Kyllian Asselin de Beauville
kyllian.asselin-de-beauville@polytechnique.edu

Lab session # 6
ALTEGRAD 2023

28/11/23

1 Question 1

In an Erdős–Rényi graph $G(n, p)$, where n is the number of nodes and p is the probability of an edge between any two nodes, each node can potentially connect with any of the other $n - 1$ nodes, and each of these connections occurs independently with probability p. This leads to the degree distribution of a node following a Binomial distribution, specifically $B(n - 1, p)$. Consequently, the expected number of degrees $\mathbb{E}[d]$ of a node in such a graph is given by the formula:

$$\mathbb{E}[d] = p \times (n - 1) \tag{1}$$

Given $n = 25$, we calculate the expected degrees for $p = 0.2$ and $p = 0.4$ as follows:

- **For $p = 0.2$:**
$$\mathbb{E}[d] = 0.2 \times (25 - 1) = 0.2 \times 24 = 4.8 \tag{2}$$
- **For $p = 0.4$:**
$$\mathbb{E}[d] = 0.4 \times (25 - 1) = 0.4 \times 24 = 9.6 \tag{3}$$

2 Question 2

Trainable linear layers, also known as fully connected layers, are not typically used as readout functions in graph level Graph Neural Networks (GNNs) [1, 2] for several reasons, particularly due to their lack of permutation invariance and inefficiency when dealing with variable-sized inputs. In the context of Figure 1, consider graphs G_1, G_2, and G_3 with respective adjacency matrices A_1, A_2, and A_3, and a shared feature matrix X.

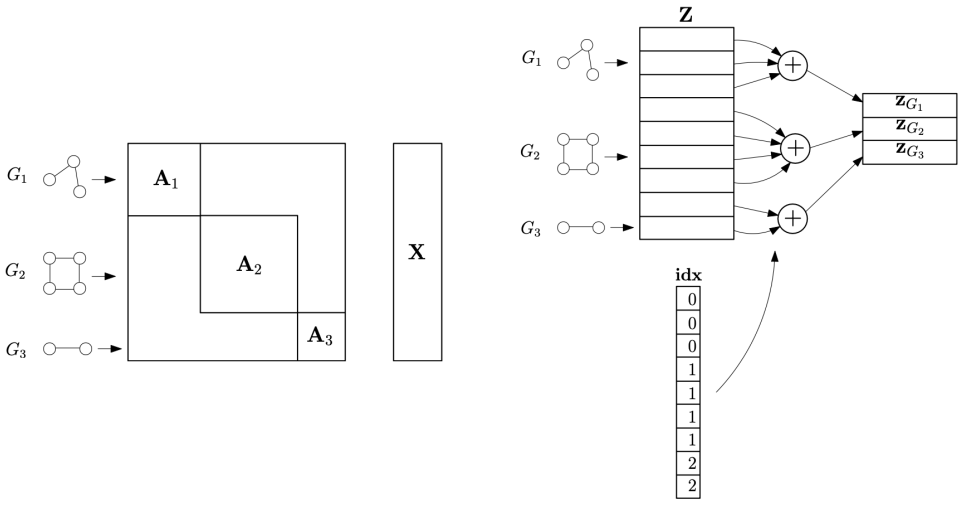

Figure 1: Implementation details of the graph neural network.

- **Permutation invariance:** Graph data is inherently unordered. Two graphs can be considered the same (isomorphic) even if their nodes are listed in different orders. A sum or mean operation over nodes respects this property, as changing the order of nodes does not change the output of these operations. In contrast, a linear layer does not handle such permutations, leading to different outputs for isomorphic graphs unless additional mechanisms are in place.

- **Variable-sized inputs:** Graphs G_1, G_2, and G_3 may have different numbers of nodes (n_1, n_2, and n_3 respectively). A linear layer requires a fixed-size input, which is not practical for graphs of varying sizes. Conversely, sum and mean operations are agnostic to the number of inputs, making them more suitable for aggregating node features into a fixed-size graph representation.

3 Question 3

`neighbor_aggr='sum', readout='sum'`

```
[[ -6.7333,  18.9288,   7.7692,  -7.2437],
 [ -7.4120,  20.8317,   8.5455,  -7.9575],
 [ -8.0907,  22.7346,   9.3218,  -8.6713],
 [ -8.7694,  24.6375,  10.0981,  -9.3851],
 [ -9.4481,  26.5404,  10.8744, -10.0989],
 [-10.1268,  28.4432,  11.6507, -10.8127],
 [-10.8055,  30.3461,  12.4270, -11.5265],
 [-11.4842,  32.2490,  13.2033, -12.2403],
 [-12.1629,  34.1519,  13.9797, -12.9541],
 [-12.8416,  36.0548,  14.7560, -13.6679]]
```

The output vectors increase in magnitude with the size of the graph. This observation aligns with the expectation that both the sum aggregation and the sum readout will accumulate features. This is the most expressive configuration among the four, as it maximizes the retention of structural information by accumulating all node features without normalization, making it particularly effective for distinguishing between non-isomorphic graphs.

