Assignment1 of CS 410/510: Mining & Learning on Graphs

Instructor: Yu Wang, **Website**: https://ml-graph.github.io/fall-2024/,

Due: 10/21/2024 23:59 PST

This homework is meant only for you to complete. In other words, you should not work with others in the course or seek help from others besides the instructor or TA. Additionally, if you use AI assistants, this is fine, but please acknowledge this somewhere in your submission (and know that you're taking full responsibility for your submission).

Please submit your solutions on Canvas. For questions 1-4, 6, and 7, please type your answers or hand-write (but it needs to be legible) and submit in pdf form. For question 5, 8, and 9 please submit either a Python notebook file that contains all the code and results (i.e., figures and calculated values), or submit a Python source file and another pdf document containing the results (i.e., figures and calculated values). Do not include the downloaded network dataset from question 5 in your submission.

There will be a deduction of 10 points if submitting other formats besides the ones mentioned above (e.g., submitting .docx instead of a pdf or .zip combining multiple files).

1. [15 points] Note that the local clustering coefficient is defined by the following:

$$C = \frac{\#\text{of links between neighbors}}{\#\text{maximum possible number of links between negihbors}} \tag{1}$$

(a) [10 points] Calculate the local clustering coefficient C for a node with degree k in an Erdos-Renyi Random Graph $G_{n,p}$.

(b) [5 points] Using the fact that the $< k >= p(n-1) \approx pn$, discuss why $C \to 0$ as $n \to \infty$ if we keep the mean degree $< k >$ fixed by using the result you obtained from 1(a).

2. [20 points] Note that in the $G_{n,p}$ Erdos-Renyi random graph model (which generates undirected graphs) we have that each edge exists with probability p. Thus, an edge (v_i, v_j) between two vertices v_i and v_j exists with probability p, and similarly for an edge (v_i, v_k) between v_i and v_k, etc. Further, remember that a triangle exists between three nodes v_i, v_j, and v_k, if and only if there exists three edges (v_i, v_j),(v_j, v_k), (v_k, v_i) in the graph.

(a) [10 points] Calculate the number of triangles that will exist in a graph generated by the Erdos-Renyi Random Graph Model $G_{n,p}$ in terms of n and $< k >$. Below is the general form of calculating the number of triangles:

$$\#\text{ of triangles} = \Delta \mathcal{S}_3 \tag{2}$$

where Δ = probability for any particular set of three nodes to form a triangle; $\mathcal{S}_3$ = number of possible sets of three nodes in a graph of n nodes.

(b) [10 points] Using the fact that the $< k >= p(n-1) \approx pn$, show that the number of triangles you calculated (from the result above in 2(a)) is approximately equal to the below as $n \to \infty$.

$$\#\text{ of triangles} = \text{your answer from 2(a) here} \approx \frac{1}{6} < k >^3 \tag{3}$$

3. [15 points] In the lecture "GraphModels-AdditionalModels", we discussed the expected number of common neighbors for the configuration model. More specifically, we discussed why it was equal to the following:

$$\sum_l \left(\frac{k_i k_l}{2m}\right)\left(\frac{k_j(k_l-1)}{2m}\right) \tag{4}$$

when assuming large number of edges m (i.e., having $p_{ij} = \frac{k_i k_j}{2(m-1)} \approx \frac{k_i k_j}{2m}$). We further noted that Eq.(2) can be simplified to the following:

$$\frac{\langle k^2\rangle - \langle k\rangle}{\langle k\rangle} p_{ij} \tag{5}$$

where we again note that $\langle k\rangle$ is the average degree of the nodes (i.e., $\frac{1}{n}\sum_i k_i$) and we use $\langle k^2\rangle$ to denote the average of the squared degrees (i.e., $\frac{1}{n}\sum_i k_i^2$).

Show that Eq.(2) equals Eq.(3). Note that you must show all necessary steps to establish this relationship.

4. [14 points] Remember that in the configuration model we were able to provide a degree sequence $D = \{k_1, k_2, \cdots, k_n\}$ representing the degree for each of the n nodes in the graph to be constructed. Assume that the degree sequence provided is the same for all nodes, i.e., $k_1 = k_2 = \ldots = k_n$ (and that the degree sequence is valid).

(a) [7 points] Describe what the degree distribution would look like for this graph.

