Xi'an Jiaotong-Liverpool University

Pattern Recognition

Lecture 18. Dimensionality Reduction: Manifold Learning

Dr. Shanshan ZHAO & Dr. Hong Seng GAN

School of AI and Advanced Computing
Xi'an Jiaotong-Liverpool University

Academic Year 2023-2024

Table of Contents

A kind reminder

- This lecture covers many algorithms.
- It's supposed to provide multiple potential choices for doing your CW.
- The details of these algorithms will NOT be assessed in the exam.

Outline

Introduction

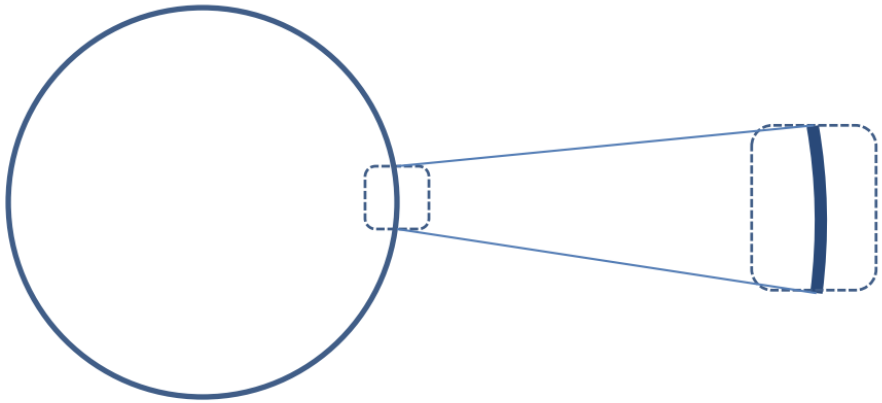
We will discuss the problem of recovering the **underlying low-dimensional structure** in a high-dimensional dataset. This structure is often assumed to be a curved manifold. So this problem is called **manifold learning**.

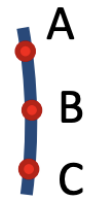
Manifold

Manifold

Roughly speaking, a **manifold** is a topological space which is locally Euclidean. One of the simplest examples is the surface of the earth, which is a curved 2D surface embedded in a 3D space. At each local point on the surface, the earth seems flat. Or a 1D line from a tiny circumference of a 2D curve.

Manifold

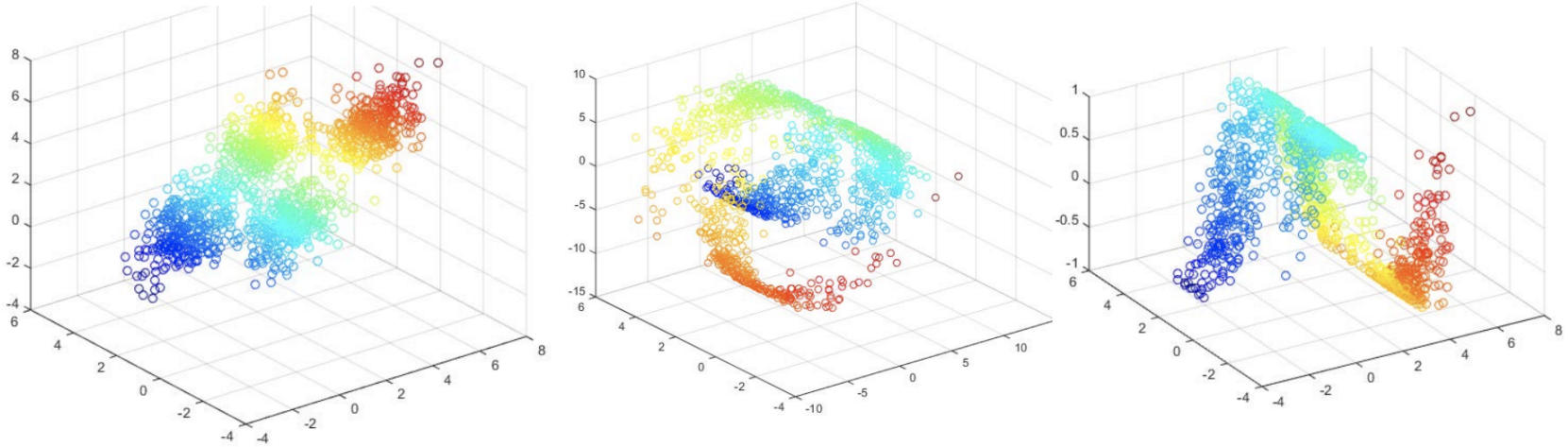




$$AC \approx AB + BC$$

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

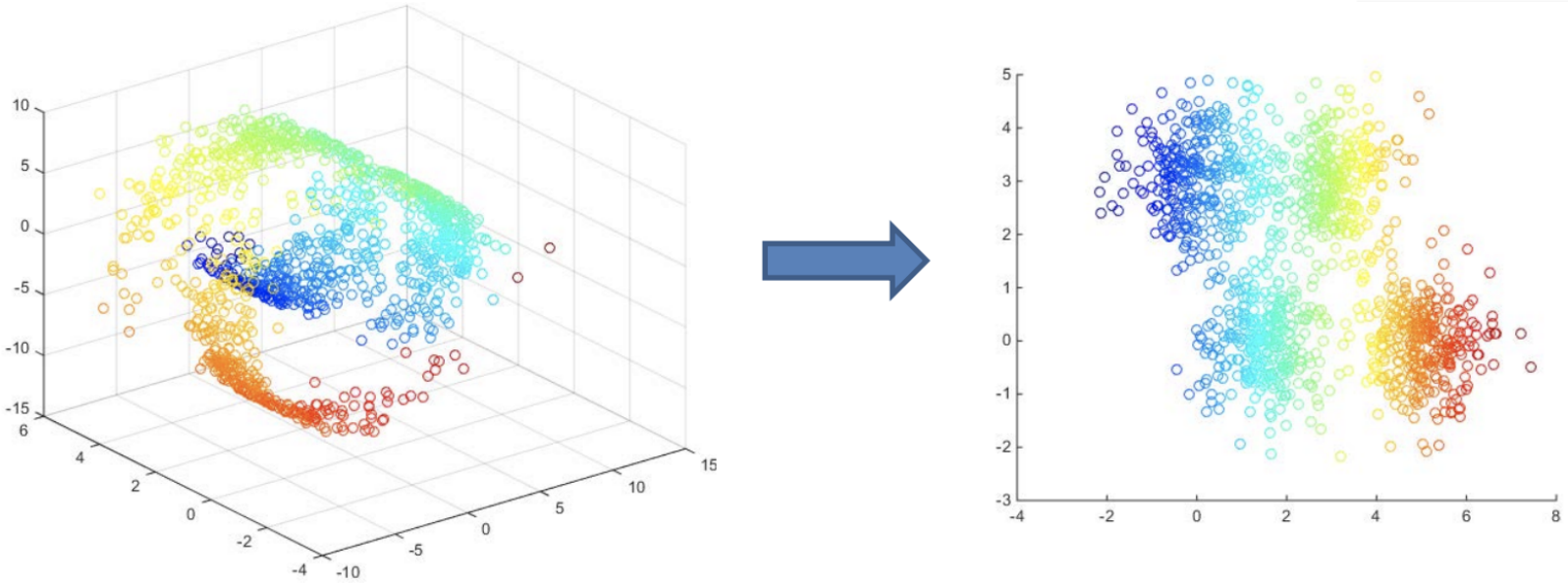
Manifold examples



- 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?

Manifold Key assumption

Most "naturally occuring" high dimensional dataset lie a low dimensional manifold. This is called the manifold hypothesis.



