Graph Neural Networks

Lecture 4

Danila Biktimirov
Applied Computer Science
Neapolis University Pafos

4 March 2025

Overview

1. Previously on...
2. PageRank
3. Unnormalized Laplacian
4. Normalized Laplacians
5. Ratio Cut

Previously on...

Embeddings

1. Mapping to the embedding space
2. Encoder-decoder paradigm
3. Various approaches to creating embeddings
   - Inner-product methods
   - Laplacian eigenmaps
   - Random walks
   - Node2Vec
4. Embeddings for graphs

Knowledge Graphs (or Multi-Relational Graphs)

Graph - $G = (V, E)$, where V are vertices, E are edges

In a regular graph: $e = (u, v)$

In a knowledge graph: $e = (u, t, v)$

In general, knowledge graphs solve the task of predicting missing links, but there are also node classification problems.

PageRank

- An algorithm used for ranking web pages on the Internet.

For network analysis, it is a clear example of utilizing all previously obtained knowledge to create a ready-made method for solving the pressing problem of search.

What are Web Pages?

We will consider the Internet in a simplified way (as it was in the early days).

The Internet consists of pages.

Pages contain links to other pages.

The **task** is to determine which pages are the most important.

Connection with Graphs

In this simplified view, the Internet can be conveniently represented as a huge graph.

Vertices - pages

Edges - links

Thus, our graph will be directed: an edge between two vertices will indicate that page A contains a link to page B.

Characteristics of Such a Graph

- The graph is enormous.
- Loops may exist (a page links to itself).
- Edges can be bidirectional (e.g., the main page links to a subpage, which in turn links back to the main page).

Intuition

The most obvious idea: a page is important if it has many links.

Immediately, the question arises: which links are more important, **outgoing** or **incoming**?

Moreover, another question arises: are all links **equally** important?

Intuition. Model

A link from an important page should indicate the importance of a page.

Let's describe the model:

- The contribution of each link should be proportional to the importance of the page it originates from.
- If page i with importance r_i has d_i links, then each outgoing link contributes $\frac{r_i}{d_i}$.
- Page j will have an importance equal to the sum of incoming links.

Somewhat recursive...

Rank

How to solve it? The Gaussian method is not very effective when dealing with billions of pages.

$$r_j = \sum_{i \to j} \frac{r_i}{d_i}$$

Matrix Form

Stochastic Adjacency Matrix:

If there is a link from i to j, then $M_{ij} = \frac{1}{d_i}$.

Thus, the equations can be rewritten in matrix form:

$$r = Mr$$

Intuition Behind the Matrix Form

Let's try to understand the meaning of $r = Mr$.

Imagine a random user browsing the Internet:

1. At time t, they land on page i.
2. At time $t + 1$, they follow a random link from page i.
3. They arrive at page j after following a link from i (**important: j can be the same as i**).
4. This process continues indefinitely.

Let $p(t)$ be a vector where the i-th element represents the probability of the user being on page i at time t.

Thus, $p(t)$ is a probability distribution over all pages.

Intuition Behind the Matrix Form

The movement through links in a random, equal-probability manner can be described as:

$$p(t+1) = Mp(t)$$

Imagine that at some moment, the equation holds:

$$p(t+1) = Mp(t) = p(t)$$

Then $p(t)$ is the stationary distribution of random walks.

Recalling $r = Mr$, we obtain that r is the stationary distribution.

Moreover...

Let's recall centrality through eigenvalues:

$$\lambda c = Ac$$

where λ is the eigenvalue, and c is the eigenvector.

Since we have $r = Mr$, setting $\lambda = 1$ gives:

$$1r = Mr$$

Conclusion

Combining all three ideas, we get:

- r is the eigenvector of the stochastic adjacency matrix M with eigenvalue 1.
- Starting from any vector and iterating $M(M(...(Mu)))$, we get the long-term distribution of random walkers.
- PageRank = Limiting Distribution = Principal Eigenvector of M.

