Balthazar Neveu
balthazarneveu@gmail.com

Lab session # 7
ALTEGRAD 2023

12/11/23

Code

More info: MVA ALTEGRAD Balthazar Neveu on Github

1 DeepSets: Learn to add

(and add to learn...)

Question 1 : LSTM are not permutation invariant, therefore not recommended for sets processing

Permutation invariance refers to the property where the model's output does not change if the order of the input data is changed. For instance, in a permutation invariant model, the input sequence [1, 2, 3] would yield the same output as [3, 2, 1]. Permuation invaraiance is a desirable property to deal with sets.
Long Short-Term Memory (LSTM) models are not permutation invariant. LSTM process sequential data in a recurrent fashion and maintain a kind of memory. Order matters for natural language processing or time series but not for sets.
LSTM are therefore not suited for sets processing.

We confirm this with the figure 1 from task 7.

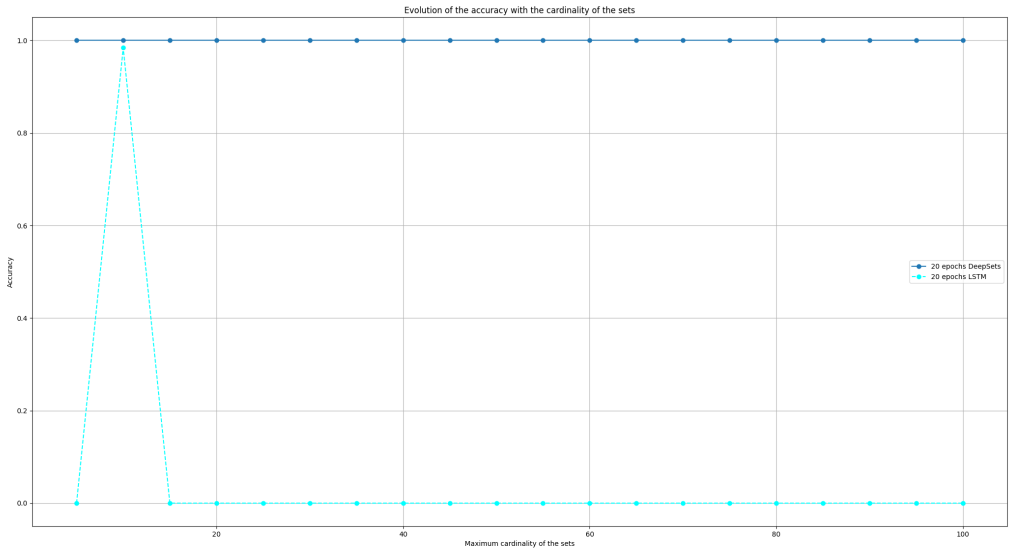

Figure 1: Comparison of accuracies of the prediction of the sum over a set of integers with regard to the cardinality of the set. Deepset is able to generalize while LSTM fails (only rougly correct when summing 10 digits, in the regime it was trained).

Task 7

Please note that this figure may be a bit misleading as the results of deepSet look almost too good to be true (100% accuracy), which is why I also report the accuracies of Deepset and LSTM while training 2. Rounding the predictions while estimating the accuracy (as a classification) also lets us think that the DeepSet model

perfectly learnt how to add integers. This is not totally true if we take a careful look at the Mean Absolute Error (MAE) which is around 0.2 when we sum 100 integers. 3.

100% accuracy = learn to add ... add to learn - Interpretation (out-of-scope)

It is important to note that one can think of learning the addition on a set as a degenerate case for the Deepset architecture.
Special embeddings like $[1, 1, 1, 1, 0, ..., 0]$ can encode the digit $4 = 1 + 1 + 1 + 1...$ since a fully connected layer can simply perform identity mapping, the pooling sum across the set can perform the actual sums on each component of the hidden vectors... which is the actual task we're learning here. The availabiltiy of the sum operation in the pooling layer is the key to this trick. The final fully connected layer can compensate the $tanh$ non linearity ($tanh(1) \approx 0.76$, $tanh(0) = 0$). There may be many other ways to achieve this, but this one is the simplest I can think of.
Multiplication would probably be much harder.

