LACOMBE Yoach
yoach.lacombe@gmail.com

Lab session # 5
ALTEGRAD 2021

01/21/22

1 Question 1

DeepWalk algorithm takes into inputs sentences made of random walk throughout the graph. Here, these sentences are very easy to determine. Indeed if we note n_1 the first random node of a sentence and n_2 its neighbor (it only has one since the connected components are complete graph K_2), the sentence will be $[n_1, n_2, n_1, n_2, ...]$. So the DeepWalk algorithm will learn to embed a node close to its only neighbor.
So the cosine similarity of these two nodes will be close to 1, whereas the cosine similarity associated to two nodes in different connected components will be close to 0.

2 Question 2

If we note c_1 and c_2 the columns of X_1, we notice that $X_2 = [c_1, -c_2]$. So the two embeddings are closely related. In particular, they have the same rank, and are produced by the same vector. They are thus both as informative as the other.

3 Question 3

GNN typically contains more than one message passing layer because one of the best advantages of deep-learning based methods is that they are highly non linear. This high non-linearity is there when there are more than 1 layer.
Using more layers than the diameter of the graph wouldn't be useful at all, since it would've already passed between every pair of nodes of the network. Indeed every new layer augment the connectivity of the message passing (k-layers means passing a message from each node to its k-related neighbors). However, using more than 2 layers could be useful, but it would require a large amount of computational power.

4 Question 4

First, let's compute the adjacency matrix.

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

Then,

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

From $\tilde{A}$, we can get $\tilde{D}$ and $\hat{A}$.

$$\tilde{D} = diag(2, 3, 3, 2)$$

$$\hat{A} = \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2} = \begin{bmatrix} 1/2 & 1/6^{1/2} & 0 & 0 \\ 1/6^{1/2} & 1/3 & 1/3 & 0 \\ 0 & 1/3 & 1/3 & 1/6^{1/2} \\ 0 & 0 & 1/6^{1/2} & 1/2 \end{bmatrix}$$

Now, compute Z^0:

$$Z^0 = f(\hat{A}XW^0) = f(\begin{bmatrix} 0.5v^T & -0.2v^T \end{bmatrix})$$

where $v = \hat{A}X = [1/2, 2/3, 2/3, 1/2] + 1/\sqrt{6}$.
So $Z^0 = f(\begin{bmatrix} 0.5v^T & 0 \end{bmatrix})$.

After computations, the last matrix is:

$$Z^1 = \begin{bmatrix} 0.0998612 & 0.133148 & 0.266297 & 0.166435 \\ 0.162997 & 0.217329 & 0.434658 & 0.271661 \\ 0.162997 & 0.217329 & 0.434658 & 0.271661 \\ 0.0998612 & 0.133148 & 0.266297 & 0.166435 \end{bmatrix}$$

Let's notice that nodes with the same degrees have the same vector embedding. The GNN successfully captured the specificities of the graph (which is symmetric).
If we'd used a classifier afterwards, it would probably could have been able to distinguish between nodes with same degree distribution.

References