**Github Link - https://github.com/sudhais/DL\_Lab06**

Q1. As the number of nodes N increases, the potential number of edges grows rapidly, but if the actual number of edges doesn't grow as fast, the graph becomes sparser, and the density value decreases as the number of nodes increases

Q2.

1. Explain the differences between supervised learning, self-supervised learning and semi-supervised learning methods

**Supervised Learning**:

In supervised learning, the model is trained on a labeled dataset where each input is paired with a known output (label). The goal is to map the input to the correct label.

Requires a large amount of labeled data

The model learns node embeddings using all labeled nodes to predict node labels.

**Self-Supervised Learning**:

In self-supervised learning, the model generates pseudo-labels from the data itself without needing explicit labels. The learning process often involves predicting part of the data from other parts, such as filling in missing words in a sentence or predicting the next frame in a video.

Uses unlabeled data and defines auxiliary tasks (pretext tasks) for representation learning.

The model might learn node embeddings by reconstructing graph structure or predicting node features without actual node labels.

**Semi-Supervised Learning**:  
 In semi-supervised learning, the model is trained with a small amount of labeled data and a large amount of unlabeled data. The labeled data guides the model, while the unlabeled data helps improve the model's performance by discovering patterns or structure in the data.

Requires both labeled and unlabeled data.

semi-supervised GCN uses a small number of labeled nodes in a graph and leverages the graph structure to infer labels for the remaining nodes.

02.

|  |  |  |
| --- | --- | --- |
| Aspect | Transductive Learning | Inductive Learning |
| Goal | Make predictions for specific test data | |  | | --- | |  |  |  | | --- | | Generalize to unseen data | |
| Test Data Available | Available during training | Not available during training |
| Learning Objective | Focused on specific examples | Focused on generalization |
| Use of Unlabeled Data | Often uses unlabeled data for improved accuracy | Does not typically use unlabeled data directly |
| Application | GCNs in node classification (KarateClub) | General machine learning tasks like classification or regression |

03.

1. 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.

**Validation Accuracy**: a gradual stabilization of validation accuracy as the number of epochs increases. Initially, there might be an increase in accuracy, but after a certain point, accuracy may plateau or even drop if the model starts overfitting.

**Training Loss**: The loss function should decrease steadily, and accuracy for training nodes should improve as the model sees more data over time.

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

**Accuracy**: I notice a drop in both training and validation accuracy as the model can no longer aggregate information from a node's own features.

**Loss**: The loss may increase slightly, indicating that removing self-loops reduces the model's effectiveness at capturing node-specific information.

1. 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.

 Accuracy: Deeper models could potentially overfit the data, especially when the number of parameters increases significantly. You might notice an initial accuracy increase, but later epochs might show reduced accuracy due to overfitting.

 Loss: The model may require more epochs to stabilize, and the loss might oscillate before convergence.

* + 1. Add skip connections between some of the GCNConv() layers and try to see if that can improve the model performance.

 Accuracy: Skip connections often help improve accuracy, especially in deeper architectures, by preventing the network from forgetting useful information from earlier layers.

 Loss: You may notice a faster convergence of the loss, as gradients flow more smoothly through the network with skip connections.