COMP4222 Final Exam

Date: Dec 12, 2022

Time: 4:30PM - 7:30PM

Instructor: Yangqiu Song

Name: ________________ **Student ID:** ________________

Question	Score	Question	Score
1	/ 10	6	/ 10
2	/ 11	7	/ 10
3	/ 12	8	/ 12
4	/ 12	9	/ 13
5	/ 10		/
Total:	**/ 100**		

The HKUST Academic Honor Code

Honesty and integrity are central to the academic work of HKUST. Students of the University must observe and uphold the highest standards of academic integrity and honesty in all the work they do throughout their program of study.

As members of the University community, students have the responsibility to help maintain the academic reputation of HKUST in its academic endeavors.

Sanctions will be imposed if students are found to have violated the regulations governing academic integrity and honesty.

Declaration of Academic Integrity I confirm that I have answered the questions using only materials specifically approved for use in this examination, that all the answers are my own work, and that I have not received any assistance durng the examination.

Signature: ________________

1 Yes/No Questions (10 Points)

1. TransE, TransR and DistMult can model transitive relations.
2. In terms of aggregating neighbor information, SUM aggregator's representation ability is stronger than AVERAGE and MAX.
3. DeepWalk model is a special case of node2vec model.
4. Stacking more graph neural network layers can make the model have larger receptive fields.
5. If we want to use curriculum learning and distance weighted sampling strategies in our link prediction model training, we need to select negative samples with small distance negative nodes first then larger and larger.
6. In the Minimum Cut algorithm, the weight of cut is directly proportional to the number of edges in the cut.
7. To collect link level features for a graph, the Local Neighborhood Overlap uses shortest path length between nodes to capture neighborhood information.
8. One of the basic assumptions of the Content-based Filtering is that users with similar tastes in the past will have similar tastes in the future.
9. To defend against malicious attacks, the Graph Purifying method augment the training set with adversarial data.
10. In terms of self-supervised learning on graph data, one of the fundamental differences found in graph domain is that nodes are connected and dependent.

Answers: FTTTF TFFFT

2 Node2Vec (11 Points)

In this question, you are required to list all the paths generated from the biased random walk process. Assume that F is the second last node which is indicated with dotted line, and D is the last node which is indicated with dashed line. Assume p is 2, q is 0.5, and the remaining walk length is 2. You should list all possible paths and their associated probabilities.

Please write down all the necessary steps to compute the probabilities. The probability should be in the format of fraction rather than decimal.

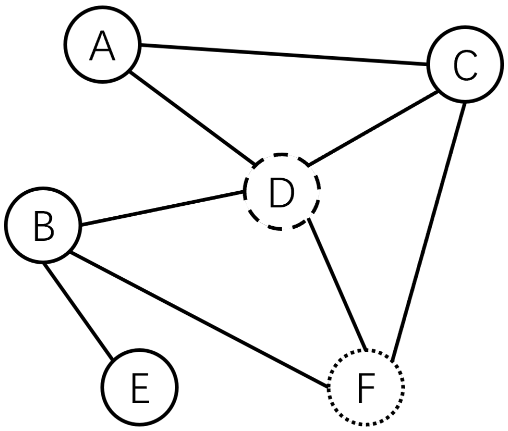


Figure 1: Input graph

Hints: your path should start with node D and contains 3 nodes.

Answer:
D - A - C: 4/9*2/3=8/27
D - A - D: 4/9*1/3=4/27
D - B - E: 2/9*4/7=8/63
D - B - F: 2/9*2/7=4/63
D - B - D: 2/9*1/7=2/63
D - C - A: 2/9*2/5=4/45
D - C - D: 2/9*1/5=2/45
D - C - F: 2/9*2/5=2/45
D - F - C: 1/9*2/5=2/45
D - F - D: 1/9*1/5=1/45
D - F - B: 1/9*2/5=1/45

3 Graph Attention Network (12 Points)

Consider Graph Attention Network (GAT).