Now we can solve the problem.

PageRank Solution

1. Initially, assign an initial rank to each node.
2. Continue until convergence (when the norm of the difference between ranks at times $t+1$ and t is smaller than a certain ϵ).

Power Iteration

- An algorithm to solve the problem:

1. Initialization: $r^0 = [1/N, \ldots, 1/N]$.
2. Update step: $r^{t+1} = Mr^t$.
3. Repeat until $\|r^{t+1} - r^t\|_1 < \varepsilon$.
   - $r := r^{t+1}$.

Problems?

What Questions Arise About PageRank?

- Dead ends (break the algorithm).
- Cycles (convergence happens, but results may not be accurate).

Escaping Cycles

Using probabilities, we can learn to escape cycles:

- At each step, with probability β, the user selects one of the d_i links on the page.
- With probability $1 - \beta$, the user teleports.

In a finite number of steps, the user will escape the cycle.

Escaping Dead Ends

- We predefine that in a dead end, a random teleport will occur.

PageRank

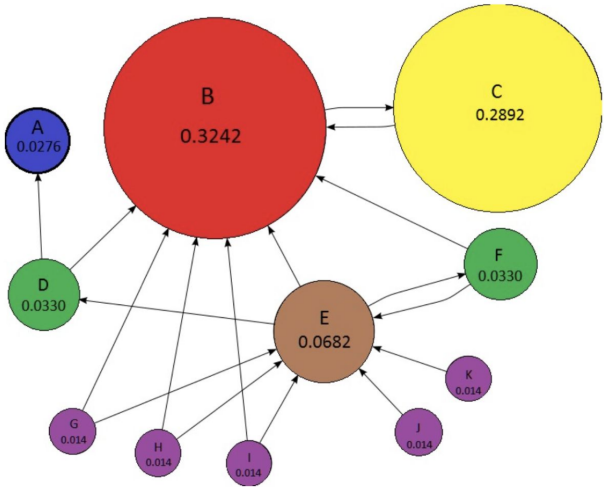
$$r_j = \sum_{i \to j} \beta \frac{r_i}{d_i} + (1 - \beta) \frac{1}{N}$$

New Matrix

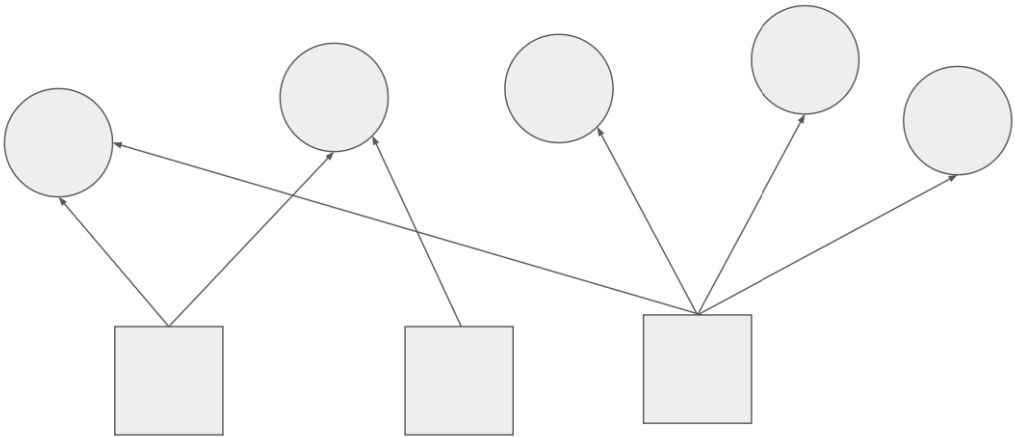
$$G = \beta M + (1 - \beta) \left[\frac{1}{N} \right]_{N \times N}$$

Thus, we obtain the equation $r = Gr$.

