linear data embedding

- Recall the example with 3D GPS coordinates that denote a car's location on a 2D surface
- In a similar example consider coordinates of items on a picnic blanket which is approximately a plane
  - In this example, the data resides on a plane embedded in 3D
- A low dimensional surface can be quite curved in a higher dimensional space
  - A plane of dough (2D) in a Swiss roll (3D)

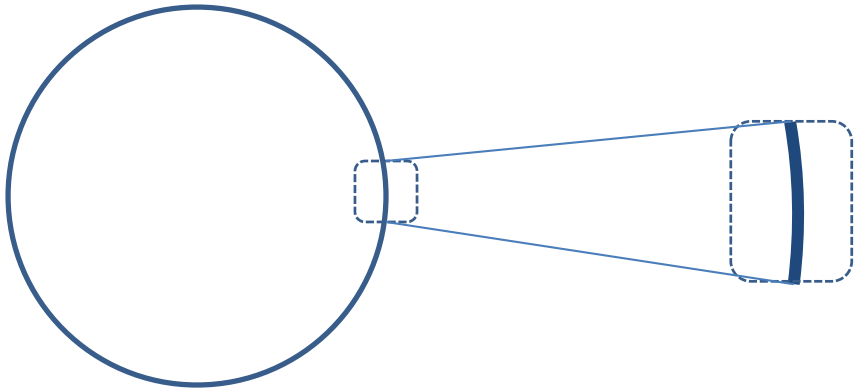
Picnic blanket image: Theo Wright, Flickr, CC2
Swiss roll image: Evan-Amos, Wikimedia Commons, CC0

Key assumption: It's simpler than it looks!



- Key assumption: High dimensional data actually resides in a lower dimensional space that is locally Euclidean
- Informally, the *manifold* is a subset of points in the high-dimensional space that locally looks like a low-dimensional space

Manifold example



$$AC \approx AB + BC$$

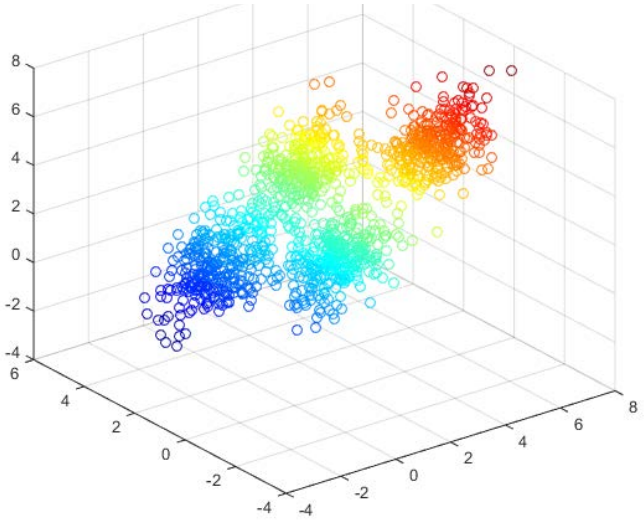
- Informally, the ***manifold*** is a subset of points in the high-dimensional space that locally looks like a low-dimensional space
- Example: arc of a circle
  - consider a tiny bit of a circumference (2D) → can treat as line (1D)

