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.



**Observation**

Graph Density has been decreased.

Degree distribution has become more varied.

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**:

* Uses **labeled data** (input-output pairs).
* The model learns from these labels to predict outputs for new inputs.

**Self-Supervised Learning**:

* No labeled data is needed.
* The model generates its own labels from the data (e.g., predicting part of the input from another part).

**Semi-Supervised Learning**:

* Uses a **small amount of labeled data** and a **large amount of unlabeled data**.
* The model learns from both to improve accuracy.
  + Explain the differences between transductive learning and inductive learning.

**Transductive Learning**:

* Learns to **predict specific outputs** only for the given training and test data (doesn't generalize beyond this data).

**Inductive Learning**:

* Learns a **general rule** from the training data that can be applied to **unseen data** in the future.

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.

When the number of epochs is 500, the loss has been decreased and the validation accuracy has been increased.

A screen shot of a graph

Description automatically generatedWhen epochs are 50

A screen shot of a graph

Description automatically generatedWhen epochs are 500

A graph showing a number of people

Description automatically generated with medium confidence

* + A close-up of a number

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

Accuracy has been decreasing.

* + 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.

Accuracy has been increased.

1. 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 GNNs allow nodes to update their features by exchanging information with neighbors iteratively. Graph Convolution Networks (GCNs) aggregate features from neighbors using a fixed weight matrix and averaging, focusing on local graph structure. Graph Attention Networks (GATs) enhance this by applying attention mechanisms to weigh the importance of neighbors' features dynamically. GraphSAGE improves scalability by sampling a fixed-size neighborhood and using various aggregation functions (mean, LSTM, pooling), making it suitable for large, inductive learning tasks. Each model has unique methods for feature aggregation and updating, affecting their performance and scalability.

**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.