DL lab 6 – Graph Neural Networks

1. Upload the NetworkX jupyter notebook file (i.e.,NetworkX\_tutorial.ipynb) to google colab root directory.
   * Run the above code and understand it.
   * Complete the code sections to get the degree matrix and Laplacian matrix of the created random graph.
   * Calculate the graph density of the random graph in the code. Use the below equation (D = graph density, |V| = number of nodes and |E| = number of edges).
   * Increase the N value from 20 (original value) to 200 with multiple N values in between and observe the change of graph density and degree distribution (i.e., histogram plot). Explain what you observe and write the answer in a word file.



1. In the KarateClub dataset based GCN code, we use semi-supervised training approach along with the transductive leaning method.
   * Explain the differences between supervised learning, self-supervised learning and semi-supervised learning methods

**Supervised Learning**: In this method, the model learns from data that has both inputs and correct outputs. The goal is to predict the correct output when given new inputs.

**Self-Supervised Learning**: This is a type of learning where the model creates its own labels from the data. There are no provided labels, but the model finds patterns and learns from them.

**Semi-Supervised Learning**: This method uses both labeled and unlabeled data. A small amount of labeled data helps guide the model, while the rest of the data is u

* + Explain the differences between transductive learning and inductive learning.

**Transductive Learning:** The model looks at both the training data and the test data it will predict on. It doesn’t try to learn patterns for new data beyond what it has seen.

**Inductive Learning:** The model learns from training data and uses that learning to predict on new, unseen data in the future. It tries to understand general patterns.

1. Upload the KarateClub dataset based GCN jupyter notebook file (i.e., KarateClub\_GCN\_introduction.ipynb ) to google colab root directory.
   * In this code, we use Zachary’s karate club network dataset.
   * Run the above code and understand it.
   * 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.
   * Experiment without self-loops added to GCNConv() layers in the GCN() model and detail the model accuracy increase/decrease in the word file.
   * 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.
     3. Detail what you observe in the word file.
2. Explain the differences between Message Passing GNN, graph convolution network (GCN), graph attention network (GAT) and GraphSAGE. Write the answers in the word file.

**Message Passing GNN**

Message Passing Graph Neural Networks (GNNs) exchange information between nodes to update their representations. Nodes gather messages from neighbors over multiple rounds, allowing them to capture complex relationships in the graph.

**Graph Convolution Network (GCN)**

GCNs are a type of message passing network that uses convolution operations on graphs. They aggregate information from neighboring nodes along with the node's own features, efficiently learning the graph's structure with a fixed number of layers.

**Graph Attention Network (GAT)**

GATs incorporate an attention mechanism, allowing nodes to weigh the importance of each neighbor differently during aggregation. This enables GATs to focus on the most relevant parts of the graph, making them suitable for complex structures.

**GraphSAGE**

GraphSAGE generates node embeddings by sampling a fixed-size neighborhood for each node and aggregating their features. This method supports different aggregation functions (like mean or pooling) and allows for scalable training using a subset of neighbors.

**Submission.**

Download the final modified notebook files (all 2 jupyter notebooks). Add these notebooks and the word file to a new zip file. Upload this zip file to the courseweb submission link. The file name should be your registration number.