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.



**Answer**:

Density decreases as N increases, because the number of possible edges grows faster than the number of actual edges grows faster than the number of actual edges (E = 2N keeps proportional to nodes, not quadratic).

Degree distribution stays narrow (most nodes have low degrees), but as N grows, the histogram becomes smoother and closer to a Poisson, like distribution.

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

**Answer:**

|  |  |  |
| --- | --- | --- |
| **Supervised Learning** | **Self – Supervised Learning** | **Sem- Supervised Learning** |
| Uses labeled data where input, output pairs(x,y) are provided for training. | A form of unsupervised learning where the model creates supervisory signals from the inputs itself by solving pretext tasks. | Combines a small, labeled dataset with a larger unlabeled dataset. |
| The model directly learns a mapping from inputs to labels optimizing a supervised loss. | Labels are generated automatically, enabling representation learning without manual annotation. | The model leverages both labeled and unlabeled data to improve performance when labeled data is scarce. |
| Performance depends heavily on quantity and quality of labeled data. |  |  |

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

**Answer**:

|  |  |
| --- | --- |
| **Transductive Learning** | **Inductive Learning** |
| The model is trained and evaluated on the same graph. | The model is trained on one set of graphs and must generalize to unseen nodes or completely new graphs at test time. |
| The model can use the entire graph structure (including nodes that will be evaluated) during training, but only a subset of node label is provided as supervision. | The model cannot rely on seeing the test nodes/ graph during training. |

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.

**Answer:**

a) Increasing the number of epochs from 50 to 500 improved validation accuracy. At 50 epochs, the model often underfits. With 500 epochs, accuracy rises and then plateaus.

b) Removing self-loops from GCNConv decreased accuracy. Self-loops are important because they allow nodes to use their own features during propagation. Without them, the model loses useful information.

c) Increasing the number of GCNConv layers from 3 to 8 showed the oversmoothing problem. As layers increase, node features become too similar, and accuracy drops. However, adding skip connections between some layers helped recover part of the lost accuracy.

d) Treating in\_channels and out\_channels as hyperparameters showed that hidden sizes in the range of 16 to 64 gave good performance. Larger hidden sizes did not bring significant improvements for this small dataset.

Overall, more epochs improved accuracy, self-loops were essential, too many layers reduced accuracy unless skip connections were added, and careful tuning of hidden sizes helped.

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.

**Answer:**

• Message Passing Neural Networks (MPNN): A general framework where nodes exchange messages with neighbors and update their states. Many GNNs fall under this category.

• Graph Convolutional Network (GCN): A special case of MPNN that uses normalized averaging of neighbor features (spectral motivation). It is simple and efficient but suffers from over smoothing when too many layers are used.

• Graph Attention Network (GAT): Extends GCN by learning attention weights for each edge, so that neighbors contribute unequally. This improves flexibility but increases computation.

• GraphSAGE: Focuses on inductive learning. It samples neighbors and applies learnable aggregators (mean, max, LSTM) so the model can generalize to unseen nodes and large graphs.

**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.