Gertner Victor
gertnervictor@gmail.com

Lab session # 6
ALTEGRAD 2023

11/28/23

1 Question 1

So when we take a Erdos-Reyni graph with n nodes, edges are randomly distributed. If we take a focus on a specific node, we have a binomial law : a node can be linked to n-1 other nodes, and for each edge possible we have a p probability for the node to exist.
Hence we can write :

$$\mathbb{P}(deg(v) = k) = \binom{n-1}{k} p^k (1-p)^{n-1-k} \tag{1}$$

So we have :

$$\mathbb{E}(deg(v)) = (n-1) * p \tag{2}$$

So in our two cases :

- **n = 25, p = 0.2** : $\mathbb{E}(deg(v)) = 4.8$

- **n = 25, p = 0.4** : $\mathbb{E}(deg(v)) = 9.6$

2 Question 2

We can see two main issue by using a fully connected layer as the readout function :

- **Permutation invariant :** A fully connected layer is not permutation invariant, for instance it exactly learn patterns out of the order of the input. And this is an issue since our graph are made of unordered nodes, we want the readout function to be permutation invariant.
- **Size Variability :** Readout function like sum or mean don't have any issue with size variability of the graph they are working with. On the other hand, the input size of a fully connected layer is fixed and cannot be adapted to what it's working with. For instance if we say we set our input size to 3, it could maybe do zero padding to be able to process G3, but it wouldn't be able to be adapted to G2 that has 4 nodes. And that's a huge issue since we are working with various size of graphs.

3 Question 3

Here is the result for all the combinations :

- **aggr = sum, readout = sum** : all the represenation of the graphs are different.
- **aggr = sum, readout = mean** : all the representation of the graphs are the same.
- **aggr = mean, readout = mean** : all the representation of the graphs are the same.
- **aggr = mean, readout = sum** : all the representation of the graphs are different.

So we see that what makes the fact that the representation are different or not is the value of the readout, when it's a sum they are different and when it's a mean they are the same. This might be explained by the fact that the mean induce a loss of information : it's not an injective function. Whereas the sum is an injective function. This is why, in our case we have only cycle graphs : even if they are bigger and bigger, the nodes have the same property (number of neighbors). Hence if we take the mean as the last readout, we have a non injective function on really close in terms of structure graphs : it cannot distinguish them. And if we take a sum as the readout function, it can.

4 Question 4

In order to find other graphs that can never be distinguished by the GNN model which uses the sum operator both for neighborhood aggregation and as the model's readout function, we could use the same logic as the lab sheet and take for G1 : two 4 cycles and for G1 an 8 cycle, this works.
But we might want to have other architecture of graphs.
We can look at regular graphs (each node has the same degree), they tend to have if they are of the same order really close local structure and WL can struggle on regular graphs and since we saw in class that the GNN we consider cannot do better than W-L we might want to go this way. Hence, if we take those two graphs :

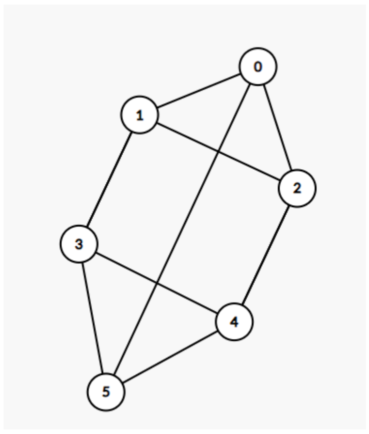

Figure 1: G1

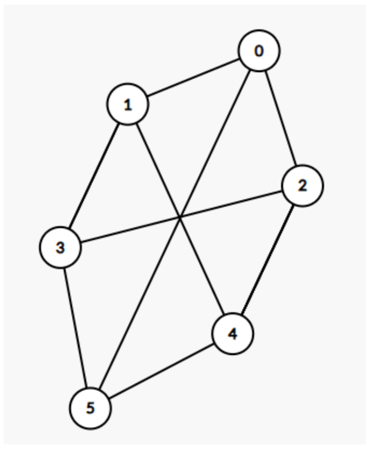

Figure 2: G2

Figure 3: The two graphs we consider

So as we can see, G1 as 3 cycle in it while G2 as a minimum size of cycle equal to 4, so we know they are not isomorphic. But when we put them in our algorithm, it gives the same representation at the end : it cannot distinguish them !