PageRank Example



Recommendations



Application to Recommendations

- We modify the teleportation format by limiting the set of nodes where a reverse jump can occur.
- The most similar element can be found by restricting the teleportation set to a single specific node.

Unnormalized Laplacian

The most basic Laplacian matrix is the unnormalized Laplacian, defined as follows:

$$\mathbf{L} = \mathbf{D} - \mathbf{A}$$

Laplacian Matrix Properties

- **Symmetric and Positive Semi-Definite:**

$$\mathbf{L}^T = \mathbf{L}, \quad \mathbf{x}^T\mathbf{L}\mathbf{x} \geq 0, \quad \forall \mathbf{x} \in \mathbb{R}^{|\mathcal{V}|}$$

- **Vector Identity:**

$$\mathbf{x}^T\mathbf{L}\mathbf{x} = \sum_{(u,v)\in\mathcal{E}} (\mathbf{x}[u] - \mathbf{x}[v])^2$$

- **Eigenvalues:** $\mathbf{L}$ has $|\mathcal{V}|$ non-negative eigenvalues:

$$0 = \lambda_{|\mathcal{V}|} \leq \lambda_{|\mathcal{V}|-1} \leq \cdots \leq \lambda_1$$

Laplacian and Connected Components

Theorem: The geometric multiplicity of the 0 eigenvalue of $\mathbf{L}$ equals the number of connected components in the graph.

Proof:

- Let $\mathbf{e}$ be an eigenvector corresponding to eigenvalue 0, then:

$$\mathbf{e}^T \mathbf{L} \mathbf{e} = 0$$

- By the definition of $\mathbf{L}$, this implies:

$$\sum_{(u,v)\in\mathcal{E}} (\mathbf{e}[u] - \mathbf{e}[v])^2 = 0.$$

- Since each term in the sum is non-negative, every term must be zero.
- This means $\mathbf{e}[u] = \mathbf{e}[v]$ for all $(u, v) \in \mathcal{E}$, so $\mathbf{e}$ is constant within each connected component.

Proof (Continued)

Block Structure of L:

$$\mathbf{L} = \begin{bmatrix} \mathbf{L}_1 & & & \\ & \mathbf{L}_2 & & \\ & & \ddots & \\ & & & \mathbf{L}_K \end{bmatrix}$$

- Each $\mathbf{L}_k$ corresponds to a connected component.
- Each block $\mathbf{L}_k$ has exactly one eigenvalue 0, with eigenvector $\mathbf{1}_k$ (a constant vector over that component).
- Since $\mathbf{L}$ is block diagonal, its eigenvalues are the union of the eigenvalues of all $\mathbf{L}_k$.
- Thus, the number of zero eigenvalues of $\mathbf{L}$ is exactly K, the number of connected components.

The geometric multiplicity of the 0 eigenvalue of $\mathbf{L}$ equals the number of connected components. □

Normalized Laplacians

Definition: Two common normalized variants of the Laplacian are:

Symmetric Normalized Laplacian:

$$\mathbf{L}_{\text{sym}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}}$$

Random Walk Laplacian:

$$\mathbf{L}_{\text{RW}} = \mathbf{D}^{-1}\mathbf{L}$$

Graph Cuts and Clustering

Definition: A *cut* in a graph partitions its nodes into disjoint subsets.

Let $\mathcal{A} \subset \mathcal{V}$ be a subset of nodes, and let $\bar{\mathcal{A}}$ be its complement:

$$\mathcal{A} \cup \bar{\mathcal{A}} = \mathcal{V}, \quad \mathcal{A} \cap \bar{\mathcal{A}} = \emptyset.$$

Given a partition of the graph into K non-overlapping subsets $\mathcal{A}_1, \ldots, \mathcal{A}_K$, the **cut value** of this partition is defined as:

$$\text{cut}(\mathcal{A}_1, \ldots, \mathcal{A}_K) = \frac{1}{2} \sum_{k=1}^{K} \left| (u, v) \in \mathcal{E} : u \in \mathcal{A}_k, v \in \bar{\mathcal{A}}_k \right|.$$

