Graph Augmentation

Till now we have seen that we take raw graph structure to create computation graph but this is not necessory. We will discuss two different technique for computation graph creation:

① Graph feature augmentation

② Graph structure augmentation

It is highly unlikely that computational graph we create from raw input graph is optiment. This is because of various reason e.g:

① some time input graph lacks feature e.g only adjacency matrix as input. So at that Graph feature augmentation improves expressive power of input graph.

② ⓐ Some time graph structure can be too sparse so the message passing becomes inefficient. or

ⓑ graph structure is too dense so that message passing is too costly e.g some celebraty has millions of followers then message passing from all followers is very costly.

or

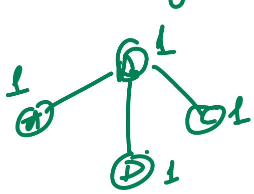
ⓒ graph structure too large so that computational graph can not fit into GPU.

To overcome all these issue we do augmentation.

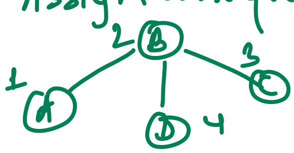
① Feature Augmentation

→ Input graph do not have node feature

→ solution 1: Assign constant value to node



→ Assign unique Id's to nodes. Convert unique Id into one hot encoding

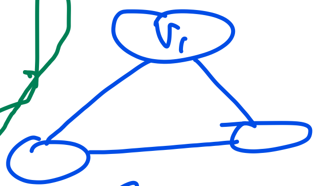


One-hot vector for Node ID-3 in 4-node graph.

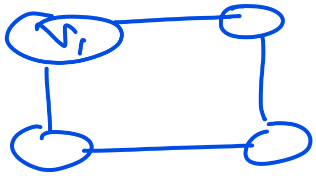
$= [0, 0, 1, 0]$, Note: Refer lec-8-GNN slide-10 for solution comparison.

→ Certain structure hard to learn by GNN

→ e.g: cycle count feature, suppose



G_1



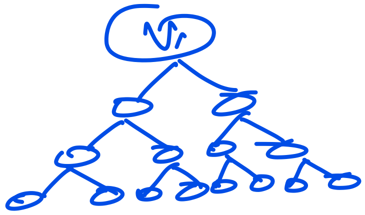
G_2

For an G_1 & G_2 can GNN count cycle length where v_1 node residing?

Ans: No.

Reason: For both computation graph will be same.

G_1



G_2

GNN will see that both graph reside in infinite cycle length graph.

This can be solved by use of cycle count as augmented feature e.g

we start from cycle count 0

[0,0,0,1,0,0] → For G_1

[0,0,0,0,1,0] → For G_2

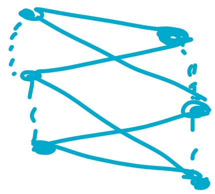
→ Other feature augmentation technique could be:

- → Node degree
- → clustering coefficient
- → Page Rank
- → Centrality etc.
- → Any other featureset used in Traditional featureset generation e.g graphlets

② Graph structure augmentation:

① when graph is too sparse

→ Add virtual edges: connect 2-hop neighbours by virtual edge. This can be done using $A + A^2$.
usecase: Bipartite graph e.g authors to papers.
2-hop neighbour virtual edge make projection graph. which will show author-author collaboration.



→ Add virtual nodes: In sparse graph suppose 2-node has shortest path length 10. So message passing will not be effective. If we add virtual node 'v' which connects to all node then all node will have distance 2 which improves the message passing.

→ when graph is too dense: In this taking message from all neighbours is too computational expensive and to reduce this at each layer we sample the $N(v)$ and take subset of this. This way it reduces computational overhead and helps in scalling for large networks.