CSCI-4974/6971: Homework 4
<v1.0> updated April 15, 2026

Vertex Classification and Subgraph Mining

Due Date: Friday 17 April 2024, 11:59pm via Submitty

For this assignment, we're going to consider GNNs for the vertex classification problem as well as a standard subgraph frequency analysis study. For vertex classification, we're going to build on the basic GCN models discussed in class. Subgraph frequency analysis is going to focus on analyzing graphs given a selection of possible null models. We'll be using the following 'real' network for a baseline comparison.

- Dolphin friendships: `http://cs.rpi.edu/~slotag/classes/SP26m/hw/dolphins.data`

We will be using Submitty for collecting homeworks. Upload a single *.py file that outputs responses for all of the below. Pay careful attention to output formatting. Your code should be runable as a script on the command line (via `bash$ python3 hw04.py`). **You can use any NetworkX/NumPy/SciPy/PyTorch functionality you wish, but do not use any other external libraries unless otherwise specified**.

1. For the first part of the assignment, we will be comparing a few different approaches for the vertex classification problem. We will be considering the standard Planetoid dataset of Cora, varying different features and architectures then analyzing the impact on final accuracy. See below for more implementation details into our the algorithms:

   - Label Propagation – This will serve as our baseline. Feel free to use the code from class or NetworkX's implementation.
   - PyTorch GCN – We will be considering multiple different feature sets, described below. To keep things simple, just use the architecture supplied in the code for Lecture 15. Make sure you normalize your features from [0,1], especially for degree centrality.

   And the feature sets that we will be comparing:

   - Baseline – Simply use the features supplied by the dataset. E.g., for the Cora citation network these would be a boolean word vector designating whether a specific dictionary word appears in the paper represented by a vertex. These are directly included in the dataset when you load it.

- Structure Only – As was shown in class, one approach is to simply learn a graph embedding by using as the feature set the identity matrix. This results in the GCN basically learning purely based on the network topology (effectively, what label propagation is doing in an unsupervised way).
- Centrality Only – We'll use the five centrality measures from HW2 (degree, closeness, betweenness, eigenvector, PageRank) as the features for each vertex.
- Structure+Centrality – Concatenate the identity matrix feature tensor along with the centrality measures features.
- Baseline+Structure+Centrality – Now concatenate all three together.

For handling training and testing data, do a 90/10 training/test split as we've done in class. We'll primarily be reporting the maximum test accuracy, so we aren't going to worry about a validation set to keep it simple. Check the Lecture 15 code to observe one way it was done. There's many other ways and built-in functions to accomplish the same thing.

Run all 6 methods 5 total times, testing accuracy after each iteration/epoch. You'll track the maximum accuracy achieved among the 5 tests for each implemented method. This is what you will report as output.

2. For the second part of the assignment, we will be doing a subgraph frequency analysis, comparing the subgraph frequencies in real data to a few different potential 'null model' random graphs. You'll read in the real graph and determine the number of vertices, number of edges, and the degree distribution. You will then use these values to generate 5 instances each of 4 random graphs:

   - Erdos-Renyi $G(n, m)$ graphs
   - Chung-Lu graphs
   - Barabasi-Albert graphs
   - Watts-Strogatz graphs with $p = 0.25$

   Next, you'll use generated instances of all possible 3- and 4-vertex connected subgraphs created from the methodology as given in class. For each network and subgraph, determine the average number of counts over each of the 5 generations for a network type. You'll output those averages for comparison of the 'null models' against the real network.