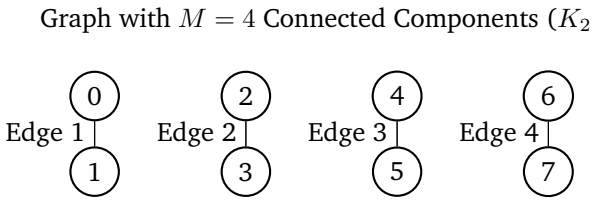
1 Question 1

1.1 Visualization of the graph

First let's visualize what is such a graph that is defined in the question. If we consider that $M = 4$, we obtain a graph like this one:



Indeed, as a recall from lecture 2, a connected component is a subgraph which is not connected to any additional vertices in the supergraph. Now, with this graph, the question is what do we expect of the cosine similarity. Therefore, let's briefly discuss of the cosine similarity.

1.2 Cosine similarity

The cosine similarity is a metric to assess how similar two vectors are. Schematically, it is defined as the cosinus of the angle between the two vectors we want to compare. The value is therefore between -1 and +1 where -1 indicates that the vectors have no similarity, +1 indicates that the vectors are identical and 0 that the vectors are orthogonals, these measurements are shown in Fig 1.

$$\text{cosine similarity}(\mathbf{A}, \mathbf{B}) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} \tag{1}$$

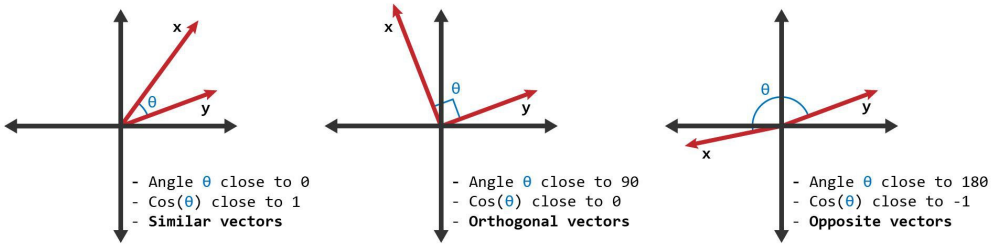
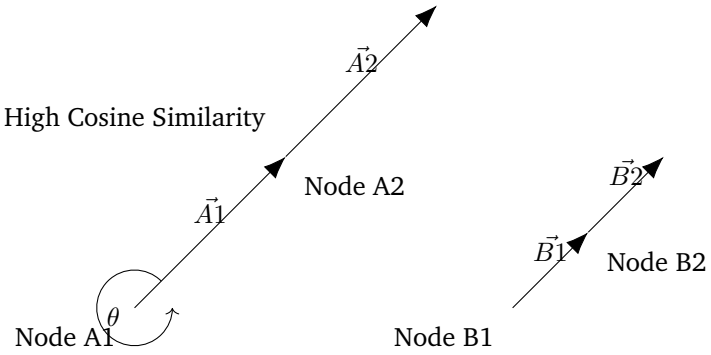


Figure 1: Cosine similarity illustrated.

1.3 Cosine similarity - Embeddings of the nodes within connected components

In each connected component, there are only the two nodes that are linked each other with one edge. Therefore, it is obvious that in the DeepWalk, the nodes will always be taken. Therefore, these two nodes will be embedded and then the cosine similarity will be close to 1.



1.4 Cosine similarity - Embeddings of the nodes in different connected components

The vertices in different connected components have no edges between them in the graph. During the random walks, nodes from different components will never be together. Therefore, the cosine similarity between the embeddings of nodes from different connected components is expected to be very low (close to 0).

2 Question 2

2.1 DeepWalk time complexity

DeepWalk's time complexity is influenced by two factors: the generation of random walks and the training of the Skip-gram model. Indeed, the deepwalk function generates random walks on the graph [2]. Each walk is a sequence of nodes, analogous to a sequence of words in a sentence. When we apply Skip-Gram, this is to learn the embeddings of the nodes. The principle of DeepWalk Algorithm is reminded in Fig 2.

