COMP4222: Machine Learning with Structured Data

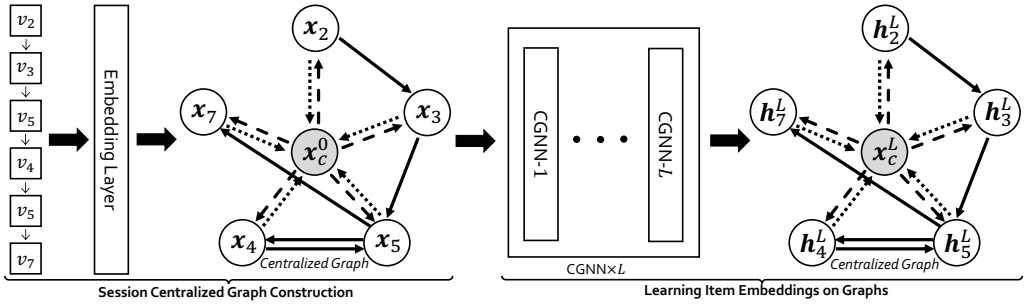
Fall 2024

Homework 3

Release Date: *Nov 19, 2024* Due Date: *Nov 30, 2024* Max mark: *50*

25 points

I. Session-based Recommendation is a task that predicts users' next actions based on their sequential behaviors in the current session. Formally, let V denote all unique items in all sessions. Given a session as $S = \{v_1, v_2, \cdots, v_n\}$ consisting of n sequential items, where $v_i \in V$ is the i-th item in the session, the goal of Session-based Recommendation is to predict v_{n+1} to click. In this question, we introduce the *Centralized Graph Neural Network (CGNN)* model to solve this problem.



Session Centralized Graph Construction For each session $S = \{v_1, v_2, \ldots, v_n\}$, we construct a centralized graph to represent the transition relationship among the items in the session. Specifically, each session is represented as $G_s = \{\mathcal{V}_s, \mathcal{E}_s\}$, where $\mathcal{V}_s = \{\{x_1, x_2, \ldots, x_m\}, x_c\}$ denotes the node set of the centralized graph. Here, the first part $\{x_1, x_2, \ldots, x_m\}$ indicates all unique items in the session, which we call *satellite nodes*, and the latter part x_c is the newly added *central node*.

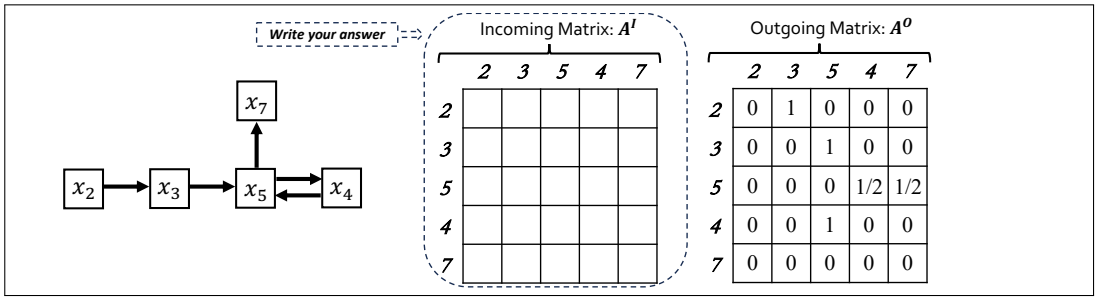
$\mathcal{E}_s$ is the edge set in the centralized graph, which consists of two types of edges, i.e., central connections and satellite connections. For the central connections, we add a bidirectional edge between the central node and each satellite node. For the satellite connections, the edge $(x_i, x_j) \in \mathcal{E}_s$, i.e., the solid lines in the centralized graphs mean that the user clicks item x_j after clicking x_i. In this way, the adjacent relationship between two items in the session can be represented by an incoming matrix $\boldsymbol{A}^I$ and an outgoing matrix $\boldsymbol{A}^O$:

$$\boldsymbol{A}^I_{(i,j)} = \begin{cases} 1/in_i, (x_j, x_i) \in \mathcal{E}_s \\ 0, (x_j, x_i) \notin \mathcal{E}_s \end{cases}, \boldsymbol{A}^O_{(i,j)} = \begin{cases} 1/out_i, (x_i, x_j) \in \mathcal{E}_s \\ 0, (x_i, x_j) \notin \mathcal{E}_s \end{cases} \tag{1}$$

where out_i represents the number of outbound edges for x_i and in_i represents the number of inbound edges for x_i. Please note that **only satellite connections** need to be considered in the adjacency matrices here.