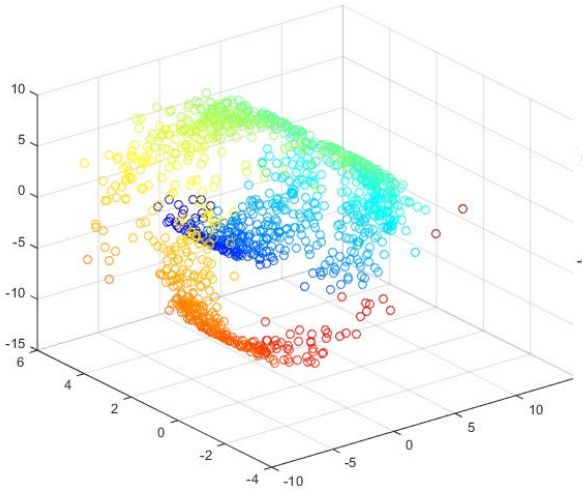
l-dimensional manifold

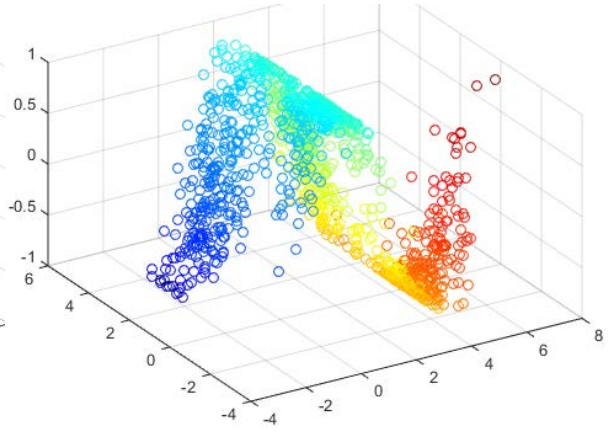
- Definition from *Guillemin and Pollack, Differential Topology, 1974*
- A mapping f on an open set $U \subset \boldsymbol{R}^m$ is called *smooth* if it has continuous partial derivatives of all orders
- A map $f: X \to \boldsymbol{R}^l$ is called *smooth* if around each point $\boldsymbol{x} \in X$ there is an open set $U \subset \boldsymbol{R}^m$ and a smooth map $F: U \to \boldsymbol{R}^l$ such that F equals f on $U \cap X$
- A smooth map $f: X \to Y$ of subsets of two Euclidean spaces is a *diffeomorphism* if it is one to one and onto, and if the inverse map $f^{-1}: Y \to X$ is also smooth. X and Y are *diffeomorphic* if such a map exists
- Suppose that X is a subset of some ambient Euclidean space $\boldsymbol{R}^m$. Then X is an l-dimensional *manifold* if each point $\boldsymbol{x} \in X$ possesses a neighbourhood $V \subset X$ which is diffeomorphic to an open set $U \subset \boldsymbol{R}^l$

Manifold examples

- A few examples of manifolds are shown below
- In all cases, the idea is that (hopefully) once the manifold is "unfolded", the analysis, such as clustering becomes easy
- How to "unfold" a manifold?



