Matias Etcheverry
matias.etcheverry9@gmail.com

Lab session # 5
ALTEGRAD 2022

11/26/22

1 Node embedding with DeepWalk

This is the embedding I obtained using DeepWalk on the French web

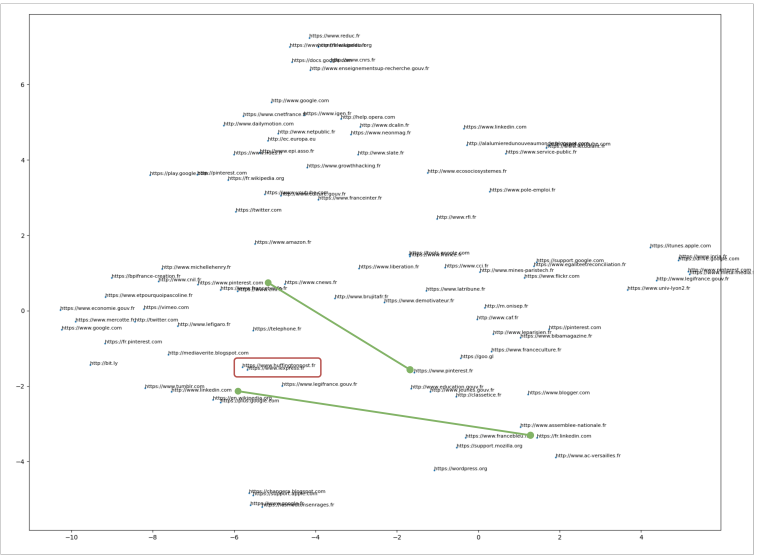

Figure 1: t-SNE visualization of node embeddings on the French Web. Red represents a consistent area while green are 2 similar relationship between 2 websites.

This embedding is not to bad. For instance, we can see that `huffingtonpost.fr` is close to `lexpress.fr` (in red). Moreover, the vector linking `linkedin.com` and `fr.linkedin.com` is similar to the vector linking `pinterest.com` and `pinterest.fr` (in green). This is exactly what we are looking for.

2 Question 1

Applying the DeepWalk architecture can easily be generalized to directed graph, without any change. Indeed, to compute a walk, we simply take a random path from a node v to its neighbors. With a directed graph, we take a random allowed path in the right direction to the neighbors of v.
With a weighted graph, the procedure would almost be the same too. Instead of randomly choosing among the neighbors of v, we choose randomly with weights which are the weights of the edges.

3 Question 2

With a graph containing only nodes and no edges, the adjacency matrix is the null matrix. Indeed, a node isn't connected to anything, as there is no propogation of information. Thus, the equations of the model become: $\tilde{A}$ vs $\widetilde{A}$

$$
\begin{aligned}
&A = \mathbf{0} \\
&\tilde{A} = \mathbf{I} \\
&\tilde{D} = \mathbf{I} \quad \text{because} \quad \tilde{D}_{ik} = \delta_{ik} \sum_j \tilde{A}_{ij} \\
&\hat{A} = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} = \mathbf{I} \\
&Z^0 = f(\hat{A} X W^0) = f(X W^0) \quad \text{without writing the bias} \\
&\dots
\end{aligned}
$$

Thus the described method becomes a fully connected network.

4 Question 3

The adjacency matrix acts like a convolution layer in the sense that it gathers a node and its neighbors, with all the same weights. Thus, when multiplying by the adjacency matrix, a node's features become the average of itself and its neighbors' features. Thus, the k^{th} layer of the GNN using the adjacency matrix will fetch the features of the neighbors at a distance k of the original node.
In the end, the current GNN as a receptive field of 2. A GNN with n layers has a receptive field of n.

