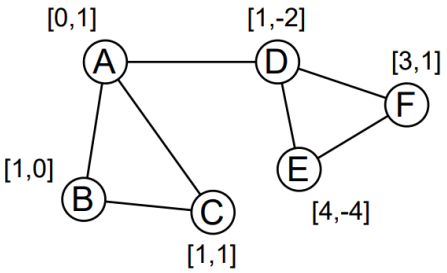
Final Exam Solutions (Graph Neural Networks –Fall 2024)

Full Name:
Student ID:

1. (10p) Consider an undirected graph G of six nodes A, B, C, D, E, and F given in the following figure. Each node has initial features that are the numbers standing next to it, as follows:



Assume that the hidden layer of an GCN model of all nodes at layer (k) can be calculated as:

$$H^{(k)} = \sigma\left(A \cdot H^{(k-1)}\right),$$

where $H^{(k)}$ denotes the output at layer k, σ is a ReLU function $\mathrm{ReLU}(x) = \max(0, x)$.
A is the adjacency matrix.

a. Calculate the output of the GCN model at layer $k = 1$.
b. What are the limitations of the GCN model compared to GraphSage?

Solutions:

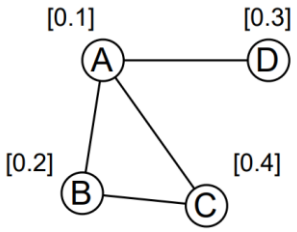
$$A = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

$$H^1 = \sigma\left(AH^0\right) = \mathrm{ReLU}\left(\begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 1 \\ 1 & -2 \\ 4 & -4 \\ 3 & 1 \end{bmatrix}\right) = \mathrm{ReLU}\left(\begin{bmatrix} 3 & -1 \\ 1 & 2 \\ 1 & 1 \\ 7 & -2 \\ 4 & -1 \\ 5 & -6 \end{bmatrix}\right) = \begin{bmatrix} 3 & 0 \\ 1 & 2 \\ 1 & 1 \\ 7 & 0 \\ 4 & 0 \\ 5 & 0 \end{bmatrix}$$

The limitations of GCN model compared to GraphSage:

- Scalability: GCNs require the entire graph adjacency matrix and feature matrix during training. This can lead to high memory usage and computational inefficiency due to matrix multiplications involving the entire graph.
- Transductive learning: GCNs assume a static, fixed graph structure during training. Any changes to the graph (e.g., adding nodes or edges) require retraining the model or re-computing the embeddings.

2. (10p) Consider an undirected graph G of five nodes A, B, C, and D given in the following figure. Each node has initial features that are the numbers standing next to it (i.e., the initial feature of node 'A' is $h_A^{(0)} = 0.1$). According to GraphSAGE model with a MAX aggregation function, the feature of a node i at layer k can be updated as:



$$h_{N(i)}^{(k)} = \text{AGGREGATE}\left(\left\{h_u^{(k-1)}, \forall u \in N(i)\right\}\right)$$

$$h_i^{(k)} = \text{ReLU}\left(h_i^{(k-1)} \,\|\, h_{N(i)}^{(k)}\right)$$

where $\|$ is a concatenation, $\text{ReLU}(x) = \max(0, x)$, *N(i)* is the neighbour nodes of node *i*.

a) Calculate the feature of each node at $k = 1$.

b) How does the GraphSage model address the scalability problem?