- Key assumption: High dimensional data actually resides in a lower dimensional space that is locally Euclidean

Outline

- The simplest approach to manifold learning is multidimensional scaling (MDS).
- This tries to find a set of low dimensional vectors $\{z_i \in \mathscr{R}^L : i = 1 : N\}$ such that the pairwise distances between these vectors is as similar as possible to a set of pairwise dissimilarities $D = \{d_{ij}\}$ provided by the user.
- There are several variants of MDS
  - Classical MDS (Equivalently to PCA)
  - Metric MDS
  - Non-metric MDS

Multidimensional Scaling (MDS)

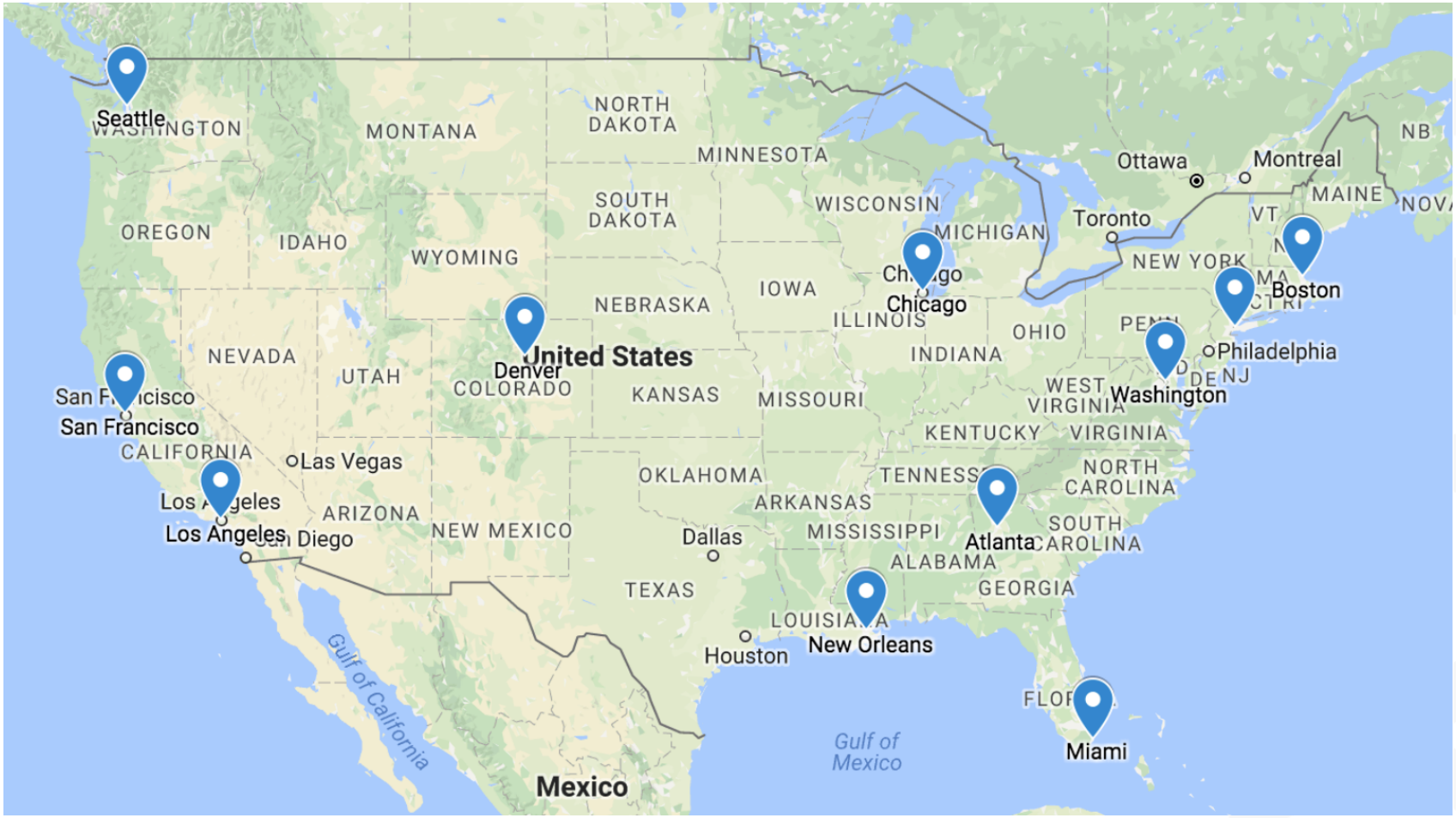
- Classical MDS assumes Euclidean distances. Metric MDS **generalize** it to allow for any dissimilarity measure by defining the stress function.

$$\mathscr{L}(Z) = \sqrt{\frac{\sum_{i<j}(d_{ij} - \hat{d}_{ij})^2}{\sum_{ij} d_{ij}^2}}$$

  where $\hat{d}_{ij} = ||z_i - z_j||$. This is called metric MDS.
- Non-metric MDS, instead of trying to match the distance between points, try to match the ranking of how similar points are. To do this, let $f(d)$ be a monotonic transformation from distances to ranks.

Multidimensional Scaling (MDS)

Example



Multidimensional scaling (MDS)

- It is easy to measure the *scaled* inter-city distances.
  - We only need a ruler and the physical map, or the coordinates of the blue dots to calculate the Euclidean distance.
  - Then the achieved distance numbers can be scaled up to kilometers based on the scale of the original map.
- Now, let's imagine the reverse process, when the inter-cities distances are available, and we want to draw a map out of them, such that the close cities locate close to each other, and far ones locate far from each other. To do so, the between-cities distances of the 11 cities is an option.

Multidimensional scaling (MDS)

distances between cities

Table1 -Approximate pair-wise distances of 11 US cities

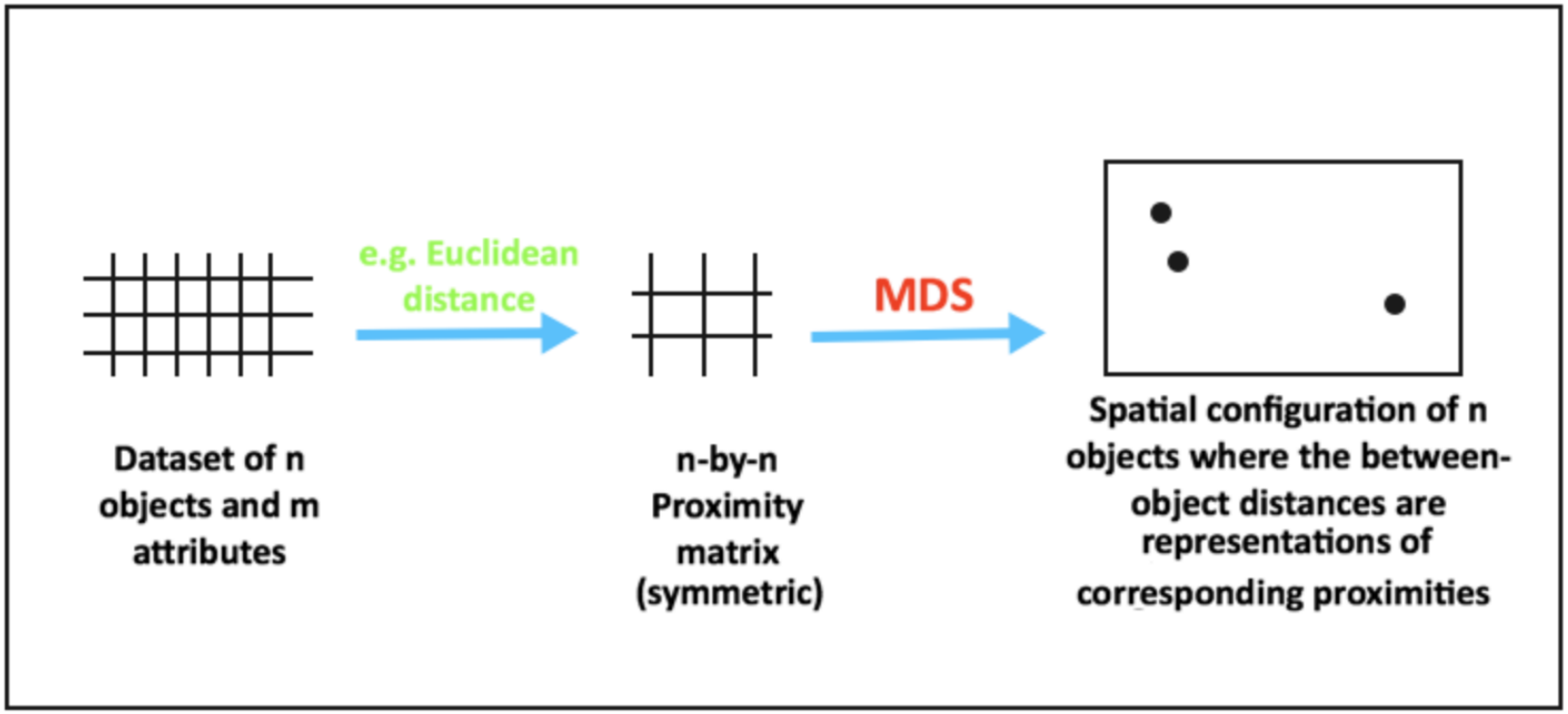
	ATL	BOS	ORD	DCA	DEN	LAX	MIA	JFK	SEA	SFO	MSY
ATL	0	934	585	542	1209	1942	605	751	2181	2139	424
BOS	934	0	853	392	1769	2601	1252	183	2492	2700	1356
ORD	585	853	0	598	918	1748	1187	720	1736	1857	830
DCA	542	392	598	0	1493	2305	922	209	2328	2442	964
DEN	1209	1769	918	1493	0	836	1723	1636	1023	951	1079
LAX	1942	2601	1748	2305	836	0	2345	2461	957	341	1679
MIA	605	1252	1187	922	1723	2345	0	1092	2733	2594	669
JFK	751	183	720	209	1636	2461	1092	0	2412	2577	1173
SEA	2181	2492	1736	2328	1023	957	2733	2412	0	681	2101
SFO	2139	2700	1857	2442	951	341	2594	2577	681	0	1925
MSY	424	1356	830	964	1079	1679	669	1173	2101	1925	0

