Korea Advanced Institute of Science and Technology
School of Electrical Engineering
EE531 Statistical Learning Theory , Fall 2019

Issued: Dec. 3 2019
Due: Dec. 19 2019 (11:59 PM)

Assignment IV

Policy

Group study is encouraged; however, assignment that you hand-in must be of your own work. Anyone suspected of copying others will be penalized. The homework will take a considerable amount of time so start early. There are PyTorch problems in the assignment, and it may require some self-study.

1. **Implementing Graph Convolutional Networks (GCNs) (PyTorch programming)**

   In this homework, students will implement a GCN for tackling a semi-supervised classification task. Specifically, students will be provided with CORA dataset which contains sparse bag-of-words feature vectors for each document and a list of citation links between documents. We treat the citation links as (undirected) edges and construct a binary, symmetric adjacency matrix. Each document has a class label that is one of seven classes: Case Based, Genetic Algorithms, Neural Networks, Probabilistic Methods, Reinforcement Learning, Rule Learning, Theory. The detailed structure of the graph is described as follows:

   - There are 2708 nodes in the graph. We use all nodes for training where 140 nodes are labeled and 2568 nodes are unlabeled. We use 300 labels for validating and 2268 labels for testing.
   - There 5429 edges are describing the connection between nodes that are given.
   - Each node is represented by a feature which is a 1433-dimensional vector.

   The purpose of a GCN is to predict the class of the unlabeled nodes given a graph consisting of labels and unlabeled nodes. A skeleton code is provided in a IPython Notebook file `2019_Fall_EE531_Assignment_4.ipynb`. This skeleton is implemented using `pytorch` deep learning framework (if you are new to `pytorch`, you can self-study deep learning `https://pytorch.org/tutorials/beginner/deep_learning_60min_blitz.html`). You can run IPython Notebook file on Google Colab, this is a free cloud service and it supports free GPU (see instructions below for using). In the skeleton code, we provide classes and utility function described as follows:

   - `adj, features, labels, idx_train, idx_val, idx_test = load_data()`: Function used to load graph dataset. It returns the adjacent matrix (`adj`) size of $N \times N$, list of nodes with their features (`features`) size of $N \times D$, list of node labels (`labels`) size of N, and the split index of data for training, validating, testing. As we mentioned above, we use 140 labels for training, 300 labels validating, 1000 labels for testing, $N = 2708, D = 1433$.
   - `GraphConvolutionLayer()`: This is a base layer class for a Graph Convolutional Network model.

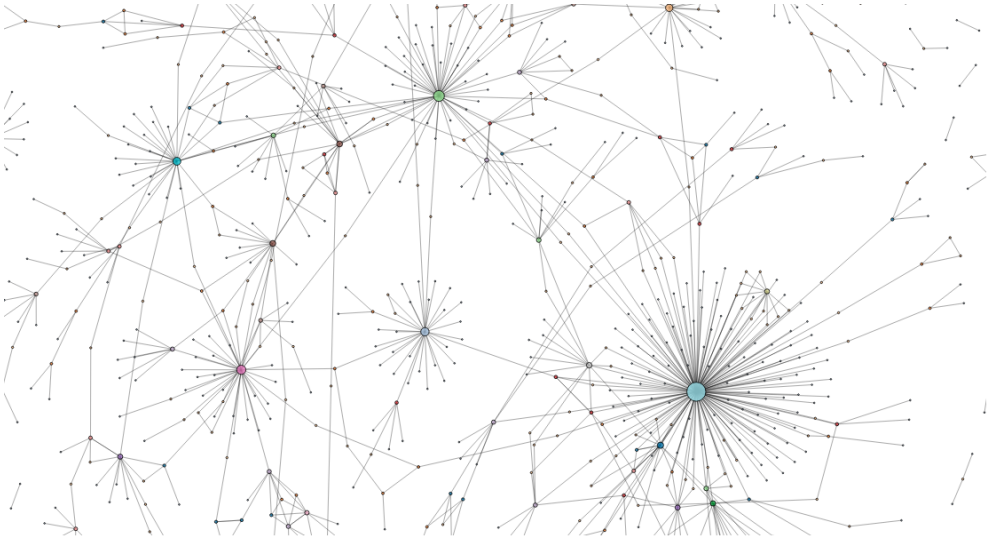


Figure 1: The graph dataset visualization

(i) In this class, function `forward()` takes nodes's feature of size $N \times D$ and adjacent matrix of size $N \times N$ as input. The matrix represents the graph structure, and the function outputs a new feature representation for the graph. *You have to complete this function.*

(ii) The function `init_parameters()` is used to initialize the weight of graph convolution layer. *You have to complete this function.*

- `GCN()`: Class for GCN model. In this class, you have to stack several `GraphConvolutionLayer` to build the network.

  (i) In this class, function `forward()` takes inputs as nodes and adjacent matrix. *You have to complete this function.*

- `train()`: The utility function contains the training loop. *You have to complete this function.*
- `test()`: The utility function for testing. *You have to complete this function.*

We describe the main components of the algorithm in the form of a skeleton code. For detailed information, please look at the code.

Requirement for report:

- Describe your implementation.

- Plot the learning curve and performance curve for training and validating over the epoch.

- Plot confusion matrix for the test set.

- Report final result on the test set, comment and conclude your results.

