Alexandre Zajac
zajac.alexandre@gmail.com

Lab session # 7
ALTEGRAD 2020

01/22/21

1 Question 1

We place ourselves in the situation where we have 4 sets with the same cardinality (2) and the same sum of elements (0). At the beginning of the training, the representation of the vectors/sets in the embedding space is going to be different since it will act as a lookup table first, and it's the same phenomenon for the 2 other weight matrices W_1 and W_2. However, after updating the 3 weights matrices W_0 (embedding), W_1 and W_2 after each epoch through back-propagation, it's likely that the representations of the sets are going to be closer and closer, even in the embedding space, and the network might also bring their representation together by picking up the symmetry characteristic between the 2 elements of each set.

2 Question 2

If we place ourselves in the graph classification problem, where we have each graph being a single sample, then we can totally use DeepSets as a module of a graph neural network architecture. In fact, if we want to classify the graphs, we have to somehow embed a representation of each the graph's nodes, and since every graph doesn't have the same number of nodes (different cardinality), we could apply a similar logic to DeepSets as an intermediary step in our model.

Instead of embedding numbers, we could reduce the dimension of our input nodes for each of the graph to a single element: $\{z_1, z_2, ..., z_n\}$ if the graph has n nodes, and we could apply a readout function to combine these representations and pass it through an MLP with m final $Dense()$ layer(s) if we are dealing with a m-class classification problem.

3 Question 3

For computing the number of sub-graphs on which we have to evaluate the influence spread, we need to know how many possible starting configurations we can have for the algorithm. In fact, let's establish that we are computing the influence spread of a set S of k nodes on a graph containing n nodes.

For each subset S of nodes, we have to evaluate the influence spread on 1 sub-graph, containing the remaining $n - k$ nodes. So in total, we have to evaluate the influence on as many sub-graphs as we have starting points, which is: $\binom{n}{k}$.

4 Question 4

The properties of the greedy algorithm are such that extracting seeds with it is always going to yield an influence spread that will be higher than selecting top-k nodes according to some centrality measure (the number of edges for example). Selecting such nodes is in fact quite random because we don't know much about the activation probabilities of each of these node's edges.

What is not random, is the process of the greedy algorithm at each iteration:

$$v = argmax_{u \in V \setminus S} f(S \cup \{u\}) - f(S) \tag{1}$$

For any given f when computing the spread influence, even with an approximation, the greedy algorithm is (by design) **guaranteed** to chose the node that maximizes the current state, according to the spread influence. If we were tor run the simulations for a number $n \to \infty$, the algorithm would converge to the exact maximum of the influence spread.