Solal Danan
solal.danan@polytechnique.edu

Lab session # 5
ALTEGRAD 2024

11/26/24

1 Question 1

To generalize the DeepWalk architecture for **directed graphs**, the random walk process must account for edge directions. In a directed graph, edges have a defined direction from one node to another, so during a random walk, only the *outgoing neighbors* of a node should be considered. Outgoing neighbors are nodes that can be reached directly via edges originating from the current node. This ensures that the walk respects the directional flow of the graph, preserving the graph's inherent structure. This approach is particularly relevant for web graphs or social networks, where edge direction carries critical information about relationships, such as hyperlinks or follower connections.

For **weighted graphs**, the random walk must incorporate edge weights, which represent the strength or importance of connections. Instead of selecting neighbors uniformly, a *weighted random walk* assigns probabilities to edges proportional to their weights, guiding the traversal based on connection strength. These weights might reflect probabilities, costs, or capacities, depending on the application. For example, in transportation networks or knowledge graphs, this adaptation ensures the random walk captures the relative significance of paths in the graph, leading to embeddings that more accurately represent the graph's structure.

2 Question 2

DeepWalk works by performing random walks on the graph and learning node embeddings using a Skip-gram model, which is analogous to word embeddings in NLP. The random walks generate sequences of nodes (similar to sentences), and the Skip-gram model learns embeddings that predict the probability of co-occurrence between nodes in these sequences.

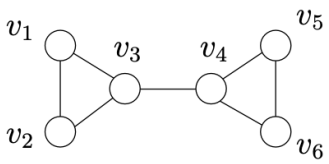

Figure 1: A synthetic graph G.

The two embedding matrices X_1 and X_2 represent the embeddings of the graph G in 2-dimensional space, obtained by running the DeepWalk algorithm twice. Each row in these matrices corresponds to the embedding of a node v_i in G. The embeddings generated by DeepWalk capture the structural relationships between nodes in the graph, aiming to place nodes with similar neighborhoods close to each other in the embedding space.

The two embeddings X_1 and X_2 are related through a **rotation** or a **reflection** because DeepWalk embeddings are not unique and are determined up to an orthogonal transformation. This behavior arises from the unconstrained nature of the embedding space: the optimization process in the Skip-gram model starts from different random initializations, causing the final embeddings to differ in orientation. While DeepWalk preserves relative distances and angles between nodes, it does not fix their absolute position or orientation in the vector space. As a result, embeddings like X_1 and X_2 encode the same graph structure but appear reflected due to the stochasticity and flexibility of the learning process.

3 Question 3

In the GCN architecture implemented in Task 10, each message passing layer aggregates information from the immediate neighbors of a node. Since the architecture consists of two message passing layers, the receptive field of each node includes all nodes within two hops, meaning all nodes connected to it by at most two edges. The prediction $\hat{Y}_i$ for a given node i therefore incorporates features from all nodes in its 2-hop neighborhood.

In general, for a GCN with k message passing layers, each layer extends the receptive field of a node by one hop. The first layer aggregates features from the 1-hop neighbors, the second from 2-hop neighbors, and so on. Thus, after k layers, the maximal distance of nodes considered in the calculation of $\hat{Y}_i$ is k edges. This means the receptive field of a node spans all nodes within its k-hop neighborhood in the graph.

4 Question 4

We compute the matrix Z^1 for both graphs step-by-step. The given weight matrices are:

$$W^0 = \begin{bmatrix} -0.8 & 0.5 \end{bmatrix}, \quad W^1 = \begin{bmatrix} 0.1 & 0.3 & -0.05 \\ -0.4 & 0.6 & 0.5 \end{bmatrix}.$$

The feature matrix X for both graphs is: $X = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}^\top$.

Complete Graph K_4

1. The adjacency matrix (which represents the connections between nodes in the graph) for K_4 is:

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

Adding self-loops and normalizing:

$$\tilde{A} = A + I = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}, \quad \hat{A} = \frac{1}{4}\tilde{A}.$$

2. First Layer Output: $Z^0 = f\left(\hat{A}XW^0\right)$.

$$\hat{A}X = \frac{1}{4}\begin{bmatrix} 4 \\ 4 \\ 4 \\ 4 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}, \quad Z^0 = f\left(\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}\begin{bmatrix} -0.8 & 0.5 \end{bmatrix}\right) = f\left(\begin{bmatrix} -0.8 & 0.5 \\ -0.8 & 0.5 \\ -0.8 & 0.5 \\ -0.8 & 0.5 \end{bmatrix}\right) = \begin{bmatrix} 0 & 0.5 \\ 0 & 0.5 \\ 0 & 0.5 \\ 0 & 0.5 \end{bmatrix}.$$