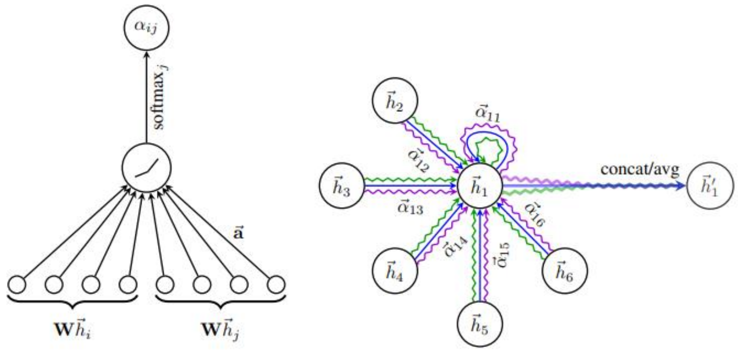


Figure 2: The multi-head attention employed in GAT

The description of a single graph attention layer as follow: The input to the layer is a set of node features, $h = \{\vec{h}_1, \vec{h}_2, \cdots, \vec{h}_N\}, \vec{h}_i \in \mathbb{R}^F$, where N is the number of nodes, and F is the number of features in each node. The layer produces a new set of node features (of potentially different cardinality F'), $\mathbf{h}' = \left\{\vec{h}'_1, \vec{h}'_2, \ldots, \vec{h}'_N\right\}, \vec{h}'_i \in \mathbb{R}^{F'}$, as its output.

In order to obtain sufficient expressive power to transform the input features into higher-level features, at least one learnable linear transformation is required. A shared linear transformation, parametrized by a weight matrix, $\mathbf{W}$ is applied to every node.

Usually, the attention mechanism a is a single-layer feedforward neural network, parametrized by a weight vector $\vec{a}$, and applying the LeakyReLU nonlinearity. Fully expanded out, the coefficients computed by the attention mechanism may then be expressed as:

$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T \left[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_j\right]\right)\right)}{\sum_{k \in \mathcal{N}_i} \exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T \left[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_k\right]\right)\right)}$$

where $\cdot^T$ represents transposition and k is the concatenation operation

Once obtained, the normalized attention coefficients are used to compute a linear combination of the features corresponding to them, to serve as the final output features for every node (after potentially applying a nonlinearity, σ)

$$\vec{h}'_i = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} \mathbf{W} \vec{h}_j\right)$$

And the equation is extended to multi-head attention:

$$\vec{h}'_i = \|_{k=1}^{K} \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \vec{h}_j\right)$$

where $\|$ represents concatenation, α_{ij}^k are normalized attention coefficients computed by the k-th attention mechanism, and $\mathbf{W}^k$ is the corresponding input linear transformation's weight matrix.

Here are some assumptions:

1. The input graph is undirected with no edge features, the node feature dimension is 100.
2. We have 2-layer GAT model, with output feature size $[10, 10]$ respectively; multi-head numbers $[5, 3]$ respectively, all the bias is ignored.

Answer the following questions:

1. Compute the number of all trainable parameters of this model. (4 pts)
2. Recently, a paper proposed that there is a problem in the standard GAT attention scoring function: the learned layers **W** and **a** are applied consecutively and thus can be collapsed into a single linear layer. To fix this limitation, they simply apply the **a** layer after the nonlinearity, and the **W** layer after the concatenation:

$$\text{Original GAT:} \quad e(h_i, h_j) = \text{LeakyReLU}\left(\overrightarrow{\mathbf{a}}^T \left[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_j\right]\right)$$

$$\text{GATv2:} \quad e(h_i, h_j) = \overrightarrow{\mathbf{a}}^T \, \text{LeakyReLU}\left(\mathbf{W}\left[\vec{h}_i \| \vec{h}_j\right]\right)$$

Please compute the number of all trainable parameters of this new GAT model. (8 pts)

Answer:

1. 1-layer: input size: 100, for each attention, $\mathbf{W} \in \mathbb{R}^{100\times 10}$, $\vec{a} \in \mathbb{R}^{20}$, parameter for 1-layer $(100 \times 10 + 20) \times 5 = 5100$
2-layer: input size: $5 \times 10 = 50$, for each attention, $\mathbf{W} \in \mathbb{R}^{50\times 10}$, $\vec{a} \in \mathbb{R}^{20}$, parameter for 1-layer $(50 \times 10 + 20) \times 3 = 1560$
Total: $1560 + 5100 = 6660$
2. 1-layer: input size: 100, for each attention, $\mathbf{W} \in \mathbb{R}^{200\times 10}$, $\vec{a} \in \mathbb{R}^{10}$, parameter for 1-layer $(200 \times 10 + 10) \times 5 = 10050$
2-layer: input size: $5 \times 10 = 50$, for each attention, $\mathbf{W} \in \mathbb{R}^{100\times 10}$, $\vec{a} \in \mathbb{R}^{10}$, parameter for 2-layer $(100 \times 10 + 10) \times 3 = 3030$
Total: $10050 + 3030 = 13080$
The question is lack of some information, we decide to give all of you full grades. paper link: https://arxiv.org/pdf/2105.14491.pdf