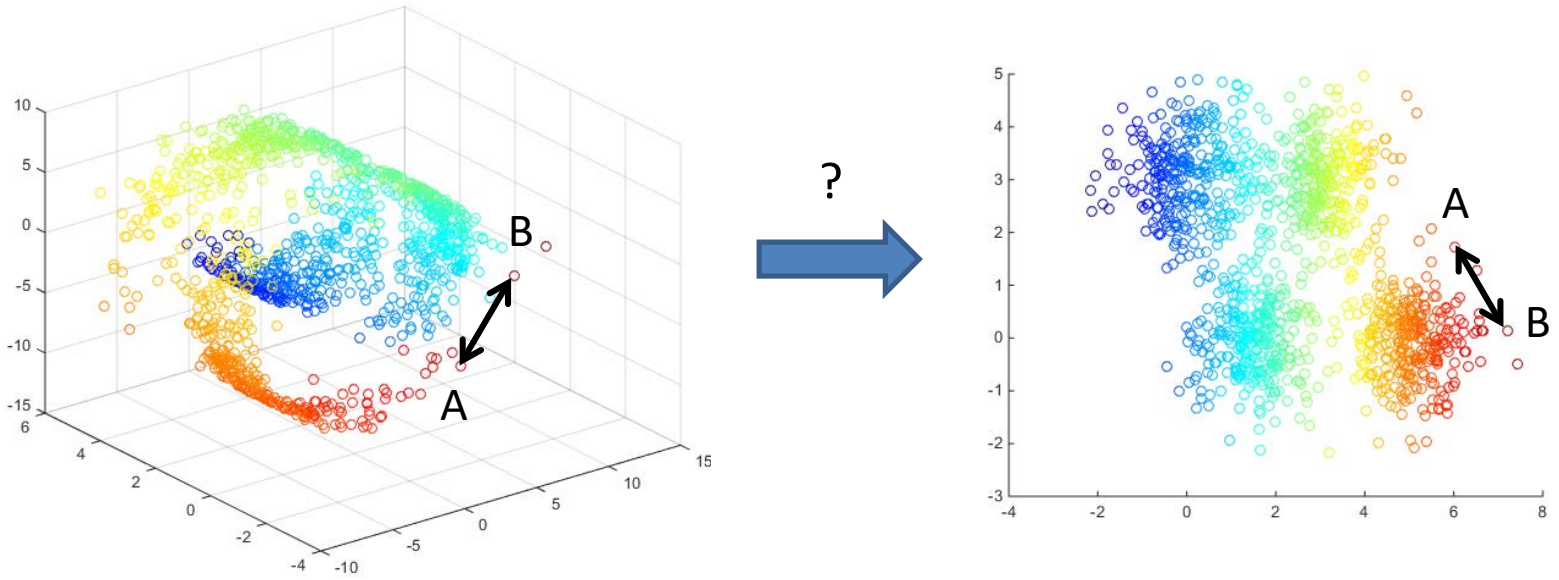




Geodesic Distances and Isomap

A non-linear dimensionality reduction algorithm that preserves locality information using geodesic distances

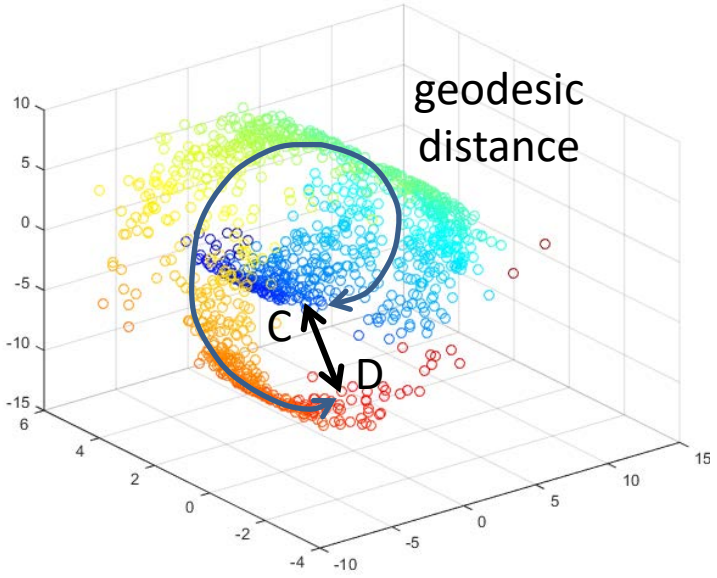
General idea: Dimensionality reduction

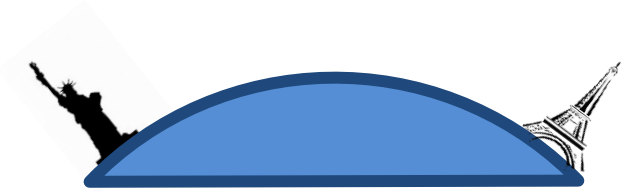


- Find a lower dimensional representation of data that preserves distances between points (MDS)
- Do visualization, clustering, etc. on lower dimensional representation

Problems?

"Global distances" VS geodesic distances





- "Global distances" cause a problem: we may not want to preserve them
- We are interested in preserving distances along the manifold (*geodesic distances*)

Images: ClkerFreeVectorImages and Kaz @pixabay.com (CC0)

MDS and similarity matrix

- In essence, "unfolding" a manifold is achieved via dimensionality reduction, using methods such as MDS
- Recall that the input of an MDS algorithm is similarity (aka proximity) matrix where each element w_{ij} denotes how similar data points i and j are
- Replacing distances with geodesic distances simply means constructing a different similarity matrix without changing the MDS algorithm
  - Compare it to the idea of modular learning in kernel methods
- As you will see shortly, there is a close connection between similarity matrices and graphs and in the next slide, we review basic definitions from graph theory

Refresher on graph terminology

- *Graph* is a tuple $G = \{V, E\}$, where V is a set of *vertices*, and $E \subseteq V \times V$ is a set of *edges*. Each edge is a pair of vertices
  - *Undirected* graph: pairs are unordered
  - *Directed* graph: pairs are ordered
- Graphs model pairwise relations between objects
  - Similarity or distance between the data points
- In a *weighted graph*, each vertex v_{ij} has an associated weight w_{ij}
  - Weights capture the strength of the relation between objects

