BARILLER Halvard
halvard.bariller@ens-paris-saclay.fr

Lab session # 7
ALTEGRAD 2023

12/05/23

1 Question 1

Please describe briefly whether the LSTM is a permutation invariant model and conclude whether you recommend the use of LSTMs on sets.

LSTMs are designed for sequential data and have an inherent assumption of temporal dependence in their architecture: they specifically leverage the sequential nature of the input data to capture long-range dependencies and patterns. The architecture includes gates that control the flow of information and decide what information to remember or forget over time.
In sets, the arrangement or order of elements does not carry any significance, and the relationships between elements are not defined by their position. Therefore, a model that is sensitive to the order of input elements, like an LSTM, may struggle to generalize across different permutations of the same set or to capture meaningful dependencies.
Moreover, by their own design, LSTMs are strongly dependant on the order of the input, and are consequently not permutation invariant: it would be impossible to use them when doing a translation task for example if they showed such a property.

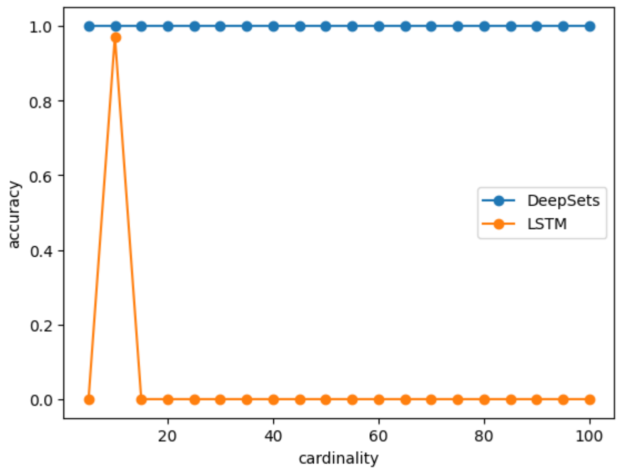

Figure 1: Accuracy comparison between DeepSets and a LSTM

We clearly see in figure 1 that the LSTM performs only when presented with test inputs similar to what it has been trained on, and does not generalize well.

2 Question 2

What are the architectural differences between the graph neural network (GNN) used for graph- level tasks that we implemented in Lab 6 and the DeepSets architecture? What is the difference between a set, such as the ones taken as input by the DeepSets architecture and a graph without edges, which could be processed using for example a GNN?

Graph Neural Networks (GNNs) for graph-level tasks utilize message-passing layers to capture information flow between interconnected nodes in a graph, leveraging the underlying graph structure. These networks then include readout functions to aggregate node-level information and produce a graph-level representation.
In contrast, DeepSets architecture is designed for sets, collections of unordered elements, and focuses on permutation invariance. In [1], it has been shown that over a countable set, a permutation invariant function takes the form $\rho(\sum_{x \in X} \phi(x))$. DeepSets therefore employs an architecture mirroring this founding: it tries to learn such representations ρ and ϕ with a MLP (learn the representation), then sums the representations and finally learns another MLP. It processes individual elements within a set, disregarding the order.
The key distinction lies in GNNs' ability to leverage explicit graph structures, while DeepSets operate on sets without assuming any inherent relationships or connectivity between elements.

While DeepSets is inherently build to learn on unstructured data, a GNN is built to understand local phenomenon through Message-Passing layers. Therefore, even if sets and graphs without edges both involve collections of non-linked elements, a set is composed of truly independant elements. In contrast, a graph without edges is making a prior on the absence of relationships between the nodes: in a GNN, the absence of edges brings as much information as the presence of edges about the graph structure as a whole.

3 Question 3

In the analysis of graphs we distinguish between homophilic and heterophilic cluster structure. Homophilic cluster structure describes a scenario in which the node set of a graph can be par- titioned into subsets, called communities, such that many edges arise between nodes in equal communities and far fewer edges arise between nodes in different communities. Contrariwise, heterophilic cluster structure describes the scenario in which most edges arise between nodes in different communities and few edges are present between nodes in the same community.
1) For a stochastic block model with $r = 2$ please specify two edge probability matrices P from which homophilic and heterophilic graphs can be sampled.
2) Please calculate, showing your workings, 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. The diagonal elements of matrix P are set equal to 0.8, while its off-diagonal elements equal to 0.05).

1) In a stochastic block model with $r = 2$, we can respectively sample an homophilic and an heterophilic graph for the following probability matrices:

$$P_{hom} = \begin{bmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{bmatrix}, \quad P_{het} = \begin{bmatrix} 0.2 & 0.8 \\ 0.8 & 0.2 \end{bmatrix},$$

as they favor respectively intra and extra-community edges.
2) In stochastic block model with $r = 4$ and 5 in each community, we have a graph with 20 nodes with the following probability matrix is:

$$P_{hom} = \begin{bmatrix} 0.8 & 0.05 & 0.05 & 0.05 \\ 0.05 & 0.8 & 0.05 & 0.05 \\ 0.05 & 0.05 & 0.8 & 0.05 \\ 0.05 & 0.05 & 0.05 & 0.8 \end{bmatrix}$$

In a graph (without self-loops) with $n = 20$ nodes, we have a maximum number of edges $|E_{max}| = \frac{n(n-1)}{2} = 190$. In each block, there is a maximum number of edges of $4 \times 5/2 = 10$ edges. Therefore, as there are 4 blocks, we have a maximum number of extra-community edges of $190 - 4 \times 10 = 150$.
The probability of each extra-community edge being $p = 0.05$, we can model the expected number of edges between nodes in different blocks as the expected value of a Binomial distribution. Finally, we have that the expected number of edges is equal to $150 \times 0.05 = 7.5$.

4 Question 4

The Binary Cross-Entropy reconstruction loss is appropriate in the context of unweighted graphs. Could you briefly propose a loss function that would be more suitable for the reconstruction of weighted graphs, i.e., graphs for which the entries of the adjacency matrix are not constrained to only take values equal to 0 or 1?

When dealing with weighted graphs and aiming to reconstruct the adjacency matrix, we need to consider a loss function that accounts for the continuous nature of the edge weights. One suitable loss function for this scenario is the Mean Squared Error Loss, which is commonly used in regression tasks. The MSE Loss is defined as the average of the squared differences between the predicted values and the ground truth. In the context of weighted graphs, it can be formulated as follows:
Let A be the ground truth adjacency matrix (with continuous edge weights) and $\hat{A}$ be the predicted adjacency matrix. The MSE Loss is given by:

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

where N is the number of nodes in the graph.
This loss function penalizes the model for deviations in edge weights, providing a measure of how well the predicted weights match the true weights in the adjacency matrix.

References

[1] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan Salakhutdinov, and Alexander Smola. Deep sets, 2017. cite arxiv:1703.06114Comment: NIPS 2017.