Balthazar Neveu
balthazarneveu@gmail.com

Lab session # 6
ALTEGRAD 2023

11/23/23

Code

More info: MVA ALTEGRAD Balthazar Neveu on Github

1 Section 1: Graph-level classification

Question 1: $\mathbb{E}(d)^{p=0.2} = 4.8$ and $\mathbb{E}(d)^{p=0.4} = 9.6$

We denote a graph $G = (V, E)$ where $v_i \in V$ denote the vertices or nodes and E are the edges.

Preliminary remark : Overall statistics of edges

The total number of undirected edges when the graph is dense is $\binom{n}{2} = \frac{n(n-1)}{2}$ When $n = 25$, the densest graph will have 300 edges (when $p = 1$). The Erdős–Rényi has in average $p * \binom{n=25}{2}$ edges.

Distribution of the degree d

For a given node vi, the probability to be connected to v_j where $i \neq j$ is a Bernoulli distribution.

$$\mathbb{P}[(v_i, v_j) \in E] \sim \mathbb{B}(p)$$

Degree of edge v_i is the number of all connected nodes (number of edges). It therefore follows a Binomial distribution. The maximum number of edges is $n - 1$ ($d \leq n - 1$) The expectation of the degree d is:

$$\mathbb{E}(d) = (n - 1).p$$

Note: Computing the degree is equivalent to counting the number of flipped coins with a tail (1 toss of a coin with probability p of having a tail is equivalent to get a connected 1 edge)
Note: Obviously, degenerate case when $p = 1$, the degree reaches the maximum as we'd expect.

Numerical application

$$\mathbb{E}(d)^{p=0.2} = 24 * 0.2 = 4.8$$
$$\mathbb{E}(d)^{p=0.4} = 24 * 0.4 = 9.6$$

Question 2: Readout operator

Readout functions shall reduce the graph's "spatial dimension" (they're the equivalent for graph to global pooling operation for image and signal processing). Therefore, there are a certain amount of guideline for the readout operators:

- permutation invariant
- size agnostic

Fully connected layers do not like good "Readout" functions.
The weight matrix of a fully connected layer is fixed, so the number of nodes is fixed.
Assume for a minute you'd train a fully connected layer to replace the readout function. If you trained it on G_1, G_2, G_3 which have a different sizes number of nodes, you could use a weight matrix of size determined by the graph which most nodes (G_2 has $|V_{G_2}| = 4$ nodes here). Weight matrix would have a fixed size $(|V_{G_2}| * h_{\text{in}}, h_{\text{out}})$ and you'd have to pad the input vectors with zeros when there are not enough nodes.
This simply feels wrong: we could assume you'd pad with zeros if a new test graph has more nodes or remove some nodes when there are less nodes.
Last but not least, a fully connected layer is not permutation invariant (shuffle input features of a linear layer and you get a different result).
Closing remark: The case where it would make sense is for instance if the graph nodes are always ordered the same way, basically meaning that there's an underlying geometric structure.

One could think of MRI connectivity between areas of the brain where the nodes are always localized at the same spatial location using an atlas of "standardized areas" like the Harvard-Oxford brain atlas. Brain MRI seem to be standardized this way and this introduces hard constraints in the nature of the data.
It seems like such readout is not such a bad idea for some special graph-level problems (drug discovery mentioned Neural readouts this is an active area of research.)

Task 3

```
Epoch: 0191 loss_train: 0.3165 acc_train: 85.39% time: 0.0434s
Optimization finished!
loss_test: 0.2615 acc_test: 88.89% time: 0.0466s
```

Section 2: Graph Neural Networks expressiveness

Question 3: influence of readout and aggregation functions while discriminating Cycle graphs with GNN

In the following, $n > 2$ denotes the number of nodes in the cycle graph C_n.

