AI506: Data Mining and Search (Spring 2023)

Homework 3: PageRank on Hypergraphs

Release: May 5, 2023,
Due: May 19, 2023, 11:59pm

1 Introduction

In this assignment, you will implement the PageRank algorithm on hypergraphs. While graphs are widely used to represent pairwise interactions, hypergraphs provide a generalization of graphs and are natural representations of group interactions. Specifically, a hypergraph $H = (V, E)$ consists of a set of nodes V and a set of hyperedges $E \in 2^V$ where a hyperedge $e \in E$ is a non-empty subset of V.

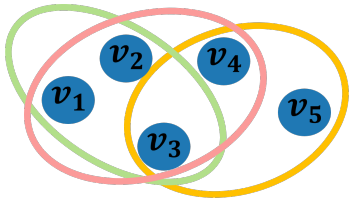

Figure 1: An example of a hypergraph with 5 nodes and 3 hyperedges.

You will first implement a naive extension of PageRank on hypergraphs. From the perspective of a random web surfer, the surfer moves between nodes through hyperedges. At any given time, the surfer is on a node (let's call it node i). To move to the next node, the surfer chooses one of the hyperedges that include node i at random (let's call it hyperedge e). The surfer then moves to a random node j within the hyperedge e.

In the next step, you will implement an advanced extension with edge-dependent node weights. In other words, the surfer selects the next node in the hyperedge differently depending on the weight of the node in the hyperedge. More specifically, a hypergraph $H = (V, E, \omega, \gamma)$ with edge-dependent node weights is a set of vertices V, a set of hyperedges $E \in 2^V$, a weight $\omega(e)$ for each hyperedge $e \in E$, and a weight $\gamma_e(v)$ for each nodes depending on hyperedge e. Instead of selecting a node in the hyperedge uniformly at random (with probability of $1/|e|$ where $|e|$ is the size of hyperedge e), the surfer selects one node v in the hyperedge e based on the edge-dependent weights $\gamma_e(v)$. The details can be found in Section 3.

2 Naive Extension on Hypergraphs (40 Points)

In a naive extension of PageRank on hypergraphs, the nodes in each hyperedge are considered equally important. The probability of the surfer moving from node u to node v is as follows:

1. Pick a hyperedge e containing u with probability $1/d(u)$, where $d(u)$ is the degree of node u, i.e., the number of hyperedges containing u.
2. Move to a node v in the hyperedge e with probability $1/|e|$, where $|e|$ is the size of hyperedge e.

Additionally, for the teleport operation, the surfer invokes the above random walk with probability `damping_factor`, and jumps from a node to any other node with probability (1 - `damping_factor`). To sum up, the one-step update of PageRank score of node v is $r_v = \sum_{e \in E(v)} \sum_{u \in e} \beta \frac{1}{|e|} \frac{r_u}{d(u)} + (1-\beta)\frac{1}{|V|}$ where $|V|$ is the number of nodes.

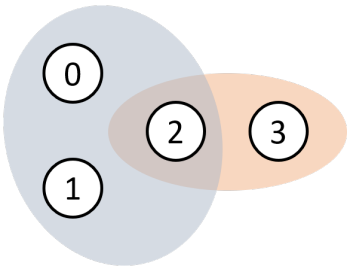

Figure 2: A toy example of the naive extension.

In **Step 1** of `hw3.ipynb`, you are required to do the following:

1. Calculate the PageRank scores of all nodes after a one-step update in the toy example shown in Figure 2 manually and store the results in `step1_toy_example_result`. Specifically, initialize PageRank scores of all nodes as $1/|V|$ and update the score of some node v by $r_v = \sum_{e \in E(v)} \sum_{u \in e} \beta \frac{1}{|e|} \frac{r_u}{d(u)} + (1-\beta)\frac{1}{|V|}$.
2. Implement the one-step update of PageRank by computing the transition probability accurately in `run_pagerank_one_step`. To be specific, one-step update indicates the one iteration of the power iteration algorithm.
3. Iteratively update the PageRank scores until convergence is achieved (i.e., the difference becomes smaller than $\epsilon = 1e-6$) in `run_pagerank`.

3 Advanced Extension with Edge-dependent node Weights (35 Points)

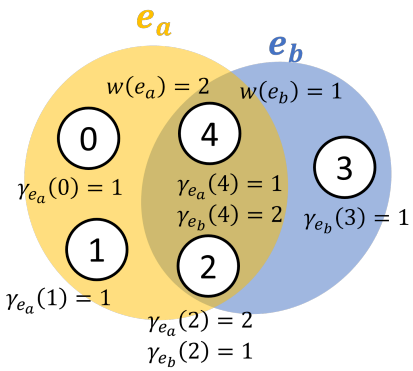

Figure 3: An example of a hypergraph with edge-dependent node weights.

In an advanced extension of PageRank for hypergraphs, we consider a hypergraph with edge-dependent node weights, $H = (V, E, \omega, \gamma)$, consisting of a set of vertices V, a set of hyperedges $E \in 2^V$, a weight $\omega(e)$ for each hyperedge $e \in E$, and a weight $\gamma_e(v)$ for each node depending on hyperedge e. Note that in the previous naive extension, we considered $\omega(e) = 1$ for every hyperedge and $\gamma_e(v) = 1$ for every node and hyperedge.

Specifically, The probability of the surfer moving from node u to node v is as follows. We use $E(v) = \{e \in E : v \in e\}$ to denote the set of hyperedges containing a node v, $d(v) = \sum_{e \in E(v)} \omega(e)$ to indicate the degree of a node v, and $\delta(e) = \sum_{v \in e} \gamma_e(v)$ to denote the degree of hyperedge e.

