EN.553.744 Data Science Methods for Large-Scale Graphs

Homework 3

Luana Ruiz, Caio F. D. Netto, Ruijia Zhang

April 3, 2025

You must use Google Colab as your development environment—your `.ipynb` notebook will be your deliverable.

1 Introduction

In this assignment, you will observe the concept of transferability in graph neural networks (GNNs) using the `ogbn-arxiv` dataset. The dataset consists of a citation graph where nodes represent scientific papers published on arXiv, and edges represent citation relationships. The node classification task is to predict the subject area (category) of each paper given its content embedding and citation context.

You will experiment with training GNN models on subgraphs of the citation network and evaluating how well they generalize to both the full graph and a temporally extended graph including future publications.

2 Data handling

To load the arXiv dataset, you will need to download the `ogb` library. An installation that avoids broken dependencies on most Google Colab machines we tested is the following:

```
!pip uninstall torch
!pip install torch==2.5.0
!pip install torch_geometric
!pip install ogb
```

From `ogb`, you will need the following class, which will be used to download the dataset as:

```
from ogb.nodeproppred import PygNodePropPredDataset
```

```
dataset = PygNodePropPredDataset(name='ogbn-arxiv')
```

Exercise 1. *After loading the* `ogbn-arxiv` *dataset using PyG, report:*

- *Total number of nodes*
- *Number of classes*
- *Total number of edges*
- *Average node degree*
- *The node features (i.e., the paper attributes)*

Exercise 2. *Filter the dataset to only retain nodes (and associated edges) corresponding to papers published between 2005 and 2015 (inclusive). This will be your* `Dataset1`*. Report:*

- *Total number of nodes*
- *Total number of edges*
- *Average node degree*

Plot histograms of node years, class labels, and node degrees. Report any imbalances.

Exercise 3. *Write a function that takes a graph and splits its nodes randomly into 80% for training and 20% for testing. Generate a split of* `Dataset1` *and save it for later use.*

3 Transferability of GNNs

Let N be the size of the graph in `Dataset1`, which we will denote $\mathbf{G}_N$. Using $\mathbf{G}_N$, we will randomly sample subgraphs $\mathbf{G}_n$ of sizes

$$n = \{1000, 2000, 5000, 10000, 20000\}.$$

For each n, we will then split the nodes of $\mathbf{G}_n$ into training and test sets, and train five GNNs Φ_n, one per subgraph.

Exercise 4. *Using PyG, write a class for generating GNN models with ChebNet convolutions ($K = 3$), 2 layers, ReLU nonlinearity in the first layer, and softmax nonlinearity in the last layer. Is this GNN convolutional? Explain.*

Exercise 5. *Train five GNN models Φ_n on the five subgraphs $\mathbf{G}_n$.*

(a) Report the ***relative absolute difference*** *between:*

  – *Accuracy of Φ_n on the test set of $\mathbf{G}_n$*

  - *Accuracy of Φ_n on the test set of $\mathbf{G}_N$, the full 2005–2015 graph*

  Repeat this for 5 random realizations of each subgraph $\mathbf{G}_n$. Plot the ***mean*** *and* ***standard deviation (as error bars)*** *of this gap versus n.*

(b) *Report the test accuracy of each Φ_n on the test set of the full 2005–2015 graph $\mathbf{G}_N$. Also repeat this for 5 realizations of each $\mathbf{G}_n$ and plot the* ***mean accuracy with error bars*** *versus n.*

Discuss your results.

Exercise 6. *Create a second dataset including all papers from 2005 to 2018. This will be your* `Dataset2` *whose graph we denote $\mathbf{G}_M$. Report:*

- *Total number of nodes*
- *Total number of edges*
- *Average node degree*

Plot histograms of node years, class labels, and node degrees. Report any imbalances.

Exercise 7. *Split $\mathbf{G}_M$ in* `Dataset2` *into a training and a test set. Using the same models Φ_n trained in Exercise 5, evaluate them on the test set of $\mathbf{G}_M$. Repeat this for 5 realizations of each $\mathbf{G}_n$ and plot the* ***mean accuracy with error bars*** *versus n. Discuss your results.*