Given an example session $S = \{v_2, v_3, v_5, v_4, v_5, v_7\}$, we can construct the incoming matrix and the outgoing matrix in the following figure.

3 points **(1)** Construct the matrix $\boldsymbol{A}^I$ for the example above. Please write the answer directly into the diagram above.



Learning Item Embeddings on Graphs We initialize the representation of satellite nodes and the central node. For the satellite nodes, we directly take the trainable embeddings of unique items in the session as the satellite nodes' representation:

$$\boldsymbol{h}^0 = [\boldsymbol{x}_1^0, \boldsymbol{x}_2^0, \ldots, \boldsymbol{x}_m^0], \tag{2}$$

where $\boldsymbol{x}_i^0 \in \mathbb{R}^d$. As for the central node $\boldsymbol{x}_c^0$, we apply an average pooling on the satellite nodes to get the initialization of the central node, i.e.,

$$\boldsymbol{x}_c^0 = \frac{1}{m}\sum_{i=1}^{m} \boldsymbol{x}_i^0 \tag{3}$$

To learn a node representation according to the graph, we update the satellite nodes and the central node as follows. First, we consider the information from adjacent nodes. For each satellite node x_i in the centralized graph at layer l, we utilize the incoming and outgoing matrices to obtain the propagation information as follows:

$$\begin{aligned}\boldsymbol{a}_i^l &= \textsc{Concat}\left(\boldsymbol{A}_i^I\left(\boldsymbol{h}^{l-1\top}\boldsymbol{W}^I + \boldsymbol{b}^I\right), \boldsymbol{A}_i^O\left(\boldsymbol{h}^{l-1\top}\boldsymbol{W}^O + \boldsymbol{b}^O\right)\right)^\top \\ &= \textsc{Concat}\left(\boldsymbol{A}_i^I\left([\boldsymbol{x}_1^{l-1}, \boldsymbol{x}_2^{l-1}, \ldots, \boldsymbol{x}_m^{l-1}]^\top\boldsymbol{W}^I + \boldsymbol{b}^I\right), \boldsymbol{A}_i^O\left([\boldsymbol{x}_1^{l-1}, \boldsymbol{x}_2^{l-1}, \ldots, \boldsymbol{x}_m^{l-1}]^\top\boldsymbol{W}^O + \boldsymbol{b}^O\right)\right)^\top\end{aligned} \tag{4}$$

where $\boldsymbol{A}_i^I, \boldsymbol{A}_i^O \in \mathbb{R}^{1\times m}$ are the incoming the outgoing weights for node x_i, i.e., the i-th row in the incoming matrix and the outgoing matrix. $\boldsymbol{W}^I, \boldsymbol{W}^O$ are the learnable parameters for the incoming edges and the outgoing edges, respectively, while $\boldsymbol{b}^I, \boldsymbol{b}^O \in \mathbb{R}^d$ are the bias vectors. Hence we obtain $\boldsymbol{a}_i^l \in \mathbb{R}^{2d}$ to represent the propagation information for node $\boldsymbol{x}_i$.

2 points **(2)** How many parameters are there in matrices $\boldsymbol{W}^I$ and $\boldsymbol{W}^O$ respectively? (Note: The answer is a polynomial with respect to d.)