Multidimensional scaling (MDS)

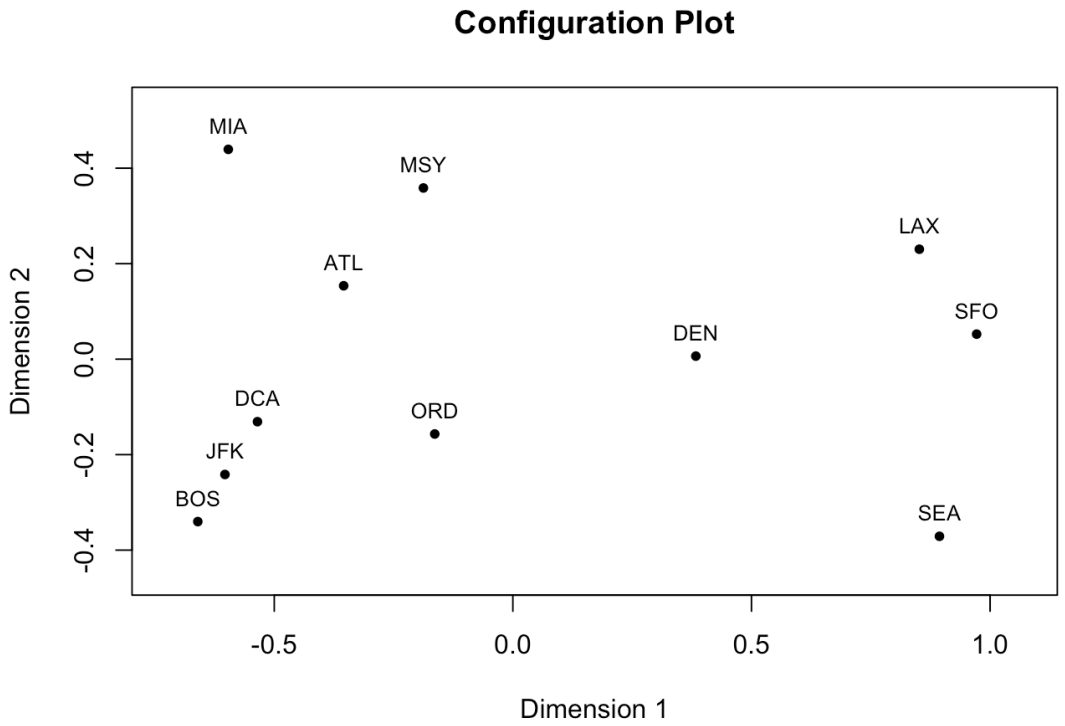
- Now we want to draw a map the major cities using the between-cities distances. It is way more difficult than the measuring distances on the map!
- This is where MDS can help.
  - MDS is able to find a configuration of the observations(here cities) in a low dimensional map(here 2d map) such that the original pair-wise between-object distances are preserved as much as possible in the map(=low dimensional projection).
  - Hence, the final map(also called configuration) is set in a way that close observations tend to be closer, and far observations locate far from each other.

- The distances are, in a broader sense, proximity scores.
  - proximity : dis/similarity scores
- How to measure the distance in general is determined by the characteristics of the data as well as the goal of the study.
  - most common distance form : Euclidean Distance

Multidimensional scaling (MDS)



Multidimensional scaling (MDS)



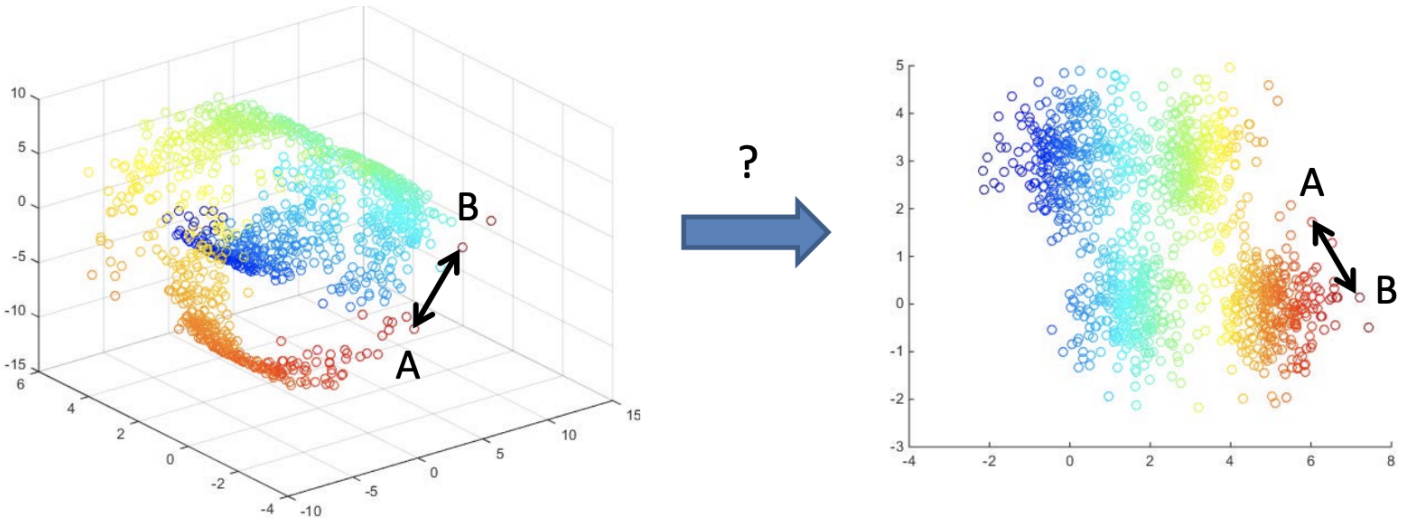
- The cities on the map are not in their expected locations.
- MDS only tries to preserve the inter-object distances