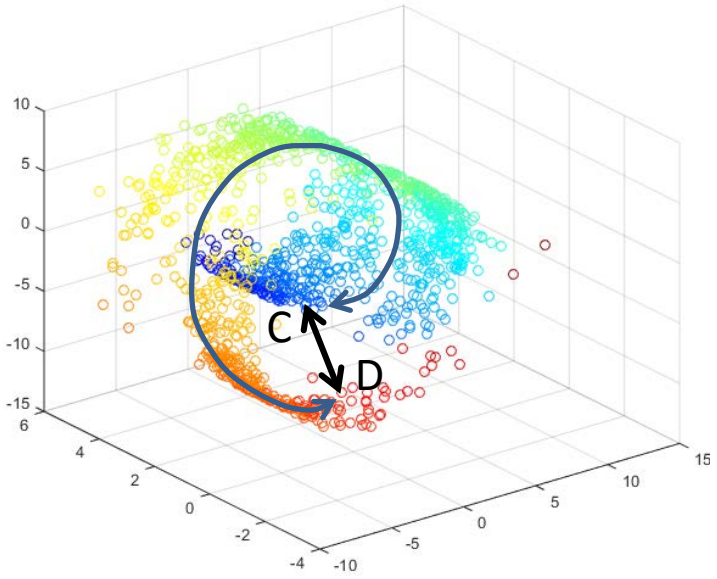
Weighted adjacency matrix

- We will consider weighted undirected graphs with non-negative weights $w_{ij} \geq 0$. Moreover, we will assume that $w_{ij} = 0$, if and only if vertices i and j are not connected
- The *degree* of a vertex $v_i \in V$ is defined as

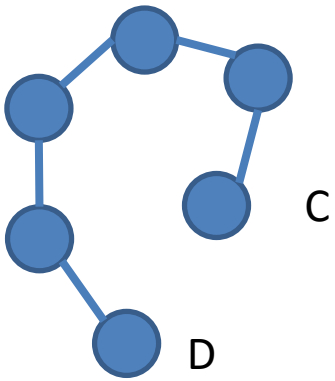
$$\deg(i) \equiv \sum_{j=1}^{n} w_{ij}$$

- A weighted undirected graph can be represented with an *weighted adjacency matrix* $\boldsymbol{W}$ that contain weights w_{ij} as its elements

Similarity graph models data geometry

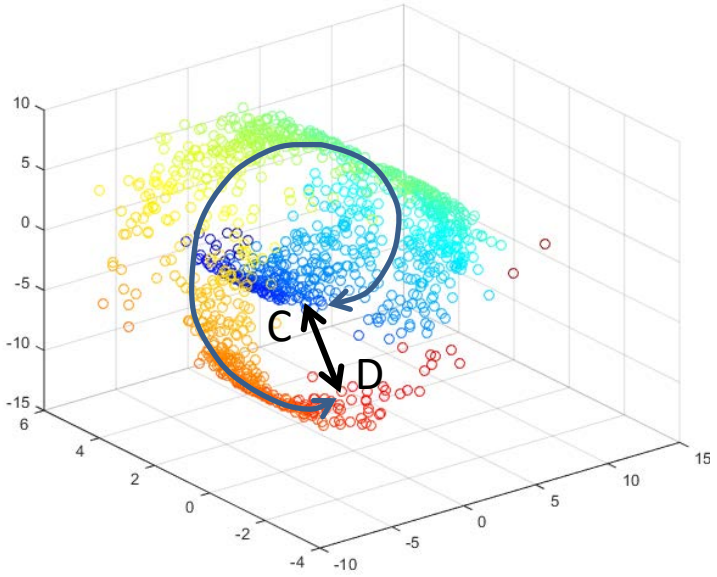


geodesic distance

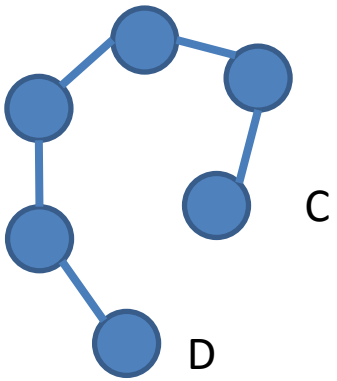


- Geodesic distances can be approximated using a graph in which vertices represent data points
- Let $d(i,j)$ be the Euclidean distance between the points in the original space
- Option 1: define some local radius ε. Connect vertices i and j with an edge if $d(i,j) \leq \varepsilon$
- Option 2: define nearest neighbor threshold k. Connect vertices i and j if i is among the k nearest neighbors of j OR j is among the k nearest neighbors of i
- Set weight for each edge to $d(i,j)$