(b) [7 points] What happens to the generated graph when $k_1 = k_2 = \ldots = k_n = 1$ (i.e., all nodes have degree equal to 1). If each node only has degree one, this means that they are all required to only have one edge connected to them, since having zero edges or more than one edge would result in a degree lower or higher than 1, respectively. Hence, in this setting, how many connected components will be in the generated graph? Note that there is an exact answer since all possible graphs that can be constructed with this degree sequence will have the same number of connected components.

Note: It is assumed the degree sequence $k_1 = k_2 = k_3 \cdots k_n$ is a valid degree sequence for the configuration model, since not all sequences are valid. For example, if $n = 1$, then we have $k_1 = 1$, but it is impossible to have a graph with a single node having degree one, since even if that node has a self-loop, its degree would be 2.

5. [35 points] Graph Modeling Programming Questions.

Please go through the Python NetworkX tutorial to familiarize yourself with the package. Write a program to generate the following types of graphs:

(a) [2 points] Erdos-Renyi $G_{n,p}$ graphs with $n = 1000$ and p equal to the following: 0.001, 0.005, 0.01 (i.e., your program will create 3 graphs here). Please see this API in NetworkX.

(b) [2 points] Barabasi-Albert preferential attachment model with $n = 1000$ and m equal to the following: 1, 2, 5 (i.e., your program will create 3 graphs here). Please see this API in NetworkX.

(c) [2 points] Watts-Strogatz small-world graphs with $n = 1000$, $k = 4$, and p equal to the following: 0, 0.1, 1 (i.e., your program will create 3 graphs here). Please see this API in NetworkX.

[5 points] Download a real social network from one of the two links below (note that the first link shows visuals of many network properties if you want to see them): link1, link2

This graph represents 59,835 private messages sent among a set of 1,899 students from the University of California, Irvine. When processing this dataset which is of the form source, destination, and time, you can ignore the time component and only use the source and destination links. Furthermore, if multiple messages were exchanged between the same pair of users, this can just be represented as a simple graph with binary edges (i.e., you do not need to consider weights here based on the number of messages they exchanged). This should collapse the 59,835 messages down to 20,296 edges.

Next, for each of the generated graphs and the downloaded real-world social network, please perform the following:

(a) [8 points] For each of these graphs plot the degree distribution similar to that found in the answer given by Dr. Brian Keegan in the below answer on StackOverflow: link3 Please note that you will need to make some adjustments as the NetworkX and/or Matplotlib packages have slightly changed.

(b) [8 points] Discover the number of components in the graphs and what percentage of the nodes are in the largest component.
Please see this API in NetworkX.

(c) [8 points] On the largest component in the graph (found in the above) compute the average shortest path length between all the nodes in that component.
Please see this API in NetworkX.

6. [10 points] **Please note here that you have read the below to receive the points for this question. You do not need to do any work for this question.**

We defined the Newman variant of the Katz centrality as:

$$c_i = \alpha \sum_j A_{ij} c_j + \beta_i \tag{6}$$

where α and β_i are positive constants. Note that for simplicity this can be rewritten as the Alpha-Centrality where we let the personalized vector $\boldsymbol{\beta} = \mathbf{1}$ (i.e, the vector of ones) as follows:

$$\mathbf{c} = \alpha \mathbf{A} \mathbf{c} + \mathbf{1} \tag{7}$$

(a) We mentioned in the lecture that the Alpha-Centrality of Eq. (7) can be rearranged to obtain the following:

$$\mathbf{c} = (\mathbf{I} - \alpha \mathbf{A})^{-1} \mathbf{1} \tag{8}$$

Please show the necessary steps on how to obtain Eq.(6) from Eq.(5).

Steps:

$$\begin{aligned}
\mathbf{c} &= \alpha \mathbf{A} \mathbf{c} + \mathbf{1} \\
\mathbf{c} - \alpha \mathbf{A} \mathbf{c} &= \mathbf{1} \\
\mathbf{I}\mathbf{c} - \alpha \mathbf{A} \mathbf{c} &= \mathbf{1} \quad \text{since } \; \mathbf{c} = \mathbf{I}\mathbf{c} \\
(\mathbf{I} - \alpha \mathbf{A})\mathbf{c} &= \mathbf{1} \quad \text{since } \; \mathbf{A}\mathbf{c} + \mathbf{B}\mathbf{c} = (\mathbf{A} + \mathbf{B})\mathbf{c} \\
(\mathbf{I} - \alpha \mathbf{A})^{-1}(\mathbf{I} - \alpha \mathbf{A})\mathbf{c} &= (\mathbf{I} - \alpha \mathbf{A})^{-1}\mathbf{1} \quad \text{sssuming the matrix is invertible in (b)} \\
\mathbf{c} &= (\mathbf{I} - \alpha \mathbf{A})^{-1}\mathbf{1}
\end{aligned}$$