5 Node classification on Karate

fig. 2 represents the ground truth classification in the karate dataset

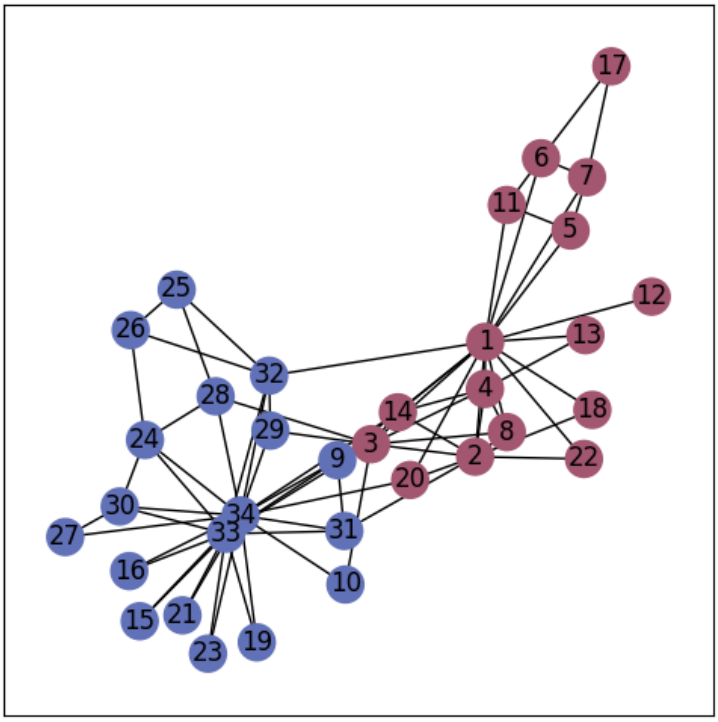

Figure 2: Ground truth classification of the 2 groups (red and blue).

We have 2 traditional methods of embedding: the DeepWalk and the Laplacian embedding which don't require any learning. We alos have 2 methods of embedding for the Graph Neural Network: either the features are all differents (like $\mathbf{I}$) or they are all equal (like $\mathbf{1}$). We obtain the following performances, using the 4 methods of embedding:

Metric	DeepWalk Embedding	Laplacian Embedding	GNN ($\mathbf{X} = \mathbf{I}$)	GNN ($\mathbf{X} = \mathbf{1}$)
Test Loss	/	/	**0.0002**	0.7792
Test Accuracy	**100%**	85.71%	**100%**	28.57%

6 Question 4

table 1 are all the matrices needed for the computation of Z^1 with 2 graphs: $\mathcal{K}_4$ and $\mathcal{S}_4$.
A few points about how we computed these matrices:

- a node is never linked to itself in the adjacency matrix
- the central node in $\mathcal{S}_4$ is the first node
- the features are a vector of 1.

The matrix Z^1 is made of rows which represent the embeddings of the nodes. We notice in the matrix Z^1 that all look alike nodes have the same activation. Indeed, since they have the same features and the same

$\mathcal{K}_4$	$\mathcal{S}_4$
$A = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$	$A = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$
$\hat{A} = \begin{bmatrix} 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \end{bmatrix}$	$\hat{A} = \begin{bmatrix} 0.25 & 0.35 & 0.35 & 0.35 \\ 0.35 & 0.5 & 0 & 0 \\ 0.35 & 0 & 0.5 & 0 \\ 0.35 & 0 & 0 & 0.5 \end{bmatrix}$
$Z^0 = \begin{bmatrix} 0 & 0.5 \\ 0 & 0.5 \\ 0 & 0.5 \\ 0 & 0.5 \end{bmatrix}$	$Z^0 = \begin{bmatrix} 0 & 0.65 \\ 0 & 0.43 \\ 0 & 0.43 \\ 0 & 0.43 \end{bmatrix}$
$Z^1 = \begin{bmatrix} 0 & 0.3 & 0.25 \\ 0 & 0.3 & 0.25 \\ 0 & 0.3 & 0.25 \\ 0 & 0.3 & 0.25 \end{bmatrix}$	$Z^1 = \begin{bmatrix} 0 & 0.37 & 0.31 \\ 0 & 0.27 & 0.22 \\ 0 & 0.27 & 0.22 \\ 0 & 0.27 & 0.22 \end{bmatrix}$

Table 1: Computation of 2^1.

adjacency, they results in the same embedding. Moreover, we also notice that the first coordinate of the embeddings is always 0. This due to the `ReLU` activation.
If we had randomly sampled node features $\mathbf{X}$, the features would all have been different, and thus the embeddings too. There would be no repeating pattern in the embedding matrix Z^1.