3. Second Layer Output: $Z^1 = f\left(\hat{A}Z^0W^1\right)$.

$$\hat{A}Z^0 = \begin{bmatrix} 0 & 0.5 \\ 0 & 0.5 \\ 0 & 0.5 \\ 0 & 0.5 \end{bmatrix}, \quad Z^1 = f\left(\begin{bmatrix} 0 & 0.5 \\ 0 & 0.5 \\ 0 & 0.5 \\ 0 & 0.5 \end{bmatrix}\begin{bmatrix} 0.1 & 0.3 & -0.05 \\ -0.4 & 0.6 & 0.5 \end{bmatrix}\right) = f\left(\begin{bmatrix} -0.2 & 0.3 & 0.25 \\ -0.2 & 0.3 & 0.25 \\ -0.2 & 0.3 & 0.25 \\ -0.2 & 0.3 & 0.25 \end{bmatrix}\right) = \begin{bmatrix} 0 & 0.3 & 0.25 \\ 0 & 0.3 & 0.25 \\ 0 & 0.3 & 0.25 \\ 0 & 0.3 & 0.25 \end{bmatrix}.$$

Due to the symmetry of the graph K_4, all nodes have identical representations in Z^1.

Star Graph S_4

1. The adjacency matrix for the star graph S_4 is:

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

adding self-loops:

$$\tilde{A} = A + I = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}.$$

The degree matrix is $D = \text{diag}(4, 2, 2, 2)$, and the normalized adjacency matrix is:

$$\hat{A} = D^{-1/2}\tilde{A}D^{-1/2} = \begin{bmatrix} 0.5 & 0.3536 & 0.3536 & 0.3536 \\ 0.3536 & 0.5 & 0 & 0 \\ 0.3536 & 0 & 0.5 & 0 \\ 0.3536 & 0 & 0 & 0.5 \end{bmatrix}.$$

2. First Layer Output: $Z^0 = f\left(\hat{A}XW^0\right)$.

$$\hat{A}X = \begin{bmatrix} 1.56 \\ 0.8536 \\ 0.8536 \\ 0.8536 \end{bmatrix}, Z^0 = f\left(\begin{bmatrix} 1.56 \\ 0.8536 \\ 0.8536 \\ 0.8536 \end{bmatrix} \begin{bmatrix} -0.8 & 0.5 \end{bmatrix}\right) = f\left(\begin{bmatrix} -1.248 & 0.78 \\ -0.68288 & 0.4268 \\ -0.68288 & 0.4268 \\ -0.68288 & 0.4268 \end{bmatrix}\right) = \begin{bmatrix} 0 & 0.78 \\ 0 & 0.4268 \\ 0 & 0.4268 \\ 0 & 0.4268 \end{bmatrix}.$$

3. Second Layer Output: $Z^1 = f\left(\hat{A}Z^0W^1\right)$.

$$\hat{A}Z^0 = \begin{bmatrix} 0 & 0.843 \\ 0 & 0.489 \\ 0 & 0.489 \\ 0 & 0.489 \end{bmatrix} Z^1 = f\left(\begin{bmatrix} 0 & 0.843 \\ 0 & 0.489 \\ 0 & 0.489 \\ 0 & 0.489 \end{bmatrix} \begin{bmatrix} 0.1 & 0.3 & -0.05 \\ -0.4 & 0.6 & 0.5 \end{bmatrix}\right) = f\left(\begin{bmatrix} -0.337 & 0.506 & 0.421 \\ -0.196 & 0.294 & 0.245 \\ -0.196 & 0.294 & 0.245 \\ -0.196 & 0.294 & 0.245 \end{bmatrix}\right) = \begin{bmatrix} 0 & 0.506 & 0.421 \\ 0 & 0.294 & 0.245 \\ 0 & 0.294 & 0.245 \\ 0 & 0.294 & 0.245 \end{bmatrix}$$

Due to the structure of S_4, the central node (node 1) has a different representation in Z^1 compared to the leaf nodes (nodes 2, 3, 4), which share identical representations. Z^1 reflect the central and leaf node differences:

$$Z^1_{\text{central}} \neq Z^1_{\text{leaf}}.$$

Observation

In K_4, all nodes have identical features in Z^1 due to symmetry. In S_4, the central node and leaf nodes have distinct features, reflecting their different structural roles.

Random Features

If the node features X were randomly sampled from a uniform distribution, the observed structure in the representations Z^1 would no longer exhibit symmetry across structurally identical nodes, as the initial features would introduce variability that propagates through the GCN layers.

The t-SNE visualization shows distinct clusters of nodes in the test set, indicating that the GNN effectively learned meaningful representations that separate nodes based on their class labels.