`neighbor_aggr='sum', readout='mean'`

```
[[-1.1792, -1.0064,  0.2699,  0.5239],
 [-1.1792, -1.0064,  0.2699,  0.5239],
 [-1.1792, -1.0064,  0.2699,  0.5239],
 [-1.1792, -1.0064,  0.2699,  0.5239],
 [-1.1792, -1.0064,  0.2699,  0.5239],
 [-1.1792, -1.0064,  0.2699,  0.5239],
 [-1.1792, -1.0064,  0.2699,  0.5239],
 [-1.1792, -1.0064,  0.2699,  0.5239],
 [-1.1792, -1.0064,  0.2699,  0.5239],
 [-1.1792, -1.0064,  0.2699,  0.5239]]
```

The output vectors are not only uniform across graphs of different sizes but also identical. This indicates that the mean readout normalizes the summed features to a fixed value, reflecting that each node has the same degree in cycle graphs.

`neighbor_aggr='mean', readout='sum'`

```
[[-2.5640,  0.1438, -1.2906,  3.5066],
 [-2.8131,  0.1476, -1.4278,  3.8703],
 [-3.0623,  0.1515, -1.5650,  4.2340],
 [-3.3114,  0.1554, -1.7022,  4.5977],
 [-3.5606,  0.1593, -1.8393,  4.9614],
 [-3.8098,  0.1631, -1.9765,  5.3251],
 [-4.0589,  0.1670, -2.1137,  5.6888],
 [-4.3081,  0.1709, -2.2509,  6.0525],
 [-4.5573,  0.1748, -2.3880,  6.4161],
```

```
 [-4.8064,  0.1787, -2.5252,  6.7798]]
```

The output vectors increase with graph size but the increments are less pronounced compared to the 'sum-sum' case. This is because mean aggregation normalizes the features per node, then the sum readout accumulates these normalized features across the graph.

`neighbor_aggr='mean', readout='mean'`

```
[[ 0.6273, -0.3965,  0.0074, -0.4173],
 [ 0.6273, -0.3965,  0.0074, -0.4173],
 [ 0.6273, -0.3965,  0.0074, -0.4173],
 [ 0.6273, -0.3965,  0.0074, -0.4173],
 [ 0.6273, -0.3965,  0.0074, -0.4173],
 [ 0.6273, -0.3965,  0.0074, -0.4173],
 [ 0.6273, -0.3965,  0.0074, -0.4173],
 [ 0.6273, -0.3965,  0.0074, -0.4173],
 [ 0.6273, -0.3965,  0.0074, -0.4173],
 [ 0.6273, -0.3965,  0.0074, -0.4173]]
```

The output vectors are the same for all graphs, which shows that the mean operations for both aggregation and readout provide size-invariant graph representations. This is desirable for tasks that require comparable graph embeddings regardless of the graph size.

4 Question 4

Figure 2 is an example of two non-isomorphic graphs that cannot be distinguished by a GNN model using the sum operator for both neighborhood aggregation and readout. G_1 is composed of two disconnected complete graphs K_4, each with 4 nodes and 6 edges. G_2, derived from G_1, has one edge removed from each K_4 and an additional edge connecting the two components, forming a bridge. Both graphs have the same number of nodes and edges.

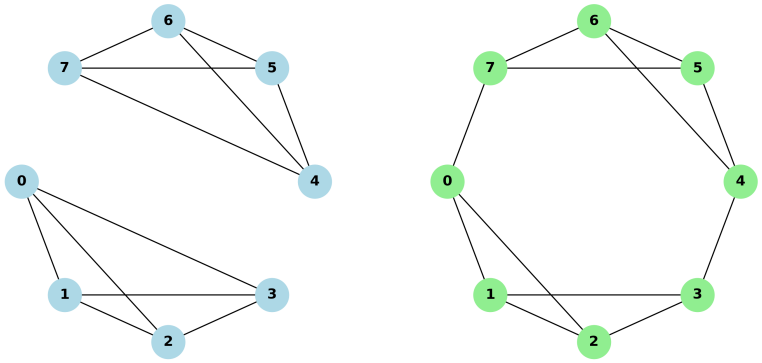

Figure 2: Non-isomorphic graphs G_1 and G_2.

```
[[ 0.8164,  4.6034, -3.0981,  3.6430],
 [ 0.8164,  4.6034, -3.0981,  3.6430]]
```

The GNN model with sum operations for both aggregation and readout produces identical embeddings for G_1 and G_2 because the sum aggregation results in the same total feature value for each node, based on their

degree, and the sum readout compiles these features into a graph-level representation. Due to the identical number of nodes with matching degrees, the GNN model cannot capture the structural differences between the two graphs.

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, 2021.

[2] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks?, 2019.