Assignment 3 - Kaggle Part
Graph Node Classification

AIST 4010: Foundation of Applied Deep Learning (Spring 2022)

DUE: **11:59PM (HKT), Apr. 12, 2022**

1 Introduction

A graph is a data structure consisting of two components: nodes (vertices) and edges. A graph G can be defined as $G = (V, E)$, where V is the set of nodes, and E is the set of edges between nodes. If there are directional dependencies between nodes, then edges are directed. Otherwise, edges are undirected. A graph can represent many things like social media networks, chemical molecules, financial transactions, etc.

Graph Neural Networks (GNNs) are a class of deep learning methods designed to perform inference on data described by graphs. GNNs can be directly applied to graphs, and provide an easy way to do node-level, edge-level, and graph-level prediction tasks. Usually, the problems that GNNs resolve includes: Node Classification, Graph Classification, Link prediction, Graph clustering, etc.

In this assignment, you will use basic Graph Convolutional Networks (GCNs) or some advanced GNNs to do **node classification**. In this task, your system will be given a whole graph including nodes with their features, edges between nodes, and labels of some nodes as input. You will have all knowledge for the graph structure while you only get part of label information for graph nodes. Here we have 7 classes, and we simply labeled them to be **'Class_0', 'Class_1', 'Class_2', 'Class_3', 'Class_4', 'Class_5', 'Class_6'**. The system should decide which class the rest nodes should belong to.

You will do your task on one graph dataset containing some labeled nodes to classify other unlabeled nodes. This is called semi-supervised learning. You will learn about graph processing, graph representation, and, of course, graph convolutional networks. The purpose of this assignment is to help you understand how we deal with graph data.

- Goal: Given a graph, and labels of some nodes, classify other nodes.
- Kaggle: `https://www.kaggle.com/c/aist4010-spring2022-a3`

2 Node Classification

2.1 Graph Encoding

This time we will meet a new data format, graph. We know graph has vertices and edges. Usually, we can encode the graph by **adjacency matrix** representing the structural information. For undirected graph, the matrix should be symmetric. Besides, every vertex (node) may have some features, and we can represent these features by feature vectors. And the edges may have some weight.

For this task, the graph is an undirected graph. Every node has its feature (already pre-processed). And edges don't have weight.

2.2 Semi-supervised learning

In this graph, some nodes have labels while the other don't. We need to classify those unlabeled nodes (test set) by information from these labeled nodes (train/val set), as well as information from the graph structure. This is an example of semi-supervised learning.

We know that the class of certain node should be determined by its features and its neighbors. And you will finally find that even if we only know a small amount of node labels (train set), we can still have good predicted results on the test set.

2.3 Multi-class Classification

In fact, this task is still a multi-class classification: the input to your system is the graph structure, node features, then your model needs to predict the classes for some nodes.

The task could be divided to three parts:

- Loading graph data and labels from raw files.
- Encoding this graph appropriately.
- Training a deep learning model for classification.

Here is a naive way to do this task:

First, build the adjacency matrix and load the node features. Concatenate the corresponding row in the adjacency matrix of one node and its features together as one input, then you can directly apply MLP on it to train a classification model. And in this way, you can easily beat the MLP baseline.

3 Dataset

The data for the assignment can be downloaded from the Kaggle competition link[1]. The data set includes one file describing edges and one file storing the feature for every node, as well as the train/val/test split files.

3.1 File Structure

The structure of the dataset is as follows:

- `edges.txt`: This file contains all the edges in the graph. Every row is an edge. For one row (edge), the two columns are the two nodes that the edge connects.
- `features.txt`: This file contains all nodes in the graph and their features. Every row is a node. For one row (node), the first column is the node name, and the group of rest column elements is the feature vector for this node.

[1] `https://www.kaggle.com/c/aist4010-spring2022-a3/data`

- `train_labels.csv`: You are supposed to use nodes and their labels in train data set to train your model for the task. The first column is the node name, and the second is the label.
- `val_labels.csv`: You are supposed to use nodes and their labels in val data set to validate the classification accuracy. The first column is the node name, and the second is the label.
- `test_idx.csv`: You are supposed to assign classes for nodes in test data set and submit your result. The first column is the node name, and you need to fill in the class in the second column.
- `sample-submission.csv`: This is a sample submission file for this competition.

3.2 Loading Graph Data

There are multiple ways to load the graph data. You may directly use `NumPy` to load the files and build the graph. Or you may choose graph packages like `Networkx` [2], `DGL`[3] to build the graph.

3.3 Classification Models

Feel free to use any model architectures including MLP, GCN, and some latest graph models.

Note that you are **NOT ALLOWED** to use **pre-trained models** in this assignment, for we would like you to learn about GNNs in detail.

But you are **ALLOWED** to reference model architectures from papers or GitHub repos.

4 System Evaluation

The evaluation metric is quite straightforward:

$$accuracy = \frac{\#\ correctly\ classified\ nodes}{\#\ total\ nodes} \tag{1}$$

You may check the `sample-submission.csv` for your reference before submission.

5 Submission

Following are the deliverables for this assignment:

- **Kaggle submission**. Please submit your results on Kaggle page with your nickname. Make sure your nickname appears on the public leaderboard. If you are better than the baseline, your marks can be above 60%. If you are better than the deep learning baseline, your marks can be above 80%.
- **Blackboard submission**. Please pack all files in one '.zip' file named as **'nickname_real name_SID'** like 'lctest_Licheng ZONG_1155123456.zip'. Please name your files as **this format**, or you will get **mark deduction**.
  - A one page report describing your model architecture, hyper parameters, the epochs you trained and any other interesting details leading to your best result for the above competition. Please limit the report to **one page**.
  - **All** your source codes as the format of '.ipynb' or '.py' in **one folder**.

[2] `https://networkx.org/`
[3] `https://www.dgl.ai/`

6 Conclusion

Nicely done! Here is the end of Assignment 3 (Kaggle Part), and the beginning of the GNN world. As always, feel free to ask on Piazza if you have any questions. We are always here to help.

Good luck and enjoy the challenge!

7 Reference

1. `https://neptune.ai/blog/graph-neural-network-and-some-of-gnn-applications`
2. `http://web.stanford.edu/class/cs224w/`
3. Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Philip, S. Y. (2020). A comprehensive survey on graph neural networks. IEEE transactions on neural networks and learning systems, 32(1), 4-24.