4 Knowledge Graph Embedding (12 points)

The task of knowledge graph embedding (KGE) tries to find appropriate representations for entities and relations and appropriate mathematical computations between the representations to approximate the symbolic and logical relationships between entities. In the lecture, we introduced several knowledge embedding methods, such as TransE, DisMult, etc. One major challenge for KGE is that the relations in real-world knowledge bases exhibit complex behaviours: they can be injective (1-1), or non-injective (1-N, N-1, N-N), symmetry or skew-symmetry; one relation may be the inversion of another relation; one relation may be the composition of other two relations. To better model these relations, recently, a new knowledge graph embedding, MQuadE was proposed.

In MQuadE, it uses a matrix E as the representation of each entity e in the knowledge base. The matrix is required to be symmetric, that is, $\boldsymbol{E} = \boldsymbol{E}^T$. For the representation of relation r, it uses a pair of matrices $< \boldsymbol{R}, \hat{\boldsymbol{R}} >$ to model its influence on the head entity (by $\boldsymbol{R}$) and tail entity (by $\hat{\boldsymbol{R}}$). MQuadE represents a fact triple (h, r, t), that is, (head entity, relation, tail entity), in the knowledge graph with a matrix quadruple ($\boldsymbol{H}$, $\boldsymbol{R}$, $\hat{\boldsymbol{R}}$, $\boldsymbol{T}$), where $\boldsymbol{H}$ and $\boldsymbol{T}$ are the representations of h and T respectively and $< \boldsymbol{R}, \hat{\boldsymbol{R}} >$ is the pair of representation of r. One matrix $\boldsymbol{R}$ post-multiplies the head entity matrix $\boldsymbol{H}$; another matrix $\hat{\boldsymbol{R}}$ pre-multiplies the tail entity matrix $\boldsymbol{T}$. It assumes that the score function $f(h, r, t) = \|\boldsymbol{HR} - \hat{\boldsymbol{R}}\boldsymbol{T}\|_F^2$ is small if the fact triple (h,r,t) is true and is large if the fact triple is false. .

1. Assume that $\boldsymbol{H}$,$\boldsymbol{R}$,$\hat{\boldsymbol{R}}$,$\boldsymbol{T} \in \mathbb{R}^{p \times p}$, and we have n_e entities and n_r relations in knowledge graph. MQuadE contains how many parameters? (4 pts)
2. Please specify the relation property with the corresponding relation matrix form. (4 pts)

$$\boldsymbol{R}_1^T = \hat{\boldsymbol{R}}_2, \hat{\boldsymbol{R}}_1^T = \boldsymbol{R}_2$$

3. Please specify the relation matrix form with the corresponding relation property: (4 pts)

   A relation r_3 is the composition of relation r_1 and r_2 (written as $r_3 = r_1 \oplus r_2$) iff the facts (a, r_1, b) and (b, r_2, c) imply the fact (a, r_3, c).

Answer: $p^2 n_e/2 + 2p^2 n_r$ or $p(p+1)n_e/2 + 2p^2 n_r$ The first one is paper's answer, the second one is what can we know from the description, we consider both of them are right.

Answer: Inversion. Or the fact triple (h, r_1, t) holds $\Longleftrightarrow$ the fact triple (t, r_2, h) holds

Answer:

$$\boldsymbol{R}_3 = \boldsymbol{R}_1 \boldsymbol{R}_2, \hat{\boldsymbol{R}}_3 = \hat{\boldsymbol{R}}_1 \hat{\boldsymbol{R}}_2$$

paper link: https://dl.acm.org/doi/pdf/10.1145/3442381.3449879

5 Graph Isomorphism (10 Points)

We consider undirected graphs $G = (V, E, X_V)$, where $V = \{1, \cdots n\} := [n]$ is the set of vertices; $E \subseteq V \times V$ is the set of vertices satisfying $(u, v) \in E$ if and only if $(v, u) \in E$; and X_V is the set of node features: For all $v \in V, X_v \in \mathbb{R}_d$. The neighbors of a vertex v is $\mathcal{N}_{G(v)} = \{w : (v, w) \in E\}$. Graphs without node features can be represented either by taking $X_v = 1$ for all $v \in V$, or assigning a unique identifier for each node. We say $G = (V, E, X_V)$ and $G' = (V', E', X'_V)$ are isomorphic if there exists a relabeling of the nodes of G that produce the graph G'. In other words, they are isomorphic if there exists a permutation $\Pi \in S_n$ so that $\Pi V = V'$, $\Pi E = E'$ where $\Pi(u, v) := (\Pi u, \Pi v)$ and $\Pi X_V = X'_V$ where $\Pi X_v = X_{\Pi v}$.