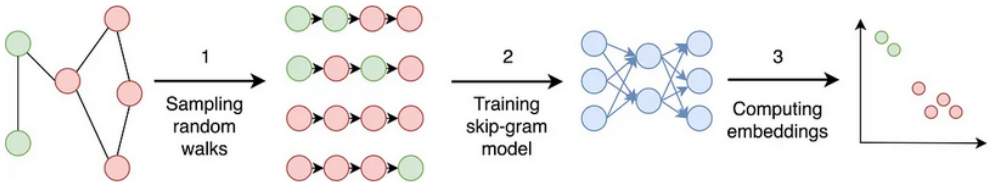


Figure 2: Principle of Deepwalk Algorithm. Considering each node being a word, this algorithm maps it into an embedding representation. Adapted from [6].

First, let's focus on the random walks. We consider that γ is the number of walks per node, l is the walk length, N is the number of nodes, then the complexity for the generation generating of random walks is $O(\gamma \times l \times N)$.

Then, there is the skip-gram model. The complexity of training the Skip-gram model or Word2Vec algorithm. It only focus on some factors that are the number of walks, the length of each walk, the dimension of the embeddings (d), the window size w and the context size. Meaning that the time complexity for it is $O((dlog(N) + d) \times \gamma \times l \times N \times w)$, where d is the embedding dimension. Its principle is shown in Fig 3.

Therefore the overall complexity of the DeepWalk algorithm is $O(\gamma \times l \times N \times (dlog(N) + d)$. By removing the constants, we obtain a complexity of $O(Nlog(N))$.

The paper that introduced this algorithm is [5]. In this one, but also in [2] and [4], they show that indeed, because of the softmax function, the complexity of the deepwalk algorithm is as proven $O(Nlog(N))$.

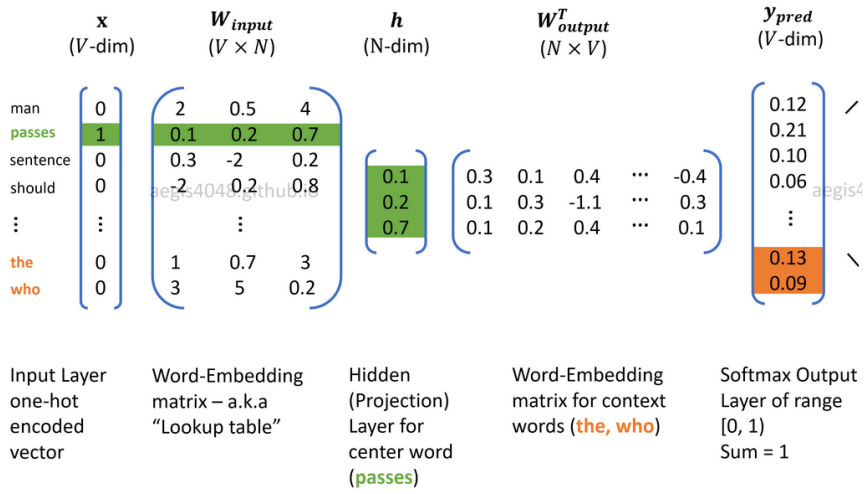
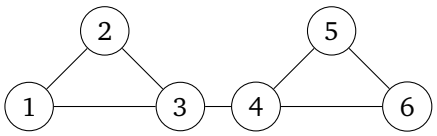


Figure 3: Skip gram neural network structure. Adapted from [4].

2.2 Spectral Embedding time complexity

Spectral embedding is also influenced by two steps; the graph Laplacian construction and the eigenvalue decomposition. The overall process was described in [1]. To understand clearly this algorithm, let's look at the graph.



Given graph with adjacency matrix A:

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

Degree matrix D and Laplacian matrix $L = D - A$:

$$D = \text{diag}(A\text{'s row sums})$$

$$L = D - A$$

We just have to compute its eigenvalues and eigenvectors now. The obtained graph embeddings is displayed in Fig 4. Indeed, the eigenvectors correponding to the smallest non-zero eigenvalues are known to capture significant structual properties of the graph.

First, the graph Laplacian Construction is the matrix L it has a complexity of $O(N^2)$.

