Assignment 3 - Kaggle Part
Graph Node Classification

AIST 4010: Foundation of Applied Deep Learning (Spring 2024)

DUE: **11:59PM (HKT), Apr. 8, 2024**

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 include: 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 the 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-spring2024-a3`

2 Node Classification

2.1 Graph Encoding

This time we will meet a new data format, graphs. We know a graph has vertices and edges. Usually, we can encode the graph by **adjacency matrix** representing the structural information. For an 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 others 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 a certain node should be determined by its features and its neighbors. And you will finally find that even if we only know a small number 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, and node features, then your model needs to predict the classes for some nodes.

The task could be divided into 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 the rest column elements is the feature vector for this node.

[1] `https://www.kaggle.com/c/aist4010-spring2024-a3/data`

- `train_labels.csv`: You are supposed to use nodes and their labels in the 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 the 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 the 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.

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 Models

Since it's an optional assignment, there is no suggestion here. Feel free to use any model architectures including GCN, GAT and some latest graph models.

But note that you are **NOT ALLOWED** to use **pre-trained models** in this assignment, because we hope you can learn about GNNs in detail.

But you are **ALLOWED** to refer to open-source codes.

6 Submission

The following are the deliverables for this assignment:

- **Kaggle submission.** Please submit your predicted results on the Kaggle page with your nickname. Keep it the same with A1. Make sure your nickname appears on the public leaderboard. If your score is higher than the **LR baseline**, you can obtain at least 60%. If your score is higher than the **MLP baseline**, you can obtain at least 80%. The final score will depend on your ranking. At least the first three students can obtain a full score.
- **Blackboard submission.** The Deadline for the Blackboard submission is **one day later** than the Kaggle submission, so you don't need to rush. Please pack all files in one '.zip' or '.tar.gz' file named 'nickname_SID'. For example, if my nickname is 'lctest' and my SID is '1155123456', I should name my submission file as 'lctest_1155123456.zip'. And it should have these contents:

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

  - A report describing your model architecture, loss function, hyperparameters, and any other interesting detail led to your best result for the above competition. And cite all references in this report. Please limit the report to **two pages** (including references) and submit it in **'.pdf' format.**
  - A sub-folder containing **all** your source codes (including data-processing, training, prediction, etc.) in '.ipynb' or '.py' format.

7 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 us on Piazza if you have any questions. We are always here to help.

I'd like to emphasize some notices here:

- A3 Kaggle part is **optional**. Although we recommend you complete it to learn about GNNs, you could choose not to do it **without** any punishment.
- A3 Kaggle is actually a bonus part. If you complete this A3-Kaggle, you can use the two highest scores among the three Kaggle-assignments as your final grade.
- Importing additional data is **NOT ALLOWED**.
- Directly calling model APIs is **NOT ALLOWED**.
- Referring to open-source codes to implement the models is **ALLOWED**. Remember to cite them.
- Loading pre-trained weights is **NOT ALLOWED**.
- The Kaggle submission deadline is **11:59 PM (HKT), Apr. 8, 2024.** The competition will close exactly at that time. You **CANNOT** use late days for the Kaggle part assignment. Grades will be deducted by 25% for each late day.
- The Blackboard submission deadline is one day later: **11:59 PM (HKT), Apr. 9, 2024.**. You must submit the files required on Blackboard, or you will face a 10% mark deduction

Good luck and enjoy the challenge!

8 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.