SOLUTIONS:

a)

$$h_{N(i)}^{(k)} = \text{AGGREGATE}\left(\left\{h_u^{(k-1)}, \forall u \in N(i)\right\}\right)$$

$$h_{N(A)}^{(1)} = MAX\left(h_B, h_C, h_D\right) = MAX\left(0.4, 0.2, 0.3\right) = 0.4$$

$$h_{N(B)}^{(1)} = MAX\left(h_A, h_C\right) = MAX\left(0.1, 0.4\right) = 0.4$$

$$h_{N(C)}^{(1)} = MAX\left(h_A, h_B\right) = MAX\left(0.2, 0.1\right) = 0.2$$

$$h_{N(D)}^{(1)} = MAX\left(h_A\right) = MAX\left(0.1\right) = 0.1$$

$$h_i^{(k)} = \text{ReLU}\left(h_i^{(k-1)} \,\|\, h_{N(i)}^{(k)}\right)$$

$$h_A^{(1)} = \text{ReLU}\left(h_A^{(0)} \,\|\, h_{N(A)}^{(0)}\right) = Max\left(0, [0.1, 0.4]\right) = [0.1, 0.4]$$

$$h_B^{(1)} = \text{ReLU}\left(h_B^{(0)} \,\|\, h_{N(B)}^{(0)}\right) = Max\left(0, [0.2, 0.4]\right) = [0.2, 0.4]$$

$$h_C^{(1)} = \text{ReLU}\left(h_C^{(0)} \,\|\, h_{N(C)}^{(0)}\right) = Max\left(0, [0.4, 0.2]\right) = [0.4, 0.2]$$

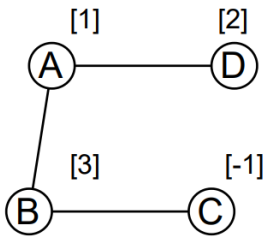
$$h_D^{(1)} = \text{ReLU}\left(h_D^{(0)} \,\|\, h_{N(D)}^{(0)}\right) = Max\left(0, [0.3, 0.1]\right) = [0.3, 0.1]$$

b) GraphSAGE model addresses the scalability problem by employing a neighborhood sampling strategy and designing the model to operate inductively.

- GraphSAGE avoids processing the entire graph at once by sampling a fixed-size set of neighbors for each node. Instead of aggregating features from all neighbors (as done in GCNs), GraphSAGE samples a fixed number of *K* neighbors for each node.

- GraphSAGE is designed to learn inductive representations, meaning it can generate embeddings for unseen nodes or graphs without retraining.

3. (10p) Consider an undirected graph G of five nodes A, B, C, and D given in the following figure. Each node has initial features that are the numbers standing next to it, as follows:



The output of an GCNII model of all nodes at layer (k) can be calculated as:

$$H^{(k)} = \sigma\left[\left((1-\beta)I_n\right)\cdot\left((1-\alpha)\tilde{A}\cdot H^{(k-1)} + \alpha H^{(0)}\right)\right]$$

where $H^{(k)}$ denotes the output at layer k, $\tilde{A}$ is the normalized matrix ($\tilde{A} = D^{-1}A$), I_n is the identity matrix, $\alpha = \beta = 0.5$, σ is a ReLU function $\text{ReLU}(x) = \max(0, x)$.

a) Calculate $\tilde{A}$ and the representations of each node at layer $k = 1$.

b) What are the reasons why the over-smoothing problem happens in graphs? And how does GCNII address the problem?

SOLUTIONS:

a)

$$\tilde{A} = D^{-1}A = \begin{bmatrix} 0 & 0.5 & 0 & 0.5 \\ 0.5 & 0 & 0.5 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

$$H^{(k)} = \sigma\left[\left((1-\beta)I_n\right)\left((1-\alpha)\tilde{A}\cdot H^{(k-1)} + \alpha H^{(0)}\right)\right]$$

$$\tilde{A}\cdot H^{(0)} = \begin{bmatrix} 0 & 0.5 & 0 & 0.5 \\ 0.5 & 0 & 0.5 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}\cdot\begin{bmatrix} 1 \\ 3 \\ -1 \\ 2 \end{bmatrix} = \begin{bmatrix} 2.5 \\ 0 \\ 3 \\ 1 \end{bmatrix}; I_n = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\left[\left((1-\alpha)\tilde{A}\cdot H^{(k-1)} + \alpha H^{(0)}\right)\cdot\left((1-\beta)I_n\right)\right]$$