Computing geodesic distances

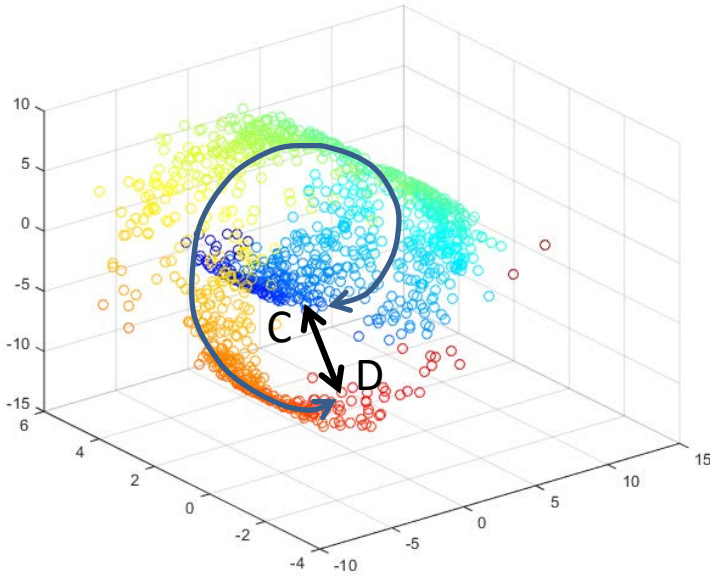


geodesic distance

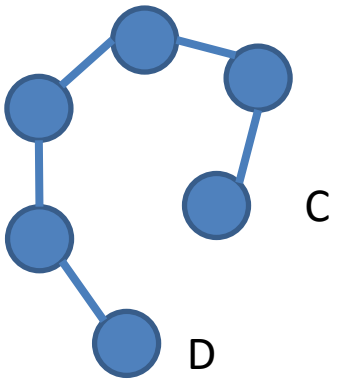


- Given the similarity graph, compute shortest paths between each pair of points
  - E.g., using Floyd-Warshall algorithm in $O(n^3)$
- Set geodesic distance between vertices i and j to the length (sum of weights) of the shortest path between them
- Define a new similarity matrix based on geodesic distances

Isomap: summary



geodesic distance



1. Construct the similarity graph
2. Compute shortest paths
3. Geodesic distances are the lengths of the shortest paths
4. Construct similarity matrix using geodesic distances
5. Apply MDS

Spectral Clustering

An spectral graph theory approach to non-linear dimensionality reduction

Data processing pipelines

- Isomap algorithm can be considered a pipeline in a sense that in combines different processing blocks, such as graph construction, and MDS
- Here MDS serves as a core sub-routine to Isomap
- Spectral clustering is similar to Isomap in that it also comprises a few standard blocks, including k-means clustering
- In contrast to Isomap, spectral clustering uses a different non-linear mapping technique called Laplacian eigenmap

Spectral clustering algorithm

1. Construct similarity graph, use the corresponding adjacency matrix as a new similarity matrix
   * Just as in Isomap, the graph captures local geometry and breaks long distance relations
   * Unlike Isomap, the adjacency matrix is used "as is", shortest paths are not used
2. Map data to a lower dimensional space using Laplacian eigenmaps on the adjacency matrix
   * This uses results from spectral graph theory
3. Apply k-means clustering to the mapped points

Similarity graph for spectral clustering

- Again, we start with constructing a similarity graph. This can be done in the same way as for Isomap (but no need to compute the shortest paths)
- Recall that option 1 was to connect points that are closer than ε, and options 2 was to connect points within k neiborhood
- There is also option 3 usually considered for spectral clustering. Here all points are connected to each other (the graph is fully connected). The weights are assigned using a Gaussian kernel (*aka* heat kernel) with width parameter σ

$$w_{ij} = \exp\left(-\frac{1}{\sigma}\left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|^2\right)$$