Ratio Cut

$$\text{RatioCut}(\mathcal{A}_1, \ldots, \mathcal{A}_K) = \frac{1}{2} \sum_{k=1}^{K} \frac{\left|(u, v) \in \mathcal{E} : u \in \mathcal{A}_k, v \in \bar{\mathcal{A}}_k\right|}{|\mathcal{A}_k|}$$

Normalized Cut (NCut)

$$\text{NCut}(\mathcal{A}_1, \ldots, \mathcal{A}_K) = \frac{1}{2} \sum_{k=1}^{K} \frac{\left|(u, v) \in \mathcal{E} : u \in \mathcal{A}_k, v \in \bar{\mathcal{A}}_k\right|}{\text{vol}(\mathcal{A}_k)}$$

where:

$$\text{vol}(\mathcal{A}) = \sum_{u \in \mathcal{A}} d_u$$

Spectral Clustering via Ratio Cut

Goal: Find a cluster assignment that minimizes the Ratio Cut using the Laplacian spectrum.

Optimization Problem:

$$\min_{\mathcal{A} \in \mathcal{V}} \text{RatioCut}(\mathcal{A}, \bar{\mathcal{A}})$$

Vector Representation for Ratio Cut

Let $\mathbf{a} \in \mathbb{R}^{|\mathcal{V}|}$ be defined as:

$$\mathbf{a}[u] = \begin{cases} \sqrt{\frac{|\bar{\mathcal{A}}|}{|\mathcal{A}|}} & \text{if } u \in \mathcal{A} \\ -\sqrt{\frac{|\mathcal{A}|}{|\bar{\mathcal{A}}|}} & \text{if } u \in \bar{\mathcal{A}} \end{cases}$$

Reformulating Ratio Cut using the Laplacian

$$\mathbf{a}^T \mathbf{L}\mathbf{a} = \sum_{(u,v)\in\mathcal{E}} (\mathbf{a}[u] - \mathbf{a}[v])^2 =$$

$$= \sum_{(u,v)\in\mathcal{E}:u\in\mathcal{A},v\in\bar{\mathcal{A}}} (\mathbf{a}[u] - \mathbf{a}[v])^2 =$$

$$= \sum_{(u,v)\in\mathcal{E}:u\in\mathcal{A},v\in\bar{\mathcal{A}}} \left(\sqrt{\frac{|\bar{\mathcal{A}}|}{|\mathcal{A}|}} - \left(-\sqrt{\frac{|\mathcal{A}|}{|\bar{\mathcal{A}}|}}\right)\right)^2 =$$

$$= \mathrm{cut}(\mathcal{A},\bar{\mathcal{A}}) \left(\frac{|\mathcal{A}|}{|\bar{\mathcal{A}}|} + \frac{|\bar{\mathcal{A}}|}{|\mathcal{A}|} + 2\right) =$$

$$= \mathrm{cut}(\mathcal{A},\bar{\mathcal{A}}) \left(\frac{|\mathcal{A}| + |\bar{\mathcal{A}}|}{|\bar{\mathcal{A}}|} + \frac{|\mathcal{A}| + |\bar{\mathcal{A}}|}{|\mathcal{A}|}\right) =$$

$$= |\mathcal{V}|\mathrm{RatioCut}(\mathcal{A},\bar{\mathcal{A}})$$

Properties of the Vector a

Zero Mean

$$\sum_{u \in \mathcal{V}} \mathbf{a}[u] = 0, \quad \text{or equivalently,} \quad \mathbf{a} \perp 1$$