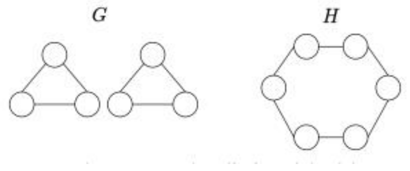


Figure 3: A canonical example of two regular non-isomorphic graphs

1. We have two graphs G, H without any node attributes shown as in figure 3. Can any message passing graph neural networks (e.g. GIN) distinguish these two graphs? Explain why. (4 pts)

2. We have introduced the 1-WL test for graph isomorphism problems. Here we extend the 1-WL test to k-WL test. The WL test keeps a state (or color) for every node (or tuples of nodes denoted by $\vec{v} = (v_1, \cdots, v_k) \in V_k$ in its k-dimensional versions). It refines the node states by aggregating the state information from their neighbors. In order to compute the update, WL uses an injective hash function defined in different objects modulo equivalence classes. In particular, for all $v, w \in V$ hash$(X_v) = hash(X_w)$ iff $X_v = X_w$. For $\vec{v} = (v_1, \cdots v_k)$, $\vec{v'} = (v'_1, \cdots v'_k)$ we define the hash function such that hash$(G[\vec{v}]) =$ hash$(G[\vec{v'}])$ iff (1) $X_{v_i} = X_{\vec{v'}} \forall i \in [k]$; and (2) $(v_i, v_j) \in E$ iff $(v'_i, v'_j) \in E', \forall i, j \in [k]$.We use the notation $\{\!\{\cdot\}\!\}$ to denote a multiset The algorithm is defined in Alg. 1. What is the smallest k that k-WL test can distinguish graph G and H? explain why. (6 pts)

```
Algorithm 1 k-WL (k ≥ 2)
Input: $G = (V, E, X_v)$
  $c^0_{\vec{v}} \leftarrow$ hash$(G[\vec{v}])$ for all $\vec{v} \in V^k$
  repeat
    $c^\ell_{\vec{v},i} \leftarrow \{\!\{c^{\ell-1}_w : w \in \mathcal{N}_i(\vec{v})\}\!\} \forall v \in V^k, i \in [k]$
    $c^\ell_{\vec{v}} \leftarrow$ hash $\left(c^{\ell-1}_{\vec{v}}, c^\ell_{\vec{v},1}, \dots c^\ell_{\vec{v},k}\right) \forall \vec{v} \in V^k$
  until $\left(c^\ell_{\vec{v}}\right)_{\vec{v} \in V^k} == \left(c^{\ell-1}_{\vec{v}}\right)_{\vec{v} \in V^k}$
  return $\{\!\{c^\ell_v : v \in V\}\!\}$
```

Answer: No, the MPGNN's representative power is bounded by 1-WL test or show the computation graph.
$k = 3$ Explanation: refer to paper https://arxiv.org/pdf/2201.07083.pdf

6 RGCN (10 points)

Relational Graph Convolutional Networks (RGCN) is related to a recent class of neural networks operating on graphs, and is developed specifically to handle the highly multi-relational data characteristic of realistic knowledge bases.

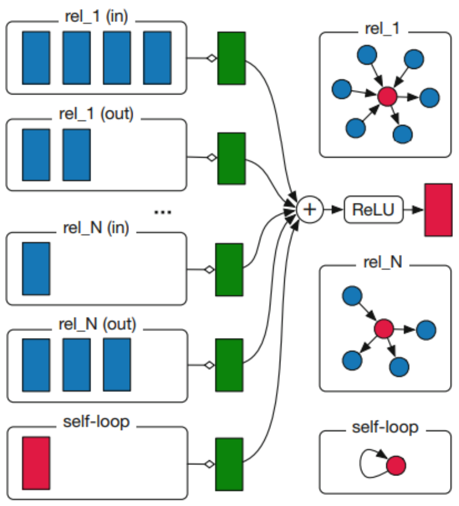


Figure 4: Single RGCN layer

The figure 4 demonstrates how to compute the update of a single graph node/entity (red) in the RGCN model. Activations (d-dimensional vectors) from neighboring nodes (dark blue) are gathered and then transformed for each relation type individually (for both in and outgoing edges). The resulting representation (green) is accumulated in a (normalized) sum and passed through an activation function (such as the ReLU). This per-node update can be computed in parallel with shared parameters across the whole graph. The definition of simple propagation model for calculating the forward-pass update of an entity or node denoted by v_i in a relational (directed and labeled) multi-graph:

