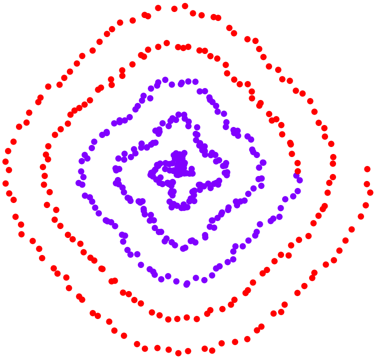
DSC 140B
Representation Learning

Lecture 13 | Part 1

Nonlinear Dimensionality Reduction

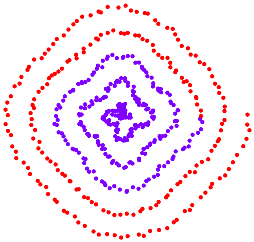
Scenario

- You want to train a classifier on this data.
- It would be easier if we could “unroll” the spiral.
- Data seems to be one-dimensional, even though in two dimensions.
- Dimensionality reduction?



PCA?

- Does PCA work here?
- Try projecting onto one principal component.



No

PCA?

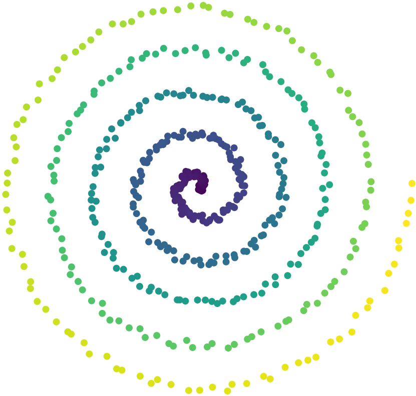
- PCA simply “rotates” the data.
- No amount of rotation will “unroll” the spiral.
- We need a fundamentally different approach that works for non-linear patterns.

Today

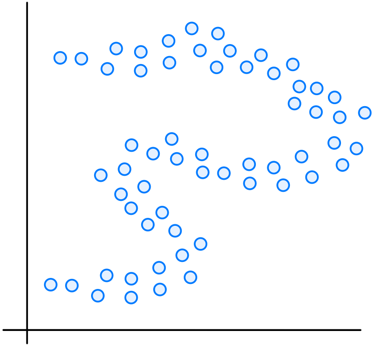
- Non-linear dimensionality reduction via **spectral embeddings**.

Rethinking Dimensionality

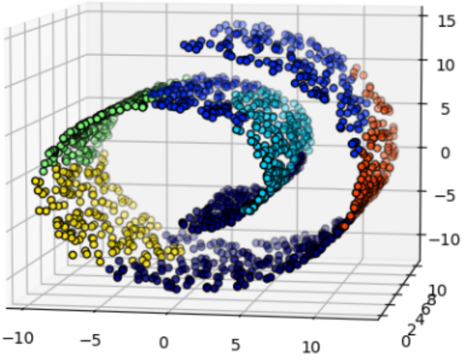
- Each point is an (x, y) coordinate in two dimensional space
- But the structure is one-dimensional
- Could (roughly) locate point using one number: distance from end.



Rethinking Dimensionality



Rethinking Dimensionality

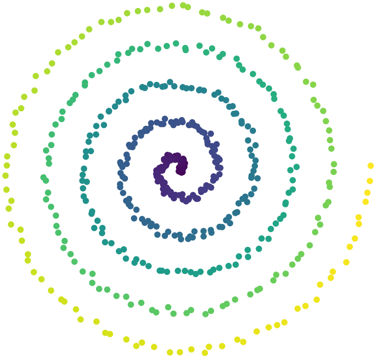


Rethinking Dimensionality

- Informally: data expressed with d dimensions, but its *really* confined to k-dimensional region
- This region is called a **manifold**
- d is the **ambient** dimension
- k is the **intrinsic** dimension

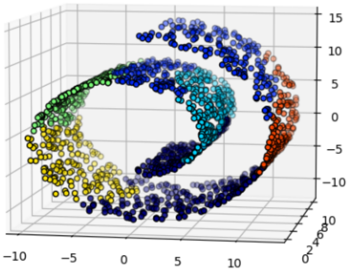
Example

- Ambient dimension: 2
- Intrinsic dimension: 1



Example

- Ambient dimension: 3
- Intrinsic dimension: 2



Example

- Ambient dimension:
- Intrinsic dimension:

Manifold Learning

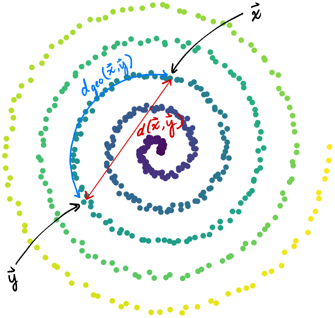
- **Given**: data in high dimensions
- **Recover**: the low-dimensional manifold

Types of Manifolds

- Manifolds can be linear
  - E.g., linear subpaces – hyperplanes
  - Learned by PCA
- Can also be non-linear (locally linear)
  - Example: the spiral data
  - Learned by **Laplacian eigenmaps**, among others

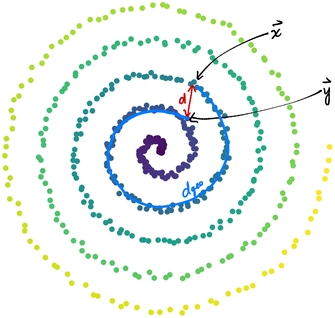
Euclidean vs. Geodesic Distances

- **Euclidean distance**: the “straight-line” distance
- **Geodesic distance**: the distance along the manifold



Euclidean vs. Geodesic Distances

- **Euclidean distance**: the "straight-line" distance
- **Geodesic distance**: the distance along the manifold



Euclidean vs. Geodesic Distances

- If data is close to a linear manifold, geodesic ≈ Euclidean

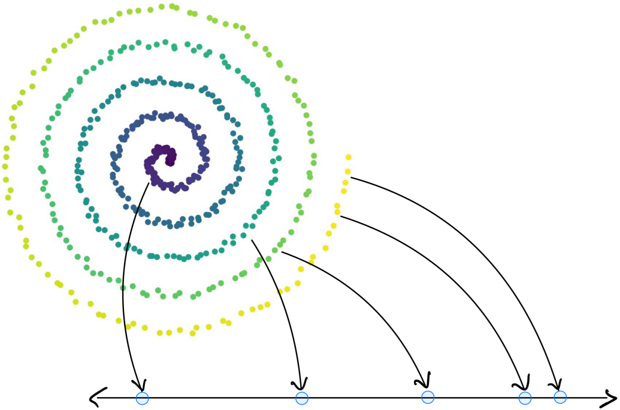
- Otherwise, can be very different

Non-Linear Dimensionality Reduction

- **Goal**: Map points in $\mathbb{R}^d$ to $\mathbb{R}^k$

- **Such that**: if $\vec{x}$ and $\vec{y}$ are close in **geodesic** distance in $\mathbb{R}^d$, they are close in **Euclidean** distance in $\mathbb{R}^k$

Embeddings



DSC 140B
Representation Learning

Lecture 13 | Part 2

Embedding Similarities

Similar Netflix Users

- Suppose you are a data scientist at Netflix
- You're given an $n \times n$ **similarity matrix** W of users
  - entry (i, j) tells you how *similar* user i and user j are
  - 1 means "very similar", 0 means "not at all"
- Goal: visualize to find patterns

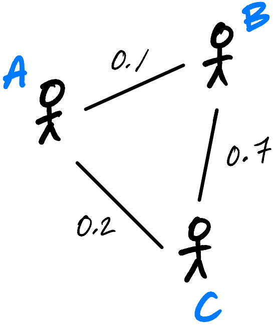
Idea

- We like scatter plots. Can we make one?
- Users are **not** vectors / points!
- They are nodes in a **similarity graph**

Similarity Graphs

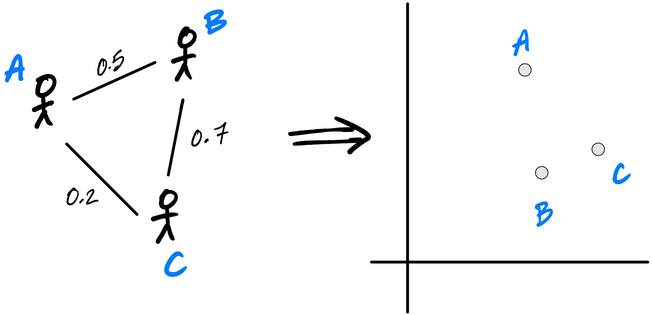
- Similarity matrices can be thought of as weighted graphs, and *vice versa*.

$$
\begin{array}{c|ccc}
 & A & B & C \\
\hline
A & 1 & 0.1 & 0.2 \\
B & 0.1 & 1 & 0.7 \\
C & 0.2 & 0.7 & 1
\end{array}
$$



Goal

- **Embed** nodes of a similarity graph as points.
- Similar nodes should map to nearby points.



