Boubacar SOW
boubacar.sow@ens-paris-saclay.fr

Lab session # 5
ALTEGRAD 2023

11/21/23

1 Node embedding and cosine similarity

The DeepWalk algorithm generates node embeddings by simulating random walks on the graph and then applying the Skip-Gram model to these walks. The Skip-Gram model tends to place nodes that co-occur frequently in the same walks close together in the embedding space.
Given this graph structure, where each connected component is a complete graph K_2, we can expect the following:
When we're inside a connected component: The two nodes in the same component (our K_2) are always going to be visited together in every stroll that includes either of them. This is because they're directly connected to each other and not connected to any other node. So, their embeddings should end up being super similar, which means a high cosine similarity.

When we're looking at different connected components: Nodes in different components will never show up in the same stroll, because there are no edges between components. So, the Skip-Gram model doesn't have any reason to put them close together in the embedding space. Their embeddings could end up being orthogonal (at right angles to each other) or even randomly oriented with respect to each other, which means a low cosine similarity

2 Complexity of Deepwalk and spectral embedding techniques

- **Deepwalk**
  Pimentel et. al. stated that DeepWalk algorithm has a time complexity bounded by $\mathcal{O}(|V|)$ [1]
- **Spectral embeddings** The time complexity of spectral embeddings is dominated by the eigen-decomposition of the Laplacian matrix which is $\mathcal{O}(|V|^3)$.

$|V|$ is the number of nodes.

3 Effect on the architecture if no self-loops were added

In our implementation, we added an identiy matrix to the original adjacency matrix. This operation results in a new adjacency matrix $\tilde{A}$. The effect of this operation is that each node is now considered a neighbor of itself which ensures that a node's own features are included when aggregating feature information from its neighbors. If no self-loops where added, the hidden states would be affected in the following ways:

1. **Single Layer GNN**: In a single layer GNN, a node's hidden state is updated based on its neighbors' features. Without self-loops, the node's own features would be excluded from this update. Consequently, the next hidden state would lack information about the node's features, which could result in losing important details, especially if the node's own features are crucial for the task.
2. **Two-Layer GNN**: In a two-layer GNN, hidden states undergo two updates. The first update is similar to a single layer GNN. Without self-loops, the node's own features would be omitted from both updates. As a result, the second hidden state would lack information about the node's features, relying only on its neighbors and potentially their neighbors. This omission may lead to a further loss of information and restrict the GNN's ability to capture more intricate patterns in the graph.

4 Star and cycle graphs

First, let's define the adjacency matrices for the two graphs:

1. **Adjacency Matrices**: We start with the adjacency matrices for the star graph S4 and the cycle graph C4.
   For the star graph S4:

$$A_{\text{star}} = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

For the cycle graph C4:

$$A_{\text{cycle}} = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

2. **Degree Matrices**: Next, we compute the degree matrices $\tilde{D}$ for both graphs.

   For the star graph S4:

$$\tilde{D}_{\text{star}} = \begin{bmatrix} 4.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 2.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 2.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 2.00 \end{bmatrix}$$

   For the cycle graph C4:

$$\tilde{D}_{\text{cycle}} = \begin{bmatrix} 3.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 3.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 3.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 3.00 \end{bmatrix}$$

3. **Inverse Square Root of Degree Matrices**: We then compute the inverse square root of the degree matrices $\tilde{D}^{-\frac{1}{2}}$ for both graphs.

   For the star graph S4:

$$\tilde{D}_{\text{star}}^{-\frac{1}{2}} = \begin{bmatrix} 0.50 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.71 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.71 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.71 \end{bmatrix}$$

   For the cycle graph C4:

$$\tilde{D}_{\text{cycle}}^{-\frac{1}{2}} = \begin{bmatrix} 0.58 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.58 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.58 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.58 \end{bmatrix}$$

4. **Normalized Adjacency Matrices**: Next, we normalize the adjacency matrices using the formula

$$\hat{A} = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$$

   where $\tilde{A} = A + I$, and $\tilde{D}$ is a diagonal matrix such that $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$.