$$h_i^{(l+1)} = \sigma\left(\sum_{r\in\mathcal{R}}\sum_{j\in\mathcal{N}_i^r}\frac{1}{c_{i,r}}W_r^{(l)}h_j^{(l)} + W_0^{(l)}h_i^{(l)}\right)$$

where $\mathcal{N}_i^r$ denotes the set of neighbor indices of node i under relation $r \in R$. $c_{i,r}$ is a problem-specific normalization constant that can either be learned or chosen in advance (here we choose $c_{i,r} = |\mathcal{N}_i^r|$). Here we consider a subgraph extracted from one knowledge graph as follow:

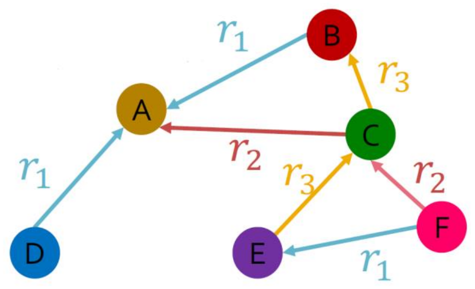


Figure 5: Subgraph of knowledge graph

Answer the following questions:

1. Draw the computation graph for target node B with 2-layer RGCN. (Hints: the computation graph should demonstrate different relations) (4 pts)
2. Assume that all relations in knowledge graph are shown in the subgraph figure 5, the input node feature dimension is 20. We use a 2-layer RGCN, and the output dimension for each layer is $[30, 10]$, please compute the number of all the trainable parameters in the model. (6 pts)

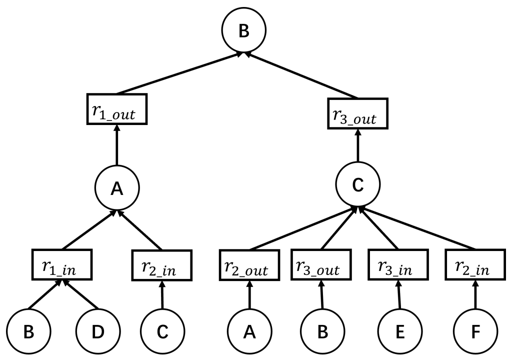


Figure 6: Answer to 1

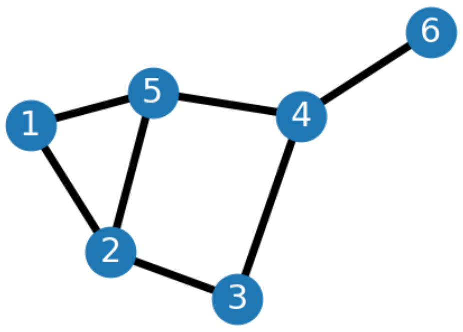
Answer:

1. Figure 6
2. $|\mathcal{R}| = 6$, $W_r^{(1)}, W_0^{(1)} \in \mathbb{R}^{20\times30}$, $W_r^{(2)}, W_0^{(2)} \in \mathbb{R}^{30\times10}$, total parameter: $7 \times (20 \times 30 + 30 \times 10) = 6300$

7 Node Centrality (10 Points)

One day, you join a peer to peer (P2P) file-sharing network $\mathcal{G}_{P2P}$, where each peer in the network connects to a number of other peers. You do not know who has the file you want. Therefore, you need to send a query to a node (peer) through the network to find out whether it has the target file.

A. Consider the following sub-graph $\mathcal{G}$ sampled from $\mathcal{G}_{P2P}$:



1. (3 pts) Write down the Normalized Degree Centrality for all the nodes in $\mathcal{G}$ (normalized by the maximum possible degree).
2. (3 pts) Write down the Normalized Closeness Centrality for all the nodes in $\mathcal{G}$.
3. (2 pts) Write down the Normalized Betweeness Centrality for node 4 and node 6.

B. When a peer p in the P2P network receives a query, it is required to act as follows:

- If p has the file, it should report back that it has the file and is ready to share it.
- If p does not have the file, then p will pass the query to all of its neighbors.
- The query will have a "time-to-live" stamp, which is decremented each time it is passed on. For instance, if time-to-live is set to 2, it will only be passed on to the original node's neighbors and those neighbor's neighbors (i.e., within 2-hops).