Graph Laplacian

- Recall that $\boldsymbol{W}$ denotes weighted adjacency matrix which contains all weights w_{ij}
- Next, *degree matrix* $\boldsymbol{D}$ is defined as a diagonal matrix with vertex degrees on the diagonal. Recall that a vertex degree is $\deg(i) = \sum_{j=1}^{n} w_{ij}$
- Finally, another special matrix associated with each graph is called unnormalised *graph Laplacian* and is defined as $\boldsymbol{L} \equiv \boldsymbol{D} - \boldsymbol{W}$
  - For simplicity, here we introduce spectral clustering using unnormalised Laplacian. In practice, it is common to use a Laplacian normalised in certain way, e.g., $\boldsymbol{L}_{nrm} \equiv \boldsymbol{I} - \boldsymbol{D}^{-1}\boldsymbol{W}$, where $\boldsymbol{I}$ is an identity matrix

Laplacian eigenmaps

- Laplacian eigenmaps, a central sub-routine of spectral clustering, is a non-linear dimensionality reduction method
- Similar to MDS, the idea is to map the original data points $\boldsymbol{x}_i \in \boldsymbol{R}^m$, $i = 1, \ldots, n$ to a set of low-dimensional points $\boldsymbol{z}_i \in \boldsymbol{R}^l$, $l < m$ that "best represent" the original data
- Laplacian eigenmaps use a similarity matrix $\boldsymbol{W}$ rather than original data coordinates as a starting point
  - Here the similarity matrix $\boldsymbol{W}$ is the weighted adjacency matrix of the similarity graph
- Earlier, we've seen examples of how "best represent" criterion is formalised in MDS methods
- Laplacian eigenmaps use a different criterion, namely the aim is to minimise (subject to some constraints)

$$\sum_{i,j} \left\| \boldsymbol{z}_i - \boldsymbol{z}_j \right\|^2 w_{ij}$$

Alternative representation of mapping

- This minimisation problem is solved using results from spectral graph theory
- Instead of the mapped points $\boldsymbol{z}_i$, the output can be viewed as a set of n −dimensional vectors $\boldsymbol{f}_j, j = 1, \ldots, l$. The solution eigenmap is expressed in terms of these $\boldsymbol{f}_j$
  - For example, if the mapping is onto 1D line, $\boldsymbol{f}_1 = \boldsymbol{f}$ is just a collection of coordinates for all n points
  - If the mapping is onto 2D, $\boldsymbol{f}_1$ is a collection of all the fist coordinates, and $\boldsymbol{f}_2$ is a collection of all the second coordinates
- For illustrative purposes, we will consider a simple example of mapping to 1D

Problem formulation for 1D eigenmap

- Given an $n \times n$ similarity matrix $\boldsymbol{W}$, our aim is to find a 1D mapping $\boldsymbol{f}$, such that f_i is the coordinate of the mapped i^{th} point. We are looking for a mapping that minimises $\frac{1}{2}\sum_{i,j}\left(f_i - f_j\right)^2 w_{ij}$
- Clearly for any $\boldsymbol{f}$, this can be minimised by multiplying $\boldsymbol{f}$ by a small constant, so we need to introduce a scaling constraint, e.g., $\|\boldsymbol{f}\|^2 = \boldsymbol{f}'\boldsymbol{f} = 1$
- Next recall that $\boldsymbol{L} \equiv \boldsymbol{D} - \boldsymbol{W}$

Re-writing the objective in vector form

- $\frac{1}{2}\sum_{i,j}(f_i - f_j)^2 w_{ij}$
- $= \frac{1}{2}\sum_{i,j}(f_i^2 w_{ij} - 2 f_i f_j w_{ij} + f_j^2 w_{ij})$
- $= \frac{1}{2}\left(\sum_{i=1}^n f_i^2 \sum_{j=1}^n w_{ij} - 2\sum_{i,j} f_i f_j w_{ij} + \sum_{j=1}^n f_j^2 \sum_{i=1}^n w_{ij}\right)$
- $= \frac{1}{2}\left(\sum_{i=1}^n f_i^2 \deg(i) - 2\sum_{i,j} f_i f_j w_{ij} + \sum_{j=1}^n f_j^2 \deg(j)\right)$
- $= \sum_{i=1}^n f_i^2 \deg(i) - \sum_{i,j} f_i f_j w_{ij}$
- $= \boldsymbol{f'Df} - \boldsymbol{f'Wf}$
- $= \boldsymbol{f'Lf}$

