ECE 381K: Machine Learning on Real World Networks

HW2: Assigned 10/24/23, DUE 11/07/23 (midnight - 11:59:59pm CDT)

Problem 1 (25 pts)

In this problem, you will simulate an infection spreading through dynamic contact networks. In these networks, a node represents a person, while an edge represents close contact between two people. To this end, you will generate D = 30 scale-free[1] networks (each network corresponds to one day) of size N =1000 nodes and seed the infection by choosing m = 5 different Patient 0s on the first day $\boldsymbol{D_1}$ according to the descriptions in part 1a and 1b. You need to assume a closed population which means that all 1000 nodes will be present on each day (i.e., network). To simulate the infection, you apply the *Susceptible–Infected–Recovered* (SIR) model to the networks as defined below:

```
Step 1: Build $D = 30$ Scale-Free networks
Step 2: Choose $m = 5$ people to be 'infected' from the first network $D_1$, all other (N-m) nodes are 'susceptible'
Step 3: for $i = 1$ to $D$:
        For each 'infected' node
            If $i$ = this node's time_to_recover:                    // Infected-to-recovered
                Change this node's status to 'recovered'

            For each 'susceptible' neighbor:                        //Susceptible-to-infected
                Randomly assign the chance of infection spreading (between 0 and 1) to its neighbors
                If chance <= $P_{transmit\_virus}$
                    Set this node's status to 'infected'
                    Set this node's time_to_recover = $i + T_{RECOVER}$
```

With the following parameters:

Parameter	Description	Value
N	Number of nodes	1000
D	Simulation length (number of days)	30
$P_{transmit\_virus}$	Probability of transmitting a virus	.015
$T_{RECOVER}$	Days it takes for an infected node to recover	7

You will be provided with skeleton code written in Python to help you get started.

a) Seed the infection with the top 5 nodes that have the *highest degree* in $\boldsymbol{D_1}$'s network, then simulate the infection, and finally plot the number of infected nodes (*#Nodes) vs time (Day)*.

b) Now seed the infection with 5 *random nodes* from $\boldsymbol{D_1}$'s network and plot the number of infected nodes per day. (Preferably, we want to see both infection curves from a) and b) on the same graph.) Comment on the differences between these two infection curves? At the end of the simulation, how many people within the N=1000 population get infected for 1a and 1b? Why do these infection curves look different? Explain.

c) Now propose and implement an infection mitigation strategy of your choice. Using both set ups from parts a) *and* b) above deploy your mitigation strategy and plot the number of infected nodes vs. day in both cases. How does your mitigation technique hold up against 1a) and 1b) outbreaks?

[1] Either use your SF implementation from HW1 or use nx.barabasi-albert-graph from NetworkX. More instructions in the skeleton code.

Problem 2 (30 pts)

The goal of this problem is to explore the LinkedIn ego-network structure using provided data. The data contains 250 ego-networks, which have user attributes and their relationships (with types). Each student in this class has a randomly assigned ego-network. Please check Canvas for your User ID. We define the ***ego-user*** to be the person (*i.e.*, node) whose ego-network is analyzed. Each ego-network dataset comes with the following information:

2.a) Complete edge list for the ego-network (*network.txt*)

Example:

U0 U0 U1

U0 U0 U2

…

U1046 U1046 U1172

U1046 U1046 U1173

Here, a record "U0 U0 U1" means that U0 is connected to U1 in U0's ego-network.

2.b) Relationships between ego users' and their friends (*label.txt*)

Example:

U3297 U3299 L4

U10305 U10351 L2

U10305 U10369 L2

Here, the record "U10305 U10369 L2" means that the type of relationship between U10305 and U10369 is L2. Of note, there can possibly exist 8 types of relationships (total) in your ego-network.

2.c) Profile attributes (location, college, employer) for the users

Example: (college.txt)

U346

2

C0

C1

Here, U346 is UserID, 2 is the number of values associated with the attribute, e.g., U346 attended two colleges (C0 and C1). *employer.txt* and *location.txt* follow the same format.

For more details on the dataset, please refer to PDF: https://arxiv.org/pdf/1309.4157.pdf.

In this problem, we want you to perform an in-depth analysis of your assigned ego-network. The following steps will allow you to understand the structure of ego-network in detail.

2.1 Preliminary analysis using Gephi

a. **Visualization**. Visualize your data using at least two layout techniques. Choose the edges to be *directed*. Color the ego-user in your network and the edges connecting to it in *red*. Report the number of nodes and edges in your ego-network. Attach screenshots of your visualizations to your writeup.

b. **Network analysis**. Report degree distribution, node degree, clustering coefficient, average path length, betweenness centrality distribution, and number of communities.