$$= \left(0.5\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\right)\cdot\left(0.5\begin{bmatrix} 2.5 \\ 0 \\ 3 \\ 1 \end{bmatrix} + 0.5\begin{bmatrix} 1 \\ 3 \\ -1 \\ 2 \end{bmatrix}\right) = 0.5\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}0.5\begin{bmatrix} 3.5 \\ 3 \\ 2 \\ 3 \end{bmatrix} = 0.25\begin{bmatrix} 3.5 \\ 3 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 0.875 \\ 0.75 \\ 0.5 \\ 0.75 \end{bmatrix}$$

$$H^{(k)} = \sigma\left(\begin{bmatrix} 0.875 \\ 0.75 \\ 0.5 \\ 0.75 \end{bmatrix}\right) = \begin{bmatrix} 0.875 \\ 0.75 \\ 0.5 \\ 0.75 \end{bmatrix}$$

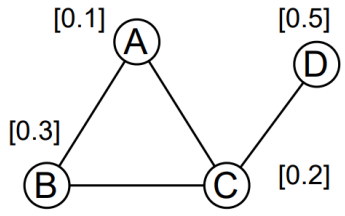
b) Over-smoothing is a common problem in graph neural networks (GNNs), where node representations become indistinguishable as the number of layers increases. The reason is the excessive neighbor aggregation problem. That is, GNNs aggregate features from neighbors at each layer. As the number of layers increases, nodes receive information from increasingly distant nodes in the graph. As a result, this aggregation process leads to all nodes in a connected component having similar embeddings, causing a loss of discriminative power.

GCNII incorporates identity mapping by adding a weighted skip connection to the node embeddings:

- GCNII uses an initial residual connection that allows each layer to combine its output with the raw input features. This can helps maintain diversity in node representations by preserving raw node features across all layers. It also ensures that even as layers increase, the representations do not lose important local information.

- GCNII incorporates identity mapping to the weight matrix, preserving the initial node features across all layers, preventing the embeddings from collapsing.

4. (10p) Consider an undirected graph G of four nodes A, B, C, and D given in the following figure. Each node has initial features that are the numbers standing next to it. According to GAT model, the weight matrix W is initialized as [0.1]. The feature of node 'i' at layer (k) can be updated as:



$$h_i^{(k)} = \sigma\left(\sum_{m \in N(i)} \alpha_{im} W h_m\right)$$

where: $\alpha_{im} = \dfrac{e_{im}}{\sum_{k \in N(i)} e_{ik}}$, and $e_{im} = \sigma\left(\text{MEAN}\left(Wh_i, Wh_m\right)\right)$

σ is a ReLU function $\text{ReLU}(x) = \max(0, x)$.

a) Calculate the attention coefficients e_{AB} and e_{AC}. Then, calculate the feature of node 'A' at $k = 1$.

b) How can the attention mechanism address the node importance problem?

SOLUTIONS:

$$h_i^{(k)} = \sigma\left(\sum_{m\in N(i)} \alpha_{im} W h_m\right)$$

where: $\alpha_{im} = \dfrac{e_{im}}{\sum_{k\in N(i)} e_{ik}}$, and $e_{im} = \sigma\left(\text{MEAN}\left(Wh_i, Wh_m\right)\right)$

