DL lab 6 – Graph Neural Networks

GitHub repo link

IT21268076/DL-Lab-exercises (github.com)

1. Change in Validation Accuracy with More Epochs

- **Experiment**: The number of training epochs was increased from 50 to 500 to evaluate the effect of prolonged training on the model's validation accuracy.
- **Observation**: Validation accuracy showed continuous improvement as the epochs increased from 50 to about 200. However, beyond 400 epochs, no further enhancement was noticed, and some minor overfitting became apparent.

2. Impact of Removing Self-Loops in GCNConv Layers

- **Experiment**: Self-loops were removed from the GCNConv() layers (by setting add_self_loops=False) to assess their impact on the model's accuracy.
- **Observation**: The validation accuracy decreased when self-loops were removed. This indicates that self-loops are essential for nodes to aggregate their own features, and without them, the model's ability to learn node-specific characteristics was diminished.

3. Effect of Increasing GCNConv Layers from 3 to 8

- **Experiment**: The number of GCNConv() layers was increased from 3 to 8 to analyze how deeper architectures affect performance.
- **Observation**: The model's accuracy improved slightly when the number of layers was increased to 5. However, increasing it to 8 led to a drop in accuracy.

4. Tuning In_channels and Out_channels in GCNConv Layers

- **Experiment**: Various values of the in_channels and out_channels hyperparameters in the GCNConv() layers were tested to determine their effect on model performance.
- **Observation**: Increasing the number of channels to higher values (e.g., 64 and 128) resulted in a slight performance decrease, likely due to overfitting or excessive complexity for the dataset. Optimal channel sizes are necessary for maximizing performance.

5. Effect of Adding Skip Connections Between GCNConv Layers

- **Experiment**: Skip connections were added between certain GCNConv() layers to observe their effect on the model.
- **Observation**: Skip connections helped the model mitigate vanishing gradients by facilitating the flow of gradients through the network. This improved performance by allowing the model to use both shallow and deep features effectively.

Differences Between Message Passing GNN, GCN, GAT, and GraphSAGE

I. Message Passing Neural Networks (MPNN):

- **Overview**: MPNNs represent a flexible framework for graph neural networks where information is exchanged between nodes through a two-step process involving message aggregation and update.
- **Key Feature**: MPNNs allow customizable aggregation and update functions, making them a generalization of various GNN models.
- **Use Case**: Suitable for tasks that require tailored aggregation strategies.

II. Graph Convolutional Network (GCN):

- **Overview**: GCN is a foundational model for graph neural networks that generalizes the convolution operation to graphs, using spectral methods to aggregate information from neighboring nodes.
- **Key Feature**: It utilizes normalized adjacency matrices to propagate node features and capture local neighborhood information.
- **Use Case**: Commonly used in semi-supervised learning, node classification, and link prediction.

III. Graph Attention Network (GAT):

- **Overview**: GAT builds on GCN by incorporating attention mechanisms, allowing the model to assign varying levels of importance to neighboring nodes during the aggregation process.
- **Key Feature**: Uses attention weights to focus on the most relevant neighboring nodes.
- **Use Case**: Effective in scenarios where the importance of neighboring nodes varies, such as social networks and recommendation systems.

IV. GraphSAGE:

- **Overview**: GraphSAGE is a scalable model that samples and aggregates information from a fixed-size local neighborhood of each node, enabling it to handle large graphs efficiently.
- **Key Feature**: It employs inductive learning, meaning it can generate node embeddings for unseen nodes by learning an aggregation function.
- **Use Case**: Well-suited for large-scale graph learning tasks and generalizing to new nodes.