Then, for the eigenvalue decomposition the time complexity can be $O(N^3)$, by looking at some references paper such as [7] or [8], we indeed have the complexity is $O(N^3)$.

2.3 Conclusion

What we observe is that $O(Nlog(N)) < O(N^3)$, we can then say that the complexity of the deepwalk is lower than the complexity of the spectral embedding. Indeed, Deepwalk works by performing random walks on the graph. These walks are computationally inexpensive. The main complexity of the model comes from the training of the Skip-Gram model. On the other hand, Spectral Embedding involves the computation of the eigenvalues and eigenvectors of the Laplacian graph. And the downside with this step is that it is computationally expensive, especially for large graphs.

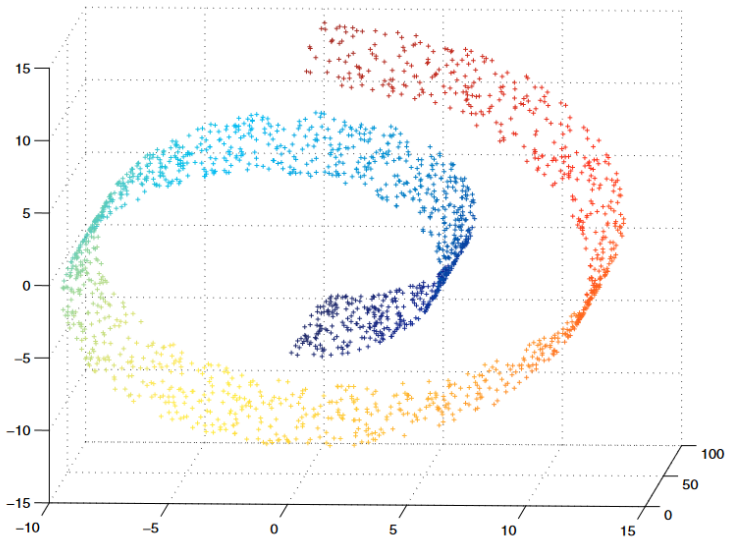


Figure 4: Spectral Embedding final representation, the 'Swiss roll'. Adapted from [1].

3 Question 3

If self-loops were not added to the graph and the original graph also contained no self-loops there would be an effect on the GNN (especially single and two layer GNN). The typical normalization used in GNNs is $\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}$ where $\tilde{A} = A+I$. If self-loops are not added, the normalization will then become $D^{-1/2}AD^{-1/2}$.

As presented in the reference paper, [9], the aggregation of the features work like in the scheme of Fig 5.

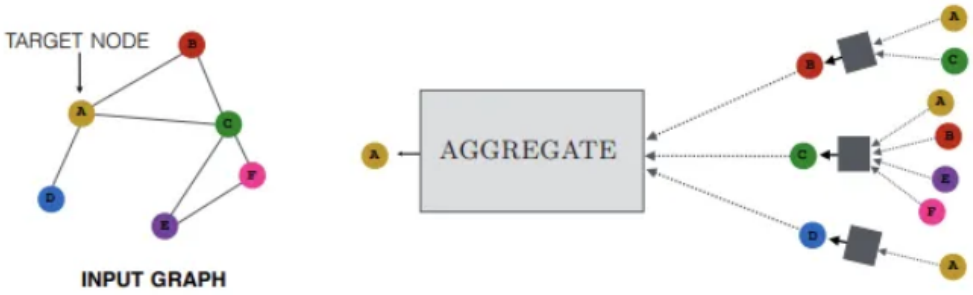


Figure 5: Graph Neural Networks aggregation. Adapted from [3].

3.1 Single Layer GNN

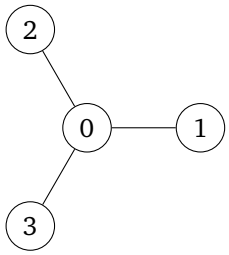
Without self-loops, each node's new features will only be calCulated with its neighbors, excluding itself. This might limit the ability of nodes to keep their original features after passing through a GNN layer. Indeed, the absence of self-loops makes the node features in the next layer entirely dependent on neighboring nodes. Furthermore, node's features will mix with its own features. Without self-loops, node's features won't mix with its own features. In addition, nodes that have no edges won't be able to update their features, as they don't have neighboring nodes to aggregate features from.