Laplace recontre Lagrange

- Our problem becomes to minimise $\boldsymbol{f}'\boldsymbol{L}\boldsymbol{f}$, subject to $\boldsymbol{f}'\boldsymbol{f} = 1$. Recall the method of Lagrange multipliers. Introduce a Lagrange multiplier λ, and set derivatives of the Lagrangian to zero
- $\mathcal{L} = \boldsymbol{f}'\boldsymbol{L}\boldsymbol{f} - \lambda(\boldsymbol{f}'\boldsymbol{f} - 1)$
- $2\boldsymbol{f}'\boldsymbol{L}' - 2\lambda\boldsymbol{f}' = 0$
- $\boldsymbol{L}\boldsymbol{f} = \lambda\boldsymbol{f}$

Déjà vu?

- The latter is precisely the definition of an eigenvector with λ being the corresponding eigenvalue!
- Critical points of our objective function $\boldsymbol{f}'\boldsymbol{L}\boldsymbol{f} = \frac{1}{2}\sum_{i,j}(f_i - f_j)^2 w_{ij}$ are eigenvectors of $\boldsymbol{L}$
- Note that the function is actually minimised using eigenvector $\mathbf{1}$, which is not useful. Therefore, for 1D mapping we use an eigenvector with the second smallest eigenvalue

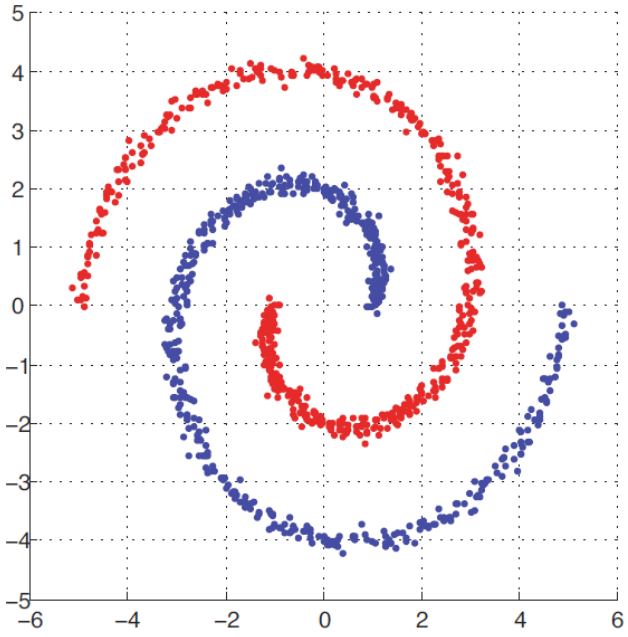
Laplacian eigenmaps: summary

- Start with points $\boldsymbol{x}_i \in \boldsymbol{R}^m$. Construct a similarity graph using one of 3 options
- Construct weighted adjacency matrix $\boldsymbol{W}$ (do not compute shortest paths) and the corresponding Laplacian matrix $\boldsymbol{L}$
- Compute eigenvectors of $\boldsymbol{L}$, and arrange them in the order of the corresponding eignevalues $0 = \lambda_1 < \lambda_2 < \cdots < \lambda_n$
- Take eigenvectors corresponding to λ_2 to λ_{l+1} as $\boldsymbol{f}_1, \ldots, \boldsymbol{f}_l$, $l < m$, where each $\boldsymbol{f}_j$ corresponds to one of the new dimensions
- Combine all vectors into an $n \times l$ matrix, with $\boldsymbol{f}_j$ in columns. The mapped points are the rows of the matrix

Spectral clustering: summary

1. Construct a similarity graph
2. Map data to a lower dimensional space using Laplacian eigenmaps on the adjacency matrix
3. Apply k-means clustering to the mapped points

spectral clustering result



This lecture

- Introduction to manifold learning
  - Motivation
  - Focus on data transformation
- Unfolding the manifold
  - Geodesic distances
  - Isomap algorithm
- Spectral clustering
  - Laplacian eigenmaps
  - Spectral clustering pipeline