Outline

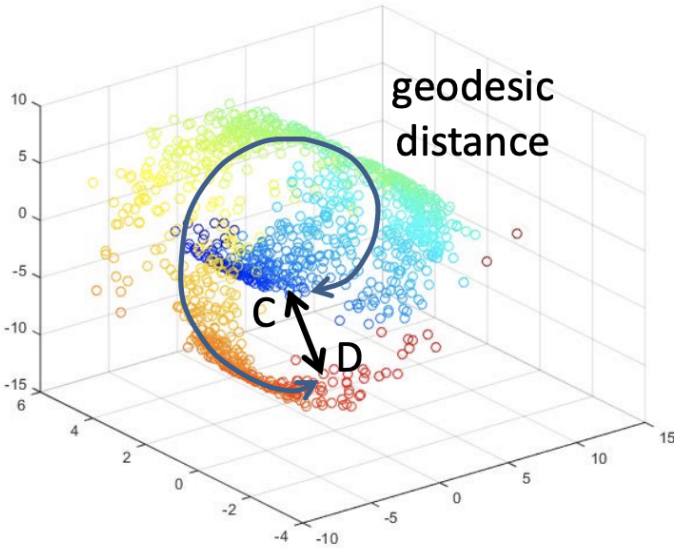
Isometric Feature Mapping (Isomap)

- The **Isometric Feature Mapping (Isomap)** algorithm combines the major algorithmic features of PCA and MDS (multi-dimensional scaling)
- 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
- problem?



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

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 extent of similarity between data points i and j.
- Replacing distances with geodesic distances simply means constructing a different similarity matrix without changing the MDS algorithm.
- a close connection between similarity matrices and graphs : basic definitions from graph theory.

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
  - Directedgraph: 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

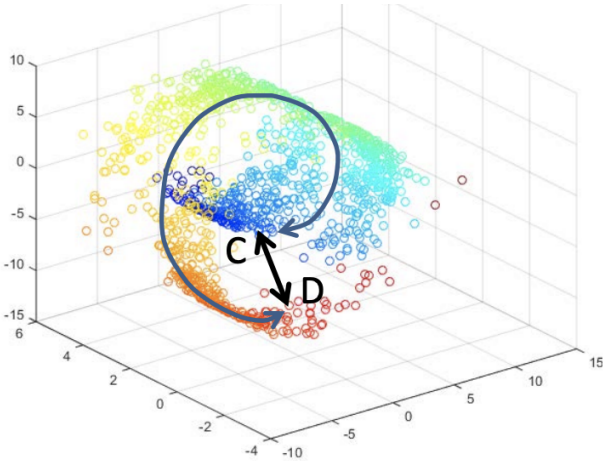
Weighted adjacency matrix

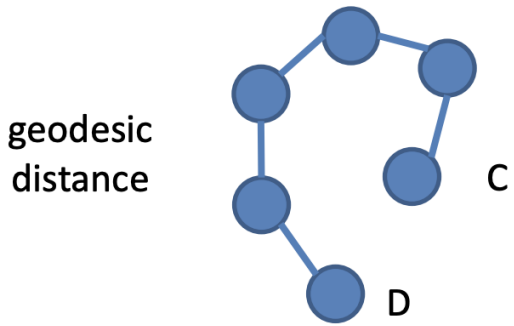
- We will consider weighted undirected graphs with non-negative weights $w_{ij} \geq 0$. Moreover, we will assume that $w_{ij} = 0$, if and only if vertices i and j are not connected
- The degree of a vertex $v_i \in V$ is defined as

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

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

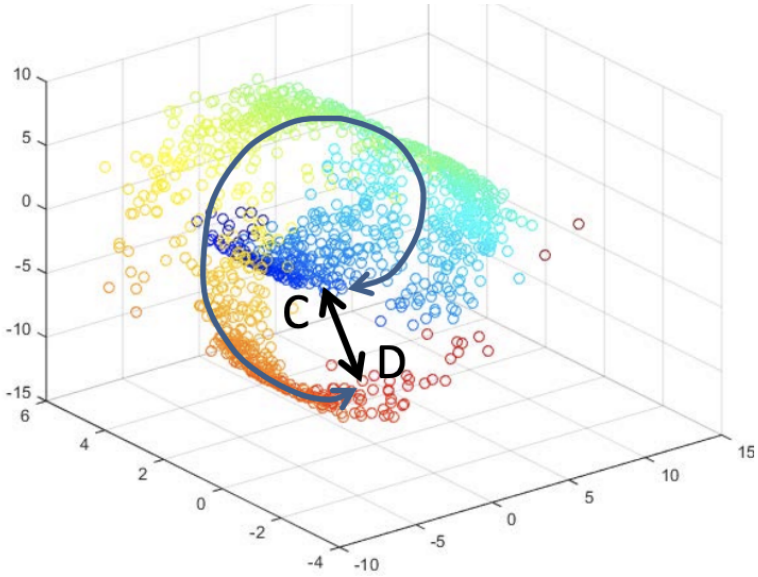
Similarity graph models data geometry

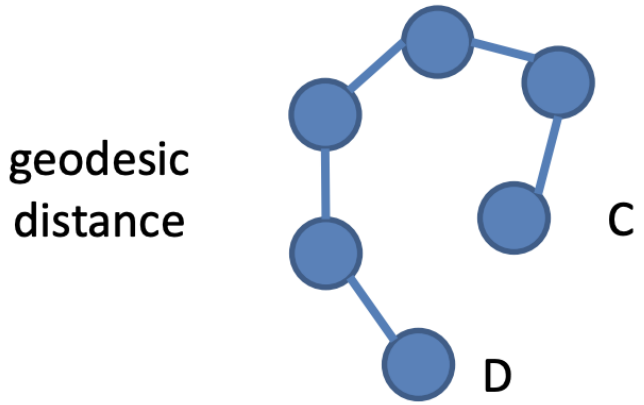




1. Geodesic distances can be approximated using a graph in which vertices represent data points
2. Let $d(i,j)$ be the Euclidean distance between the points in the original space
3. Option 1: define some local radius *epsilon*. Connect vertices i and j with an edge if $d(i,j) \leq \epsilon$.
4. Option 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.
5. Set weight for each edge to $d(i,j)$.

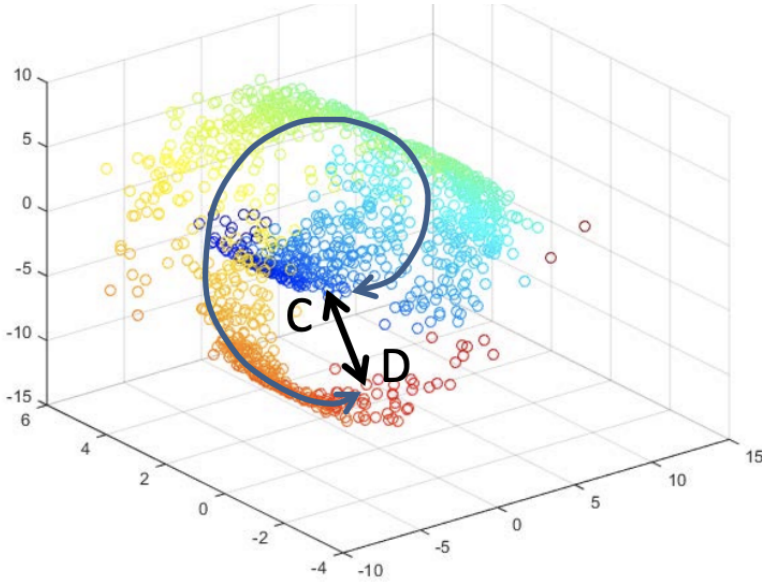
Computing geodesic distances

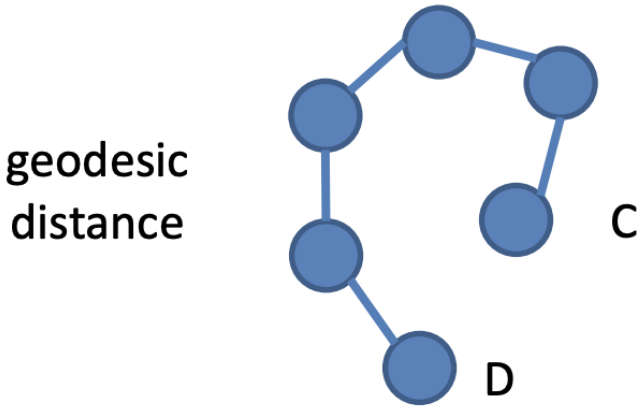




1. Given the similarity graph, compute shortest paths between each pair of points.
2. Set geodesic distance between vertices i and j to the length (sum of weights) of the shortest path between them.
3. Define a new similarity matrix based on geodesic distances.