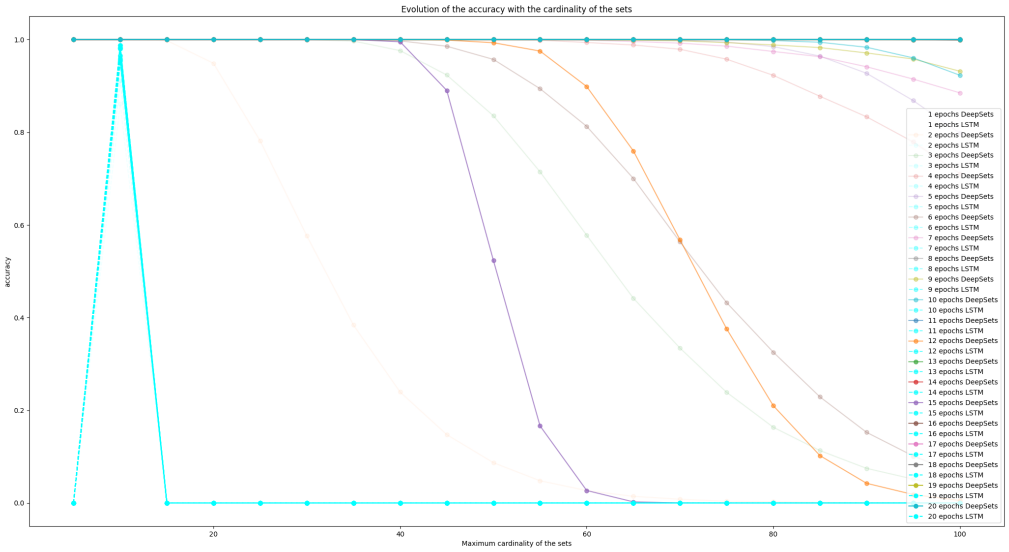

Figure 2: Evolution of the accuracies regarding cardinality during training

Question 2 : GNN vs DeepSet

Graph Neural Networks for graph-level tasks use graph structures to detect patterns (like graph classification) or properties. Deepsets do not consider edges between nodes except self loops. DeepSet is a permutation invariant architecture. It is not suited for graph-level tasks where structure matters.
GNN use aggregation/message passing layers to propagate information between nodes. Deepset does not.
What GNN for graph-level tasks and DeepSet have in common is the readout function (a.k.a. pooling layer).
Although there's no explicit ordering between the nodes, the relationship between nodes is considered in the GNN architecture: aggregation layers relies on the permutation equivariance.
We can consider of DeepSet as a degenerate instance of GNN. 2 options to understand the concept.

- Deepset is an instance of GNN where the aggregation layers are discarded. The fully connected layers, non linear activations and readout function (global pooling) are the layers in common.
- Deepset is an instance of GNN where the agrregation layers are used but with a degenerate graph construction where nodes are isolated (degree 0, no edges except self loops). The adjacency matrix becomes the identity matrix.

Processing a set using DeepSet or a GNN with a degenerate edgeless graph construction with only self loops is mathematically equivalent. *Using the GNN code from lab6 would be unefficient on the set.*

