Basile Terver
basile.terver@polytechnique.edu



# Question 1

## Litteral expression

In the $G(n, p)$ model, each of the $\frac{n(n-1)}{2}$ possible edges is considered independently from the others and added with probability $p$.
Thus, for each node $v$, being connected to any of the $n-1$ potential neighbors is of probability $p$ and are disjoint events. Therefore, by additivity of the probability, we get, that the expected degree $d_v$ of any $v \in V$ is

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

## Numerical application

- If $p = 0.2$, we get $\mathbb{E}[d_v] = 24 \times (0.2) = 4.8$ for all $v \in V$.
- If $p = 0.4$, we get $\mathbb{E}[d_v] = 24 \times (0.4) = 9.6$ for all $v \in V$.

# Question 2

For graph-level classification, the readout function must aggregate the information among **all** nodes. This is called pooling and is the graph equivalent of pooling layers of CNNs for example. What is expected for a readout function is

- To be invariant to permutation of nodes
- To not depend on the size of the input graph

In the example given with $G_1, G_2, G_3$, suppose we use a trainable fully connected layer as readout.
Since a weight matrix is not permutation invariant. Therefore, permutating the nodes in one of the 3 graphs or permutating the graphs in the minibatch would yield a different prediction, which is not wanted.

# Task 3

The quality of the training and therefore of the test accuracy generally varies between 0.7778 and 1. Here are some training results upon relaunching the whole gnn.py script.

```
Epoch: 0191 loss_train: 0.4050 acc_train: 0.8090 time: 0.0450s
Optimization finished!
loss_test: 0.3406 acc_test: 0.8889 time: 0.0520s
```

Second run:

```
Epoch: 0191 loss_train: 0.1617 acc_train: 0.9326 time: 0.0451s
Optimization finished!
loss_test: 0.0968 acc_test: 1.0000 time: 0.0511s
```

Third run:

```
Epoch: 0191 loss_train: 0.1807 acc_train: 0.9326 time: 0.0440s
Optimization finished!
loss_test: 0.5820 acc_test: 0.7778 time: 0.0450s
```

# Task 6

Contrary to what we did in the previous lab, the message passing layer we implement in this part of the lab does not add self-loops to the adjacency matrix.
The reason is we do not want the weights associated to a node's own features (here $W_1$) to be the same as the weights associated to its neighbors (here $W_0$).

# Question 3

Here are what the four combinations of neighborhood aggregation and readout functions yield in terms of embeddings of the 10 cycle graphs $\{C_{10}, \dots, C_{19}\}$, which surely are not isomorphic:

- neighbor aggregation= mean, readout=mean : equal embeddings.
- neighbor aggregation= mean, readout=sum : different embeddings.
- neighbor aggregation= sum, readout=mean : equal embeddings.
- neighbor aggregation= sum, readout=sum : different embeddings.

Let us explain the reason for this situation following the processing of the input by the GNN:

- input $X \in \mathbb{R}^{n \times d}$ filled with ones ($d = 1$).
- $XW \in \mathbb{R}^{n \times h_1}$ has equal rows because $X$ had equal rows for any weight matrix $W \in \mathbb{R}^{d \times h_1}$.
- In all cycle graphs, the degree of each node is 2, and since $XW$ had equal rows, we get $AXW_0 = 2XW_0$. Similarly, we get $D^{-1}AXW_0 = XW_0$. Thus, $AXW_0, D^{-1}AXW_0$ have equal rows and $MP(A, X)$ has equal rows, whether the neighborhood aggregation function is mean or sum.
- Since the ReLU function is applied componentwise, we have that $Z^0$ has equal rows.
- By the same reasoning as for the first message passing layer, we get that $Z^1$ has equal rows.

Now, the choice of readout function yields two situations:

- readout=sum : the 10 cycle graphs have a different number $n$ of nodes, thus a different number of rows, this unique row is denoted $v \in \mathbb{R}^{h_2}$. The sum readout function sums these $n$ rows and yields a graph feature $z_G^0 = nv \in \mathbb{R}^{h_2}$.
- readout=mean : $z_G^0 = v \in \mathbb{R}^{h_2}$.