   The normalized adjacency matrix for the star graph S4 is:

$$\hat{A}_{\text{star}} = \begin{bmatrix} 0.25 & 0.35 & 0.35 & 0.35 \\ 0.35 & 0.50 & 0.00 & 0.00 \\ 0.35 & 0.00 & 0.50 & 0.00 \\ 0.35 & 0.00 & 0.00 & 0.50 \end{bmatrix}$$

   The normalized adjacency matrix for the cycle graph C4 is:

$$\hat{A}_{\text{cycle}} = \begin{bmatrix} 0.33 & 0.33 & 0.00 & 0.33 \\ 0.33 & 0.33 & 0.33 & 0.00 \\ 0.00 & 0.33 & 0.33 & 0.33 \\ 0.33 & 0.00 & 0.33 & 0.33 \end{bmatrix}$$

5. **Weight Matrices and Feature Matrix**: We define the weight matrices and the feature matrix.

   The weight matrix W0 is:

$$W0 = \begin{bmatrix} 0.50 & -0.20 \end{bmatrix}$$

   The weight matrix W1 is:

$$W1 = \begin{bmatrix} 0.30 & -0.40 & 0.80 & 0.50 \\ -1.10 & 0.60 & -0.10 & 0.70 \end{bmatrix}$$

   The feature matrix X is:

$$X = \begin{bmatrix} 1.00 \\ 1.00 \\ 1.00 \\ 1.00 \end{bmatrix}$$

6. **Z Matrices**: We compute the matrices Z0 and Z1 for the star graph S4 and the cycle graph C4 using the formula
$$Z = f(\hat{A}ZW)$$
where f is the ReLU function.

The matrix Z0 for the star graph S4 is:
$$Z0_{\text{star}} = \begin{bmatrix} 0.66 & 0.00 \\ 0.43 & 0.00 \\ 0.43 & 0.00 \\ 0.43 & 0.00 \end{bmatrix}$$
The matrix Z1 for the star graph S4 is:
$$Z1_{\text{star}} = \begin{bmatrix} 0.18 & 0.00 & 0.49 & 0.31 \\ 0.13 & 0.00 & 0.36 & 0.22 \\ 0.13 & 0.00 & 0.36 & 0.22 \\ 0.13 & 0.00 & 0.36 & 0.22 \end{bmatrix}$$
The matrix Z0 for the cycle graph C4 is:
$$Z0_{\text{cycle}} = \begin{bmatrix} 0.50 & 0.00 \\ 0.50 & 0.00 \\ 0.50 & 0.00 \\ 0.50 & 0.00 \end{bmatrix}$$
The matrix Z1 for the cycle graph C4 is:
$$Z1_{\text{cycle}} = \begin{bmatrix} 0.15 & 0.00 & 0.40 & 0.25 \\ 0.15 & 0.00 & 0.40 & 0.25 \\ 0.15 & 0.00 & 0.40 & 0.25 \\ 0.15 & 0.00 & 0.40 & 0.25 \end{bmatrix}$$

In the representations Z1, we observe that for the star graph S4, the central node (first row) has a different representation from the other nodes. This is expected as the central node has a different structural role in the graph (it is connected to all other nodes). For the cycle graph C4, all nodes have the same representation in Z1. This is also expected as all nodes in C4 have the same structural role (each node is connected to two other nodes).
In the representations Z1, we observe that for the star graph S4, the central node (first row) has a different representation from the other nodes. This is expected as the central node has a different structural role in the graph (it is connected to all other nodes). For the cycle graph C4, all nodes have the same representation in Z1. This is also expected as all nodes in C4 have the same structural role (each node is connected to two other nodes).

References

[1] Tiago Pimentel, Rafael Castro, Adriano Veloso, and Nivio Ziviani. Efficient estimation of node representations in large graphs using linear contexts. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2019.