Then we feed $\boldsymbol{a}_i^l$ and node x_i's previous state $\boldsymbol{h}_i^{l-1}$, i.e., the i-th column in $\boldsymbol{h}^{l-1}$, into the

gated graph neural networks:

$$\begin{aligned}
\boldsymbol{z}_i^l &= \sigma\left(\boldsymbol{W}_z \boldsymbol{a}_i^l + \boldsymbol{U}_z \boldsymbol{h}_i^{l-1}\right), \\
\boldsymbol{r}_i^l &= \sigma\left(\boldsymbol{W}_r \boldsymbol{a}_i^l + \boldsymbol{U}_r \boldsymbol{h}_i^{l-1}\right), \\
\tilde{\boldsymbol{h}}_i^l &= \tanh\left(\boldsymbol{W}_h \boldsymbol{a}_i^l + \boldsymbol{U}_h\left(\boldsymbol{r}_i^l \odot \boldsymbol{h}_i^{l-1}\right)\right), \\
\hat{\boldsymbol{h}}_i^l &= \left(1 - \boldsymbol{z}_i^l\right) \odot \boldsymbol{h}_i^{l-1} + \boldsymbol{z}_i^l \odot \tilde{\boldsymbol{h}}_i^l,
\end{aligned} \tag{5}$$

where $\boldsymbol{W}_z, \boldsymbol{W}_r, \boldsymbol{W}_h$ and $\boldsymbol{U}_z, \boldsymbol{U}_r, \boldsymbol{U}_h$ are trainable parameters in the network. σ represents the *sigmoid* function and $\odot$ is the element-wise multiplication. Note that $\boldsymbol{z}_i^l, \boldsymbol{r}_i^l, \tilde{\boldsymbol{h}}_i^l, \hat{\boldsymbol{h}}_i^l \in \mathbb{R}^d$.

6 points **(3)** How many parameters are there in $\boldsymbol{W}_z, \boldsymbol{W}_r, \boldsymbol{W}_h, \boldsymbol{U}_z, \boldsymbol{U}_r$ and $\boldsymbol{U}_h$, respectively? (Note: The answer is a polynomial with respect to d.)

Next, we consider information from the central node. In CGNN, the central node can represent the overall information from all items in a session. For each satellite node, we apply a gating network to decide how much information should be propagated from the central node and the adjacent nodes, respectively. Specifically, we calculate the similarity $\alpha_i^l \in \mathbb{R}$ of each satellite node x_i and the central node x_c with a self-attention mechanism as:

$$\alpha_i^l = \frac{\left(\boldsymbol{W}_{q1} \hat{\boldsymbol{h}}_i^l\right)^\top \boldsymbol{W}_{k1} \boldsymbol{x}_c^{l-1}}{\sqrt{d}} \tag{6}$$

where $\boldsymbol{W}_{q1}, \boldsymbol{W}_{k1}$ are trainable parameters. Then we apply a gating network to selectively integrate the information from the adjacent node $\hat{\boldsymbol{h}}_i^l$ and from the central node $\boldsymbol{x}_c^{l-1}$ as follows:

$$\boldsymbol{h}_i^l = \left(1 - \alpha_i^l\right) \hat{\boldsymbol{h}}_i^l + \alpha_i^l \boldsymbol{x}_c^{l-1} \tag{7}$$

For updating the central node, we introduce a self-attention mechanism to assign different degrees of importance to the satellite nodes $\boldsymbol{\beta} \in \mathbb{R}^m$ as follows:

$$\boldsymbol{\beta} = \operatorname{softmax}\left(\frac{\left(\boldsymbol{W}_{k2} \boldsymbol{h}^l\right)^\top \boldsymbol{W}_{q2} \boldsymbol{x}_c^{l-1}}{\sqrt{d}}\right) \tag{8}$$

where $\boldsymbol{W}_{q2}, \boldsymbol{W}_{k2}$ are the trainable parameters. Then, we combine the satellite nodes using a linear combination as a new representation of the central node:

$$\boldsymbol{x}_c^l = \boldsymbol{h}^l \boldsymbol{\beta} \tag{9}$$

By updating the satellite nodes and the central node iteratively, we can obtain the item rep-

resentations. Moreover, to accurately represent the transition relationship between items, multi-layer CGNNs can be stacked, where the l-layer of CGNN can be denoted as:

$$\boldsymbol{h}^l, \boldsymbol{x}_c^l = \text{CGNN}\left(\boldsymbol{h}^{l-1}, \boldsymbol{x}_c^{l-1}, \boldsymbol{A}^I, \boldsymbol{A}^O\right) \tag{10}$$

Finally, we obtain the new representations of the satellite nodes $\boldsymbol{h}^L$ and the central node $\boldsymbol{x}_c^L$, as we employ an L-layer CGNN.

