HW1 – Networks

TA in charge – Sagitt Efrat

Due date 14/05/19

In this exercise you will practice some of the network concepts learned in class. You will receive a dataset that represents a social network such that users rate the level of trust they have in other users, upon exchanging goods with each other. 3,072 nodes, 18,102 edges

The rate is on a scale of -10 to +10 (excluding 0). A rating of -10 considered fraudsters while +10 means "you trust the person as you trust yourself".

You will create a directed graph where each directed edge represents a rating of a transaction between two edges. Edges and nodes can be added when time passes. We will name this directed graph – G. The dataset is attached and can be found on the Moodle under the name 'data_students'. There are 4 columns:

- source: unique Id of the rater
- target: unique id of the rated
- rating: the opinion source gave on target in [–10, 10] scale
- time: time the rating was made in unix-epoch

We will define a weighted directed graph G_t as: All edges from G that were created until time t (including).

We will construct an undirected weighted graph H_t from G_t where:

- There is a weak edge {x,y} at time t if at there was at least one rating from x to y or from y to x in G.
- There is a strong edge {x,y} at time t is there was a bidirectional rating between x and y in G.

Part A (30 points)

1. Compute the distribution of nodes degree in the graph G.
    a. Create a histogram as in fig. 18.2 in the book, of the out-degree for each node. On the same graph, plot the Power Law distribution with the most similar parameter a and c (using the notation of Section 18.2 in the book).
    b. Is there a bow-tie structure in the graph? Explain

For question 1 attach the histogram and write the answers in your pdf file submission.

2. Compute for every node in G_t:
    a. Closeness centrality
    b. Betweenness centrality

Write a function G_features(int time) that receives time t, creates G_t for that time (ignore edge weights) and returns for every node in G_t, parts 2.a-2.b in a dictionary of dictionary format. You should follow the format in the file hw1_part1.py.

See example in the appendix.

You can use *networkx* library in order to find shortest distances, however not to measure centrality directly. Compute the features only for the nodes in the largest (Strongly) CC .

Part B (60 points)

In this part, we will test some methods for completion of edges, and compare them using Precision and Recall models.

Split 'data_students' to:

1. train – contains 3 columns: source, target, rating with a matching header (see example)
2. test – contains 3 columns: source, target, rating with a matching header

The train should contain all edges added before (including) certain time t (test should be the complementary – after time t)
All the learning process will be done only on the train while the predictions will be on both, in order to measure the test and train error.
To measure the training error, make predictions on train [(source, target)] and compare them with train [rating].
To measure the test error, make predictions on the test [(source, target)] and compare them with test [rating].

Create graph G from train_set.

3. Unweighted Undirected Graph. Create graph H_t, ignore weights on edges. Predict the rest of edges using the "soft" triadic closure property:
   Any open triangle will be closed with probability of exactly p=0.03 between time period (t, t+1). Thus, if nodes x, y, are disconnected and have k neighbors in common at time t, the probability they will be connected at time t+1 is $1-(1-0.03)^k$. Run the algorithm N time periods. Return a list of lists of the added edges (both (x,y) and (y,x)). Compare the results with the test set using Precision and Recall models.

4. Weighted Undirected Graph. Create graph H_t. Predict the rest of edges using the "soft strong" triadic closure property: If nodes x, y, are disconnected and have k neighbors:
   - if the i^{th} common neighbor is weak neigh of x and strong neighbor of y (wlog) then the probability an edge will be constructed is p=0.02
   - if the i^{th} common neighbor is strong neigh of x and strong neighbor of y then the probability an edge will be constructed is 2*p

   The new edge connection will be weak

   Run the algorithm N time periods. Compare the results with test set using Precision and Recall models
5. Directed Graph. Predict the rest of directed edges in G_t using the "soft directed" triadic closure property: Any edge will be created between time period (t, t+1) depending on the number of shortest paths between x,y that are smaller equal to 4. Thus:
   Let L be the distance d(x,y) (length of the shortest directed path).
   If L>4 no edge is created. Otherwise, let m be the number of x-y paths of length exactly L. Create an edge with probability min{1, m/(5^L)}
   Run the algorithm N time periods. Compare the results with test set using Precision and Recall models.

Your code will be tested for different times t (that actually exists in the given G graph). You can assume part A will be tested on the same time t for each pair of students, so you can compute them all at once. Same for part B.

Part A should finish run in 5 minutes. Part B should finish run in 30 minutes (for k=6 and time~1306021600). You should run your code on your VM and check that it run well, and within the time limit.

Part C - competitive part (15 points).

In this part you should try to predict the network spreading by a certain time, Heuristically/by principles learned in class.

Create a CSV file that holds rows of weighted edges you think will be added to the directed graph by the time = 1453438800, submit a csv file with the name 'hw1_part2.csv'. We will compare this list to the actual list of next edges weight in the graph with same structure as 'data_students.txt'. If you want to consider a new user that will be added call it '9876543'. The rating of a non-existing edge will be considered 1. The predictions accuracy will be estimated by RMSE.

Submission instructions:

Attach one **zip** file (not rar or tar or gz or you got it..) that carries the name hw1_studentID1_studentID2.zip for only one submitter than the name should be hw1_studentID1.zip (fill in your own ID's).

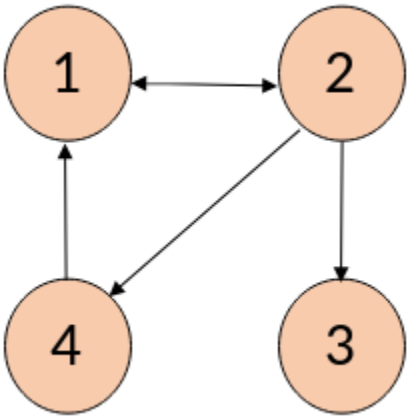
the zip file should include:

1. hw1_part1.py
2. hw1_part2.csv
3. hw1_studentID1_studentID2.pdf
4. data_students.txt (as attached in Moodle, make no changes in this file)

Make sure there are no sub folders in the zip file. Make sure your file name does not have the name hw1_studentID1_studentID2.zip.zip (funny but happens a lot).

Only one student in a group should submit

We will use similar(but not the same) Main.py file in order to test your code.



for the graph above :

{‘a':{1: 0.667, 2: 0.444, 3: 0.5, 4: 0.444}, ’b’: {1: 0.333, 2: 0.5, 3: 0.0, 4: 0.0} }