Homework#4: Citation Prediction with Graph Convolution Neural Networks (GCNs) using PyTorch

Objective: The goal of this assignment is to practice implementing Graph Neural Networks (GNNs) using PyTorch for the task of link prediction. You will specifically predict citations between research papers in the Cora dataset. [Due: 05/28/2025, 11:59PM]

Dataset: The provided dataset contains research papers and citation information. Each node represents a paper, and each edge represents a citation between two papers. Node features are also included. Please download the dataset from iCampus.

Problem Description: You will develop a Graph Neural Network (GNN) model using PyTorch to predict potential citations (links) between papers in the Cora dataset.

Tasks:

1. **Installation of Libraries:**
   Install the following essential libraries using PyTorch, Pandas, NetworkX, Matplotlib, and scikit-learn. Usage of libraries other than those listed above is not allowed.
2. **Data Loading and Preprocessing:**
   Load and preprocess the Cora dataset to obtain adjacency matrix and node features matrix.
3. **Graph Construction:**
   Convert the dataset into an undirected graph using NetworkX. Clearly represent papers as nodes, citations as edges, and node features as node attributes.
4. **Implementation of the Graph Convolutional Layer (GraphConv):**
   Implement a custom PyTorch layer named GraphConv to perform the graph convolution operation. The layer should accept node features and the adjacency matrix as inputs and perform a linear transformation followed by a ReLU activation function.
5. **Building the GCN Model:**
   Implement a GCN consisting of two *GraphConv* layers followed by a linear layer for edge classification.
6. Hyperparameters:
   - Hidden dimension (both GCN layers): 16
   - Dropout rate: 0.1
   - Learning rate: 0.01
   - Weight decay: 4e-5
   - Batch size: 256
   - Random seed: 42 (To ensure reproducibility by setting random seed (random_state=42 or np.random.seed(42)) explicitly when creating your training (70%), validation (10%), and test (20%) splits.)
   - Epochs: 10
   - Negative edges (Q): 2
   - Threshold for the positive link prediction: 0.5

7. **Loss function:** In order to learn predictive representations in a fully unsupervised settings, we apply a graph-based loss function to the final output representation, $\{z_u, \forall u \in V\}$, where V is the set of nodes. The graph-based loss function encourages nearby nodes to have similar representations, while enforcing that the representations of disparate nodes are highly distinct. The loss for a node *u* in a batch is defined as:
$$J_{\mathcal{G}}(\mathbf{z}_u) = -\log\left(\sigma(\mathbf{z}_u^\top \mathbf{z}_v)\right) - Q \cdot \mathbb{E}_{v_n \sim P_n(v)} \log\left(\sigma(-\mathbf{z}_u^\top \mathbf{z}_{v_n})\right),$$
where *v* is a node that co-occurs on the positive edges in a training dataset, σ is the sigmoid function. $P_n(v)$ is a negative sampling distribution and here you can uniformly sample *Q* number of nodes from all non-existing (i.e., negative) edges. *Q* defines the number of negative samples.
8. **Data Splitting and Early Stopping:** Randomly split edges into training (70%), validation (10%), and testing (20%) sets with fixed random seed (*random_state=42*). Use validation AUC for early stopping. If validation AUC doesn't improve for 10 consecutive epochs, stop training to avoid overfitting.
9. **Evaluate the model:** Report the Area Under ROC Curve (AUC) clearly for validation and test datasets. The ROC curve plots True Positive Rate vs False Positive Rate at various thresholds. Higher AUC indicates better model capability to distinguish between positive (existing citations) and negative edges (non-existent citations). Refer to https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc for further understanding.
10. **Visualize the results:** Use NetworkX to visualize the original graph and the predicted papers after training the GNN model. Please use blue for correct predictions and red for incorrect predictions for the edges in the visualization. Discuss about how your algorithm performs (e.g., what are missing and why?).
11. Change hyperparameters to maximize your AUC score using the training/validation split. The final AUC score should be computed from the test split and compare with the AUC of Task 9.

Please submit your code as a Jupyter Notebook or a Python script along with a report describing your approach, the performance of the model, and any interesting observations you made during the process. **Report should describe the details and design criteria about the Task 5-11.**

Grading
Your submission will be graded based on the following criteria:
1. Correctness and completeness of the implementation of the GNN algorithm.
2. Clarity and quality of the report.

Note
You are free to use any external resources or libraries that you think might be helpful for this assignment, as long as you cite them properly in your report and do not violate any academic integrity policies. However, you should not use any pre-existing implementations or AI-based generations. We note that we run AI-based plagiarism checker.