
Assignment 1

Deadline : 11:59 P.M 12 Jan 2023

Important details

- You have to use Python version 3.6, PyTorch version 1.9.0 and PyTorch geometric version 2.0.3.
- This is an **individual** assignment.
- You have to submit your code on a PRIVATE github repo and share the github link on moodle submission link. You need to add 'sahilm1992' as collaborator to your github repo. If your submission is modified post due-date, the submission will not be evaluated.
- Your submission will be evaluated using an automated script. Please make sure you follow submission instructions.
- Plagiarism results in F grade.
- You might not necessarily need a GPU for running these expts. A good laptop should be ok. Hence, you need not necessarily use the cuda version of PyTorch. You can run the expts. on CPU also. During evaluation, for each run, we will set a maximum limit of 1 hour. Your code should produce the desired plots/output files within that time.
- Unethical attempts to modify the data loader etc. to gain advantage in terms of performance etc. will lead to strict penalty.
- You can find some tutorials here.
- Note that you are not supposed to import GNN Conv layers like GCNConv, GATConv etc. The submissions might be made public to all students post evaluation. If you have any doubts please confirm from us.
- We may conduct demos for your evaluation. In the demo, students will be asked to explain their code and make modifications on the fly. The performance in the evaluation will reflect on the overall score of the assignment. Skipping the demo would lead to 0 in the assignment.
- The deadline is strict. No extensions(even with penalty) will be given.

1 Assignment

In this assignment we will work on the task node classification using a specific GNN architecture.

Node classification task: We are given a graph $G = (V, E)$. In the graph G , each node $v \in V$ has a label assigned to it. Further, nodes in the graph G are associated with a feature set $\mathbf{V} \in \mathbb{R}^{|V| \times d}$ i.e each node v has features of dimension d . For a node-set $V_l \subset V$ the labels are available to us for training a Graph Neural Network. Let $Y_v \in \{1..C\}$ denote the label for node $v \in V_l$ and C denote the no. of classes.

Goal: To learn parameters θ of a graph neural network (GNN_θ) on an input graph G using the training node labels V_l . The trained model should be capable to predict labels of remaining unlabeled $V - V_l$ nodes.

1.1 Graph Neural Network

In this section, we define the graph neural network architecture that will be used by us.

Let $\mathbf{x}_v^0 \in \mathbb{R}^d$ refer to the input node features for a node v . In order to exchange information between neighbors, we create a GNN whose message aggregator is defined as below:

$$\mathbf{x}_v^k = g_\theta \left(\mathbf{x}_v^{k-1} + \sum_{j \in \mathcal{N}(v)} \mathbf{x}_j^{k-1} \right)$$

The above equation generates feature representation for a node v at k^{th} layer of the GNN using node representations of nodes at previous layer.

Here, $\mathcal{N}(v)$ refers to neighbors of the v^{th} node. The term g_θ represents a neural network, i.e. an MLP. Your implementation should be capable to handle different values of k .

1.1.1 Loss function:

The loss $\mathcal{L}(v)$ for a node v is defined using the following equations.

$$\tilde{y}_{vc} = \frac{\exp \mathbf{w}^T \mathbf{x}_v^k}{\sum_{i=1}^{i=C} \exp \mathbf{w}^T \mathbf{x}_v^k} \quad (1)$$

$$\mathcal{L}(v) = - \sum_{c=1}^{c=C} y_{vc} \log (\tilde{y}_{vc}) \quad (2)$$

where \mathbf{w} is a trainable parameter.

We learn the parameters of our graph neural network by minimizing the above loss function¹.

1.2 Splitting of datasets:

For the given datasets we use the default train/val/test split as provided by PyTorch Geometric loaders. You need to use the training nodes to train the model, the validation nodes to choose the best model(the epoch where the validation loss is minimum) and the test nodes to test the model.

1.3 Details of dataset and code:

You are provided with skeleton code in 2 files named ‘run.py’ and ‘data_utils.py’ at this [Link](#). Do not write any code in data_utils.py as it will be replaced during evaluation. Do not remove the import of ‘data_utils’ in ‘run.py’. The run.py file accepts a command line argument –dataset (dataset name) and –k (number of GNN layers).