Normalization

$$\|\mathbf{a}\|^2 = |\mathcal{V}|$$

Rewriting the Ratio Cut Minimization Problem

Optimization Reformulation:

$$\min_{\mathcal{A} \subset \mathcal{V}} \mathbf{a}^T \mathbf{L} \mathbf{a}$$

Subject to:

$$\mathbf{a} \perp 1$$

$$\|\mathbf{a}\|^2 = |\mathcal{V}|$$

$\mathbf{a}$ defined as above

Spectral Relaxation of Ratio Cut

Relaxed Optimization Problem:

$$\min_{\mathbf{a}\in\mathbb{R}^{|\mathcal{V}|}} \mathbf{a}^T\mathbf{L}\mathbf{a}$$

Subject to:

$$\begin{aligned} \mathbf{a} \perp 1 \quad &\text{(Orthogonality Condition)} \\ \|\mathbf{a}\|^2 = |\mathcal{V}| \quad &\text{(Normalization Condition)} \end{aligned}$$

From Eigenvector to Clusters:

- We obtain a discrete partition by thresholding the eigenvector:

$$\begin{cases} u \in \mathcal{A}, & \text{if } \mathbf{a}[u] \geq 0 \\ u \in \bar{\mathcal{A}}, & \text{if } \mathbf{a}[u] < 0 \end{cases}$$

Generalized Spectral Clustering

Extending Spectral Clustering to K Clusters:

- Previously, we used the second-smallest eigenvector of $\mathbf{L}$ to partition the graph into two clusters.
- This idea generalizes to K clusters using the K smallest eigenvectors.

Algorithm Steps:

1. Find the K smallest eigenvectors of $\mathbf{L}$ (excluding the smallest):

$$\mathbf{e}_{|\mathcal{V}|-1}, \mathbf{e}_{|\mathcal{V}|-2}, \ldots, \mathbf{e}_{|\mathcal{V}|-K}$$

2. Construct the matrix $\mathbf{U} \in \mathbb{R}^{|\mathcal{V}| \times (K-1)}$ with these eigenvectors as columns.
3. Represent each node u using its corresponding row in $\mathbf{U}$:

$$\mathbf{z}_u = \mathbf{U}[u], \quad \forall u \in \mathcal{V}$$

4. Run K-means clustering on the embeddings $\mathbf{z}_u$ to obtain the final cluster assignments.

Results

Method	Decoder	Similarity measure	Loss function
Lap. Eigenmaps	$\|\mathbf{z}_u - \mathbf{z}_v\|_2^2$	general	$\text{DEC}(\mathbf{z}_u, \mathbf{z}_v) \cdot \mathbf{S}[u, v]$
Graph Fact.	$\mathbf{z}_u^\top \mathbf{z}_v$	$\mathbf{A}[u, v]$	$\|\text{DEC}(\mathbf{z}_u, \mathbf{z}_v) - \mathbf{S}[u, v]\|_2^2$
GraRep	$\mathbf{z}_u^\top \mathbf{z}_v$	$\mathbf{A}[u, v], \ldots, \mathbf{A}^k[u, v]$	$\|\text{DEC}(\mathbf{z}_u, \mathbf{z}_v) - \mathbf{S}[u, v]\|_2^2$
HOPE	$\mathbf{z}_u^\top \mathbf{z}_v$	general	$\|\text{DEC}(\mathbf{z}_u, \mathbf{z}_v) - \mathbf{S}[u, v]\|_2^2$
DeepWalk	$\frac{e^{\mathbf{z}_u^\top \mathbf{z}_v}}{\sum_{k \in V} e^{\mathbf{z}_u^\top \mathbf{z}_k}}$	$p_G(v\|u)$	$-\mathbf{S}[u, v] \log(\text{DEC}(\mathbf{z}_u, \mathbf{z}_v))$
node2vec	$\frac{e^{\mathbf{z}_u^\top \mathbf{z}_v}}{\sum_{k \in V} e^{\mathbf{z}_u^\top \mathbf{z}_k}}$	$p_G(v\|u)$ (biased)	$-\mathbf{S}[u, v] \log(\text{DEC}(\mathbf{z}_u, \mathbf{z}_v))$

The End?