CPSC 483/583: Deep Learning on Graph-structured Data **Fall 2022**

Homework 3

Due 11:59pm ET Tuesday November 8 2022

This problem set should be completed individually.

General Instructions

These questions require thought, but do not require long answers. Please be as concise as possible. You are allowed to take a maximum of 1 late period (see the course website or slides about the definition of a late period).

Submission instructions: You should submit your answers in a PDF file. LaTeX is highly preferred due to the need of formatting equations.

Submitting answers: Prepare answers to your homework in a single PDF file. Make sure that the answer to each sub-question is on a *separate page*. The number of the question should be at the top of each page.

Honor Code: When submitting the assignment, you agree to adhere to the Yale Honor Code. Please read carefully to understand what it entails!

Homework survey: After submitting your homework, please fill out the Homework 3 Feedback Form. 0.5% overall extra credit will be given if you take time to reflect on the nature of the problems and possibly provide feedbacks for the benefit of future offering of this course, by filling in the form for each written and coding assignment.

1 Graph Isomorphism

1. Graph neural networks update node embeddings through the computation graph defined by the neighborhood. Draw the computation graph of *Node 1* in *Graph 1* in Figure 1 for 2-layer GNNs (i.e., aggregate neighbor nodes twice).

   Note: you can draw in ways you like, take a picture or a screenshot and insert the figure into your submitted answer.

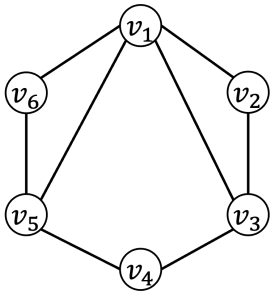

Figure 1: Graph 1

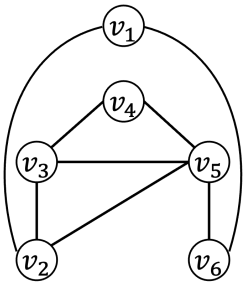

Figure 2: Graph 2

2. If every node in *Graph 1* has the same initial features, and no IDs (indices for nodes) will be used as inputs to GNNs, which pairs of nodes cannot be distinguished by typical message passing GNNs? Explain why. (**Note:** v_i is only for you to indicate the node and cannot be seen by the network)

3. We can use the Position-aware GNN introduced in Lecture 10 to distinguish the node pairs you figure out in Question 1.2. Select one node as your *Anchor* and explain why this *Anchor* works. (**Hint:** demonstrate by calculating the different relative distances between the indistinguishable node pairs and the *Anchor*.)

4. In graph theory, graph G and graph H are **isomorphic** if there is a bijection between the vertex sets of G and H: $f : V(G) \to V(H)$ such that any two vertices u and v of G are adjacent in G if and only if $f(u)$ and $f(v)$ are adjacent in H.

   Are *Graph 1* in Figure 1 and *Graph 2* in Figure 2 isomorphic? If so, demonstrate the bijection function f between *Graph 1* and *Graph 2*. If not, prove it by finding a structure in one graph but is not present in the other. (**Hint:** use color refinement algorithm to extract the structure of K-hop neighborhood. Two isomorphic graphs should share the same K-hop neighboring information for any K.)

2 Graph Algorithm and Message Passing

Let's explore the relation between message passing mechanism and graph algorithms, such as graph random walk and breadth-first search.

1. **Random Walk:** Message-passing GNNs can simulate the graph random walk algorithm, where the number of layers corresponds to the time step t. In the graph random walk setting, each node in the graph G has an embedding at each time step. Let $h_i^{(t)}$ denote the embedding

of node i at time step t and $h_i^{(0)}$ is a random initial embedding. $\mathcal{N}(i)$ denotes the neighbor nodes of node i. Assume we are at node i at time step t. At the next time step $t+1$, we move from the current node i to one of its neighbor nodes $j \in \mathcal{N}(i)$. The probability of moving to each neighbor node is equal, which is called **uniform random walk**. After each move, the embedding of nodes will change. The expectation of the embedding of the node we move to is exactly the embedding of node i in the next time step $t+1$. Therefore, $h_i^{(t+1)} = \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} h_j^{(t)}$.

Let $H^{(t)} \in \mathbb{R}^{n \times d}$, where n is the number of nodes and d is the embedding dimension. The i-th row of $H^{(t)}$ is the embedding of node i, i.e., $h_i^{(t)}$. We have $H^{(t+1)} = PH^{(t)}$, where P is called *transition matrix*. Derive the *transition matrix* using the adjacency matrix A and degree matrix D of the graph G.

2. **Breadth-first Search (BFS):** In BFS setting, there are two types of nodes with 1-dimensional embedding in $\{0, 1\}$ for a graph G. Let $h_i^{(t)}$ denote the embedding of node i at time step t, and $\mathcal{N}(i)$ denotes the neighbor nodes of i. The first type of node is visited nodes with an embedding of 1. Others are unvisited nodes with an embedding of 0. Initially, all nodes have embedding 0, except a starting node with an embedding of 1. At every step, nodes that are connected to any visited nodes become visited and change its embedding from 0 to 1. Nodes not visited keep the embedding 0.

   Message-passing GNNs can learn BFS algorithm well, where the number of layers corresponds to the time step. Write out a message function and an aggregation function with the notations given in the previous paragraph to update the embedding of node i at time step t to simulate this BFS algorithm. (**Hint:** Firstly, write out the update function of $h_i^{(t+1)}$, using the embedding of neighbor nodes at time step t (i.e., $h_j^{(t)}$ for $j \in \mathcal{N}(i)$). Figure out the message terms passed by the neighbor nodes. You can use min, max, sum, or any other well-defined non-linear functions to realize the aggregation)