Now, you are told that the files you want are distributed randomly throughout the network $\mathcal{G}_{P2P}$. Also, you already know the structure of $\mathcal{G}_{P2P}$. However, the P2P protocol only allows you to send queries to your neighbors upon joining the network, and the time-to-live is set to 2. You are allowed to choose a node position inside $\mathcal{G}_{P2P}$ as your starting position upon joining. To maximize your chances of finding the files you would like to download, which one of the following three centrality measures of the starting node position would you consider and why: betweeness, or closeness? (2 pts)

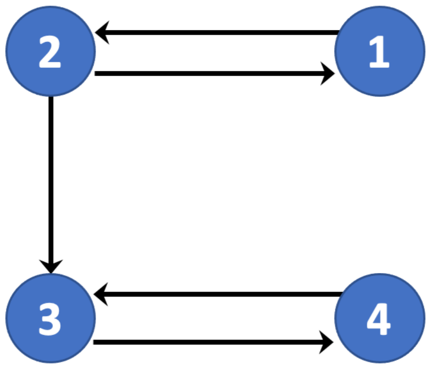
A.

1. (1: 0.4, 2: 0.6, 3: 0.4, 4: 0.6, 5: 0.6, 6: 0.2);
2. (1: 5/9; 2,3: 5/8; 4,5/7, 5: 5/7; 6: 5/11);
3. (4: 0.45, 6: 0.0);

B. Since my messages will only go out to a given radius specified by the time-to-live, I will want to maximize my closeness, so that my average shortest path to nodes in the rest of the network is the smallest and I have the greatest chance of my queries reaching them. (or any other ones with reasonable explanation)

8 PageRank (12 Points)

Consider the following directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, 2, 3, 4\}$ is the node set and $\mathcal{E} = \{(1, 2), (2, 1), (2, 3), (3, 4), (4, 3)\}$ is the edge set.



Answptthe following questions:

1. (1 pt) Write down the column stochastic matrix M for $\mathcal{G}$.
2. (3 pts) We employ the Power Iteration method to iteratively compute the target page rank vector r. Given an initial allocation of $r_0 = [\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}]^T$, stop when $\|r_k - r_{k-1}\|_1 < 0.1$, ($\|x\|_1 = \sum_{1 \le i \le N} |x_{[i]}|$ is the L_1 norm). Write down the result r_k.
3. (1 pt) Is there any spider trap in this example? If yes, write down the set of nodes which forms a spider trap. If no, explain why there is no spider trap.
4. (3 pts) Instead of the original PageRank, we now employ random teleports with $\beta = 0.8$. Given an initial allocation of $r_0 = [\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}]^T$, stop when $\|r_k - r_{k-1}\|_1 < 0.1$, ($\|x\|_1 = \sum_{1 \le i \le N} |x_{[i]}|$ is the L_1 norm). Write down the result r_k. Also, write down the resulting ranking of the four nodes.
5. (3 pts) Now, let's adjust β to 0.2 and do the iteration again. Write down the result r_k and the resulting ranking of the four nodes.

   (1 pt) What do you find compared to the result of question 4? Explain your findings.

Answers:

1. The target matrix M is shown in table 1.

0	0.5	0	0
1	0	0	0
0	0.5	0	1
0	0	1	0

Table 1: stochastic matrix M

2. $r_1 = [0.125, 0.25, 0.375, 0.25]^T$ (diff=0.25), $r_2 = [0.125, 0.125, 0.375, 0.375]^T$ (diff=0.25), $r_3 = [0.0625, 0.125, 0.4375, 0.375]^T$ (diff=0.125), $r_4 = [0.0625, 0.0625, 0.4375, 0.4375]^T$ (diff=0.125), $r_5 = [0.03125, 0.0625, 0.46875, 0.4375]^T$ (diff=0.0625).

3. Yes. Node $\{3, 4\}$ form a spider trap.

4. $r_1 = [0.15, 0.25, 0.35, 0.25]^T$ (diff=0.2), $r_2 = [0.15, 0.17, 0.35, 0.33]^T$ (diff=0.16), $r_3 = [0.118, 0.17, 0.382, 0.33]^T$ (diff=0.064). Ranking: $Node3 > Node4 > Node2 > Node1$.

5. $r_1 = [0.225, 0.25, 0.275, 0.25]$ (diff=0.05). Ranking: $Node3 > Node2 = Node4 > Node1$.

   Here the ranking of Node 2 equals that of Node 4, but previously Node 4 has higher ranking.

   Explanation: When β is smaller, we have higher probability to randomly jump to some other pages. Then, the ranking of Node 2 and 4 are less obvious. (Any other reasonable findings & explanations will also be given full marks.)

