Boubacar SOW
boubacar.sow@ens-paris-saclay.fr

Lab session # 6
ALTEGRAD 2023

11/28/23

1 Expected number of degree of a node in Erdos-Renyi random graphs with $n = 25$ and edge probabilities $p = 0.2$ and $p = 0.4$

In an Erdos-Renyi graph model, we start with n isolated nodes. Then, for each pair of nodes, we draw an edge between them with probability p, independently of the other edges. Each node can potentially connect with any of the remaining $n-1$ nodes. The process of deciding whether an edge exists between two nodes is akin to a Bernoulli trial. When we consider all $n-1$ potential connections for a given node, we're conducting multiple Bernoulli trials. This scenario is described by a Binomial distribution The expected number of degree of a node is the expected value of this Binomial distribution, which is the product of the number of trials and the probability of success, i.e., $(n-1) \times p$.

- For $p = 0.2$, the expected number of degree of a node is $n - 1 * 0.2 = 24 * 0.2 = 4.8$
- For $p = 0.4$, the expected number of degree of a node is $n - 1 * 0.4 = 24 * 0.4 = 9.6$

2 Why trainable linear layer are not commonly used as readout functions instead of the sum or mean operation in graph level GNNs

In the graph level GNNs, the graph-level readout is the step where we aggregate the features of all nodes in each graph to generate a single vector representation of the entire graph (all the graphs considered as a single graph with many connected components). In this step we use commonly sum or mean operations for some reasons:

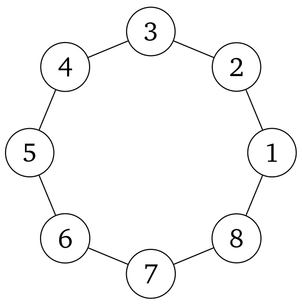
1. Permutation Invariance: Sum and mean operations are also permutation-invariant, which makes them suitable for graph-level readout. However, a fully connected layer is not permutation-invariant, and different node orders would lead to different outputs. This matters because graphs are inherently permutation-invariant, meaning that the order of nodes does not matter
2. Computational Efficiency: Sum and mean operations are more computationally efficient than fully connected layers. They do not introduce any additional parameters, unlike fully connected layers which would significantly increase the model complexity and computational cost with large graphs.
3. Variable Input Size: In many real-world scenarios, graphs can have a variable number of nodes and edges. A fully connected layer requires a fixed-size input, so it cannot handle graphs of different sizes unless some form of padding or truncation is used. On the other hand, sum and mean operations can naturally handle inputs of any size, as they operate element-wise and do not require the input to have a specific size.

3 Comments on the output produced in task 8

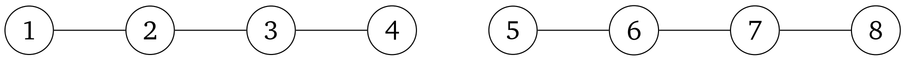
In Task 8, the Graph Neural Network (GNN) model was tested with different combinations of neighbor aggregation and readout functions. The output of the model is a tensor that represents the class scores for each graph in the batch.
The results show that the choice of neighbor aggregation and readout functions can significantly affect the output of the model. For example, when both the neighbor aggregation and readout functions are "mean", the model produces identical class scores for all graphs. This is because the "mean" operation is invariant to the number of elements, so the size of the graph (i.e., the number of nodes) does not affect the output.
On the other hand, when the readout function is "sum" (regardless of the neighbor aggregation function), the model produces different class scores for graphs of different sizes. This is because the "sum" operation is sensitive to the number of elements, so the size of the graph affects the output.

4 Example of two non-isomorphic which can never be distinguished by the GNN model which uses the sum operator both for neighborhood aggregation and as the model's readout function.

Let's have the following graphs:



(a) Graph G1: A cycle graph of length 8, with 8 vertices



(b) Graph G2: Union of two cycle graphs of length 4, with 8 vertices

Both G1 and G2 have 8 nodes and 8 edges, but they have different structures. However, a GNN model that uses the sum operator for both neighborhood aggregation and readout function would produce the same output for both graphs.
This is because the sum operator simply adds up the features of the nodes, without taking into account the specific connections between them. In both G1 and G2, each node has exactly two neighbors, so the sum of the neighbor features will be the same for all nodes in both graphs. Similarly, the sum of all node features (the readout) will also be the same for both graphs

```
tensor([[-3.8905, -0.9120,  5.3966,  2.2837],
        [-3.8905, -0.9120,  5.3966,  2.2837]], grad_fn=<AddmmBackward0>)
```

References