1. Pick a hyperedge e containing u with probability $\omega(e)/d(u)$.
2. Move to a node v in the hyperedge e with probability $\gamma_e(v)/\delta(e)$.

As in the naive extension, the teleport operation involves invoking the random walk above with probability `damping_factor`, and jumping from a node to any other node with probability (1 - `damping_factor`). To sum up, the one-step update of PageRank score of node v is $r_v = \sum_{e \in E(v)} \sum_{u \in e} \beta \frac{\gamma_e(v)}{\delta(e)} \frac{\omega(e) \cdot r_u}{d(u)} + (1 - \beta) \frac{1}{|V|}$.

For example, in the hypergraph in Figure3, the probabilities of the surfer moving (1) from node 1 to node 4 and (2) from node 2 to node 4 are computed as follows:

- Node 1 $\rightarrow$ Node 4: The surfer chooses the hyperedge e_a with probability of $\frac{\omega(e_a)}{d(1)} = \frac{\omega(e_a)}{\omega(e_a)} = 1$. Then, the surfer chooses the node 4 with probability of $\frac{\gamma_{e_a}(4)}{\delta(e_a)} = \frac{1}{1+1+2+1} = \frac{1}{5}$.
- Node 2 $\rightarrow$ Node 4: The surfer chooses the hyperedge e_b with probability of $\frac{\omega(e_b)}{d(2)} = \frac{\omega(e_b)}{\omega(e_a)+\omega(e_b)} = \frac{1}{3}$. Then, the surfer chooses the node 4 with probability of $\frac{\gamma_{e_b}(4)}{\delta(e_b)} = \frac{2}{1+1+2} = \frac{1}{4}$.

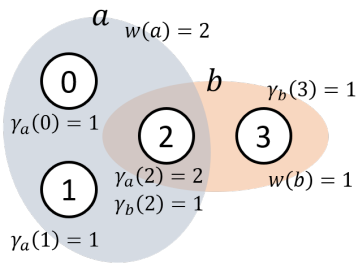

Figure 4: A toy example of the advanced extension.

In **Step 2** of hw3.ipynb, you are required to do the following:

1. Calculate the PageRank scores of all nodes after a one-step update in the toy example shown in Figure 4 manually and store the results in `step2_toy_example_result`. Specifically, initialize PageRank scores of all nodes as $1/|V|$ and update the score of some node v by $r_v = \sum_{e \in E(v)} \sum_{u \in e} \beta \frac{\gamma_e(v)}{\delta(e)} \frac{\omega(e) \cdot r_u}{d(u)} + (1-\beta)\frac{1}{|V|}$.
2. Implement the one-step update of PageRank by computing the transition probability accurately in `run_advanced_pagerank_one_step`. To be specific, one-step update indicates the one iteration of the power iteration algorithm.
3. Extend `run_pagerank` function to consider the iterative update of the PageRank scores until convergence with `use_weight=True` setting. The original implementation for the naive extension should still work with the `use_weight=False` setting.

4 Case Studies on a Collaboration Hypergraph (25 Points)

In this section, you will analyze the results of two algorithms on a co-authorship hypergraph. The hypergraph represents a collection of $10,000$ papers with $17,053$ co-authors. Each hyperedge represents a paper, and the nodes in the hyperedge represent the co-authors of the paper. The weight $\omega(e)$ of each hyperedge e is set to the number of citations of the corresponding paper plus 1, and the weight $\gamma_e(v)$ of each node in the hyperedge is set to 2 if the node v is the first or last author of the corresponding paper and 1 otherwise.

In **Step 3** of `hw3.ipynb`, you initialize PageRank scores of all nodes as $1/|V|$ and then run the `run_pagerank` function on the given hypergraph twice: once with the `use_weight=False` setting and once with the `use_weight=True` setting. This will generate two sets of PageRank results for each setting. Your task is to compare and analyze the **top** 10 authors based on each set of results.

To facilitate your analysis, we provide you with `node_info.txt`, which contains information on the count of publications, number of citations, HIndex, PIndex, and AIndex of each node i, separated by a comma. (Note that node index starts from 0, and the $i+1$ line indicates the i-th node.) You may also use various metrics, such as degree centrality, to justify the PageRank scores or identify any differences between the two versions.

5 Notes

- You may encounter some subtleties when it comes to implementation. Come up with your own design and/or contact Jihoon Ko (jihoonko@kaist.ac.kr) and Minyoung Choe (minyoung.choe@kaist.ac.kr) for discussion. Any ideas can be taken into consideration when grading if they are written in the *readme* file.
- You are **NOT** allowed to import any external libraries (e.g., NumPy, SciPy, NetworkX) in your implementation. But you are allowed to additionally import any built-in libraries in Python.

6 How to submit your assignment

1. Create hw3-[your student id].tar.gz, which should contain the following files:
   - **hw3.ipynb, pagerank.py, utils.py**: these file should contain your writing and implementation.
   - **readme.txt**: this file should contain the names of any individuals from whom you received help and the nature of the help that you received. That includes help from friends, classmates, lab TAs, course staff members, etc. In this file, you are also welcome to write any comments that can help us grade your assignment better, your evaluation of this assignment, and your ideas.
2. Make sure that no other files are included in the tar.gz file.
3. Submit the tar.gz file at KLMS (`http://klms.kaist.ac.kr`).