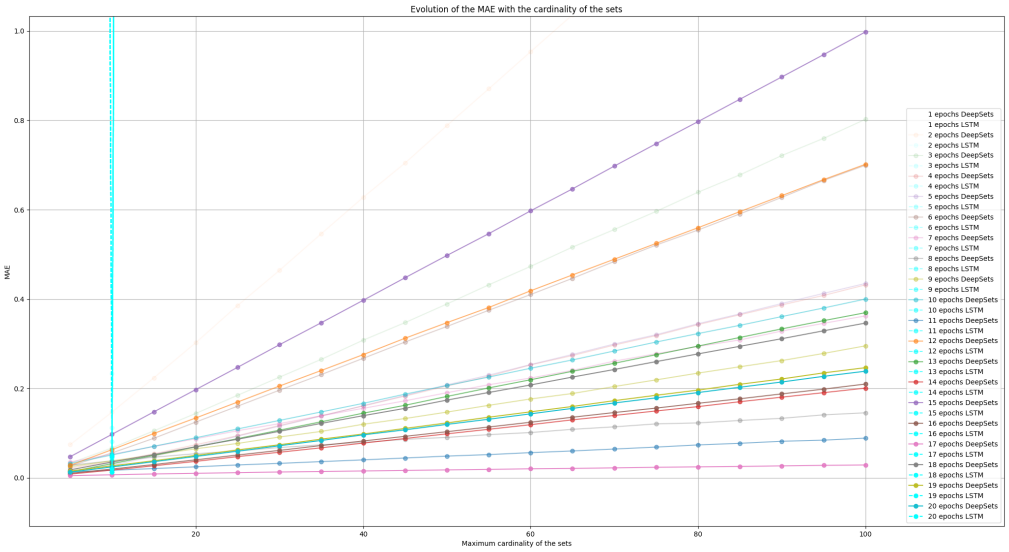

Figure 3: Evolution of Mean Absolute error regarding cardinality, during training

Note: Some graphs are only defined by their connectivity, no node features are required (same feature vector for all nodes). In a set, we can't consider that elements (edgeless nodes) are featureless. A feature is what inherently describes an element of a set!

2 Graph generation using graph variational auto-encoders

Question 3 : stochastic block model (SBM)

Stochastic Block Model with $r = 2$ communities/blocks

Edge probability matrix $P \in [0,1]^{2\times 2}$ is a symmetric matrix.

$$P = \begin{pmatrix} p_{\text{in}} & p_{\text{out}} \\ p_{\text{out}} & p_{\text{in}} \end{pmatrix}$$

- **Homophilic Graph**: In this scenario, the probability of forming edges within the same community is higher than forming edges between different communities: $p_{\text{in}} > p_{\text{out}}$ (e.g., $p_{\text{in}} = 0.8$, $p_{\text{out}} = 0.1$).

$$P_{\text{hom}} = \begin{pmatrix} 0.8 & 0.1 \\ 0.1 & 0.8 \end{pmatrix}$$

- **Heterophilic Graph** The probability of forming edges between different communities is higher than within the same community. $p_{\text{out}} > p_{\text{in}}$ (e.g., $p_{\text{in}} = 0.1$, $p_{\text{out}} = 0.8$).

$$P_{\text{het}} = \begin{pmatrix} 0.1 & 0.8 \\ 0.8 & 0.1 \end{pmatrix}$$

Expected number of edges between different blocks

Case study: $n = 20$, $r = 4$, $p_{\text{out}} = 0.05$ ($k = \frac{n}{r} = 5$ nodes per block)
The $r = 4$ stochastic block model describes graphs with $n = 20$ nodes, with P has off-diagonal elements $p_{out} = 0.05$.
The expected number of edges between different blocks is given by the following formula:

$$E = \text{Number of pairs between different blocks} \times p_{\text{out}}$$

$$E = \frac{n^2 - r \times k^2}{2} \times p_{\text{out}} = \frac{20^2 - 4 \times 5^2}{2} \times p_{\text{out}} = 150 * 0.05 = 7.5$$

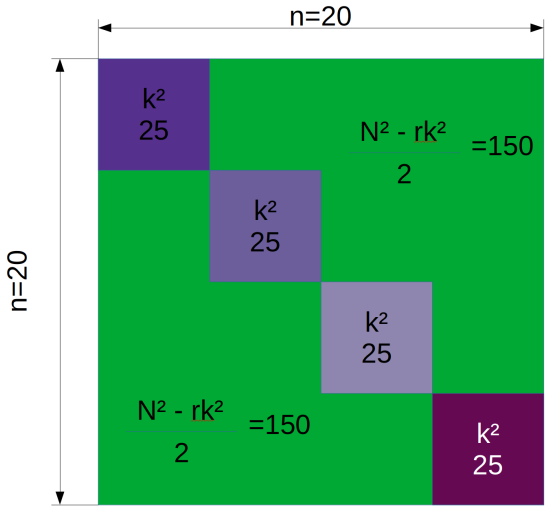

Figure 4: Adjacency matrix has a total of n^2 elements. r blocks of size $k^2 = 25$ describe all "intra community" elements. The total number of edges between different blocks is $\frac{n^2 - r \times k^2}{2}$ *(we divide by 2 because the adjacency matrix is symmetric since we deal with undirected graphs)*

The expected number of edges between nodes in different blocks of the stochastic block model, with the given parameters, is **7.5**.