Isomap : Summary





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

Isometric Feature Mapping

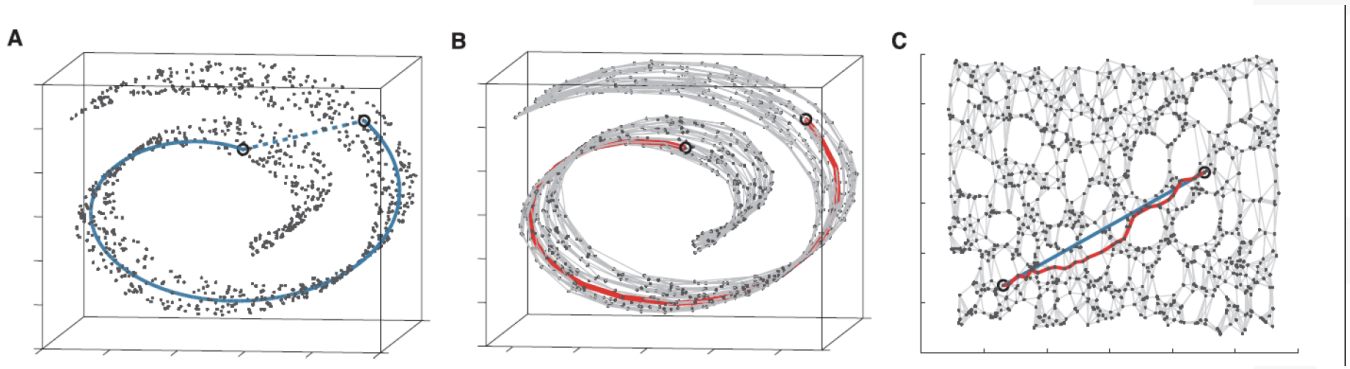


Figure 1: The "Swiss roll" data set. (A) The Euclidean distance between two points in the high-dimensional input space (length of dashed line) may not accurately reflect their intrinsic similarity, as measured by geodesic distance along the low-dimensional manifold (length of solid curve). (B) The neighborhood graph G constructed with $K = 7$ allows an approximation (red segments) to the true geodesic path with the shortest path in G. (C) The two-dimensional embedding recovered by Isomap preserves the shortest path distances in the neighborhood graph. Straight lines in the embedding (blue) now represent cleaner approximations to the true geodesic paths than do the corresponding graph paths (red).

Isometric Feature Mapping

- pros
  - A noniterative, polynomial time procedure with a guarantee of global optimality.
  - A guarantee of asymptotic convergence to the true structure for manifolds whose intrinsic geometry is that of a convex region of Euclidean space.
  - Single free parameter (ϵ or K).
- cons
  - Sensitive to noise.
  - Computationally expensive (dense matrix eigenreduction).

Outline

Locally Linear Embedding (LLE)

- The locally linear embedding (LLE) algorithm is based on simple geometric intuitions.
- Suppose that the data consist of N real-valued vectors x_i, each of dimensionality d, sampled from some underlying manifold.
- Provided there is sufficient data (such that the manifold is well-sampled), each data point and its neighbors are expected to lie on or close to a locally linear patch of the manifold.

Locally Linear Embedding(LLE)

- The local geometry of these patches is characterized by linear coefficients that reconstruct each data point from its neighbors.
- The reconstruction errors are measured by the cost function.
- Provided there is sufficient data (such that the manifold is well-sampled), each data point and its neighbors are expected to lie on or close to a locally linear patch of the manifold.

$$\epsilon(W) = \sum_i ||x_i - \sum_j W_{ij} x_j||^2$$

which adds up the squared distances between all data points and their reconstructions.

- The weights W_{ij} summarize the contribution of the j'th data point to the i'th reconstruction.

LLE

- To compute the weights W_{ij} , the cost function is minimized subject to two constraints:
  - each data point x_i is reconstructed only from its neighbors, enforcing $W_{ij} = 0$ if x_j does not belong to the set of neighbors of x_i,
  - the rows of the weight matrix sum to one: $\sum W_{ij} = 1$

- LLE constructs a neighborhood-preserving mapping based on this idea.
- In the final step of the algorithm, each high-dimensional observation x_i is mapped to a low-dimensional vector y_i representing global internal coordinates on the manifold.
- This is done by choosing d'-dimensional coordinates y_i to minimize the embedding cost function

$$\Phi(Y) = \sum_i ||y_i - \sum_j W_{ij} y_j||^2$$

Locally Linear Embedding(LLE)

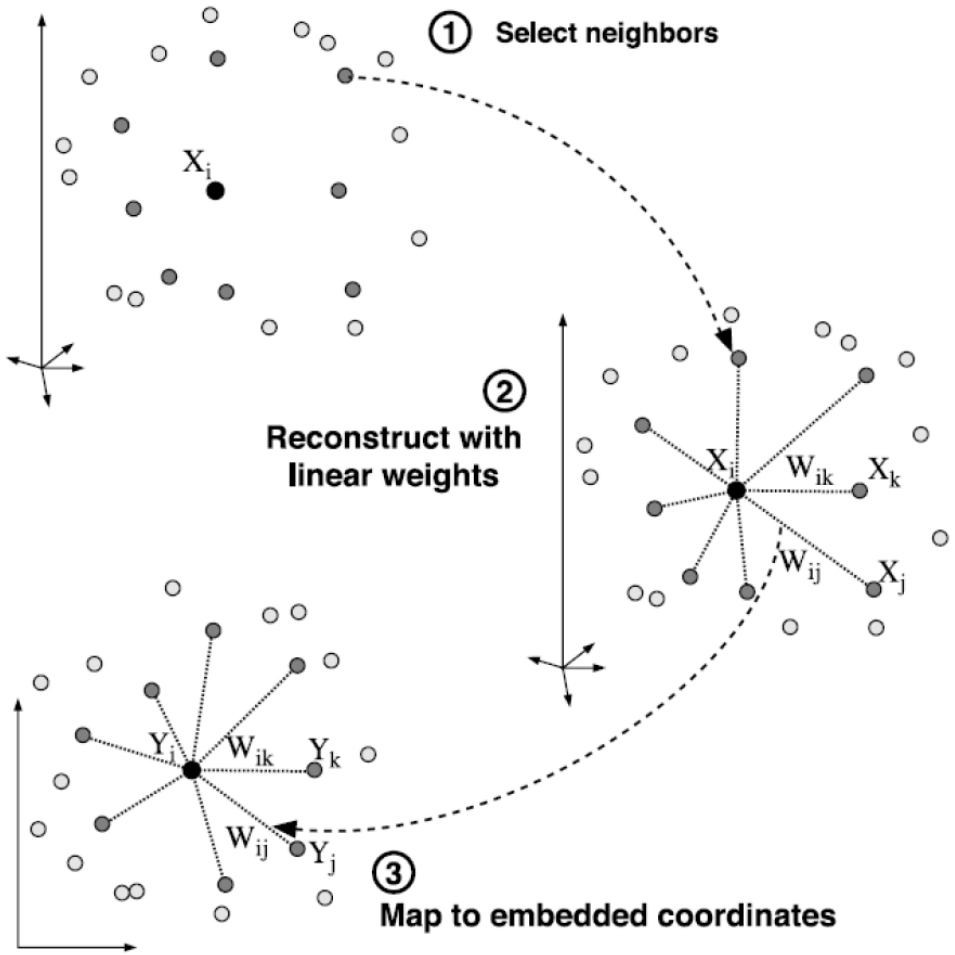


Figure 2: Steps of LLE. (1) Assign neighbors to data point x_i. (2) Compute the weights W_{ij} that best reconstruct x_i from its neighbors. (3) Compute the low-dimensional embedding vectors y_i best reconstructed by W_{ij} .