9 LightGCN and NGCF (13 Points)

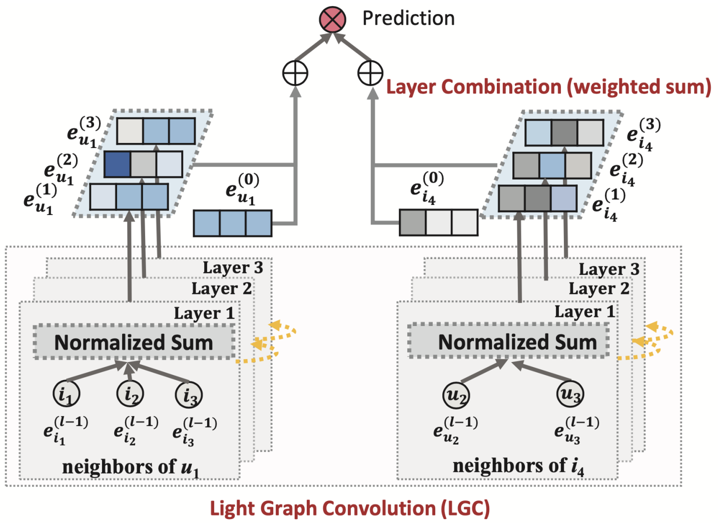


Figure 7: An illustration of LightGCN model architecture.

The Neural Graph Collaborative Filtering (NGCF) model was one of the strongest GCN-based recommendation model. On top of it, LightGCN simplifies the design of GCN to make it more concise and appropriate for recommendation. It includes only the essential component in GCN, neighborhood aggregation, for collaborative filtering. The design of the two models can be illustrated as follows:

1. Embedding layer: users and items are represented by d-dimension vectors. Both embeddings are trainable. Let $e_u^{(0)}$ denote the embedding of user u and $e_i^{(0)}$ denote the embedding of item i.

2. Embedding propagation layer: this layer makes use of graph structures to refine user and item embeddings. **This is where NGCF and LightGCN differs.**

   - The NGCF model leverages the user-item interaction graph to propagate embeddings as:

$$e_u^{(l+1)} = \sigma\big(W_1^{(l+1)} e_u^{(l)} + \sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{|N_u|\,|N_i|}} (W_1^{(l+1)} e_i^{(l)} + W_2^{(l+1)} (e_i^{(l)} \odot e_u^{(l)}))\big),$$

$$e_i^{(l+1)} = \sigma\big(W_1^{(l+1)} e_i^{(l)} + \sum_{u \in \mathcal{N}_i} \frac{1}{\sqrt{|N_u|\,|N_i|}} (W_1^{(l+1)} e_u^{(l)} + W_2^{(l+1)} (e_u^{(l)} \odot e_i^{(l)}))\big),$$

   where $e_u^{(l)}$ and $e_i^{(l)}$ respectively denote the refined embedding of user u and item i after l layers propagation ($l = 0, 1, 2, ..., L$), σ is the nonlinear activation function, $\mathcal{N}_u$ denotes the set of items that are interacted by user u, $\mathcal{N}_i$ denotes the set of users that interact with item i, and $W_1^{(l+1)}$ and $W_2^{(l+1)}$ are trainable weight matrices to perform feature transformation in each layer. By propagating L layers, NGCF obtains $L+1$ embeddings to describe a user $(e_u^{(0)}, e_u^{(1)}, ..., e_u^{(L)})$ and an item $(e_i^{(0)}, e_i^{(1)}, ..., e_i^{(L)})$. The representations learned by different layers are concatenated to as the final embeddings: $e_u = e_u^{(0)}|| \cdots ||e_u^{(L)}$,$e_i = e_i^{(0)}|| \cdots ||e_i^{(L)}$where $\|$ is the concatenation operation.

- The LightGCN model learns user and item representation by smoothing features over the graph with the simple weighted sum aggregator. It abandons the use of feature transformation and nonlinear activation. The corresponding propagation rule in LightGCN is defined as:

$$e_u^{(l+1)} = \sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{|N_u|}\sqrt{|N_i|}} e_i^{(l)},$$

$$e_i^{(l+1)} = \sum_{u \in \mathcal{N}_i} \frac{1}{\sqrt{|N_i|}\sqrt{|N_u|}} e_u^{(l)},$$

  After L layers, LightGCN obtains $L+1$ embeddings to describe a user $(e_u^{(0)}, e_u^{(1)}, ..., e_u^{(L)})$ and an item $(e_i^{(0)}, e_i^{(1)}, ..., e_i^{(L)})$. They are combined as follows to get the final user and item embeddings e_u and e_i:

$$e_u = \sum_{l=0}^{L} \alpha_l e_u^{(l)}; \qquad e_i = \sum_{l=0}^{L} \alpha_l e_i^{(l)},$$

  where $\alpha_l \geq 0$ is a hyper-parameter that denotes the importance of the k-th layer embedding in constituting the final embedding.

