Data Mining (COL761) A4

March 2024

Instructions

- Submission deadline is **19 April 23:59 IST 2024**.
- You need to do the homework in your already formed teams. You will upload the code and report to the GitHub repository mentioned in A0.
- **Your code must compile and execute on HPC.**
- You will submit a script to **moodle** (details in submission instructions section).
- Do not copy the code from your friends or from the internet. **Plagiarism will result in -X marks where X is the total of the assignment.**
- This assignment has to be done in Python using PyTorch Geometric.
- Please read the full document clearly.
- This assignment is competitive.
- Datasets are available here.

Q.1 Graph Classification. [30 points] In this question you are required to learn a mapping that maps a graph to its classes based on data provided to you. Formally, your input will be a list of graphs $[G_1, \ldots, G_n], G_i = (V_i, E_i)$ with additional node features X_i. In the dataset with C classes represented by numbers in the set $\{0, 1, \ldots, C-1\}$, you will also have a mapping $y : G \to \{0, 1, \ldots, C-1\}$ that maps each graph to its class. Your task is to learn the parameters $\boldsymbol{\theta}$ of a parametric function $f_{\boldsymbol{\theta}}$ such that it maximizes the accuracy of prediction of classes on an unseen set of Graphs G_t. Accuracy is defined as

$$\text{accuracy} := \frac{\sum_{g \in G_t} \mathbf{1}\{f_{\boldsymbol{\theta}}(g) = y(g)\}}{|G_t|}, \text{ where } \mathbf{1}\{x\} = \begin{cases} 1, & \text{if } x \text{ is a true statement} \\ 0, & \text{otherwise} \end{cases}$$

For training you will only be given a graphs with class labels. You must create unseen partitions from this to test the performance of your chosen model for the function $f_{\boldsymbol{\theta}}$. For testing your code will be given the unseen graphs and your code must generate class labels for all the graphs in order. We will evaluate your learned function's performance based on it.

Note that since model training is expensive, we expect you to provide a trained model with best performance, and we will train the model on our end only if needed whether to verify the model code and performance agree.

`interface.sh` can be run in the following formats.

- `$ bash interface.sh train </path/to/dataset> </path/to/output/model/in>`. This trains the model with the provided dataset path and puts the trained model in the model output path.
- `$ bash interface.sh test <path/to/model> </path/to/test/dataset> </path/to/output/labels.txt>`. This takes the test dataset and outputs the labels in a file in plaintext formatted. The plaintext format is as defined in the following specifications.
  - Each line should contain just one number. The number in line n represents the predicted class of graph n in the test dataset.
  - There should be no other lines (no empty lines, no lines containing comments or any form of text, no header lines, *etc.*).

Q.2 Node Classification. [30 points] In this question you are required to learn a mapping that maps nodes of a graph to their classes based on data provided to you. Formally, your input will be a graph $G = (V, E)$ with additional node features X. In the dataset with C classes represented by numbers in the set $\{0, 1, \ldots, C-1\}$, you will also have a mapping $y : V \to \{0, 1, \ldots, C-1\}$ that maps each node to its class. Your task is to learn the parameters $\boldsymbol{\theta}$ of a parametric function $f_{\boldsymbol{\theta}}$ such that it maximizes the accuracy of prediction of classes on an unseen set of nodes V_t (and corresponding E_t, X_t). Accuracy is defined as

$$\text{accuracy} := \frac{\sum_{u \in V_t} \mathbf{1}\{f_{\boldsymbol{\theta}}(u) = y(u)\}}{|V_t|}, \text{ where } \mathbf{1}\{x\} = \begin{cases} 1, & \text{if } x \text{ is a true statement} \\ 0, & \text{otherwise} \end{cases}$$

For training you will only be given a subset of the full graph with all the known labels. You must create unseen partitions from this to test the performance of your chosen model for the function $f_{\boldsymbol{\theta}}$. For testing your code will be given the full graph and your code must generate class labels for all the nodes. From this we will extract the labels of only those nodes that weren't shared for training and evaluate your learned function's performance based on it.

Note that since model training is expensive, we expect you to provide a trained model with best performance, and we will train the model on our end only if needed whether to verify the model code and performance agree.

`interface.sh` can be run in the following formats.