Locally Linear Embedding(LLE)

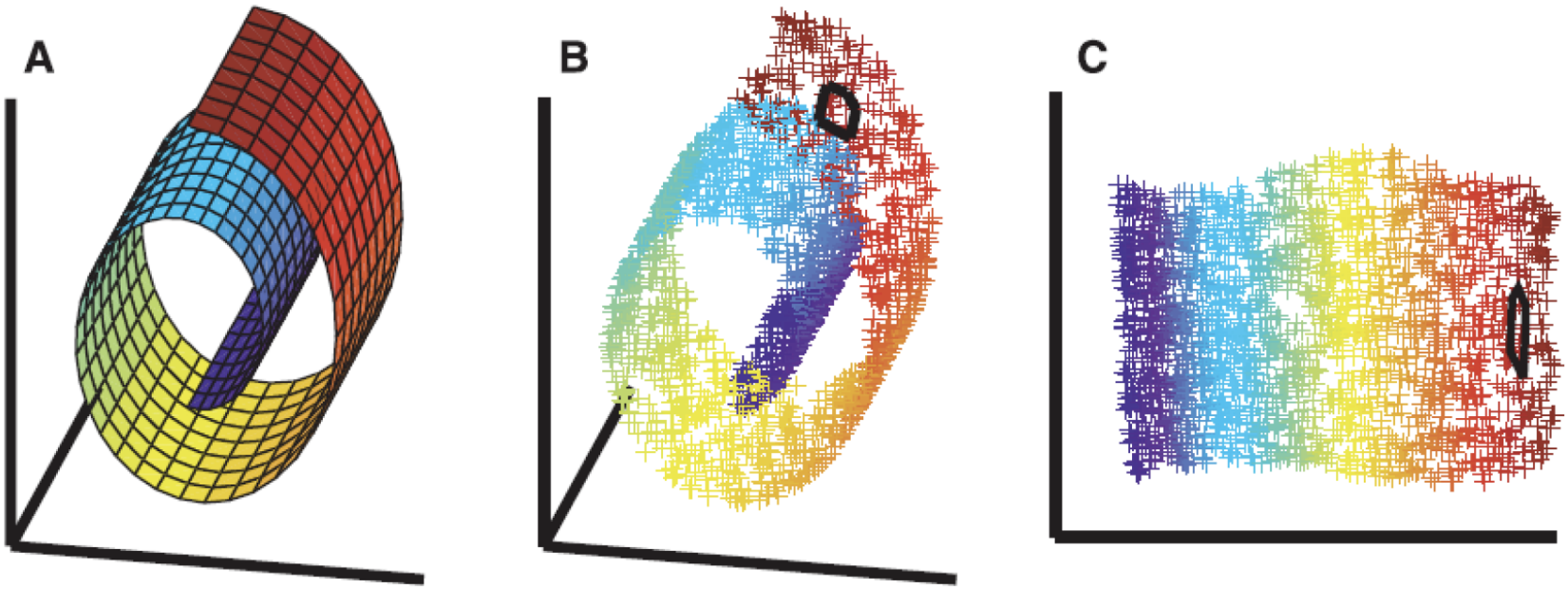


Figure 3: The "Swiss roll" data set. The color coding illustrates the neighborhood-preserving mapping discovered by LLE. Black outlines in (B) and (C) show the neighborhood of a single point.

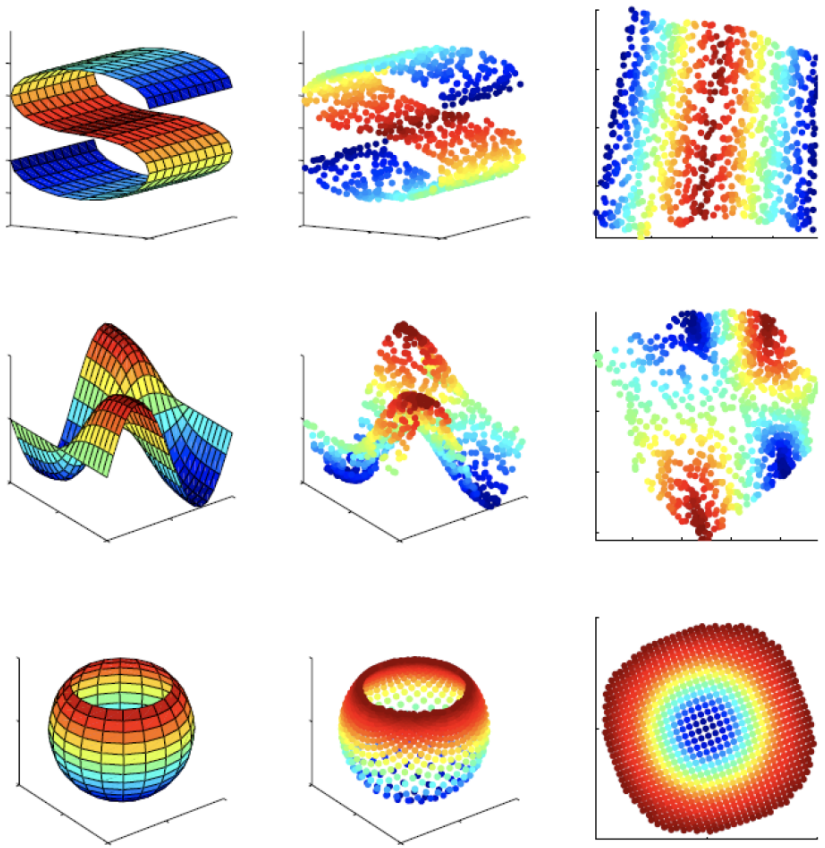


Figure 4: Other examples for three-dimensional data sampled from two dimensional manifolds.

Locally Linear Embedding(LLE)

- pros
  - Globally optimal result.
  - Single free parameter (number of neighbors, K).
  - Simple linear algebra operations using sparse matrices.
- cons
  - Sensitive to noise.
  - No theoretical guarantees.

Manifold learning code

demos

https://scikit-learn.org/stable/modules/manifold.html

Outline

t-SNE

- **t-Distributed Stochastic Neighbor Embedding (t-SNE)** is an unsupervised, non-linear technique primarily used for data exploration and visualizing high-dimensional data.
- In simpler terms, t-SNE gives you a feel or intuition of how the data is arranged in a high-dimensional space. It was developed by Laurens van der Maatens and Geoffrey Hinton in 2008.