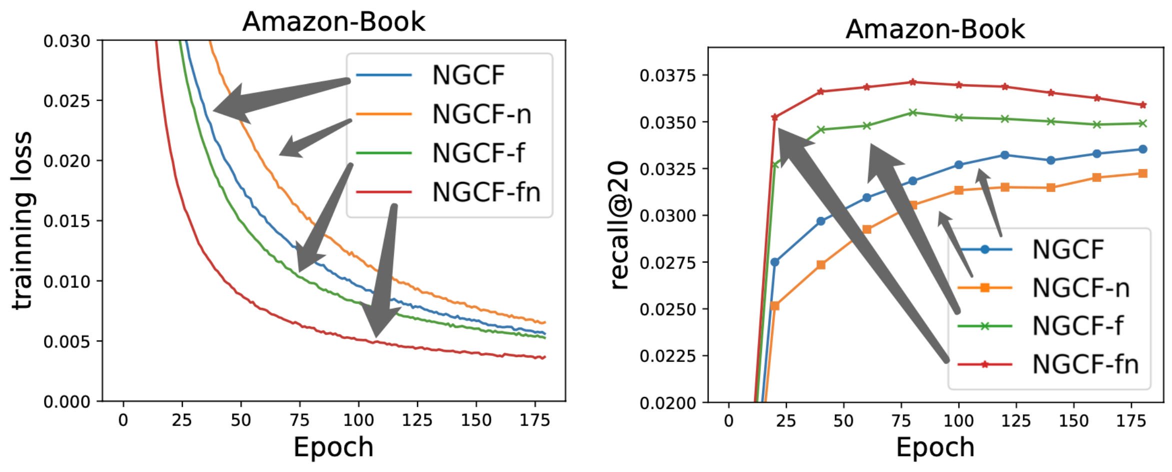
3. Model prediction: The model prediction is defined as the inner product of user and item final representations: $\hat{y}_{(u,i)} = {e_u}^\top e_i$.

Assume that:

- There are 400 unique users and 1000 unique items in the training set.
- The dimensions of user and item embeddings are both 200.
- The number of embedding propagation layer (L) is 3.
- For NGCF, the dimension of intermediate output of each embedding propagation layer is [100, 60, 30] respectively.

1. (5 pts) Compute the number of all trainable parameters of NGCF.
2. (2 pts) Compute the number of all trainable parameters of LightGCN.
3. (2 pts) What difference do you find in the result? Which design(s) do you think causes the difference?
4. (4 pts) To further explore the effect of nonlinear activation and feature transformation, researchers implemented three simplified variants of NGCF: (1) NGCF-f, which removes the feature transformation matrices $W_1^{(l)}$ and $W_2^{(l)}$; (2) NGCF-n, which removes

the non-linear activation function σ; (3) NGCF-fn, which removes both the feature transformation matrices and non-linear activations. Based on the experimental results shown in Figure 8, could you elaborate on the effect of (a) feature transformation and (b) nonlinear activation in NGCF by comparing the curves of different variants?



Training loss on Amazon-Book   Testing recall on Amazon-Book

Figure 8: Training curves (training loss and testing recall) of NGCF and its three simplified variants. We add arrows to help you identify which curve comes from which variant.

Solution:

1. Embedding: $(400 + 1000) \times 200 = 280000$

   Projection: $W_1^{(1)} : (100, 200)$, $W_2^{(1)} : (100, 200)$

   $W_1^{(2)} : (60, 100)$, $W_2^{(2)} : (60, 100)$

   $W_1^{(3)} : (30, 60)$, $W_2^{(3)} : (30, 60)$

   All parameters: 335600.

2. The trainable parameters of LightGCN are in the user and item embeddings. All parameters $= 280000$

3. LightGCN has relatively smaller number of parameters.

   LightGCN abandons the use of feature transformation and nonlinear activation. Therefore, the trainable weights in the GNNs used in NGCF are left out.

4.
   - Adding feature transformation has negative effect on NGCF, since removing it in both models of NGCF and NGCF-n improves the performance.
   - Adding nonlinear activation slightly affects the performance when feature transformation is included, but it imposes negative effect when feature transformation is disabled.