Lab 06

Question 1

After running the above code, you will observe the following:

- Graph Density: As NNN increases, the graph density tends to decrease because the number of possible edges grows quadratically, while the number of edges grows linearly (with the probability fixed at 0.5).
- Degree Distribution: The degree distribution tends to become more uniform as NNN increases**.**

Question 2

Supervised Learning

In supervised learning, the model is trained on a labeled dataset, meaning each input is paired with a corresponding correct output (label). The goal is for the model to learn the mapping between inputs and outputs so that it can predict labels for new, unseen data.

Self-Supervised Learning

Self-supervised learning is a type of unsupervised learning where the model learns to predict parts of the data from other parts. The training labels are automatically generated from the data itself without requiring human annotation. The model creates its own "pseudo-labels" by solving a pretext task.

Semi-Supervised Learning

Semi-supervised learning is a method where the model is trained using a small amount of labeled data and a large amount of unlabeled data. The idea is to leverage the limited labeled data to guide the learning from the much larger unlabeled dataset, improving model performance while reducing labeling costs.

Inductive Learning

Inductive learning involves building a general model or function from a set of training data that can be used to make predictions on unseen data. The goal is to learn patterns from the training set and apply the learned model to new, unseen data.

Transductive Learning

Transductive learning is a type of machine learning that makes specific predictions for given test examples, rather than trying to learn a general model. The key difference is that transductive learning assumes the test data is available during training and the model focuses on making the best predictions for that specific test data.

Question 3

Increase the number of epochs from 50 to 500 and observe the change in validation accuracy and write what you observe in the word file.

Epoch 50

Training Accuracy – 100%

Validation Accuracy – 64.71%

Loss – 0.7593

Epoch 500

Training Accuracy – 100%

Validation Accuracy – 82.35%

Loss – 0.0181

When I changed the epoch from 50 to 500, Validation accuracy improved, Also the loss is decreased.

Experiment without self-loops added to GCNConv() layers in the GCN() model and detail the model accuracy increase/decrease in the word file.

Epoch 500

Loss – 0.0179

Training Accuracy – 100%

Validation Accuracy – 73.53%

When I run without self-loops added to GCNConv() layers in the GCN() model with 500, Validation accuracy decreased, Also the loss is decreased.

Analysis

Adding self-loops to the GCNConv layers appears to enhance the model's ability to generalize, as evidenced by the higher validation accuracy. While the loss is slightly lower without self-loops, the trade-off is beneficial for improved validation performance with self-loops.

More layers and skip connections did not improve generalization and led to overfitting. Fewer layers and potential regularization might offer better performance.

Increase the number of GCNConv() layers in the GCN() model upto 8 layers from original 3 layers. Detail the accuracy increase/decrease in the word file.

1. ***In_channels and out_channels in GCNConv() can be considered as hyper-parameters and you can use the best performing values you find.***
2. ***Add skip connections between some of the GCNConv() layers and try to see if that can improve the model performance.***

Epoch 500

Loss – 0.0004

Training Accuracy 100%

Validation Accuracy – 58.82%

Analysis: Increasing the layers to 8 and adding skip connections resulted in perfect training accuracy but a significant drop in validation accuracy and a very low loss. This suggests overfitting, where the model performs well on training data but poorly on validation data.

Part 4

Message Passing GNN (MP-GNN)

Message Passing GNN is a general framework for graph neural networks where nodes communicate information (messages) with their neighbors. This process iteratively updates the node representations by aggregating information from neighboring nodes.

Graph Convolutional Network (GCN):

GCN is a specific type of graph neural network that generalizes convolutional neural networks (CNNs) to graph structures. It aggregates the feature information from a node's neighbors using weighted sums.

Graph Attention Network (GAT):

GAT introduces attention mechanisms into graph neural networks, allowing nodes to assign different importance (attention) to their neighbors during the feature aggregation step.

GraphSAGE (Graph Sample and Aggregation):

GraphSAGE is a framework for scalable graph neural networks that generates embeddings by sampling a fixed number of neighbors for each node and aggregating information from those sampled neighbors.

Aspect	Message Passing GNN	GCN	GAT	GraphSAGE
Core Idea	Nodes exchange messages with neighbors	Fixed-weight aggregation of neighbors	Attention-weighted aggregation of neighbors	Sampled neighbors and aggregators
Aggregation	General framework, varies across models	Fixed weights (neighbors treated equally)	Attention mechanism for neighbors	Sampling a subset of neighbors
Strengths	Flexible, used in many GNNs	Simplicity, computational efficiency	Flexible, learns importance dynamically	Scalable, handles large graphs
Weaknesses	Can be slow on large graphs	Cannot differentiate between important/unimportant neighbors	Higher computational cost	May depend on sampling strategy