How t-SNE works

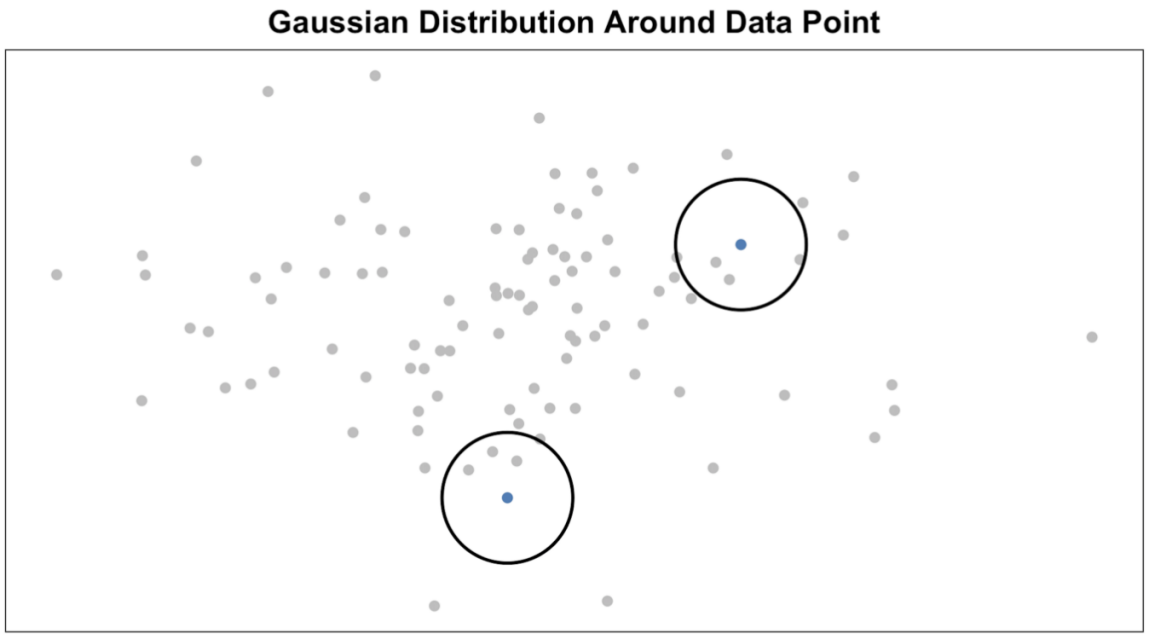
- The t-SNE algorithm calculates a **similarity** measure between pairs of instances in the high dimensional space and in the low dimensional space.
- It then tries to optimize these two similarity measures using a cost function.

How t-SNE works

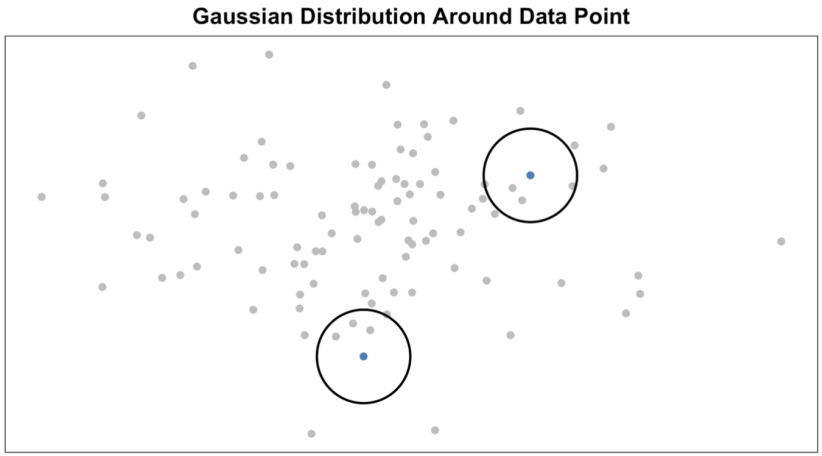
Let's break that down into 3 basic steps.

step 1

measure similarities between points in the high dimensional space.



How t-SNE works



- For each data point (x_i) we'll center a Gaussian distribution over that point.
- Then we measure the density of all points (x_j) under that Gaussian distribution. Then renormalize for all points. This gives us a set of probabilities (P_{ij}) for all points.

t-SNE

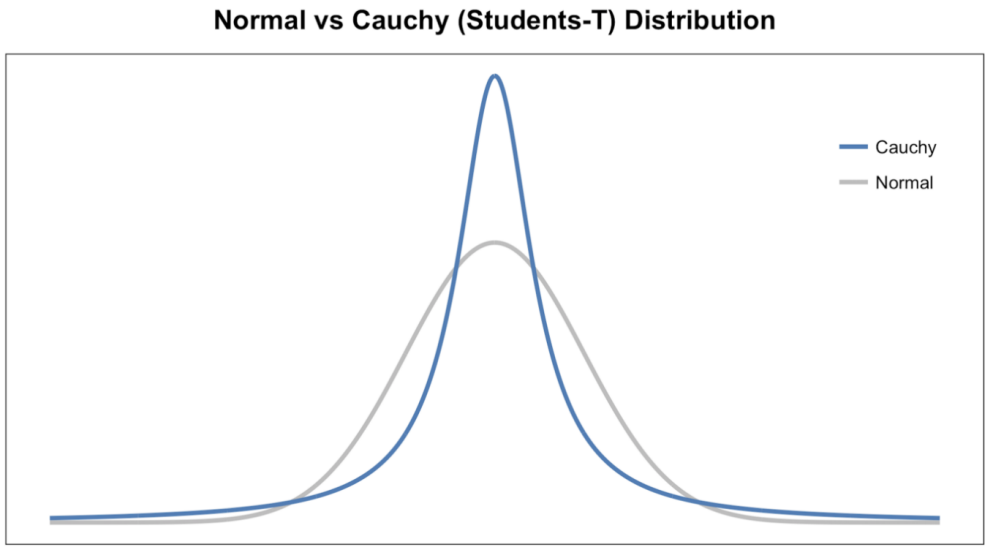
- Those probabilities are proportional to the similarities.
  - If data points x_1 and x_2 have equal values under this gaussian circle then their proportions and similarities are equal and hence you have local similarities in the structure of this high-dimensional space.
  - The Gaussian distribution or circle can be manipulated using what's called **perplexity**, which influences the variance of the distribution (circle size) and essentially the number of nearest neighbors.
    - variance depends on Gaussian and the number of points surrounding the center of it. This is the part where perplexity value comes.
    - A perplexity is more or less a target number of neighbors for our central point. Basically, the higher the perplexity is the higher value variance has.

https://apiumhub.com/tech-blog-barcelona/dimensionality-reduction-tsne/

How t-SNE works

step 2

use a Student t-distribution with one degree of freedom



This gives us a second set of probabilities (Q_{ij}) in the low dimensional space. As you can see the Student t-distribution has heavier tails than the normal distribution. The heavy tails allow for better modeling of far apart distances.

How t-SNE works

step 3

aim: the set of probabilities from the low-dimensional space (Q_{ij}) to reflect those of the high dimensional space (P_{ij}) as best as possible.

We want the two map structures to be similar.

- measure the difference between the probability distributions of the two-dimensional spaces using **Kullback-Liebler divergence (KL)** (an asymmetrical approach that efficiently compares large P_{ij} and Q_{ij} values).

$$C = KL(P||Q) = \sum_i \sum_j P_{ij} \log \frac{P_{ij}}{Q_{ij}}$$

- use gradient descent to minimize our KL cost function.

t-SNE

Steps of t-SNE algorithm:

1. compute pairwise similarity P_{ij} for every i and j.

$$P_{j|i} = \frac{exp\{-\|x_i - x_j\|\}^2/2\delta_i^2}{\sum_{k \neq i} exp\{-\|x_j - x_k\|^2/2\delta_i^2\}}$$

   define $P_{i|i} = 0$.