- *neighbor aggregation = mean*,**readout= mean: same features** (same graph description $\forall n$)
- *neighbor aggregation = sum*, **readout= mean: same features** (same graph description $\forall n$)
- *neighbor aggregation = mean, readout= sum*: different features
- *neighbor aggregation = sum, readout= sum*: different features

Using the mean readout gives similar characteristics describing all the different cycle graphs. The expressiveness of the GNN increases with the sum readout which leads to different representation when n varies. Using the mean or sum in neighbor aggregation does not change anything in the resulting representation (simply a scaling factor). We cannot make generalities
An handwavy explanation is that the sum in the readout allows "counting" features (here the underlying count refers to the number of nodes).
Let's break things down.

- X input are all ones.
- In the message passing, $MP(A, X)$
  - projected inputs features will be mapped to the same vectors $X.W_0$ has all equal rows.
  - since all nodes in a cycle graph have 2 neighbors
  - sum aggregation: $A.(X.W_0) = 2(X.W_0)$ in this special case... will also have all equal rows.
  - mean aggregation: $D^{-1}.A.(X.W_0) = (X.W_0)$ will also have all equal rows.
  - by extension $Z^0 = ReLu(MP(A, X))$ has equal rows for the sum or mean aggregation
- most importantly, $\forall n > 2$ the rows of $Z0$ are even the same no matter the length of the cycle graph.
- The same reasoning goes for Z^1, let's define z^1 a single row of Z^1.
- Readout:

  Sum readout(Z^1): sum over all the rows which are equal. We end up with $n * u$

  Average readout(Z^1): average over all the rows which are equal. We end up with a single row of u, no matter the length n

Below is a quick numerical validation of the proportionality in n for the sum readout.

We remark the proportionality property for the sum Readout.($Z_G^1 \propto n$)

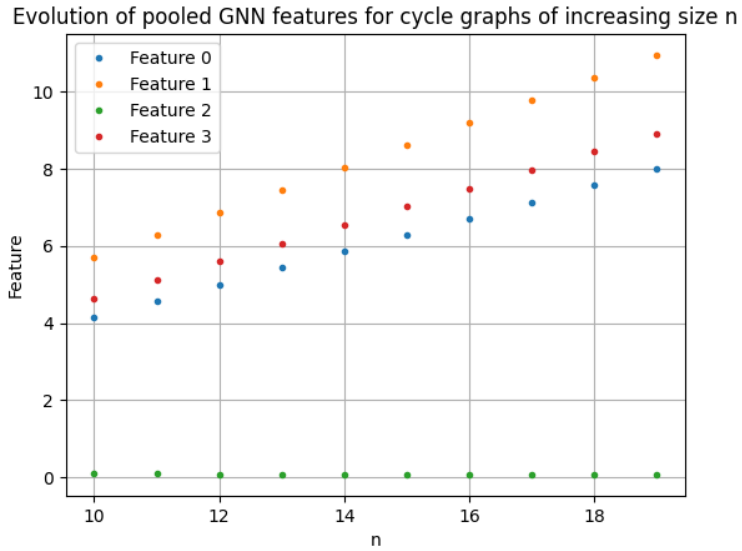

Figure 1: Evolution of graph descriptors using the sum readout. We can validate the linear evolution in n

```
neighbor_aggr='sum', readout='sum'
[
    [ 4.146614    5.709204    0.09333157  4.6471424 ]
    [ 4.5737143   6.290244    0.09161585  5.1209064 ]
    [ 5.000814    6.8712826   0.08990037  5.594671  ]
    [ 5.4279146   7.452322    0.08818489  6.068435  ]
    [ 5.8550153   8.033361    0.08646905  6.542199  ]
    [ 6.282116    8.614401    0.08475303  7.015964  ]
    [ 6.709216    9.195441    0.08303796  7.4897275 ]
    [ 7.1363163   9.77648     0.08132147  7.963493  ]
    [ 7.563416   10.357518    0.07960581  8.437258  ]
    [ 7.9905176  10.938558    0.07789027  8.911022  ]
]
```

