Assignment 4. Graph Neural Networks

DATA304: Big Data Analysis

Due Date: November 24, 2025

General Instructions

The programming assignments in this course are designed to foster your ability to implement and apply the concepts introduced during class. All datasets and resources for the assignments are provided at the following link: **Link**. You do not need to download new data for each assignment. Instead, simply place the newly provided Jupyter notebook files for each assignment into your existing `assignment_release` directory (or into the renamed directory, if you have changed it).

- **How to submit:**
  - You need to submit your code via LMS for all assignments. For each assignment, submit **All** the released `.ipynb` files, including all modifications you have made.
  - Assignments #1, #2, and #3 also require Kaggle submissions.
  - Assignments #4 and #5 do not require Kaggle submissions.

  Make sure you submit both LMS and Kaggle on time to receive the credit. All submissions are due by midnight (23:59 KST) on the deadline day.

- **How the assignments are graded:**
  - For Assignments #1, #2, and #3, you will receive full credit as long as you outperform our baseline code.
  - For Assignments #4 and #5, you will receive full credit as long as you submit your code on time.

  This policy is intended to encourage you to solve the problem by yourself and to compare your results with others. Use the Q&A board on LMS for all assignment-related questions.

Assignment Instruction

This assignment has two notebooks: `P5a.intro-to-GNN.ipynb` and `P5b.category-classification-with-GNN.ipynb`.

P5a: Introduction to GNNs

We will learn about Graph Neural Networks (GNNs) through a hands-on exercise using the Cora citation network dataset. Our goal is to develop an intuitive understanding of how GNNs work on graph-structured data and to implement a simple Graph Convolutional Network (GCN) for node classification. We will compare two approaches:

1. [**Part A**]: MLP (multi-layer perceptron) classifier
2. [**Part B**]: GNN (Graph Neural Network) classifier

⌨ **Your Task:** Carefully read and understand the provided code. Examine intermediate values by printing them out, and make sure you understand what operations are being performed at each step and how they fit together in the overall process. No submission required.

P5b: Product category classification with GNNs

This notebook builds on the previous assignment `P2c: train-with-BERT-embedding-limited-label`. The setup remains similar, but here we take a step further by incorporating label structure information into the model.

1. [**Part A**]: Revisit Classification with label embedding
2. [**Part B**]: Label GCN and GCN-Enhanced Classifier

You will implement two components in this assignment.

1. Label GCN You will implement a Graph Convolutional Network to refine label embeddings by propagating information over the label graph. This process allows semantically related labels to share contextual information, leading to more expressive and robust label representations.

The model takes as input two components: (1) the initial label embeddings $E \in \mathbb{R}^{|C| \times d}$, where $|C|$ is the total number of labels, and (2) the normalized adjacency matrix $\hat{A} \in \mathbb{R}^{|C| \times |C|}$, which encodes the connections between related labels in the hierarchy.

At each layer, the model performs message passing and transformation steps, propagating information across the label graph and producing refined label embeddings E'.

2. GCN-Enhanced Classifier The goal of this model is to classify each document by comparing its embedding with graph-enriched label embeddings produced by the LabelGCN. The process consists of three main steps. First, input document embeddings are projected into the same space as label embeddings. Next, the label embeddings are refined using the GCN described above. Finally, classification logits are computed by taking the inner product between the projected document embeddings and the refined label embeddings:

$$\text{logits}(x) = \text{Proj}(x) \cdot (E')^T.$$

⌨ **Your Task:** Start from the provided skeleton code and fill in all missing parts marked with `# TODO`. No Kaggle submission required this time.

Submission Guidelines

LMS Submission

- Submit your source code (all `.ipynb` files) to the LMS.
- Make sure the submitted code can be executed without errors.