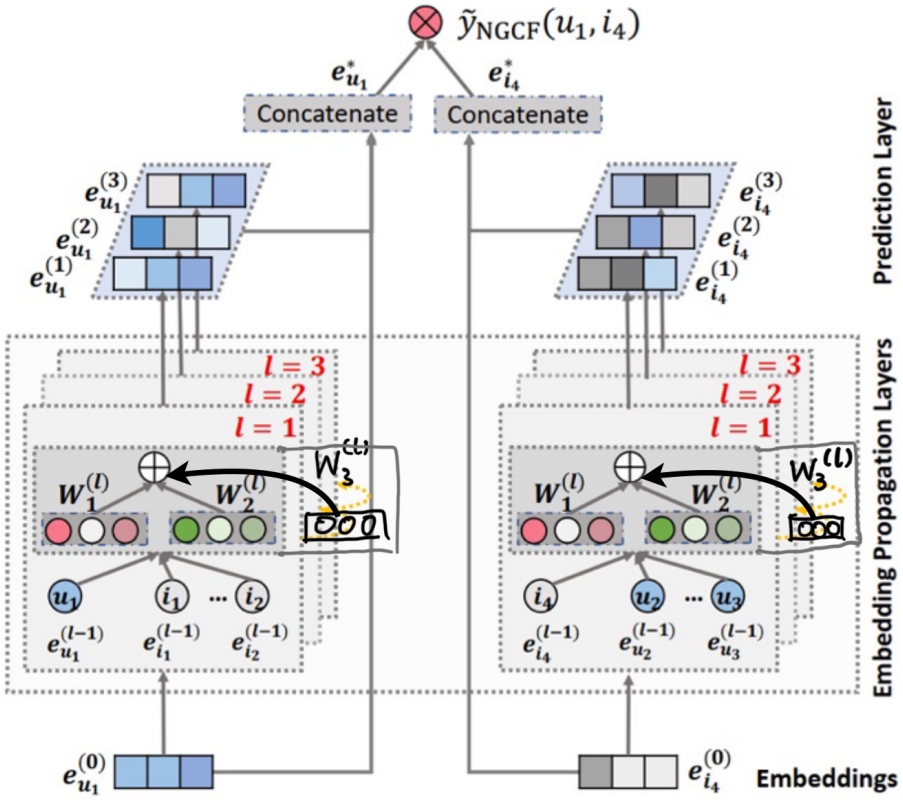
(4) Please briefly answer the following two questions.

2 points **(i)** If you want to demonstrate the effectiveness of CGNN, what experiments would you conduct to illustrate it?

4 points **(ii)** When L is large, multiple layers of graph layers propagate a substantial amount of information between nodes, which can easily lead to overfitting. What can be done to improve the model? Please describe two potential approaches.

8 points **(5)** We define $\boldsymbol{W}_{k1} \in \mathbb{R}^{d\times d}$, $\boldsymbol{W}_{k2} \in \mathbb{R}^{2d\times d}$. Given $n = 27, d = 100, L = 1$, and V consists of 2,520 unique items, please compute the number of all trainable parameters of this network.

25 points **II**. Consider a variant of the neural graph collaborative filtering (NGCF) model.



Here are some structural explanations of the NGCF-variant:

1. Embedding layer: we describe the user and item with a d-dimension vector. Both embeddings are trainable.
2. Embedding propagation layer: we build upon the message-passing architecture of GNNs in order to capture CF signal along the graph structure and refine the embeddings of users and items.

   First-order propagation. We build upon preference basis to perform embedding propagation between the connected users and items, formulating the process with two major operations: message construction and message aggregation.