- `$ bash interface.sh train </path/to/dataset> </path/to/output/model/in>`. This trains the model with the provided dataset path and puts the trained model in the model output path.
- `$ bash interface.sh test <path/to/model> </path/to/test/dataset> </path/to/output/labels.txt>`. This takes the test dataset and outputs the labels in a file in plaintext formatted. The plaintext format is as defined in the following specifications.
  - Each line should contain just one number. The number in line n represents the predicted class of node n in the test dataset.
  - There should be no other lines (no empty lines, no lines containing comments or any form of text, no header lines, *etc.*).

Q.3 Link Prediction. [40 points] In this question you are required to learn a mapping that maps pairs of nodes u and v of a graph to a binary number representing the answer to the question "whether there should be an edge between u and v based on the edges present in the rest of the graphs?" Basically, the graph has edges based on some unsaid rules, but these edges have not been made exhaustively, and your task is to find more pairs where the edges should be according to the same unsaid rule.

Formally, your input will be a graph $G = (V, E)$ with additional node features X. Imagine that you also have a function $e : V \times V \to \{0, 1\}$ which assigns 1 to a pair of nodes that have an edge between them and 0 to a pair of nodes that don't have edges between them. Your task is to learn the parameters $\boldsymbol{\theta}$ of a parametric function $f_{\boldsymbol{\theta}}$ such that it maximizes the accuracy of prediction of edges. Accuracy is defined as

$$\text{accuracy} := \frac{\sum_{u,v \in V} \mathbf{1}\{f_{\boldsymbol{\theta}}(u, v) = e(u, v)\}}{|V|^2}, \text{ where } \mathbf{1}\{x\} = \begin{cases} 1, & \text{if } x \text{ is a true statement} \\ 0, & \text{otherwise} \end{cases}$$

For testing, your code will be given test edges (pairs of nodes) and you need to predict, in order, whether there should be an edge between the two nodes or not. This generated output should be saved in a file with the format specified below.

Note that since model training is expensive, we expect you to provide a trained model with best performance, and we will train the model on our end only if needed whether to verify the model code and performance agree.

`interface.sh` can be run in the following formats.

- `$ bash interface.sh train </path/to/dataset> </path/to/output/model/in>`. This trains the model with the provided dataset path and puts the trained model in the model output path.
- `$ bash interface.sh test <path/to/model> </path/to/test/dataset> </path/to/output/labels.txt>`. This takes the test dataset and outputs the labels in a file in plaintext formatted. The plaintext format is as defined in the following specifications.
  - Each line should contain $|V|$ numbers separated by commas. Each number should either be 0 or 1. The numbers should appear in order of nodes in V. This file is basically a csv representation of the predicted adjacency matrix.
  - There should be no other lines (no empty lines, no lines containing comments or any form of text, no header lines, *etc.*).

Submission instructions.
These instructions will be constant across assignments, and have already been shared in A1. So, make sure that you have structured your A4 assignment accordingly. Like other assignments, you will be submitting

- `clone.sh` to moodle. Should contain just one line similar to `git clone https://github.com/my_repo` if your repository is named "my_repo". **Do not include personal access tokens, and don't put ssh clone commands.**
- This should clone your repository. Say your repository is named "my_repo", then we should find `my_repo/A4`.
- Within "A4" directory there should be
  - ★ Code with respect to each question should be within its own directory (Q1, Q2, and Q3, respectively).
  - ★ Within each directory, there should be an `interface.sh` file as described above.
- The time allotted to each code will be $\frac{1}{2}$ hours per dataset for node classification and link prediction. Time allotted for graph classification dataset `GC_D1.pt` will be 1 hour, but for `GC_D2.pt` it's still $\frac{1}{2}$ hours only.
- While running the code, assume that your code will run within an anaconda environment with PyTorch 1.13, and PyTorch Geometric 2.3.1 installed along with other possible scientific and ML libraries like NumPy, SciPy, Pandas, scikit-learn, NetworkX installed in it.
- Note that while saving models it is recommended to save the model weights using `torch.save` and `state_dict` of a model. The `state_dict` object has keys according to names of parameters in the model, which may change across different versions of PyTorch Geometric, so ensure you use 2.3.1, and PyTorch 1.13 for maximum compatibility. Also ensure to use `map_location='cpu'` option of `torch.load` when loading model weights from `state_dict`.