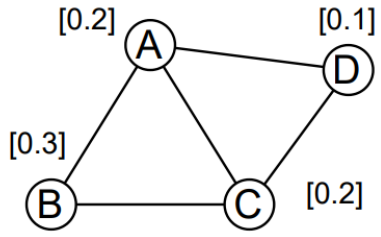
$$e_{AC} = \sigma\left(\text{MEAN}\left(Wh_A, Wh_C\right)\right) = \sigma\left(\text{MEAN}\left(0.1*0.1, 0.1*0.2\right)\right)$$
$$= \sigma\left(\text{MEAN}\left(0.01, 0.02\right)\right) = 0.015$$
$$e_{AB} = \sigma\left(\text{MEAN}\left(Wh_A, Wh_B\right)\right) = \sigma\left(\text{MEAN}\left(0.1*0.1, 0.1*0.3\right)\right)$$
$$= \sigma\left(\text{MEAN}\left(0.01, 0.03\right)\right) = 0.02$$

$$\alpha_{AB} = \frac{e_{AB}}{e_{AB} + e_{AC}} = \frac{0.02}{0.02 + 0.015} = 0.57$$
$$\alpha_{AC} = 0.43$$

$$h_A^{(1)} = \sigma\left(\sum_{m\in N(i)} \alpha_{Am} W h_m\right) = \sigma\left(\alpha_{AB} W h_B + \alpha_{AC} W h_C\right)$$
$$= \sigma\left(0.57*0.1*0.3 + 0.43*0.1*0.2\right)$$
$$= \text{Re}\,LU\left(0.57*0.1*0.3 + 0.43*0.1*0.2\right)$$
$$= 0.0171 + 0.0086 = 0.0257$$

b) The attention mechanism assigns a unique weight (attention score) to each neighboring node based on its relevance to the target node. The relevance is computed using a learned function that evaluates the relationship between the target node's features and each neighbor's features. That is, neighbors that are more relevant to target node will receive higher weights , and their features contribute more during aggregation. Meanwhile, less important or irrelevant neighbors contribute less, reducing noise in the aggregated representation.

5. (10p) Consider an undirected graph G of four nodes A, B, C, and D given in the following figure. Each node has initial features that are the numbers standing next to it. According to GIN model, the parameter is a fixed scalar $\varepsilon = 0.5$, the feature of a node i at layer k can be updated as:



$$h_i^{(k)} = \left(1+\varepsilon\right)\cdot h_i^{(k-1)} + \sum_{j\in N(i)} h_j^{(k-1)}$$

a. Calculate the feature of each node at $k = 1$.
b. What are the reasons why GIN model achieves 1-d Weisfeiler-Lehman?

SOLUTIONS:

$$h_i^{(k)} = \left(1+\varepsilon\right)\cdot h_i^{(k-1)} + \sum_{j\in N(i)} h_j^{(k-1)}$$

$$h_A^{(1)} = \left(1+\varepsilon\right)\cdot h_A^{(0)} + \left(h_B^{(0)} + h_C^{(0)} + h_D^{(0)}\right) = \left(1+0.5\right)*0.2 + \left(0.3+0.2+0.1\right) = 0.3+0.6 = 0.9$$

$$h_B^{(1)} = \left(1+\varepsilon\right)\cdot h_B^{(0)} + \left(h_A^{(0)} + h_C^{(0)}\right) = \left(1+0.5\right)*0.3 + \left(0.2+0.2\right) = 0.45+0.4 = 0.85$$

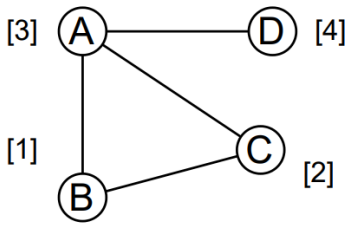
$$h_C^{(1)} = \left(1+\varepsilon\right)\cdot h_C^{(0)} + \left(h_A^{(0)} + h_B^{(0)} + h_D^{(0)}\right) = \left(1+0.5\right)*0.2 + \left(0.2+0.3+0.1\right) = 0.3+0.6 = 0.9$$

$$h_D^{(1)} = \left(1+\varepsilon\right)\cdot h_D^{(0)} + \left(h_A^{(0)} + h_C^{(0)}\right) = \left(1+0.5\right)*0.1 + \left(0.2+0.2\right) = 0.15+0.4 = 0.55$$

GIN uses a sum aggregation function to combine the features of neighboring nodes. Unlike mean or max-pooling used in other GNNs, the sum operation ensures that the aggregation function is injective (i.e., distinct input features produce distinct outputs).