Question 4 : Generate non binary graphs - use the Frobenius norm

The binary cross entropy loss is a good choice for classification problems, but is not suited for regression problems. If we want to generalize to non binary edges graphs generations, instead of using the binary cross entropy loss, we can use the **mean squared error** loss on all elements of the error matrix $\tilde{A} - A$

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

Task 11: Graph generation with VAE

See figure 5

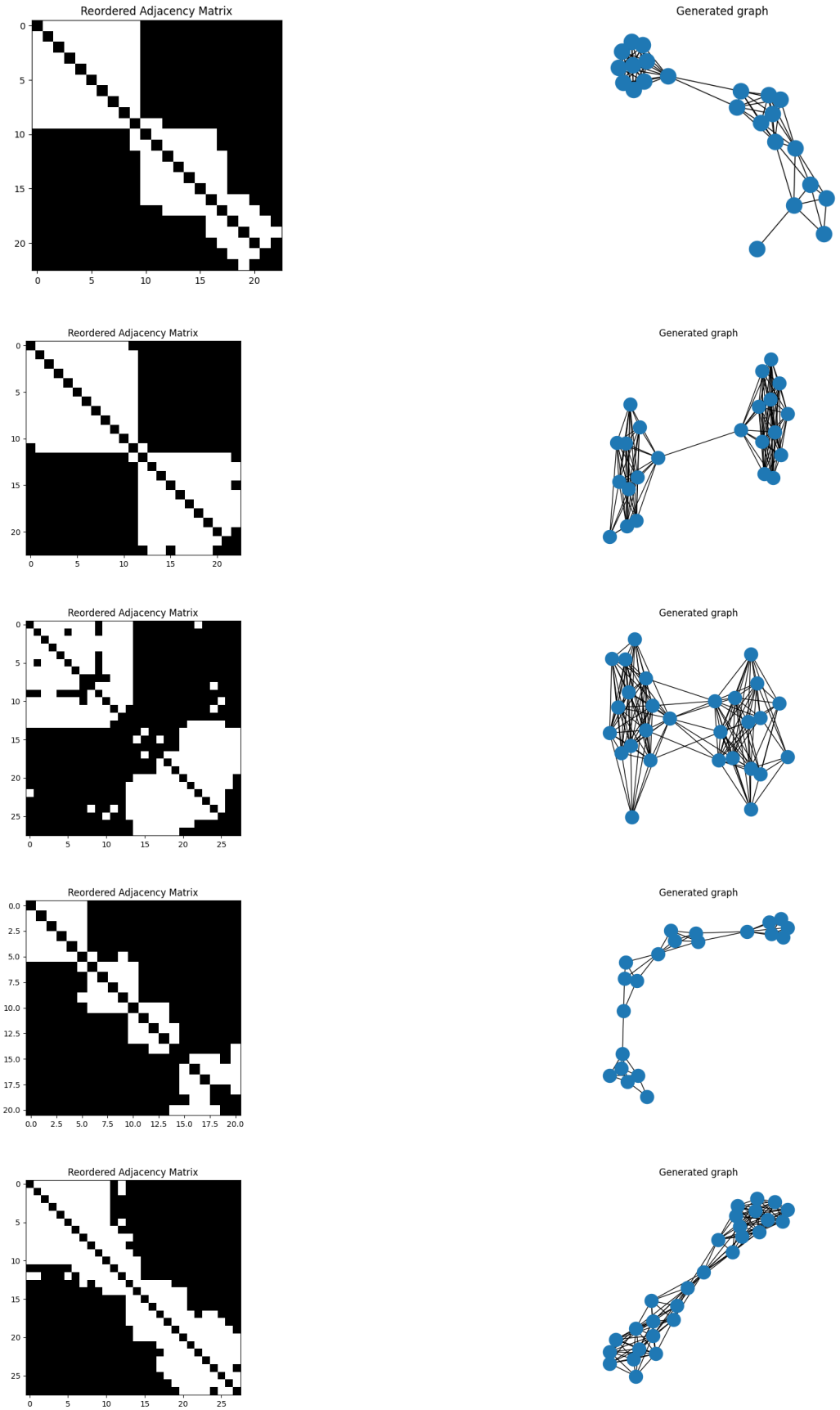

Figure 5: Generation of 5 graphs using the Variational graph autoencoder inference. The adjacency matrix is on the left, the generated graph on the right. The sampled graphs generally look like they'd been sampled from the SBM model as expected.