3.2 Two-Layer GNN

In a two-layer GNN, the features of a node are influenced by its neighbors and the neighbors of its neighbors. Without self-loops, the node won't directly influence its representation in the second layer, leading to more importance put on the distant neighborhood. Multiple layers in GNNs can then lead to over-smoothing, where node representations become indistinguishable. The absence of self-loops might make this issue less severe as each layer's aggregation doesn't include the node's own features, potentially preserving more diversity in the features.

4 Question 4

4.1 star graph S_4



4.1.1 Calculations of $\hat{A}$

$$\text{Adjacency Matrix for } S_4 : A = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

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

$$\tilde{D} = \begin{pmatrix} 4 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix}$$

$$\tilde{D}^{-1/2} = \begin{pmatrix} 1/2 & 0 & 0 & 0 \\ 0 & 1/\sqrt{2} & 0 & 0 \\ 0 & 0 & 1/\sqrt{2} & 0 \\ 0 & 0 & 0 & 1/\sqrt{2} \end{pmatrix}$$

$$\hat{A} = \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2} = \begin{pmatrix} 1/4 & 1/2\sqrt{2} & 1/2\sqrt{2} & 1/2\sqrt{2} \\ 1/2\sqrt{2} & 1/2 & 0 & 0 \\ 1/2\sqrt{2} & 0 & 1/2 & 0 \\ 1/2\sqrt{2} & 0 & 0 & 1/2 \end{pmatrix}$$

4.1.2 Calculations of Z^0

$$X = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

$$W^0 = \begin{pmatrix} 1/2 & -1/5 \end{pmatrix}$$

$$\hat{A}XW^0 = \frac{1}{2} \cdot \begin{pmatrix} \frac{1}{2}(\frac{1}{2} + \frac{3}{\sqrt{2}}) & -\frac{1}{5}(\frac{1}{2} + \frac{3}{\sqrt{2}}) \\ \frac{1}{2}(1 + \frac{1}{\sqrt{2}}) & -\frac{1}{5}(1 + \frac{1}{\sqrt{2}}) \\ \frac{1}{2}(1 + \frac{1}{\sqrt{2}}) & -\frac{1}{5}(1 + \frac{1}{\sqrt{2}}) \\ \frac{1}{2}(1 + \frac{1}{\sqrt{2}}) & -\frac{1}{5}(1 + \frac{1}{\sqrt{2}}) \end{pmatrix}$$

$$Z^0 = ReLu(\hat{A}XW^0) = \frac{1}{4} \cdot \begin{pmatrix} \frac{1}{2} + \frac{3}{\sqrt{2}} & 0 \\ 1 + \frac{1}{\sqrt{2}} & 0 \\ 1 + \frac{1}{\sqrt{2}} & 0) \\ 1 + \frac{1}{\sqrt{2}} & 0) \end{pmatrix}$$

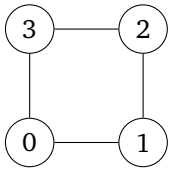
4.1.3 Calculations of Z^1

$$W^1 = \begin{pmatrix} 0.3 & -0.4 & 0.8 & 0.5 \\ -1.1 & 0.6 & -0.1 & 0.7 \end{pmatrix}$$

$$\hat{A}Z^0W^1 \cong \begin{pmatrix} 0.18 & -0.25 & 0.49 & 0.31 \\ 0.13 & -0.18 & 0.36 & 0.22 \\ 0.13 & -0.18 & 0.36 & 0.22 \\ 0.13 & -0.18 & 0.36 & 0.22 \end{pmatrix}$$

$$Z^1 = ReLu(\hat{A}Z^0W^1) \cong \begin{pmatrix} 0.18 & 0 & 0.49 & 0.31 \\ 0.13 & 0 & 0.36 & 0.22 \\ 0.13 & 0 & 0.36 & 0.22 \\ 0.13 & 0 & 0.36 & 0.22 \end{pmatrix}$$