2. make P_{ij} symmetric.

$$P_{ij} = \frac{P_{i|j} + P_{j|i}}{2N}$$

   N is the number of samples

3. choose a random solution $Y^{(0)} = \{y_1, y_2, ..., y_n\}$.

t-SNE

4. While not done:
   - compute pairwise similairites for y_i

$$Q_{ij} = \frac{(1 + \|y_j - y_i\|^2)^{-1}}{\sum_{k \neq l}(1 + \|y_j - y_i\|^2)^{-1}}$$

     define $Q_{i|i} = 0$.
   - compute the gradient of the cost function C
   - update the solution
   - if $i >$max_iter break
   - else i = i+1

examples

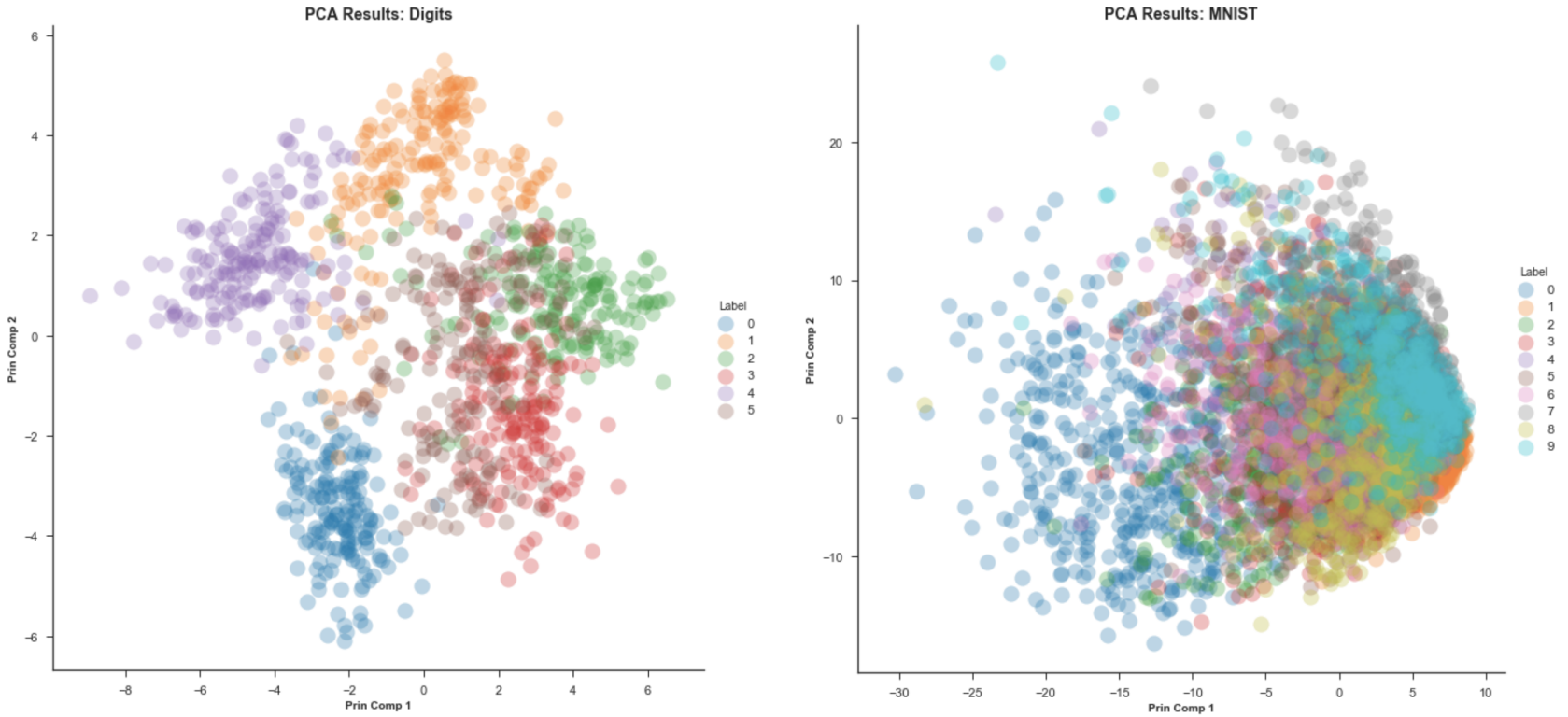


Figure 5: PCA results for both Digits and MNIST

https://towardsdatascience.com/an-introduction-to-t-sne-with-python-example-5a3a293108d1?gi=50e2b0c81d9a

examples

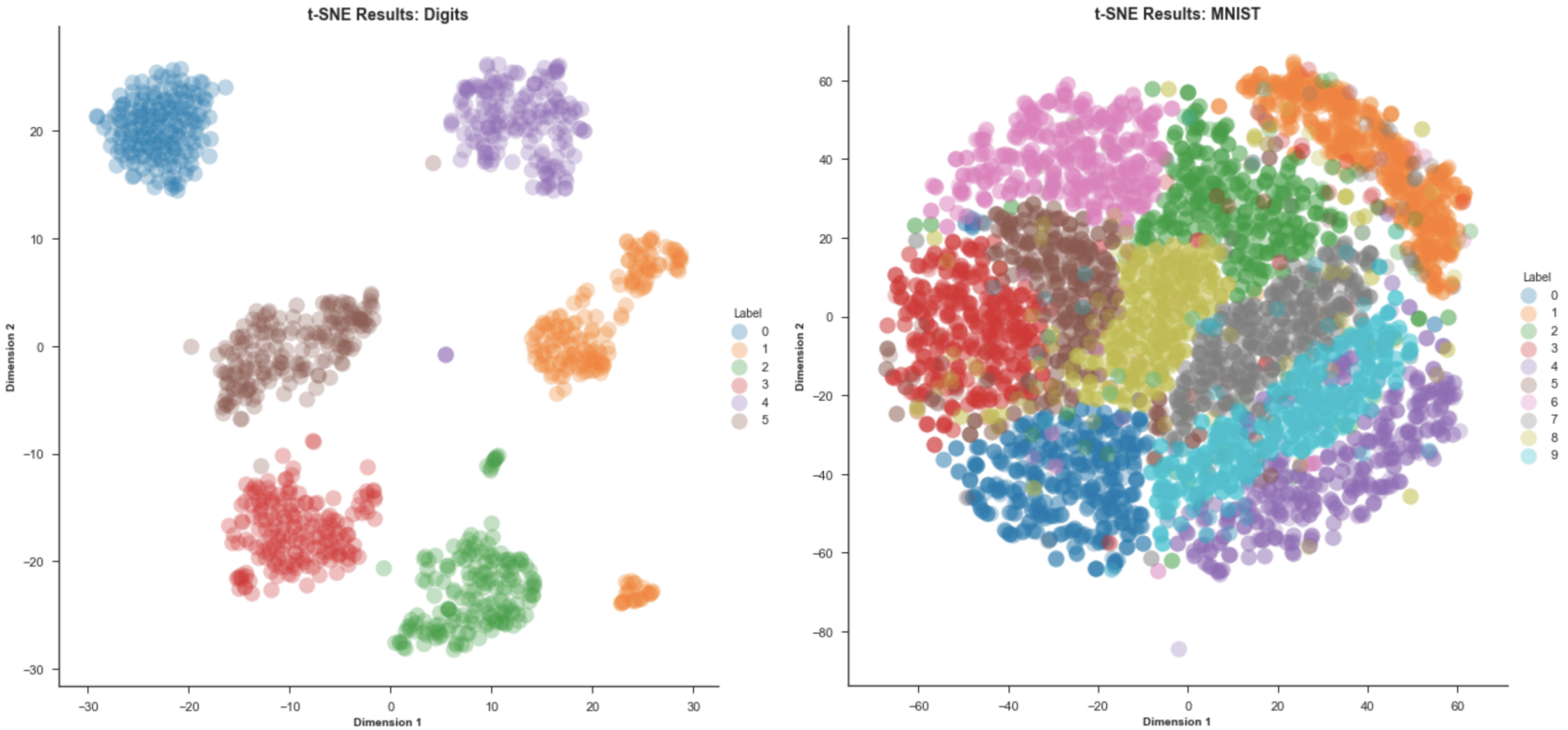


Figure 6: t-SNE algorithm results for both Digits and MNIST

Drawbacks of t-SNE

- Problems with t-SNE arise when intrinsic dimensions are higher i.e. more than 2-3 dimensions.
- t-SNE has the tendency to get stuck in local optima like other gradient descent based algorithms.
- The basic t-SNE algorithm is slow due to nearest neighbor search queries.

codes

- https://github.com/asdspal/dimRed/blob/master/tsne.ipynb
- https://github.com/asdspal/dimRed

Thank You !

Q & A