Gertner Victor
gertnervictor@gmail.com

Lab session # 7
ALTEGRAD 2023

12/05/23

1 Question 1

LSTM are not permutation invariant, for instance they are an 'extension' of rnn, hence it's made to process sequential data and are sensitive to the order of the input sequence. In order to see it, we are going to show the key equations of LSTMs :

$$\begin{aligned}
i_t &= \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}) \\
f_t &= \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}) \\
o_t &= \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) \\
g_t &= \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}) \\
C_t &= f_t \odot C_{t-1} + i_t \odot g_t \\
h_t &= o_t \odot \tanh(C_t)
\end{aligned}$$

As we can see for the computation of i_t, f_t, o_t, g_t, x_t and h_{t-1} are used. Hence if you do a permutation of the entry, the hidden state and every thing would be different. Hence since we want a model that is permutation invariant, we wouldn't recommand the use of LSTM on sets.

2 Question 2

So the main conceptual differences between Deepset and the architecture we saw in the lab 6 are :

- **The data** that is considered : for sets we have data that as no clear link, whereas graphs, the relationship between the data is clearly represented in the edges and node information.
- **How the data is treated** : In the case of graphs, we use the system of message passing to propagate information of neigbourgs, this is possible because of the local structure graph create. On the other hand, sets don't have that, this is why an embedding layer is needed to capture the similarity of the data. This is more a global approach than the local one GNN imply.
- **The task** : GNN are used when the task relies on explicit graph relationship. On the other hand Deepset is used for a task where there is no clear relationship or order in the elements, or that its relation is irrelevant.

The difference is that in a set used by deepset, each element of the set is treated equally. On the other hand, even if there is no clear connection between nodes in a graph without edge setup, we still provide the GNN a feature matrix. Hence there is a difference between the two.

3 Question 3

Part One :
so we are going to tacke the two cases one by one :

- **homophilic :** in this case, we want to have more edges inside the community yhat those going from one community to another. So for r = 2, any matrix where we have for all i $\neq$ j $P_{ii} > P_{ij}$ METTRE LA SOMME. As an exemple we can propose :

$$P = \begin{bmatrix} 0.8 & 0.15 \\ 0.15 & 0.8 \end{bmatrix} \tag{1}$$

- **heterophilic :** in this case , we want to have more edge between the communities than edges inside the communities. So for r = 2, any matrix where we have for all i $\neq$ j $P_{ii} < P_{ij}$. As an example, we can propose :

$$P = \begin{bmatrix} 0.3 & 0.55 \\ 0.55 & 0.3 \end{bmatrix} \tag{2}$$

This solution works because we ask our blocks to be of the same size (+- 1).

Part two :

Let's denote each block of nodes by $B_1, B_2, ..., B_m$ and their respective number of node by $n_1, n_2 ..., n_m$. First in order to understand how we are going to find the whole expectation, we are going to focus on a specific node.
Let's say we take a node from B_i and we want to find the expectation of the number of edge from this node to another block $B_j, j \neq i$. We can model this experience with a binomial law of parameters (n_j, p_{ij}). Hence the expected number of edge between this node and the block B_j is $n_j * p_{ij}$. Hence, the formula for the expected number of nodes between any two blocks B_j, B_i is $\sum_{h=1}^{n_i} n_j * p_{ij} = n_i * n_j * p_{ij}$.
Thus, we have now the formula we want :

$$\text{Expected number of edges between blocks : } \sum_{i=1}^{m} \sum_{j=i+1}^{m} n_i n_j p_{ij} \tag{3}$$

We can now apply it in our case : we have m=4, n_j = 5 , $pii = 0.8, p_{i,j} = 0.05$ for all j≠i.
It gives us **Expected number of edges between blocks =** $(1+1+1)*5*5*0.05+(1+1)*5*5*0.05+(1)*5*5*0.05 = 7.5$

4 Question 4 :

The previous loss that was used for unweigthed graphs, was the binary cross entropy, it's made to be used in a classification setup. If now, we take weighted edges, we don't have discrete values and we cannot use this kind of loss. Our problem here is more a regression problem, so we can use a simple MSE in order to have a loss function.

$$MSE(A, \hat{A}) = \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} (A_{ij} - \hat{A}_{ij})^2 \tag{4}$$

One issue we have there, is that as stated in the lab sheet, $\hat{A}$ takes values between 0 and 1. This is because we use a sigmoid function as the activation function of our decoder. Hence, we might want to change the last layer of our model in a simple linear in order to be able to do regression.