The combination of the sum operator with an injective MLP ensures that GIN can distinguish graphs as powerful as the 1-WL testing.

6. (10p) Consider an undirected graph G of four nodes A, B, C, and D given in the following figure. According to GNN-AK model (From start to subgraph) with a GCN model and MEAN aggregation function, the feature of a node i at layer k can be updated as:



$$h_v^{(l+1)} = AGG^{(l)}\left(G\left[N_k(v)\right]\right), l = 0,1,...,L-1$$

$$h_v = MEAN\left(h_u^{(L)} \mid u \in G\left[N_k(v)\right]\right)$$

where $G\left[N_k(v)\right]$ is the subgraph rooted at node v with k-hop distance including v, and AGG(.) is a SUM function.

a. Calculate the feature of each node at $k = 1$ and $l = 0$.
b. Why considering a set of k-hop subgraphs will make the model more expressive than only neighbor aggregation?

SOLUTIONS:

$$h_A^{(1)} = AGG^{(0)}\left(G\left[A,B,C,D\right]\right) = SUM\left(1+2+3+4\right) = 10$$

$$h_B^{(1)} = SUM\left(1,2,3\right) = 6$$

$$h_C^{(1)} = SUM\left(1,2,3\right) = 6$$

$$h_D^{(1)} = SUM\left(3,4\right) = 7$$

$$h_A = MEAN\left(h_u^{(L)} \mid u \in G\left[N_k(v)\right]\right) = MEAN\left(10+6+6+6+7\right) = 7$$

– – –

$$h_B^{(1)} = AGG^{(0)}\left(G\left[A,B,C\right]\right) = SUM\left(1,2,3\right) = 6$$

$$h_C^{(1)} = SUM\left(1,2,3\right) = 6$$

$$h_A^{(1)} = SUM\left(1,2,3\right) = 6$$

$$h_B = MEAN\left(h_u^{(L)} \mid u \in G\left[N_k(v)\right]\right) = MEAN\left(6+6+6\right) = 6$$

– – –

$$h_C^{(1)} = AGG^{(0)}\left(G\left[A,B,C\right]\right) = SUM\left(1,2,3\right) = 6$$

$$h_B^{(1)} = SUM\left(1,2,3\right) = 6$$

$$h_A^{(1)} = SUM\left(1,2,3\right) = 6$$

$$h_C = MEAN\left(h_u^{(L)} \mid u \in G\left[N_k(v)\right]\right) = MEAN\left(6+6+6\right) = 6$$

– – –

$$h_D^{(1)} = AGG^{(0)}\left(G\left[A,D\right]\right) = SUM\left(3,4\right) = 7$$

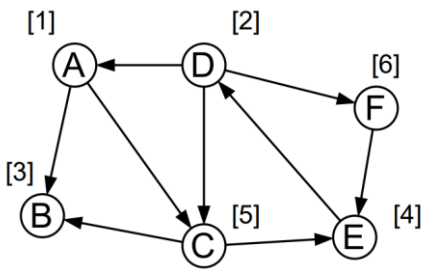
$$h_A^{(1)} = SUM\left(3,4\right) = 7$$

$$h_D = MEAN\left(h_u^{(L)} \mid u \in G\left[N_1(D)\right]\right) = MEAN\left(7,7\right) = 7$$

A k-hop subgraph retains the entire subgraph's structural information (e.g., edge arrangements, motifs, and surrounding connectivity) within the k-hop distance, providing a more complete local representation. The higher-order structures, such as cycles, or cliques, are inherently captured separately within k-hop subgraphs, allowing models to leverage complex structural patterns.

