1 Question 1

What is the expected number of degree of a node in Erdős–Rényi random graphs with $n = 25$ nodes and edge probabilities $p = 0.2$ and $p = 0.4$?

We consider an Erdős–Rényi random graph, where an edge has a probability p of being included in the graph. We assume the graph undirected and without any self-loop.
Let $E_{i,j} \sim \mathcal{B}(p)$ be the random variable denoting the existence of an edge between a node i and a node j. We define the random variable N counting the number of edges as follows:

$$N = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} E_{i,j}$$

By independance of the $(E_{i,j})_{i,j}$, we have that $N \sim \mathcal{B}in(\frac{n(n-1)}{2}, p)$, and consequently $\mathbb{E}[N] = \frac{n(n-1)}{2}p$.
For $n = 25$, we have:

$$\mathbb{E}[N] = \begin{cases} 60 & \text{for } p = 0.2 \\ 120 & \text{for } p = 0.4 \end{cases}$$

2 Question 2

Could you briefly comment on why trainable linear layers, i.e., fully connected layers, are not commonly used as readout functions instead of the sum or mean operation in graph level GNNs?

We recall the following elements:

- when performing graph classification, since different graphs may have different number of nodes from each other, possibly the best approach to create mini-batches is to concatenate the respective feature matrices and build a (sparse) block-diagonal matrix where each block corresponds to the adjacency matrix of one graph (c.f. figure 1)
- in the readout phase, for each graph, we need to apply the readout function to the rows of the feature matrix that correspond to the nodes of that graph. To achieve that, we can create a membership indicator vector which for each node indicates the graph to which this nodes belongs, and then to use an aggregation function.

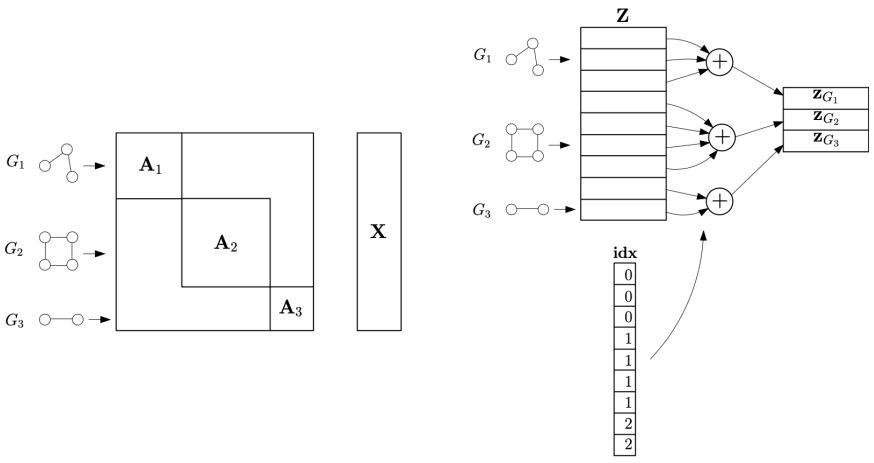

Figure 1: Implementation details of the graph neural network

Building on this approach, it would not make sense to use trainable linear layers as readout functions in graph level GNNs for two reasons.

The first one is that this whole problem setting (requiring the construction of a sparse block-diagonal matrix) is enforced by the fact that different graphs may have different number of nodes. This means that the block-diagonal matrix composed of the adjacency matrices, the feature matrix X and therefore the output Z will have different sizes depending on the mini-batch: this makes the training of a layer irrelevant since such layer would have a changing size at each batch.
The second argument advocating against such practice is that even if all graphs had the same number of nodes, using a trainable layer would raise the issue that the network would no longer be permutation invariant. Indeed, using a fully-trained linear layer as readout function, the output given for a specific graph would then be different depending on the construction of the adjacency matrix: if one were to change the order in which the nodes are considered, the network would have a different output.

3 Question 3

Please comment in a few sentences what you observe in the output produced in Task 8. How can the obtained results be explained?

After comparing all the possible combinations combinations of the sum and mean operators for neighborhood aggregation and for readout, we note two things.

The first one is that no matter the neighborhood aggregation, the MessagePassing layers will output the same tensors for each graph. This makes sense given the shared cyclic structure of the graphs: whether we compare the sum of the neighboring nodes or the mean, we will get similar results since each node has two neighbors and we use identical node features. The local structure around each node being identical, we output the same result for each cycle.

The second observation is based on the readout function. When we use the mean we get identical outputs for each graph, whereas we have different representations when using the sum. Indeed, the output of the MessagePassing layers being the same for each of the cycles, the readout functions take the same input. When computing the sum, the node representations of each graph are being summed, thus differing by a factor corresponding to the difference between the node numbers (let z be the identical vector of representation composing the rows of Z obtained after the MessagePassing layers, if a graph has x nodes the its readout will be computed as $x \times z$). In our case, since the cycles have different lengths, we get different graph representation. However, when using the mean as readout, the representations obtained after applying the sum are divided by the number of nodes: thus, we get once again the same representation for each cycle.

4 Question 4

Give an example of two non-isomorphic graphs G1 and G2 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.

In Task 11, we observe that the representations obtained for the two graphs G1 and G2 using the *sum* operator as neighborhood aggregation and readout function are the same: considering the fact that the two graphs are non-isomorphic, obtaining the same representation shows the limited expressive power of the GNN built as such.

The graphs in figure 2 represent another limit case of the expressiveness of GNNs. It was expected, as those two graphs cannot be distinguished by the Weisfeiler-Lehman graph isomorphism test, and it has been shown that Graph Neural Networks are at most as powerful as the WL test of isomorphism in terms of distinguishing between non-isomorphic graphs [1, 2].
After testing the GNN on both, we indeed get the same representation when using the sum operator as neighborhood aggregation and as readout function.

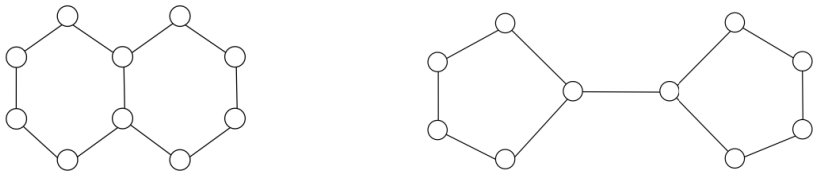

Figure 2: Non-isomorphic graphs indistinguishable by a GNN

References

[1] Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *AAAI Conference on Artificial Intelligence*, 2018.

[2] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.