CSE891 Project Part 2: Data Analysis

This mini project accounts for 10% of your final grade. The project must be done individually, without collaboration with other students. You are required to implement and apply machine learning methods for analyzing social network data. Your code must be implemented in Python on Jupyter notebook so that each step can be verified for correctness. You are prohibited from using any Python libraries for network analysis or online source code from other authors for this project. For part 2, you may use networkx library to display the network as well as kmeans clustering and logistic regression from scikit-learn library. Please check with the instructor first if you want to use other packages besides pandas, numpy, and other packages described below. The project due date is Sunday, Dec 13, 2020 (before midnight). This is the second part of the project description, which accounts for 50% of the project grade. You are strongly encouraged to start the project early.

1 Project Overview

The goal of this project is to provide you with hands-on experience applying machine learning methods to real-world data. The project involves all aspects of the data analysis pipeline—from dataset creation to data exploration and analysis. For this project, you will use a sample dataset from DBLP, a bibliography database of computer science publications (including journals, conference proceedings, book chapters, technical reports, etc). Your goal in part 2 is to analyze the co-authorship network you have generated in Part 1 of the project.

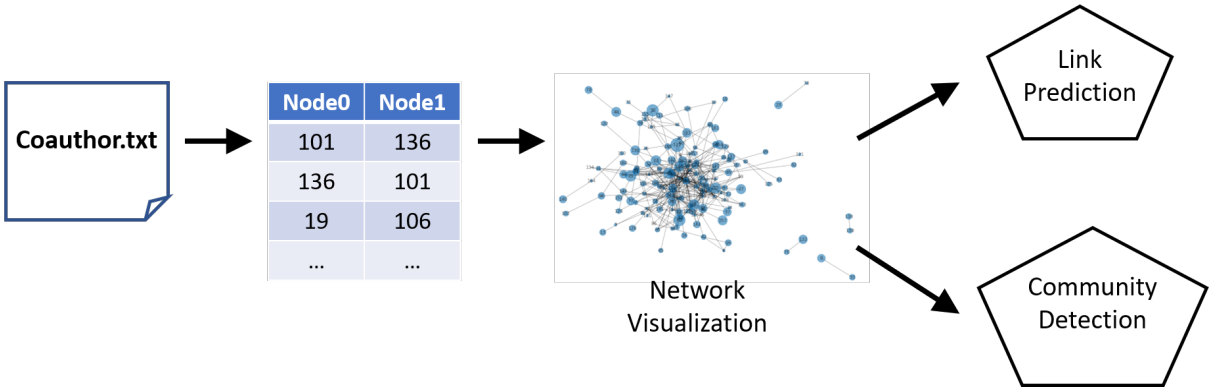

Figure 1: Summary of network analysis part of the project.

The specific tasks to be performed on this project are as follows (see Figure 1 for task overview):

1. **Link prediction**. The objective of link prediction in network analysis is to infer missing links or recommend new links in a network. For this step, your tasks include

   (a) Remove some of the links in the original network to create a new, modified network.

   (b) Use the modified network to train a logistic regression model.

   (c) Apply the logistic regression model to predict the links that were removed in part (a).

   As part of this task, you need to create a predictor matrix $\mathbf{X}$ and target class vector $\mathbf{y}$ for each node pair of a given input network. To do this, you will first extract a feature vector representation of each node in the network. The feature vector corresponds to the eigenvectors

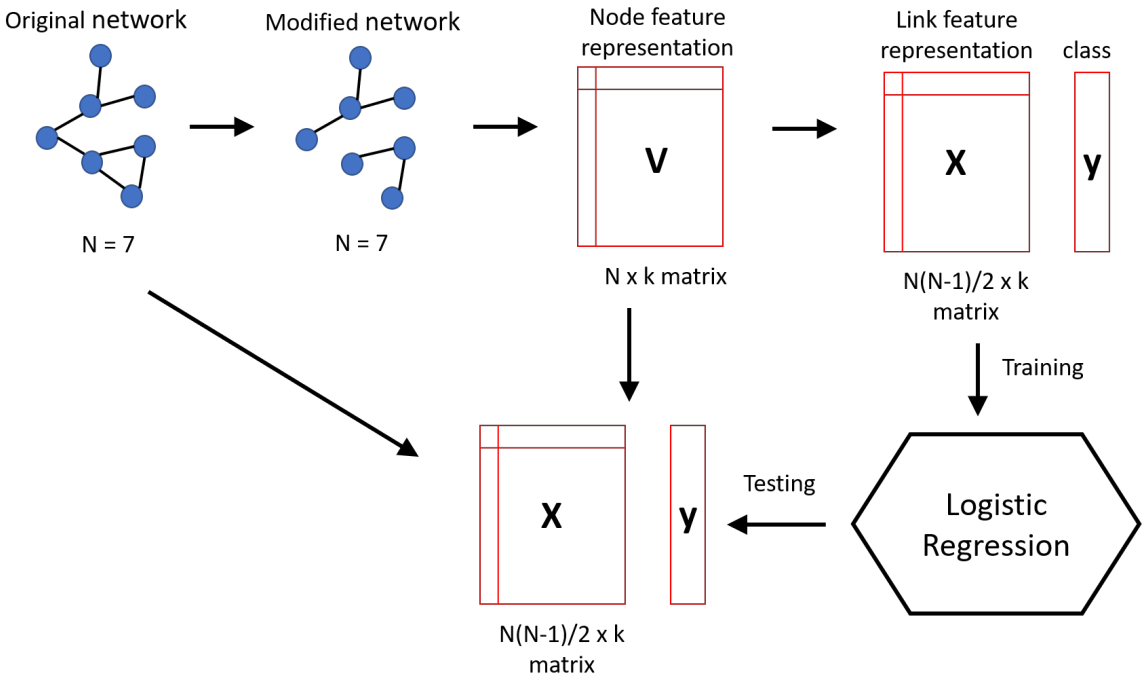

Figure 2: Steps for link prediction

of the adjacency matrix. Once you have extracted the feature vector of each node, you will compute the link feature vector for each node pair (i, j) by taking the absolute difference between their feature vectors, i.e.,