Neighbor aggregation cannot distinguish nodes in different structural roles or higher-order graph features, as it focuses only on immediate neighbors. K-hop subgraphs, on the other hand, explicitly incorporate these relationships.

7. (10p) Given a directed graph G of six nodes A, B, C, D, E, and F given in the following figure



a. According to the Graphormer model, calculate the initial node features of each node with Centrality Encoding. The definition of a Lookup table for node degree is as follows: T = {"0": 0.0, "1": 0.1, “2”: 0.2, “3”: 0.3, “4”: 0.4}.
b. Why encode the structural information of a graph in the model to make the model more powerful in capturing the graph structures? What kind of structural information needs to be considered?
SOLUTIONS

$$h_A^0 = h_A + d_A^- + d_A^+ = 1 + T[1] + T[2] = 1 + 0.1 + 0.2 = 1.3$$
$$h_B^0 = h_B + d_B^- + d_B^+ = 3 + T[0] + T[2] = 3 + 0.2 = 3.2$$
$$h_C^0 = 5 + T[2] + T[2] = 5 + 0.2 + 0.2 = 5.4$$
$$h_D^0 = 2 + T[3] + T[1] = 2 + 0.3 + 0.1 = 2.4$$
$$h_E^0 = 4 + T[1] + T[2] = 4 + 0.1 + 0.2 = 4.3$$
$$h_F^0 = 6 + T[1] + T[1] = 6.2$$

Structural information helps the model distinguish similar graphs when considering only node-level features or simple neighbor aggregation, allowing the model to identify non-isomorphic graphs. Encoding structural properties like degree, subgraph structures, or motifs allows the model to identify such distinctions and increase the power of graph isomorphism tests as the Weisfeiler-Lehman (WL) test.

There are several kinds of structural information, which can be integrated into the model: Laplacian Eigenvectors, SPD, Random Walk, and structural identity.

8. (10p) What are the main reasons why BPR loss (Bayesian Personalized Ranking loss) is commonly utilized in recommendation system models?
SOLUTIONS:

The BPR loss is commonly used in recommendation systems because it is designed for pairwise ranking optimization. Unlike pointwise loss functions (e.g., mean squared error), which predict absolute scores, BPR focuses on the relative order between items. For example, BPR works by comparing pairs of items: for a given user, it ensures that the predicted preference for an observed (positive) item i is higher than for an unobserved item j.

BPR addresses implicit feedback: It assumes all unobserved interactions are less preferred than observed ones. It effectively learns to differentiate between interacted and non-interacted items without requiring explicit negative labels.

Furthermore, BPR also can handle the scalability problem: It is computationally efficient because it simplifies the optimization problem to pairwise comparisons, avoiding the need for a full ranking of all items during training.

9. (10p) Why the TransR loss is critical in knowledge graph (KG) embedding models?

   TransR projects entities into a relation vector space, allowing each relationship to have its semantic interpretation. This flexibility makes TransR better suited for multi-relational KGs compared to models like TransE, which assume a single embedding space for all relations. This allows the model to better represent: one-to-many, many-to-one, and many-to-many relationships and asymmetric relations.

10. (10p) What are the limitations of contrastive learning methods in molecular structure learning? And why are the subgraph-based methods better at molecular structure learning?

    Contrastive methods rely heavily on data augmentations (e.g., node dropping, edge perturbation, subgraph sampling). However, random augmentations can destroy the molecular structures, and disrupt the chemical properties. Moreover, designing augmentations that preserve chemical validity is challenging for molecular graphs.

    Subgraph-based methods explicitly extract and utilize localized structural patterns, making them better suited for molecular structure learning. Subgraph-based methods capture functional groups, rings, and other motifs, which are fundamental to molecular properties. That is, Subgraph-based methods can incorporate domain-specific knowledge about functional groups or reaction mechanisms, enhancing their relevance to molecular tasks. Moreover, Subgraph-based methods are inherently more interpretable because they explicitly focus on molecular substructures.