2.2 One step further

a. **Community structure**. Use the modularity-based community detection embedded in Gephi to detect the communities in your ego-network. Visualize and color communities using different colors.

b. **Clustering**. Explore the type of relationships among users in your assigned ego-network. Group users together if their edge type (relationship) is the same. For instance, in example 2.b) in the beginning of this problem, U10305, U10351 and U10369 should belong to the same group L2, while U3297 and U3299 should belong to another group L4. We call these groups the *ground truth circles.*[2] Compute the average clustering coefficient for each ground truth circle in your network and compare it to the average clustering coefficient of the entire ego-network. Do you see any significant differences?

c. **Homophily**[3]. Explore various features (*e.g.*, college, employer, location) of users in different ground truth circles. Use entropy[4] to measure the homophily of each circle under different features, and then compare it with the average clustering coefficient of each circle. Report your results in a table as shown below. Replace '*' with the value of clustering coefficients or entropy measured homophily; fill in the corresponding row(s) as 'NA' if your ego-network does not have certain circle(s).

	clustering coefficient	*Homo college*	*Homo employer*	*Homo location*
circle 1	$*$	$*$	$*$	$*$
circle 2	$*$	$*$	$*$	$*$
circle 3	NA	NA	NA	NA
circle 4	$*$	$*$	$*$	$*$
circle 5	$*$	$*$	$*$	$*$
circle 6	$*$	$*$	$*$	$*$
circle 7	$*$	$*$	$*$	$*$
circle 8	$*$	$*$	$*$	$*$

2.3 Getting even deeper

a. How homogeneous are communities in your ego-network? Do they tend to share similar characteristics? Do similar people belong to the same communities?

b. How interconnected is your LinkedIn ego-network? Is there a high degree of overlap between communities in your ego-network?

[2] Note that although there may exist 8 types of relationship among users, some type of relationships may be missing in your assigned ego-network, and some edges may not be labeled as well, so just work on whatever is available for your ego-network.

[3] Homophily (i.e., "love of the same") is the tendency of individuals to associate and bond with similar individuals.

[4] Entropy is defined as $-\sum p(x_i)\log p(x_i)$, where $p(x_i)$ is the probability of feature x_i belonging to users in that circle.

Problem 3 (10 pts)

In this problem, you need to analyze the *community structure* in research collaboration. Towards this end, you need to follow a few steps: First, you need to download the *arXiv* collaboration network used in Homework 1 (*i.e., arXiv_lcc.gml*). Second, you need to use the community detection algorithm provided in Gephi (https://gephi.org/) to identify the communities in this network.

You need to supply a wide range of resolutions (a user-specified parameter) to obtain the resulting modularity and number of communities. Note that, a lower *resolution* in Gephi's community detection algorithm corresponds to a higher number of communities.

a) Plot the modularity as a function of number of communities. Of note, you should report a substantial number of points in the plot.

b) Using modularity resolutions of 0.1 and 0.75, visualize this network while coloring each node according to the community it belongs to. Based on these visualizations, does the result of the community detection algorithm make intuitive sense? Why or why not?

Problem 4 (15 pts)

In this problem, we analyze the connectivity of a concrete transportation system, more specifically the Light Rail and bus routes of Pittsburgh, PA. You will find a transportation network 'bus_stops.gexf' and CSV files on Canvas that have the station information for both lines of the Light Rail, and the stops for the 28X bus line. The entries are listed in the sequential order of stops; also, provided are maps for reference.

a) Visualize the transportation network, referencing the 'Station Name' field for node identification. Color the nodes according to the 'blue line' (blue), 'red line' (red), and '28xline'(black).

b) Analyze the 'small worldness' of the network you obtain by calculating the average path length, nodes degree, and clustering coefficient.

c) Insert several long-range links (but do not connect neighboring nodes) to create 'triangles' within the network in order to reduce the path length by at least 30%. Comment on your results and explain them within the context of 'small world' networks.

Problem 5 (20 pts)

In this problem, we explore the possibility of using node embeddings to identify communities from a citation network. More specifically, we investigate whether node embeddings are sufficient to identify the paper category from citations alone. You need to familiarize yourself with the Deep Graph Library (DGL) API (install here: https://www.dgl.ai/pages/start.html) to analyze citations within the scientific research community. In this Pubmed Graph Dataset, a node represents a scientific paper, while an edge represents a paper citing another paper.

(a) Download the DGL library and perform *node2vec* on the Pubmed citation network using the **dgl.sampling.node2vec_random_walk**. What *p* and *q* values do you use? How would you go about tuning the *p* and *q* parameters?

(b) Deploy t-SNE from scikit-learn to reduce the dimensionality into 2D, and then plot the embeddings. Color the embeddings according to the class labels.

(c) Do these embeddings clearly separate the labeled classes? Explain in words how you could improve interpretability of the features.

(d) Use a GNN architecture of your choice to train a classifier and report the accuracy you obtain.

Note on submission:

For all problems in this homework, everything is handled electronically. Prepare the answers using either Word or Latex and ***create a single PDF file for your submission***. Also, put the source code and all related files for each problem (as needed) in a separate folder. Please include all your answers in the write up; we will not be opening your Jupyter notebooks to find the answer. Name your file with *'first_name_last_name_hw2_write_up.pdf'*.

Make sure you have labels for *all* axes on your graphs and plots; also, make sure they are at the appropriate scale.

Finally, compress everything and name it as *"yourEID_hw2.zip"*, and deposit it on the Canvas under Assignments > HW2 Submission. In this zip file, include the write-up describing your solution and organize all the relevant files to each problem under a separate folder labelled suggestively Problem 1, Problem 2, etc.

Work *individually* on all problems.

Good luck!