   Message construction. For a connected user-item pair (u, i), we define the message from i to u as:

$$\boldsymbol{m}_{u \leftarrow i} = \frac{1}{\sqrt{|\mathcal{N}_u|\,|\mathcal{N}_i|}} \left(\boldsymbol{W}_1 \boldsymbol{e}_i + \boldsymbol{W}_2 \left(\boldsymbol{e}_i \odot \boldsymbol{e}_u\right) + \boldsymbol{W}_3 \left(\boldsymbol{e}_i || \boldsymbol{e}_u\right)\right),$$

   where $\boldsymbol{W}_1, \boldsymbol{W}_2, \boldsymbol{W}_3$ are the trainable weight matrices to distill useful information for propagation, $\odot$ denotes the element-wise product, $||$ denotes the concatenation operation. $\mathcal{N}_u$ and $\mathcal{N}_i$ denote the first-hop neighbors of user u and item i.

   Message Aggregation. In this stage, we aggregate the messages propagated from u's neighborhood to refine u's representation. Specifically, we define the aggregation function as:

$$\boldsymbol{e}_u^{(1)} = \text{ReLU}\left(\boldsymbol{W}_1 \boldsymbol{e}_u + \sum_{i \in N_u} \boldsymbol{m}_{u \leftarrow i}\right)$$

where $\boldsymbol{e}_u^{(1)}$ denotes the representation of user u obtained after the first embedding propagation layer. $\boldsymbol{W}_1$ is the weight matrix shared with the one used in message construction. Analogously, we can obtain the representation $\boldsymbol{e}_i^{(1)}$ for the item i by propagating information from its connected users using a shared weight matrix.

High-order propagation. With the representations augmented by first-order connectivity modeling, we can stack more embedding propagation layers to explore the high-order connectivity information. In the l-th step, the representation of user u is recursively formulated as:

$$\boldsymbol{e}_u^l = \text{ReLU}\left(\boldsymbol{m}_{u\leftarrow u}^{(l)} + \sum_{i\in N_u} \boldsymbol{m}_{u\leftarrow i}^{(l)}\right),$$

wherein the messages being propagated are defined as:

$$\boldsymbol{m}_{u\leftarrow i}^{(l)} = \frac{1}{\sqrt{|\mathcal{N}_u|\,|\mathcal{N}_i|}}\left(\boldsymbol{W}_1^{(l)}\boldsymbol{e}_i^{(l-1)} + \boldsymbol{W}_2^{(l)}\left(\boldsymbol{e}_i^{(l-1)} \odot \boldsymbol{e}_u^{(l-1)}\right) + \boldsymbol{W}_3^{(l)}\left(\boldsymbol{e}_i^{(l-1)} || \boldsymbol{e}_u^{(l-1)}\right)\right),$$

$$\boldsymbol{m}_{u\leftarrow u}^{(l)} = \boldsymbol{W}_1^{(l)}\boldsymbol{e}_u^{(l-1)},$$

where subscript l denotes the variable in layer l.

Model prediction: After propagating with L layers, we obtain multiple representations for the user and item. We concatenate the representations learned by different layers to get the final embeddings: $\boldsymbol{e}_u^* = \boldsymbol{e}_u^{(0)} || \ \cdots || \boldsymbol{e}_u^{(L)}$,$\boldsymbol{e}_i^* = \boldsymbol{e}_i^{(0)} || \cdots || \boldsymbol{e}_i^{(L)}$where $\|$ is the concatenation operation. Finally, we conduct the inner product to estimate the user's preference towards the target item: $\hat{y}_{NGCF(u,i)} = \boldsymbol{e}_u^{*\top}\boldsymbol{e}_i^*$.

Here are some assumptions:

1. There are 1,000 unique users and 520 unique items in the training set.
2. The dimensions of user and item embeddings are both 100.
3. The number of embedding propagation layer is 3.
4. The dimension of intermediate output of each embedding propagation layer is [80, 40, 20] respectively.

Compute the number of all trainable parameters of this model.