

1 Is the LSTM model permutation invariant?

The LSTM (Long Short-Term Memory) model is not permutation invariant. This is because LSTMs, as a type of Recurrent Neural Network (RNN), process input sequences in a specific order. The hidden state of the LSTM at any given step depends on the current input and the previous hidden state. So, if we change the order of the inputs, the outputs can also change. When we're dealing with sets, the order of elements doesn't matter. Any permutation of a set is still the same set. So, if we use an LSTM to model sets, we might get inconsistent results because the LSTM could give different outputs for different permutations of the same set. So, while LSTMs are great for sequence data, I wouldn't recommend using them on sets. They're sensitive to the order of inputs, which isn't ideal for sets. For sets, it would be better to use models that are permutation invariant, like DeepSets.

2 Architectural differences between GNN for graph-level tasks implemented in lab 6 and the DeepSets model

- **GNN for graph-level tasks (Lab 6):**

- Designed for graph-level tasks.
- Uses two message passing layers to aggregate information from neighboring nodes in a graph.
- Followed by a sum readout function to produce a vector representation of the entire graph.
- Captures the structure of the graph and the relationships between nodes.

- **DeepSets model:**

- Designed for set-structured data.
- Uses an embedding layer to project each element of the set into a higher-dimensional space.
- Followed by a fully connected layer and a sum operation to aggregate the transformed elements.
- Handles unordered sets, where the order of elements does not matter (Permutation invariant).
- More suitable for processing a set, which is a collection of distinct elements with no particular order or structure. In a set, there is no concept of individual nodes or their relationships.

Sets are unordered collections of distinct elements without relationships or connections between them (e.g., 1, 2, 3). On the other hand, graphs without edges consist of isolated nodes (e.g., V1, V2, V3) with attributes but no connections. Sets highlight the absence of order and relationships, while graphs without edges involve nodes without connections.

3 Exercise on stochastic block models

1) For a stochastic block model with $r = 2$, we can specify two edge probability matrices P for **homophilic** and **heterophilic** graphs as follows:

- **Homophilic Graph:** In a homophilic graph, many edges arise between nodes in the same community and far fewer edges arise between nodes in different communities. Therefore, the edge probability matrix P could be:

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

- **Heterophilic Graph:** In a heterophilic graph, most edges arise between nodes in different communities and few edges are present between nodes in the same community. Therefore, the edge probability matrix P could be:

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

2) To calculate the expected number of edges between nodes in different blocks of a stochastic block model with $n = 20$, containing 4 blocks of 5 nodes each, we can use the following formula:

Expected Number of Edges = Number of Pairs of Nodes in Different Blocks * Probability of an Edge Between Them

The number of pairs of nodes in different blocks can be calculated as follows:

- Each block has 5 nodes. - There are 4 blocks, so there are 6 pairs of blocks (**4 choose 2**). - Each pair of blocks can form 25 pairs of nodes (**5*5**).

So, the total number of pairs of nodes in different blocks is **6 * 25 = 150**.

The probability of an edge between nodes in different blocks is given as 0.05.

Therefore, **the expected number of edges between nodes in different blocks is $150 * 0.05 = 7.5$** . Since the number of edges must be an integer, we can round this to 8. So, we expect approximately 8 edges between nodes in different blocks.

4 A loss function that would be more suitable for the reconstruction of weighted graphs

For weighted graphs, where the entries of the adjacency matrix can take any real value, Mean Squared Error (MSE) loss would be a more suitable choice for the reconstruction loss. The MSE loss measures the average squared difference between the actual and predicted values, which makes it suitable for continuous outputs. The MSE loss can be defined as follows:

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

Here, A_{ij} is the actual value from the adjacency matrix and \hat{A}_{ij} is the predicted value. This loss function will encourage the model to minimize the difference between the actual and predicted weights of the edges in the graph.

References