Task 11

```
neighbor_aggr='sum', readout='sum'
G1 representation: [-2.1668684  4.630975  -3.4005415  1.7028651]
G2 representation: [-2.1668684  4.630975  -3.4005415  1.7028651]
```

Same representation although the graphs are different (not isomorphic!). We note that $G^2 = C^6$ being a cycle graph, we shown in question 3 that it will have a representation $z^1_{C^6} = 2z^1_{C^3}$.. Now we nottice that $G^1 = C^3 \cup C^3$ (2 connected components C^3). The readout will sum the features of these 2 graphs, ending up with $z^1_{C^6} = 2.z^1_{C^3}$. So this was expected.

Question 4

Example 1 and 2 are similar to the trick used in task 11 although they're different.

Example 0

G^1 is a square with 4 nodes, G^2 is made 2 connected components made of pairs (path of 2 nodes P^2) so no cycle involved in one of them

- $G^1 = C^4$ as shown in 2
- $G^2 = P^2 \cup P^2$ shown in 3

are not isomorphic but their GNN representations are the same.

```
G1 representation: [-2.7711809  2.5115702  2.4466345  2.0798717]
G2 representation: [-2.7711809  2.5115702  2.4466345  2.0798717]
```

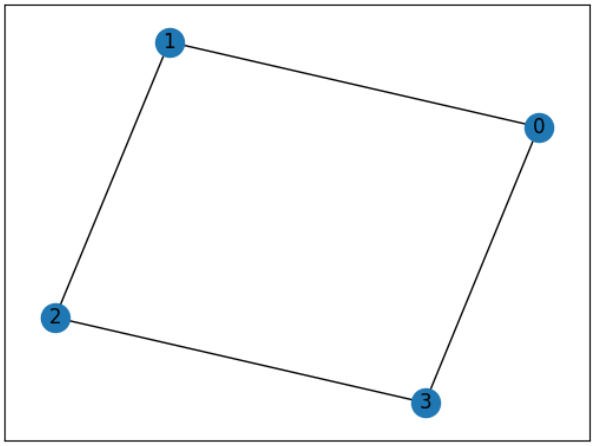

Figure 2: $G^1 = C^4$

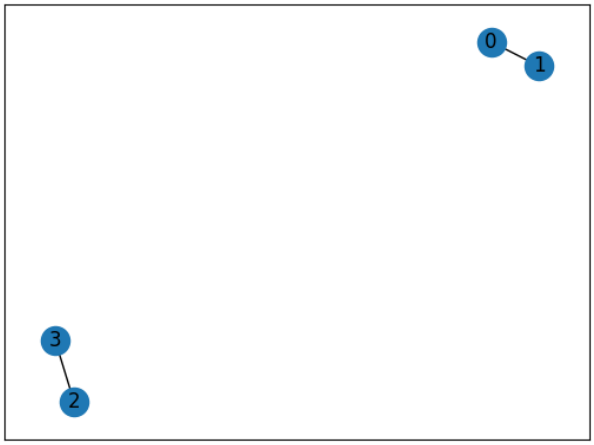

Figure 3: $G^2 = P^2 \cup P^2$

Example 1

- $G^1 = C^5 \cup C^7$ shown in 4
- $G^2 = C^{12}$ shown in 5

are not isomorphic but their GNN representation are the same.

```
G1 representation: [6.511315   1.9150434  0.20283365 3.483223  ]
G2 representation: [6.511315   1.9150434  0.20283365 3.483223  ]
```