The example command to execute the run.py for 2 datasets ‘Cora’ for 2 GNN layers and ‘CiteSeer’ for 1 GNN layer is as follows:

¹In PyTorch you can use the CrossEntropy loss function on the training nodes. Make sure to check whether you need to pass logits or softmax probability scores depending upon the loss function you choose.

```
python run.py --dataset Cora -k 2
python run.py --dataset CiteSeer -k 1
```

Note that during evaluation we might use different datasets apart from the above two.

Inside the run.py you will find an object named 'data' on line #24. The data object has the following attributes:

x: node features of shape [number of nodes, number of node features]

edge_index: edge index matrix of shape [2, number of edges]

y: node labels of shape [number of nodes]

train_mask: Type bool of shape [number of nodes]

val_mask: Type bool of shape [number of nodes]

test_mask: Type bool of shape [number of nodes]

The mask variables(eg:- train_mask) is used to identify whether a node is present in train/test/validation. For example if train_mask[i] is True then i^{th} node will be used in training.

2 Hyperparameters

We do not necessarily restrict you to choose certain set of hyper parameters, however please keep them reasonable. For example don't use a learning rate of 500 or a hidden dimension size of 2000. You are free to add more MLP layers if you want, for example to project input features to different dimensions. You could also use activation functions like ReLU etc. Using unreasonable hyper-parameters and getting very different performance/ incorrect conclusions cannot be justified. The instructions will not be able to help you much on this aspect.

3 Evaluation:

Conceptual understanding: The evaluation will focus on whether you have learned/already know how to implement a given message aggregation function. Further, we will also test whether you are able to use the correct loss function in the right manner etc. It is recommended that you add comments to the portions of the code deemed necessary.

Empirical evaluation: In terms of empirical evaluation, you need to compute the Macro F1-score on train/val/test datasets provided to you.

Impact of topology Design an experiment to show the impact of incorporating topology on performance(Macro F1-score). How much topology should you incorporate? Please do an empirical study and answer. Use Cora dataset for this part.

Break up of marks:

- 45% for correct implementation of the convolution operator.
- 15% for correctly using the right loss function and integrating it with the GNN code.
- 20% Designing experiment and analyzing impact of topology
- 20% Obtaining correct plots/performance(with a permitted error threshold).

Note: No Marks will be awarded if the code does not implement the above specified GNN.

The next section describes the format of submission.

3.1 Submission instructions






Name
 data_utils.py
 run.sh
 run.py
 EntryNumber-summary.pdf
 run_topology.sh

Figure 1: An example submission

You need to provide the github url to your code on moodle submission link. Your submission needs to provide the following functionality:

```
sh run.sh dataset-name k
```

Here 'dataset-name' is the name of the dataset and 'k' is the number of GNN layers. The 'run.sh' script should be in the home folder of your github repo and should contain commands to execute your code.

Apart from code files, you need to submit a file named **EntryNumber-summary.pdf** in the home folder of your submission. The file should contain the following:

- You need to write in plain english in 1/2 paragraphs of what you did to create the new GNN.
- You need to write about the experiment you designed for impact of topology, the results you obtained, your analysis and the conclusion you obtained.

The 'run_topology.sh' script should contain commands to run experiments and save plots etc. in a new folder named topology_generated/ in the home folder. This script is for impact of topology experiment.

The 'sh run.sh dataset-name k' command should do the following(including your training, validation, testing):

- Generate a plot named EntryNumber-train-val-dataset-name-k-perf.png in the current folder showing epoch (on x-axis) vs validation and train Macro F1-score on (y -axis) (for atleast 2000 epochs). Time interval 20 epochs.
- Generate a plot named EntryNumber-train-val-dataset-name-k-loss.png in the current folder showing epoch (on x-axis) vs train and validation loss on (y-axis) (for atleast 2000 epochs). Time interval 20 epochs.
- Create a file named EntryNumber-dataset-name-k-results.txt in the current folder. The first line of the file contains the test Macro F1-score obtained on the epoch where we obtain the lowest validation loss. The second line contains the best validation Macro F1-score obtained.

'k' and 'dataset-name' in the above file names represent number of GNN layers and the name of the dataset respectively.

Note: You are free to add as many files as you want in the submission. However, kindly limit your submission to < 20 MB.

Plagiarism policy: The code provided to you by us will be excluded from plagiarism check. Rest all other code submitted by you will be checked for plagiarism. Copy cases will be given F grade.