Assignment-1

GMLFA (AI60007) - Autumn,2024 - IIT Kharagpur

Release Date: [09/08/2024]
Submission Date: [23/08/2024]
Total Marks: 21

Instructions:

- All graded questions are compulsory to solve, non-graded questions are optional.
- Each group has to ***submit only one file*** named 'group_number_assignment1.ipynb'.
- ***Negative marking*** will be there as per our ***plagiarism policy*** given in the course webpage.
- You can use any language for coding questions, but ***'python'*** is preferred.
- Frameworks like Pytorch, Tensorflow are encouraged to construct deeper neural network architectures.
- You will be provided with one supporting code notebook (.ipynb) file with pseudocode if required.

Q-1:

Supporting Theory

Erdős-Rényi Random Graph: $G(n, p)$

An Erdős-Rényi random graph $G(n, p)$ is an undirected graph with n nodes, where each possible edge between two nodes exists with probability p, independent of all other edges.

Let V be the set of n nodes: $V = \{1, 2, \ldots n\}$. For each pair of nodes i and j (where $i \neq j$), we have: $P((i, j) \in E) = p$, where E is the set of edges, and (i, j) represents an edge between nodes i and j.

The probability of generating any specific graph G with m edges is:

$P(G) = p^m * (1 - p)^{(C(n,2)-m)}$

Where, $C(n, 2) = \frac{n(n-1)}{2}$ is the total number of possible edges in a graph with n nodes.

Expected number of edges: $E[|E|] = p * C(n, 2) = p * \frac{n(n-1)}{2}$

Expected degree of a node: $E[|deg(v)|] = p(n - 1)$

Part 1 : Write a Python function to generate the specified graph of 50 nodes (p=0.1) and plot the graph. (without using any external library) **[3 Marks].**

Part 2 : Generate the graph of flexible nodes i.e. (100, 200 upto 1000) and vary the probability value $p = 0.1, 0.2\ upto\ 0.6$ and **visualize the node degree distribution of the graphs [2 marks].**

Q-2: Node classification task based on centrality measures.

Objective:

Design a simple Neural Network model for multi-class node classification task using handcrafted designed features on the following benchmark Air Traffic datasets.

[You have to perform the experiments using the **'American air-traffic network'** ; the rest of the datasets are optional to explore.]

Dataset Details -

Brazilian air-traffic network: Data collected from the National Civil Aviation Agency (ANAC) from January to December 2016. The network has 131 nodes, 1,038 edges (diameter is 5). Airport activity is measured by the total number of landings plus takeoffs in the corresponding year.

American air-traffic network: Data collected from the Bureau of Transportation Statistics from January to October, 2016. The network has 1,190 nodes, 13,599 edges (diameter is 8). Airport activity is measured by the total number of people that passed (arrived plus departed) the airport in the corresponding period.

European air-traffic network: Data collected from the Statistical Office of the European Union (Eurostat) from January to November 2016. The network has 399 nodes, 5,995 edges (diameter is 5). Airport activity is measured by the total number of landings plus takeoffs in the corresponding period.

Link - https://github.com/leoribeiro/struc2vec/tree/master/graph

Part 1: Load the Graph and prepare the labels

1.1. Load the graph dataset.

1.2. Prepare one hot label for each node. **[2 Marks]**

Part 2: Feature Engineering and Augmentation

2.1. Compute the following features for each node:

2.1.1. Degree centrality

2.1.2. Eigenvector centrality

2.1.3. Betweenness centrality

2.1.4. Local clustering coefficient

2.1.5. Closeness centrality

2.1.6. PageRank

2.1.7. Katz Centrality

2.2. Concatenate these features with existing node features (if available) in the dataset. Prepare an augmented feature for each node. **[2 Marks]**

Part 3: Neural Network for Node Classification **[2 Marks]**

3.1. Preprocess the data and split into training (60%), validation (20%), and test (20%) sets.

3.2. Implement a neural network for multi-class classification. Design your own architecture to improve the accuracy. [minimum 3 layers]

3.3. Train the model for a minimum of 200 epochs using the ADAM optimizer (tune the learning rate) and evaluate its performance.

Part 4: Report the model's performance metrics.(test accuracy,precision,recall and F1-score) for all three datasets. Visualize the graph and the classification results. **[2 marks]**

Q-3: Graph Classification based on Graphlet Degree Vector.

Objective:

Perform a graph classification task based on graphlet degree vector count for the NCI109 dataset from the TUDataset collection. The NCI109 dataset is a collection of 4,127 chemical compounds, where each graph represents a compound, and the nodes and edges represent atoms and bonds. Each graph in the dataset is labelled as either "active" or "inactive" against a specific cancer cell line, indicating whether the chemical compound is bioactive or not.

Part 1 : Load the dataset and preprocess it. Extract the specific graphlet degree vector counts for each graph and take them as feature vectors for performing the graph classification task.

- **Types of Graphlets**

1. G1: Triangle
2. G2: Hexagon
3. G3: Pentagon
4. G4: K4 (complete subgraphs of size 4)
5. G5: K5 (complete subgraphs of size 5)
6. G6: 6-node clique
7. G7: 4-cycle (C4)

8. G8: Star (a centre node with more than 2 connections).
9. G9: Chains(linear sequences of 3 nodes or more)

For each graph, report the graphlet counts. **[4 Marks]**

Part 2 : Implement a 3-layer simple feedforward neural network with 64 hidden layer dimensions. Standardise the features of the dataset. Split the dataset into 80% train set and 20% test set. Use the ADAM optimizer with a learning rate of 0.001. Train the model for 200 epochs and report the model's performance metrics. Visualize the classification results and the features of the graphs that are important for the prediction of a chemical compound that is bioactive or not. **[3+1 Marks]**