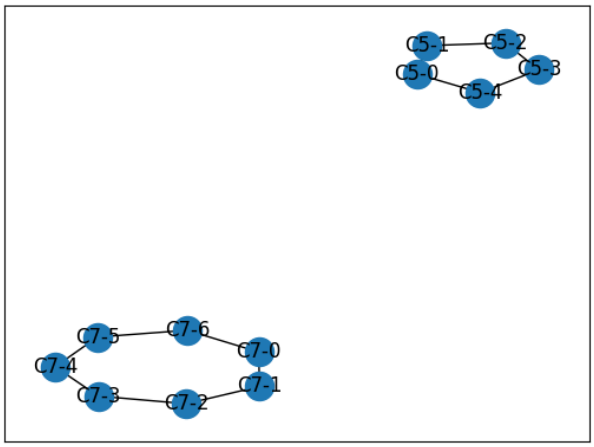

Figure 4: $G^1 = C^5 \cup C^7$

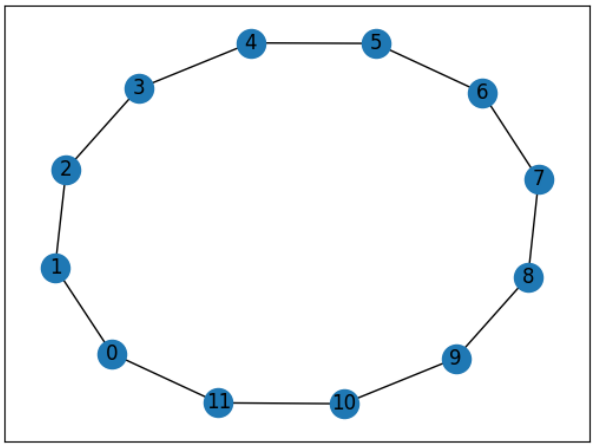

Figure 5: $G^2 = C^{12}$

Example 2

$G^1 = C^4 \cup C^4$ shown in 6 and $G^2 = C^8$ shown in 7 are not isomorphic but their GNN representation are the same.

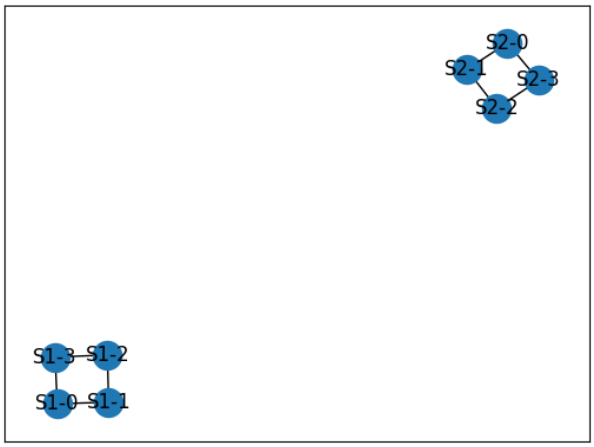

Figure 6: $G^1 = C^4 \cup C^4$

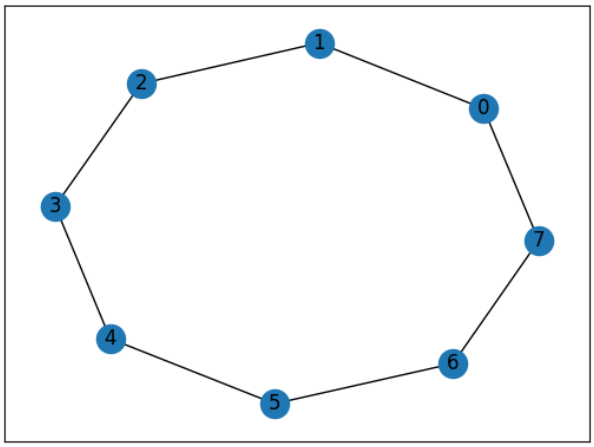

Figure 7: $G^2 = C^8$

```
G1 representation: [-0.3330682 -1.2836217  4.3689184  2.9682496]
G2 representation: [-0.3330682 -1.2836217  4.3689184  2.9682496]
These are the same although the graphs are not isomorphic
```