$$\mathbf{x}(i, j) = |\mathbf{v}_i - \mathbf{v}_j|$$

Note that $|\cdot|$ denotes element-by-element absolute difference between 2 vectors. For example, if node i has feature vector $\mathbf{v}_i = [1, 2, 3]$ and node j has feature vector $[2, 2, 1]$, the link feature vector for the node pair will be $\mathbf{x}(i, j) = [1, 0, 2]$. Figure 2 summarizes the steps you need to implement for link prediction.

2. **Community Detection**. The objective of community detection is to partition the network into a set of connected components (communities) such that nodes in the same community are more similar (connected) to each other compared to those belonging to different communities. For this project, you will implement a community detection algorithm that uses the following network modularity measure Q to determine the quality of the communities:

$$Q = \frac{1}{4m} \sum_{ij} \left(A_{ij} - \frac{d_i d_j}{2m} \right) c_i c_j,$$

where A is the adjacency matrix, d is a vector of the node degrees, m is the total number of links in the network (i.e., $\sum_{ij} A_{ij} = 2m$ for undirected network), and c is a vector that represents the community membership of each node. The factor $\frac{d_i d_j}{2m}$ represents the probability that a pair of nodes is connected to each other via random chance. Thus, a higher modularity value Q means the nodes in the same community are more connected to each other that what was expected by random chance.

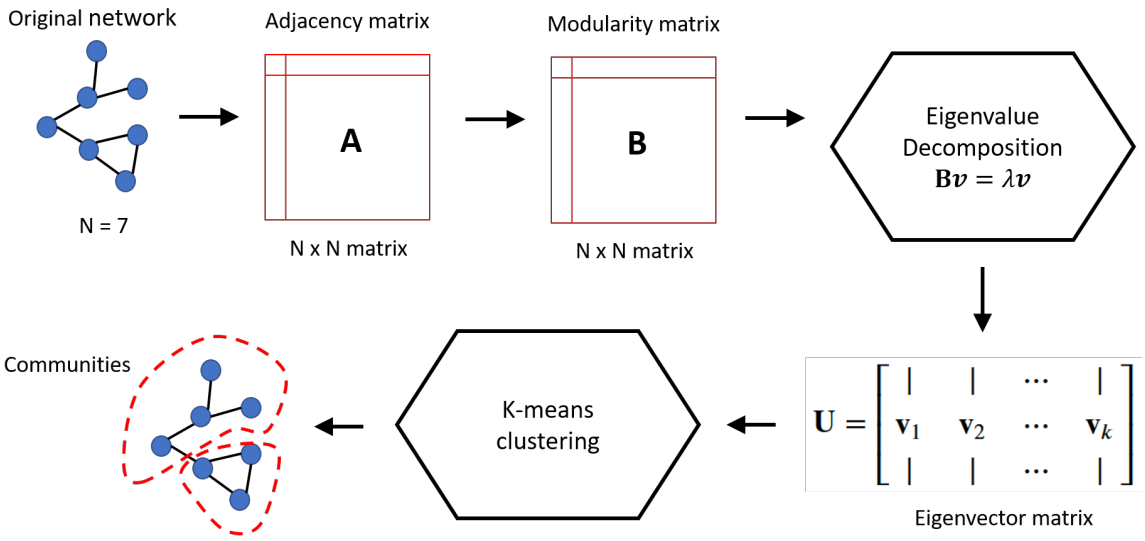

Figure 3: Steps for community detection

Thus, the community detection problem reduces to the following optimization problem:

$$\mathbf{c} = \arg\max_{\mathbf{c}} Q$$

For this formulation, each element in $\mathbf{c}$ is either +1 or -1 to indicate the cluster membership. Thus, the preceding formulation is applicable to generate 2 communities. To extend the formulation to k communities, $\mathbf{c}$ is an $N \times k$ matrix instead of an N-dimensional vector. Furthermore, the modularity function can be expressed in matrix notation as $Q = \frac{1}{4m}\mathbf{c}^T\mathbf{B}\mathbf{c}$, where $B_{ij} = A_{ij} - \frac{d_i d_j}{2m}$. It can be shown that the solution to the modularity maximization problem is equivalent to finding the top k eigenvectors of the matrix $\mathbf{B}$. Since the values of the eigenvectors are non-binary (even though $\mathbf{c}$ is supposed to be binary-valued), we can treat the eigenvectors of $\mathbf{B}$ as a feature representation of the nodes and then apply a subsequent algorithm, such as k-means clustering, to generate the final clusters. The steps that need to be implemented for the community detection task is summarized in Figure 3.

Deliverables: Submit your Jupyter notebook (`part2.ipynb`) along with its HTML version to D2L.