2. **Submit Instructions for Programming Assignment**

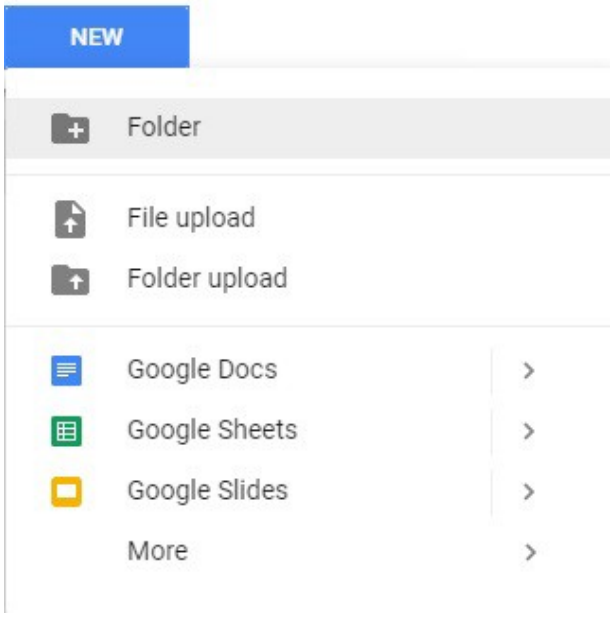
   Please submit the homework in .zip folder into KLMS named `EE531_Assignment4_ID.zip`, for example, `EE531_Assignment4_20181234.zip`. This folder should contain your completed IPython Notebook file `2019_Fall_EE531_Assignment_4.ipynb` and **a report** (it should be in.pdf format) reported the results, and explanation of the result.

In python code, the comment explaining your code **must be** included, or you will not get a full grade even if your code works fine. Do not change the name of the provided classes, functions, and comments should be written in English. Additionally submitting unexecutable code will receive **no points**.

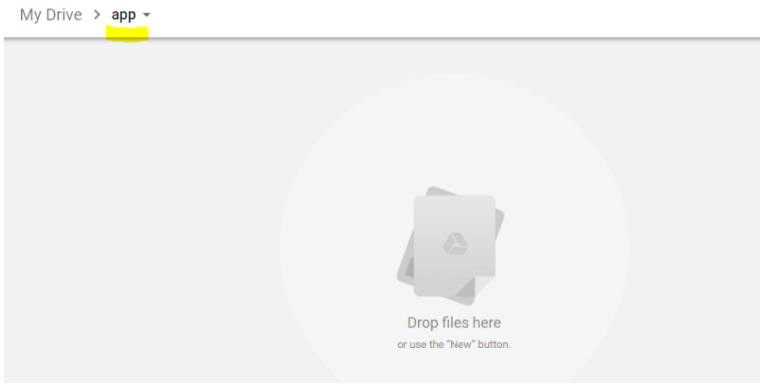
*** Google Colab Guideline**

Google Colab is a free cloud service and it supports free GPU. To start using Google Colab, follow the step below:

- Creating Folder on Google Drive.

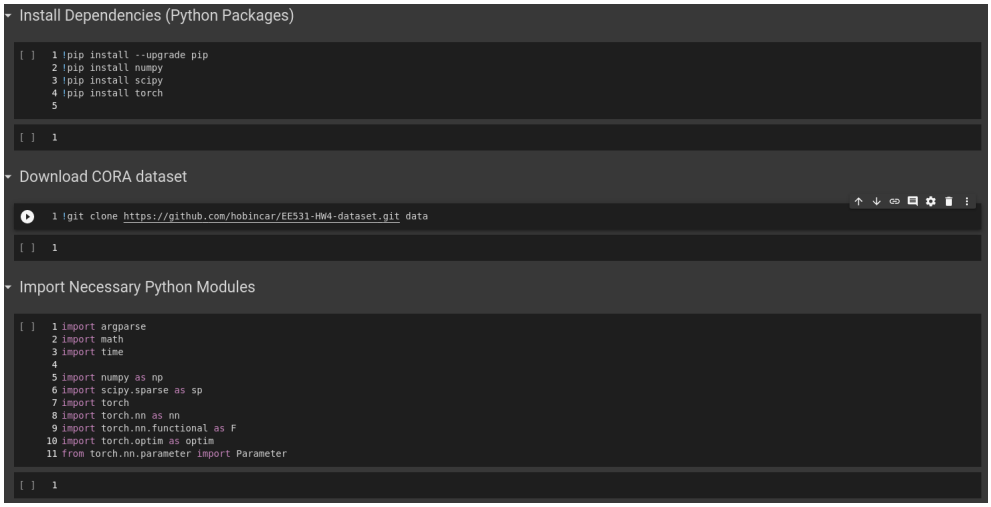


- Upload the IPython Notebook file (`2019_Fall_EE531_Assignment_4.ipynb`) into the new created folded. You can do it quickly creating a new folder, and draging the file from your computer and dropping it into *'Drop files here'* area. Look at the figure below.



- Now open it by Google Colab by right-click → Open With → Google Colaboratory, or just simple, double click to open file!
- The Ipython Notebook is organized by many cell as figure below.

  To run the program, you can either run each cell sequentially or run the whole file once (it can be slow). Keep in mind that if you modify some cells, let's run that cells again to make sure



that it is updated. To run the selected cell, clicking Runtime → Run the focused cell (or Ctrl + Enter). To whole file, clicking Runtime → Run all (or Ctrl + F9). There are many other options, try to figure it by yourself.

Reference

[1] Kipf, Thomas N. and Welling Max, Semi-Supervised Classification with Graph Convolutional Networks, ICLR 2017, `https://arxiv.org/pdf/1609.02907.pdf`