7 Node classification and representation on Cora

We then apply a GNN on the cora dataset. The features of the nodes are included in the dataset. We obtain the following metrics:

Metric	GNN
Test Loss	0.5819
Test Accuracy	85.98%

fig. 3 represents the classification obtained with a GNN in the Cora dataset. The classification is very good !

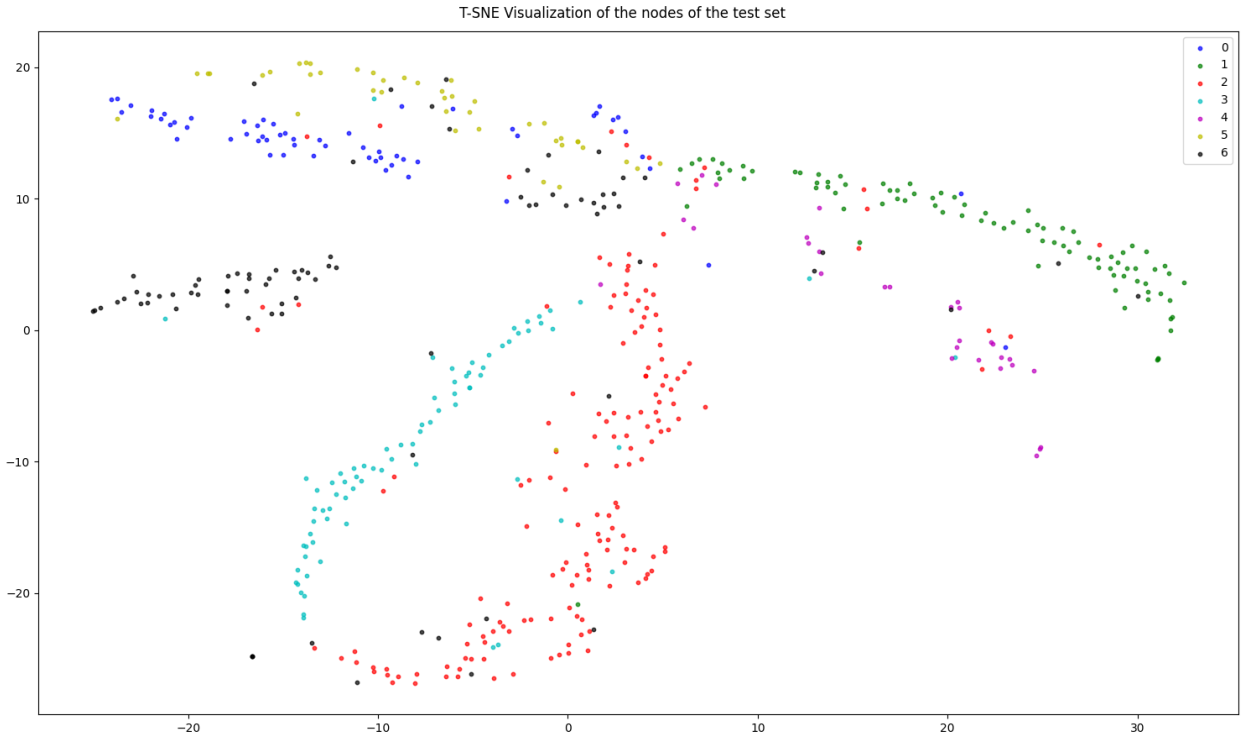

Figure 3: Classification on Cora dataset with GNN and t-SNE.