4.2 cycle graph C_4



4.2.1 Calculations of $\hat{A}$

$$\text{Adjacency Matrix for } C_4 : A = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

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

$$\tilde{D} = \begin{pmatrix} 3 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 3 \end{pmatrix}$$

$$\tilde{D}^{-1/2} = \begin{pmatrix} 1/\sqrt{3} & 0 & 0 & 0 \\ 0 & 1/\sqrt{3} & 0 & 0 \\ 0 & 0 & 1/\sqrt{3} & 0 \\ 0 & 0 & 0 & 1/\sqrt{3} \end{pmatrix}$$

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

4.2.2 Calculations of Z^0

$$X = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

$$W^0 = \begin{pmatrix} \frac{1}{2} & -\frac{1}{5} \end{pmatrix}$$

$$\hat{A}XW^0 = \frac{1}{3} \cdot \begin{pmatrix} \frac{1}{2} & -\frac{1}{5} \\ \frac{1}{2} & -\frac{1}{5} \\ \frac{1}{2} & -\frac{1}{5} \\ \frac{1}{2} & -\frac{1}{5} \end{pmatrix}$$

$$Z^0 = ReLu(\hat{A}XW^0) = \frac{1}{3} \cdot \begin{pmatrix} \frac{1}{2} & 0 \\ \frac{1}{2} & 0 \\ \frac{1}{2} & 0 \\ \frac{1}{2} & 0 \end{pmatrix}$$

4.2.3 Calculations of Z^1

$$W^1 = \begin{pmatrix} 0.3 & -0.4 & 0.8 & 0.5 \\ -1.1 & 0.6 & -0.1 & 0.7 \end{pmatrix}$$

$$\hat{A}Z^0W^1 = \frac{1}{2} \cdot \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \end{pmatrix} \cdot W^1$$

$$\hat{A}Z^0W^1 = \begin{pmatrix} 0.15 & -0.20 & 0.40 & 0.25 \\ 0.15 & -0.20 & 0.40 & 0.25 \\ 0.15 & -0.20 & 0.40 & 0.25 \\ 0.15 & -0.20 & 0.40 & 0.25 \end{pmatrix}$$

$$Z^1 = ReLu(\hat{A}Z^0W^1) = \begin{pmatrix} 0.15 & 0 & 0.40 & 0.25 \\ 0.15 & 0 & 0.40 & 0.25 \\ 0.15 & 0 & 0.40 & 0.25 \\ 0.15 & 0 & 0.40 & 0.25 \end{pmatrix}$$

4.3 Structure analysis

In both Z^1 matrix we computed, we can observe that the second column is null. Then, for S_4, the three last lines are equal. It is not the case for C_4 where all the four lines are equal. It only means that each nodes can be replaced in C_4, they all have the same structure. It is not the case in S_4 where the central node will have a different coefficient for each node.

References

[1] Mikhail Belkin and Partha Niyogi. Laplacian Eigenmaps for Dimensionality Reduction and Data Representation. *Neural Computation*, 15(6):1373–1396, 06 2003.

[2] Yifan Hu Steven Skiena Haochen Chen, Bryan Perozzi. Harp: Hierarchical representation learning for networks. In *arXiv:1706.07845v2*, 2017.

[3] Omar Hussein. The gnns, message passing over-smoothing. In *Medium / The Modern Scientist*, 2023.

[4] Eric Kim. Optimize computational efficiency of skip-gram with negative sampling. In *Pythonic Excursions*, 2019.

[5] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. DeepWalk. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, aug 2014.

[6] Anh Tuan. Paper review: “deepwalk: Online learning of social representations”. In *Medium*, 2022.

[7] Wikipedia. Eigenvalue algorithm. In *https://en.m.wikipedia.org/wiki/Eigenvalue*$_a lgorithm, 2023.$

[8] Wikipedia. Spectral clustering. In *https://en.wikipedia.org/wiki/Spectral*$_c lustering, 2023.$

[9] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks?, 2019.