COL870: Assignment 1

A. GNN Details

For this assignment, we implemented a custom GCN layer with add-pool as the aggregation mechanism. Further to implement the given embedding computation equation, we removed the normalization steps, and changed the update step to apply an MLP post message computation and aggregation. To incorporate the given loss equation in the assignment, we used the standard CrossEntropyLoss (https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html) from Pytorch.

Hyperparameters used:
Cora – gnn_layers (2), mlp_layers(1), hidden (32), learning rate (0.0001), dropout(0.2), epochs (2000)
Citeseer - gnn_layers (3), mlp_layers(1), hidden (32), learning rate (0.0001), dropout(0.5), epochs (2000)

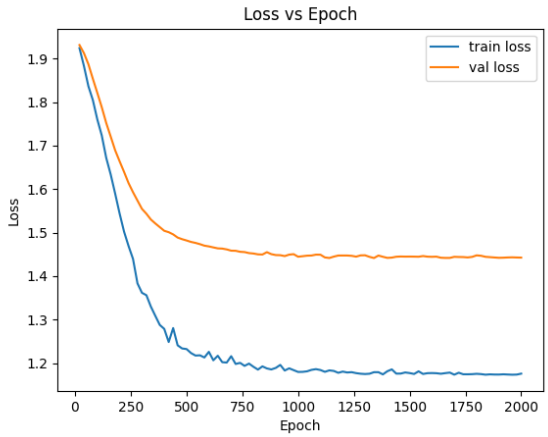
Results:

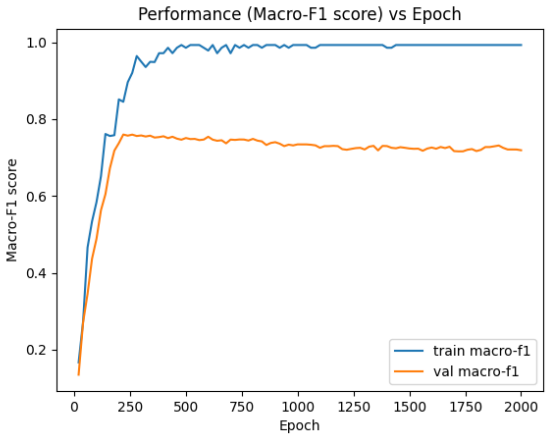
Dataset	Val (Macro-F1)	Test (Macro-F1)
Cora	76.47%	72.70%
Citeseer	55.70%	53.41%

Macro-F1 scores on train, validation, and test data

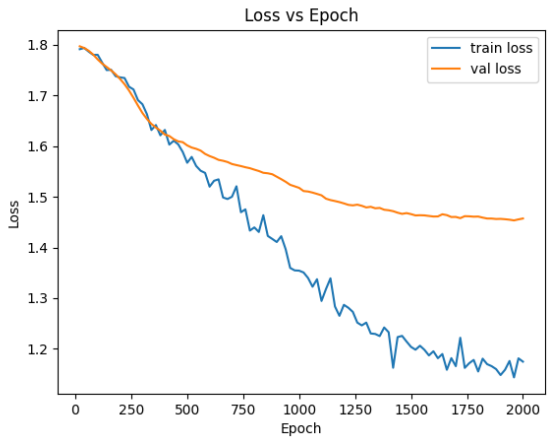
The plots of epochs vs loss/macro-F1 score for both the datasets are given below.

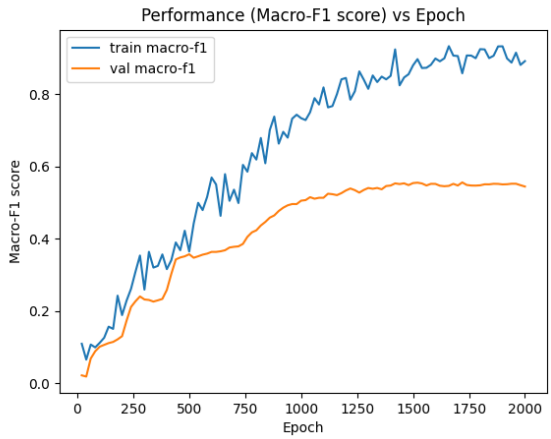
Cora





Citeseer





B. Impact of Topology

We conduct the experiments for studying the impact of topology on **Cora** dataset.

Experiment 1: GNNs aggregate the messages from the neighbors of a given target node to compute its embeddings, and then uses them to make predictions, and that is where topology gets incorporated along with the features. MLPs on the other hand only use features to generate embeddings and not topology to make predictions. So, we conduct an experiment where we present macro-F1 score on validation set, and test set using the model that had minimum validation loss for node classification when the model used is a GNN vs MLP.

Hyperparameters:

MLP – layers (2), hidden (32), learning rate (0.0001), epochs (3000)

GNN – gnn_layers (2), mlp_layers(1), hidden (32), learning rate (0.01), dropout(0.2), epochs (200)

Results:

Performance of MLP vs GNN on node classification for Cora dataset.

Model	Val (Macro-F1)	Test (Macro-F1)
MLP	53.52%	52.53%
GNN	74.79%	73.75%

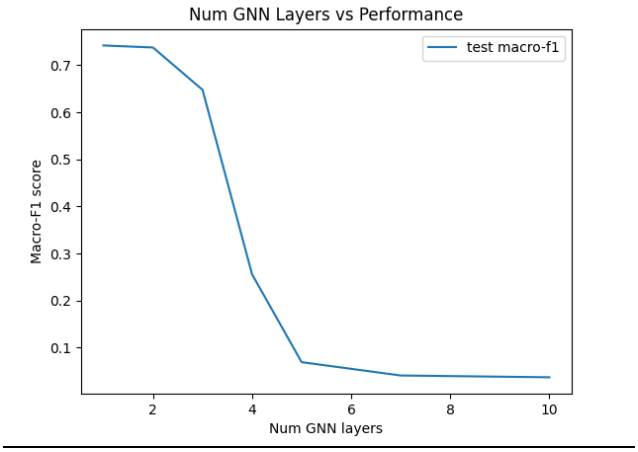
Observation: From the above table we observe that for graph data, incorporating topology along with features boosts performance.

Experiment 2: To assess, how much topology should be incorporated while aggregating messages from the neighbors, we vary the number of layers of the GNN and plot the number of layers vs test macro-F1 score. The number of GNN layers are directly proportional to the number of hops from which information is aggregated, thus it controls how much topology is being incorporated while making predictions for a given node on any given downstream task.

Hyperparameters:

GNN – gnn_layers (varying), mlp_layers(1), hidden (32), learning rate (0.01), dropout(0.2), epochs (200)

Results:



Observation: From the above plot observe that at first by increasing number of layers performance improves, for Cora we get best results at number of layers=2, then onwards the macro-f1 score keeps decreasing. This happens because of oversmoothing of the embeddings as we gather more and more information from far-away neighbors.

Conclusion: From the study above, we conclude that GNNs are effective for prediction tasks on graph data as they incorporate topology on top of existing features. Further, we see that we must choose the hyperparameter, number of GNN layers wisely to avoid oversmoothing of embeddings. Mostly, this parameter is dataset dependent, i.e., it is likely to depend on the diameter or the homophily level of the network being studied.