

DSC 140B

Representation Learning

Lecture 13 | Part 1

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

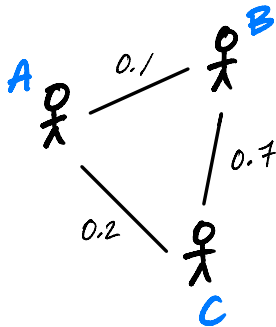
Idea

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

Similarity Graphs

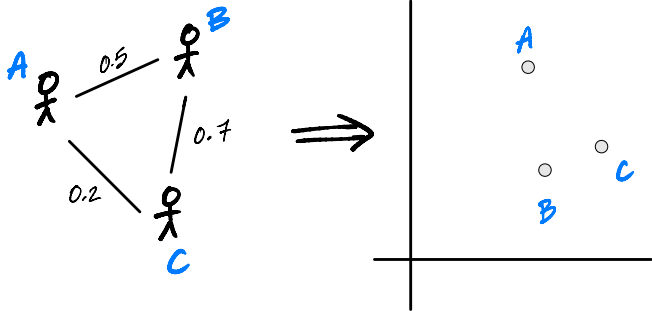
- Similarity matrices can be thought of as weighted graphs, and vice versa.

$$\begin{matrix} & \begin{matrix} A & B & C \end{matrix} \\ \begin{matrix} A \\ B \\ C \end{matrix} & \begin{pmatrix} 1 & 0.1 & 0.2 \\ 0.1 & 1 & 0.7 \\ 0.2 & 0.7 & 1 \end{pmatrix} \end{matrix}$$



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 $1, 2, \dots, n$
- ▶ 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

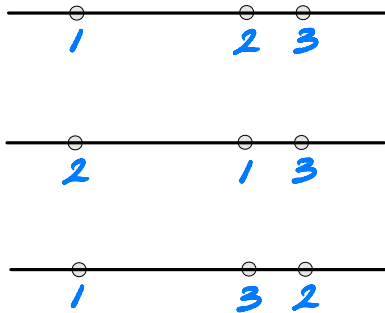
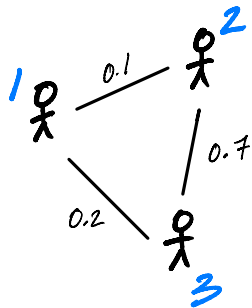
$$\vec{f} = (1, 3, 2, -4)^T$$

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 $\vec{f} \perp (1, 1, \dots, 1)^T$

How?

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

DSC 140B

Representation Learning

Lecture 13 | Part 2

The Graph Laplacian

The Problem

- **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 $\vec{f} \perp (1, 1, \dots, 1)^T$

- Now: write the cost function as a matrix expression.

The Degree Matrix

- ▶ Recall: in an unweighted graph, the degree of node i equals number of neighbors.
- ▶ Equivalently (where A is the adjacency matrix):

$$\text{degree}(i) = \sum_{j=1}^n A_{ij}$$

- ▶ Since $A_{ij} = 1$ only if j is a neighbor of i

The Degree Matrix

- ▶ In a weighted graph, define **degree** of node i similarly:

$$\text{degree}(i) = \sum_{j=1}^n w_{ij}$$

- ▶ That is, it is the total weight of all neighbors.

The Degree Matrix

- ▶ The **degree matrix** D of a weighted graph is the diagonal matrix where entry (i, i) is given by:

$$\begin{aligned}d_{ii} &= \text{degree}(i) \\ &= \sum_{j=1}^n w_{ij}\end{aligned}$$

The Graph Laplacian

- ▶ Define $L = D - W$
 - ▶ D is the degree matrix
 - ▶ W is the similarity matrix (weighted adjacency)
- ▶ L is called the **Graph Laplacian** matrix.
- ▶ It is a very useful object

Very Important Fact

► Claim:

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

► Proof: expand both sides ¹

¹Note that there was originally a $\frac{1}{2}$ in front of $\vec{f}^T L \vec{f}$, but this was not correct as written. See Problem 06 in the Midterm 02 practice for a longer explanation.

Proof