Today

- We will design a graph embedding approach:
  - **Spectral embeddings** via **Laplacian eigenmaps**

More Formally

- **Given**:
  - A **similarity graph** with n nodes
  - a number of dimensions, k

- **Compute**: an **embedding** of the n points into $\mathbb{R}^k$ so that similar objects are placed nearby

To Start

- **Given**:
  - A **similarity graph** with n nodes
- **Compute**: an **embedding** of the n points into $\mathbb{R}^1$ so that similar objects are placed nearby

Vectors as Embeddings into $\mathbb{R}^1$

- Suppose we have n nodes (objects) to embed
- Assume they are numbered 1, 2, ..., n
- Let $f_1, f_2, \dots, f_n \in \mathbb{R}$ be the embeddings
- We can pack them all into a vector: $\vec{f}$.
- Goal: find a good set of embeddings, $\vec{f}$.

Example

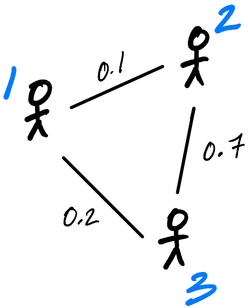
$\vec{f} = (1, 3, 2, -4)^T$

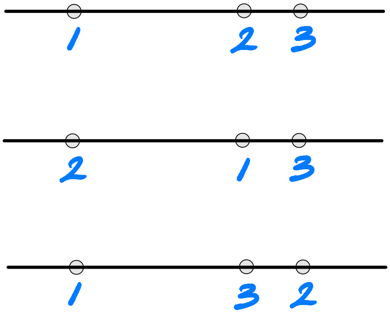
An Optimization Problem

- We'll turn it into an optimization problem:
- **Step 1**: Design a cost function quantifying how good a particular embedding $\vec{f}$ is
- **Step 2**: Minimize the cost

Example

- Which is the best embedding?





Cost Function for Embeddings

- Idea: cost is low if similar points are close
- Here is one approach:

$$\text{Cost}(\vec{f}) = \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij}(f_i - f_j)^2$$

- where w_{ij} is the weight between i and j.

Interpreting the Cost

$$\text{Cost}(\vec{f}) = \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij}(f_i - f_j)^2$$

- If $w_{ij} \approx 0$, that pair can be placed very far apart without increasing cost
- If $w_{ij} \approx 1$, the pair should be placed close together in order to have small cost.

Exercise

Do you see a problem with the cost function?

$$\text{Cost}(\vec{f}) = \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij}(f_i - f_j)^2$$

Hint: what embedding $\vec{f}$ minimizes it?

Problem

- The cost is **always** minimized by taking $\vec{f} = 0$.
- This is a "**trivial**" solution. Not useful.
- **Fix**: require $\|\vec{f}\| = 1$
  - Really, any number would work. 1 is convenient.

Exercise

Do you see **another** problem with the cost function, even if we require $\vec{f}$ to be a unit vector?

$$\text{Cost}(\vec{f}) = \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij}(f_i - f_j)^2$$

Hint: what other choice of $\vec{f}$ will **always** make this zero?

Problem

- The cost is **always** minimized by taking $\vec{f} = \frac{1}{\sqrt{n}}(1, 1, \dots, 1)^T$.
- This is a "**trivial**" solution. Again, not useful.
- **Fix**: require $\vec{f}$ to be orthogonal to $(1, 1, \dots, 1)^T$.
  - Written: $\vec{f} \perp (1, 1, \dots, 1)^T$
  - Ensures that solution is not close to trivial solution
  - Might seem strange, but it will work!

The New Optimization Problem

- **Given**: an $n \times n$ similarity matrix W

- **Compute**: embedding vector $\vec{f}$ minimizing

$$\text{Cost}(\vec{f}) = \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij}(f_i - f_j)^2$$

subject to $\|\vec{f}\| = 1$ and $\vec{f} \perp (1, 1, \ldots, 1)^T$

How?

- This looks difficult.
- Let’s write it in matrix form.
- We’ll see that it is actually (hopefully) familiar.