Complex Network Theory (Assignment – 2)

Report

Part1: Node2Vec and Logistic Regression

	Train	Test
Precision	0.87	0.82
Recall	0.85	0.79
F1-Score	0.86	0.80
Accuracy	0.87	0.78

Logistic Regression performs well on Node2Vec embeddings but shows an overfitting tendency.

Part2: Graph Convolutional Networks (GCN)

	Train	Test
Precision	0.92	0.92
Recall	0.89	0.90
F1-Score	0.90	0.91
Accuracy	0.91	0.91

Implementation Approach

- **Data Preparation**

The CORA dataset consists of scientific publications classified into one of seven classes. The dataset is represented as a graph where nodes are documents, and edges represent citation links. The code begins by loading and preprocessing the dataset:

  - ***Feature and Label Encoding:*** The load_cora_content function loads the document features and encodes the class labels as integers. Features are normalized row-wise to ensure numerical stability and effective training.
  - ***Adjacency Matrix Construction:*** The load_cora_cites function constructs the adjacency matrix representing the citation network. It adds self-loops to ensure every node can pass its own features forward, following standard practice in GCN implementations.

- **Model Definition**

The GCN model comprises two main components:

  - ***GCN Layer:*** Defined in the GCNLayer class, it performs the core operation of graph convolution, which involves multiplying the input features by a weight matrix

(learnable parameters), followed by multiplication with the normalized adjacency matrix to aggregate neighbor features.
  - ***GCN Model:*** The GCN class implements a two-layer GCN model. It applies a ReLU activation function after the first layer and includes a dropout layer for regularization. The output of the second GCN layer is passed through a log-softmax function to generate predictions for each class.

- **Normalization Techniques**

Two normalization techniques are applied:

  - ***Feature Matrix Normalization:*** Ensures that each feature has equal importance during the initial feature aggregation.
  - ***Adjacency Matrix Normalization:*** Facilitates the smooth flow of gradients during backpropagation by ensuring that the magnitude of the feature vectors does not explode.

- **Training Process**
  - The train function orchestrates the model's training process over a specified number of epochs. For each epoch, it performs the following steps:
  - Generate predictions by passing the features and the adjacency matrix through the model.
  - Calculate the loss using negative log-likelihood (NLL) loss, comparing the predictions to the true labels of the training set.
  - Perform backpropagation and adjust the model weights using the Adam optimizer.
  - Print the loss and accuracy at regular intervals to monitor the training progress.

- **Evaluation Results**

After training, the evaluate function assesses the model's performance on both the training and test sets. It computes the following metrics:

  - ***Accuracy:*** The proportion of correctly predicted labels to the total number of predictions.
  - ***Precision, Recall, and F1-Score:*** These metrics provide a more detailed insight into the model's performance, especially in cases of class imbalance. Precision measures the model's accuracy in identifying positive instances, recall measures the ability to find all positive instances, and F1-score provides a balance between precision and recall.

Insights

The comparison between Node2Vec with Logistic Regression and Graph Convolutional Networks (GCN) reveals key insights:

- **Node2Vec and Logistic Regression:** Shows good performance on training data but experiences a notable performance drop on test data (Precision: 0.87 to 0.82, Recall:

0.85 to 0.79, F1-Score: 0.86 to 0.80, Accuracy: 0.87 to 0.78), suggesting some overfitting but still a decent ability to generalize.

- **Graph Convolutional Networks (GCN):** Exhibits excellent and consistent performance across both training and test data (Precision: 0.92, Recall: 0.89 to 0.90, F1-Score: 0.90 to 0.91, Accuracy: 0.91), indicating a strong ability to learn and generalize well without overfitting.

<u>**Overall Insights**</u>

- **Model Suitability:** The GCN model appears to be more suited for the task at hand, given its higher and more consistent performance metrics. This could be attributed to GCN's ability to directly leverage the graph structure in its learning process, whereas Node2Vec requires a preprocessing step to generate node embeddings, which are then used in a traditional machine learning model like Logistic Regression.
- **Generalization Capability:** GCN shows a stronger capability to generalize from training data to unseen test data without a significant drop in performance, which is an essential characteristic of a robust model.