HW3

Due: 12th November

Total: 60 points

Submission Instructions:

- You need to do the homework in your already formed team of 3.
- Only one submission per team will be accepted. Make sure no duplicate submissions are made by another team member.
- Upload all code to GitHub.
- Your GitHub repo must contain all scripts and code you used in this assignment (including preprocessing scripts and code, if any).
- Your submission should be as follows :
    - Submit an install.sh file on moodle such that running **'sh install.sh'** should clone your repository and load required modules.
    - After unzipping, we should get a folder of the name 'HW3_kerberosid'. Eg: If submitter's entry number is 2019CSZ8763, your folder name will be "HW3_csz198763".
    - Inside the main folder, there should be a subfolder named Q2. Include all code/scripts for Q2 in this sub-folder. The main folder should also contain a writeup file and a readme file (refer to the last 3 points for details about the writeup and readme file).
    - Running **'sh Q2.sh <saved>'** should run all the scripts/code corresponding to Q2 and print the RMSE score on the test data in the terminal. Note that **<saved>** will be either 0/1. If saved is 0, then training should occur from scratch, and if saved is 1 it should load the saved model that you would have trained already, and we will evaluate the model based on RMSE on the test data. Further instructions will be given in Q2. Only you can refer to the link given below for saving and loading pytorch models. https://pytorch.org/tutorials/beginner/saving_loading_models.html
    - **Please do not submit data files.**
    - Include a **single writeup file (in pdf format),** that contains the answers to Q1. Also for Q2, you should mention which GNN architecture you used and why, draw an architecture diagram of your model, mention the loss function used and also the parameters of the model. Writeup file name must be "**report_kerberosid.pdf**" . Also mention any assumptions you have made specific to each question.
    - Include a **readme file.** Filename should be **readme_kerborosid.txt**. The file should give all instructions to run your code on hpc. However, instructions for Q2.sh should be as specified above. Also include steps on how to run additional code/scripts (if any). Just in case, any ambiguity arises, we will refer to the readme instructions to resolve.
    - If submitter's entry number is 2019CSZ8763, your readme file name will be "readme_csz198763.txt" and writeup file name will be "report_csz198763.pdf". All letters in file name will be in small. Follow this convention strictly.

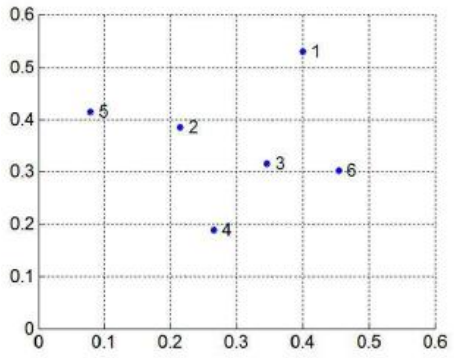
Note:

- Students need to test their code on HPC before submission. Code not running on HPC will be given zero marks.
- Submission time will be latter of submission time on moodle and time of last github commit.
- Late policy will be as informed at the beginning of the course.

Anti-Plagiarism Policy for Homework 3

Any detected attempts at plagiarism either from parallel/past submissions or the Internet will risk an F-grade in the course.

1. [20 points]
   a. Draw the dendrogram for single linkage clustering on the data below. Show all steps. (5 points):



Point	x	y
1	0.40	0.53
2	0.22	0.38
3	0.35	0.32
4	0.26	0.19
5	0.08	0.41
6	0.45	0.30

   b. What is the computation complexity of single-linkage hierarchical clustering of n points without using the support of any other data structure? Explain your reasoning. (5 points)
   c. What is the complexity of the fastest possible algorithm? Give pseudocode of your algorithm and its complexity analysis. [10 points]

2. Use Pytorch-geometric to train a Graph Neural Network where you embed nodes in a feature space such that the concatenation of the node embeddings when passed through an MLP is the RWR distance between the two nodes. Specifically, given two nodes u and v, with embeddings z_u, z_v, we want to learn embeddings such that, passing the concatenated embeddings of u and v through an MLP should result as follows:

$$z = z_u \text{ concat } z_v$$
$$\text{MLP}(\square) = \square\square\square(\square,\square)$$

$RWR(u, v)$ will be computed with fixed length random walks where from u we do random walks of length 4 after which we deterministically restart to the source node. You are free to use any GNN architecture as long as it is supported by Pytorch-Geometric. Your architecture must be inductive in nature, i.e., generalizable to unseen data.

*Note : Number of layers in the MLP is a hyperparameter that you can tune. Use an **appropriate** activation function of your choice*.

Dataset: https://pytorch-geometric.readthedocs.io/en/latest/modules/datasets.html#torch_geometric.datasets.CitationFull

Use Cora dataset from class Planetoid. Set num_train_per_class:50, split:"random". Treat the 350 nodes (50 corresponding to each of the 7 classes) as u's. Corresponding to each u, sample 30 random v's (v not equal to u) from the above training nodes. Finally, your (u,v) pairs and their corresponding RWR scores would comprise of the supervised data that you should use during your training. Hint: Use data.train_mask to sample out the training nodes, from data.x.

Similarly, for testing, set num_test:1000, and treat those 1000 nodes as u. Corresponding to each u, sample 10 random v's (v not equal to u) from the above test nodes. Compute actual RWR scores corresponding to each (u,v). Report the RMSE between the actual and the predicted scores. Hint: Use data.test_mask to sample out the test nodes, from data.x.

You can also use validation data similarly to tune any parameters, or for early stopping.

Please do not save the test data, we will test on another random subset of the node pairs.

Your marks will be **(Best RMSE / Your RMSE * 40)**.

Added remarks: You can use some sampling strategy instead of random for making the pairs (u,v) for training if most of the RWR scores for training pairs are turning out to be 0. Please mention that in your report. You can even try with all pairs from training nodes if it scales.