The fact that the embedding of the cycle graph is proportional to the number of nodes when the readout function is the sum is confirmed when we print the embeddings of the 10 graphs:

```
neighbor_aggr='mean', readout='sum'
 [[ 0.3679224  -3.4970803  -4.3724146  -0.51223105]
 [ 0.39076364 -3.8398468  -4.8184648  -0.57906353]
 [ 0.41360497 -4.182613   -5.264515   -0.64589655]
 [ 0.4364463  -4.5253797  -5.7105646  -0.71272933]
 [ 0.45928758 -4.8681464  -6.1566143  -0.77956206]
 [ 0.48212898 -5.2109127  -6.602664   -0.84639484]
 [ 0.50497013 -5.5536795  -7.0487137  -0.9132281 ]
 [ 0.5278115  -5.8964467  -7.4947634  -0.9800605 ]
 [ 0.55065256 -6.2392125  -7.9408135  -1.0468936 ]
 [ 0.5734942  -6.5819793  -8.386864   -1.1137257 ]]

neighbor_aggr='sum', readout='sum'
 [[ 6.2862887 11.654835   7.5293593  6.9657407]
 [ 6.899348  12.812969   8.280235   7.6682854]
 [ 7.5124073 13.971107   9.031115   8.3708315]
 [ 8.125467  15.129243   9.781992   9.073376 ]
 [ 8.738525  16.287378  10.532871   9.775922 ]
 [ 9.351584  17.445515  11.283748  10.478464 ]
 [ 9.964644  18.603651  12.034627  11.181011 ]
 [10.577703  19.761786  12.785504  11.883557 ]
 [11.190763  20.919922  13.536384  12.586103 ]
 [11.803824  22.078058  14.287261  13.288649 ]]
```

# Task 9

Figure 1 is the result of the drawing of the graph $G_1$ and Figure 2 is the result of the drawing of the graph $G_2$.

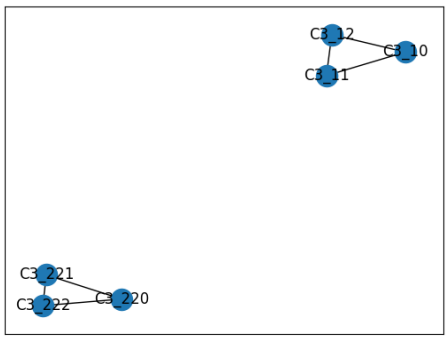

Figure 1: Drawing of the graph $G_1 = C_3 \cup C_3$.

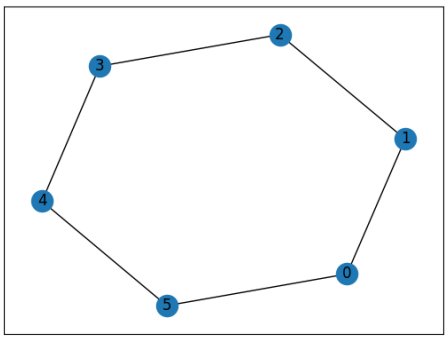


Figure 2: Drawing of the graph $G_2 = C_6$.

## Task 11

```
[[ 7.616647    0.01271515  8.505569   -0.91865665]
 [ 7.616647    0.01271515  8.505569   -0.91865665]]
```

The embeddings of $G_1$ and $G_2$ are equal. They are clearly non-isomorphic but the GNN we initialize in this task is not able to differentiate them.
Since $X$ has equal rows (all equal to 1) and all nodes have same degree equal to 2 in both cases, we can follow the reasoning of question to explain this equality of embeddings of $G_1$ and $G_2$. Even though the readout function is sum, the GNN cannot differentiate $G_1$ and $G_2$ because they have same number of nodes $n = 6$.

## Question 4

All pairs of graphs (with same number of nodes, otherwise it's less interesting) where all the nodes have same degree works. Here is an example in Figures 3 and 4. They indeed have same node embeddings:

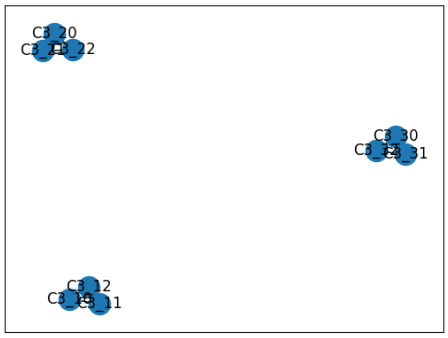

Figure 3: Drawing of the graph $G_3 = C_3 \cup C_3 \cup C_3$.

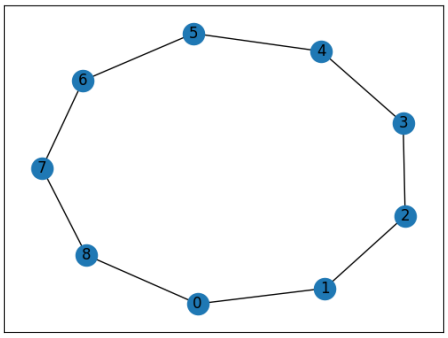


Figure 4: Drawing of the graph $G_4 = C_9$.

```
G_3 and G_4 representations
 neighbor_aggr='sum', readout='sum'
 [[ -7.648325  -14.837895   -5.0386887  -9.315835 ]
 [ -7.648325  -14.837895   -5.0386887  -9.315835 ]]
```

## References