Bastien LE CHENADEC
bastien.le-chenadec@eleves.enpc.fr

1 Question 1

An LSTM model is absolutely not permutation invariant. In fact the goal of the LSTM model is to learn information sequentially, which means that the order of the data is very important.
In our case, the model has to learn this invariance by itself, making it very difficult to train. Therefore, I would not recommend to use LSTM models on sets, but only on data where the order of the elements matters.

2 Question 2

The two models are inherently different :

- The GNN model works on a graph, which models relationships (edges) between elements (nodes).
- The DeepSets model works on a set, which is a collection of elements without any relationship between them.

However there are also similarities between the two models, as they are both permutation invariant, and work by aggregating element-level representations into a set-level representation.
In fact if we consider a graph with no edges, and where the feature of each node is the number of the element in the set (in our case $1, 2, \dots$), then the GNN model with one message passing layer and a sum readout is equivalent to the DeepSets model.
One other key difference is that the GNN allows two elements to have the same features, whereas in the DeepSets model two elements with the same features will be considered as the same element.

3 Question 3

Let us define :

$$P_1 = \begin{pmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{pmatrix} \quad P_2 = \begin{pmatrix} 0.1 & 0.9 \\ 0.9 & 0.1 \end{pmatrix}$$

It is clear that P_1 will lead to a homophilic graph and P_2 to a heterophilic graph. However they are both equal to a permutation of the clusters, which means that the stochastic block model will lead to the same distribution of graphs for both matrices (assuming the same repartition of nodes between each cluster). Therefore the notion of homophily and heterophily is not captured by the stochastic block model.
Let us compute the expected number of edges between nodes in different blocks with the matrix :

$$P = \begin{pmatrix} 0.8 & 0.05 & 0.05 & 0.05 \\ 0.05 & 0.8 & 0.05 & 0.05 \\ 0.05 & 0.05 & 0.8 & 0.05 \\ 0.05 & 0.05 & 0.05 & 0.8 \end{pmatrix}$$

Let's consider a block i and another block j. The expected number of edges between nodes of these two blocks is $5 \times 5 \times 0.05 = 1.25$ (there are 5 nodes in each block, so 5×5 possible edges, and each edge has a probability of 0.05 to exist). There are 4 blocks in total, so the expected number of edges between nodes of different blocks is $\binom{4}{2} \times 1.25 = 7.5$.

4 Question 4

The binary cross entropy is only adapted to binary classification problems, which is the case if we consider unweighted graphs. However, if we consider weighted graphs, the binary cross entropy loss is not adapted anymore. The easiest answer would be to use an MSE or MAE loss. However this would give the same importance to an error of estimation of a positive weight and to an error of estimation of a non-existing edge. Therefore we propose these adapted losses (which are to be minimized):

$$\mathcal{L}^1 = \frac{1}{n^2} \sum_{i,j} \mathbb{1}_{A_{ij} \neq 0} |A_{ij} - \hat{A}_{ij}| - \mathbb{1}_{A_{ij}=0} \log(1 - \hat{A}_{ij})$$

$$\mathcal{L}^2 = \frac{1}{n^2} \sum_{i,j} \mathbb{1}_{A_{ij} \neq 0} (A_{ij} - \hat{A}_{ij})^2 - \mathbb{1}_{A_{ij}=0} \log(1 - \hat{A}_{ij})$$

We are assuming that A_{ij} and $\hat{A}_{ij}$ are between 0 and 1. If the weights are not between 0 and 1, we can normalize them to return to this case. We are basically using an L^1 or L^2 loss on the existing edges, and a binary cross entropy loss on the non-existing edges. This way we give an exponential importance to an error on a non-existing edge and a linear or quadratic importance to an error of estimation of an existing weight. These losses are differentiable with respect to $\hat{A}_{ij}$, so we can use them to train our model.

References