(b) In Eq.(5) we can see that the parameter α controls the contribution of the eigenvector term and constant term. When choosing the value for α we must ensure that the value is not too large, but if we set $\alpha = 0$ then we only have the constant term remaining (i.e., all nodes would have the same centrality of 1). Hence, if we were to start with α from zero and increasing the value, we note that eventually this would cause the centrality values to diverge (which can be more easily seen from the perspective of Eq.(4)). This divergence happens when $(I - \alpha A)^{-1}$ diverges (i.e., $det(A - \alpha^{-1} I) = 0$ where det denoting the determinant of a matrix and based on the definition of matrix inverse). Note that this is the characteristic equation having roots α^{-1}, which means that $\lambda = \frac{1}{\alpha}$. Then, given we seek to the smallest value for α, this relates to finding the largest value for λ, which is the largest eigenvalue λ_1 of the matrix of A.

7. [30 points] We defined the PageRank matrix in the lecture as follows:

$$\mathbf{P}^{''} = \alpha\mathbf{P}^{'} + (1-\alpha)\frac{\mathbf{e}\mathbf{e}^T}{n} \tag{9}$$

where it was mentioned that Google utilized $\alpha = 0.85$.

(a) [15 points] Discuss what happens to the centrality values if we set $\alpha = 0$ and whether any of the three requirements for the Perron-Frobenius theorem are violated with this choice of α.

(b) [15 points] Discuss what happens to the centrality values if we set $\alpha = 1$ and whether any of the three requirements for the Perron-Frobenius theorem are violated with this choice of α.

8. [41 points] Centrality Time Complexity and Correlations. Using NetworkX, please calculate the centrality measures and record the running time for each of the methods on each of the graph types (e.g., using the Python time module). The first component is to evaluate the running times empirically. Note that if a (graph, centrality) pair below does not finish running in roughly 5 minutes on your machine then you can note $>$ 5min. The second part is to look at the correlation between the centrality measures.

(a) [2.5 points] Erdos-Renyi $G_{n,p}$ graphs with ($n = 500$, $p = 0.002$) and ($n = 5000$, $p = 0.0002$) (i.e., your program will create 2 graphs here). Please see API in NetworkX.

(b) [2.5 points] Barabasi-Albert preferential attachment model with ($n = 500$, $m = 2$) and ($n = 5000$, $m = 2$) (i.e., your program will create 2 graphs here). Please see API in NetworkX.

(c) [4 points] Closeness Centrality. Please see API in NetworkX.

(d) [4 points] Betweenness Centrality. Please see API in NetworkX.

(e) [4 points] Eigenvector Centrality. Please see API in NetworkX.

Using the SciPy stats calculate both the correlation coefficients comparing the following centrality scores.

(a) [4 points] Pearson correlation with Closeness-Betweenness Centralities on Barabasi-Albert network ($n = 5000$, $m = 2$)

(b) [4 points] Pearson correlation with Closeness-Eigenvector Centralities on Barabasi-Albert network ($n = 5000$, $m = 2$)

(c) [4 points] Pearson correlation with Betweenness-Eigenvector Centralities on Barabasi-Albert network ($n = 5000$, $m = 2$)

(d) [4 points] Spearman correlation with Closeness-Betweenness Centralities on Barabasi-Albert network ($n = 5000$, $m = 2$)

(e) [4 points] Spearman correlation with Closeness-Eigenvector Centralities on Barabasi-Albert network ($n = 5000$, $m = 2$)

(f) [4 points] Spearman correlation with Betweenness-Eigenvector Centralities on Barabasi-Albert network ($n = 5000$, $m = 2$)

Please see "scipy.stats.spearmanr" and "scipy.stats.pearsonr".

9. [20 points] Using NetworkX, write a program to calculate the graph-theoretic centrality measure CG we discussed in the first centrality lecture. The program should take as input a NetworkX graph, find the center of the graph, then for each node discover the distance to the center, finally calculate the values of CG(i) for each node i in the graph. To demonstrage the correctness of your program, please also execute your algorithm for the Karate Club graph found at “networkx.generators.social.karate club graph”. Note that for this question you can either choose to select a random center if there are more than one center node, or calculate the distance values for all and take the average, but should state in your submission which you decided to implement.