Exercise 1

Due date: 26/04/2021, submission in pairs

Appointed TA: Tsviel Ben Shabat

Purpose

The first exercise aims to experience network analysis, stochastic and deterministic network contagion models as taught in class.

Disclaimer: Any information mentioned in this exercise is purely for educational purposes.

Recall Exercise 0 is 5% of this exercise.

Data files

In this exercise, you will explore three different data sets:

1. PartA1.csv/PartA2.csv – Two datasets representing a similar social network from different countries. **You will use them in part A of this exercise** (hence their names). Each row represents an **undirected** edge (connection) between two persons, represented by their ids.

from	To
6194	255
6194	980
⋮	⋮
7069	7110
7075	7109

2. PartB-C.csv This dataset represents a mutual following relation in "Deezer," a music platform similar to Spotify. **You will use this data set for Parts B and C.** Each row represents an undirected edge between two users and weight $\in [0,1]$, representing the edge's strength.

from	to	W
0	14270	0.410088
0	16976	0.270293
⋮	⋮	⋮
28194	28246	0.686496
28212	28259	0.760566

Code file

The only code file is main.py; **Do not import any external** (that need to be downloaded and installed) **python packages** except Pandas, NumPy, matplotlib, seaborn, networkx. **The code will be tested automatically on the VM environment of HW0 with 10GB RAM and 4 CPUs**. Hence, unauthorized packages will cause **automatic failure**. You may add any auxiliary functions of your own to main.py.

GOOD LUCK!

Part A – Graph Stats (30%)

In the first part of the exercise, you will investigate two different social networks and set the ground for Parts B and C.

We first start by representing a graph in python by using the excellent and straightforward networkx package.

1. Complete the function build_graph to handle both weighted and unweighted graphs.
   ```
   def build_graph(filename: str) -> networkx.Graph
   ```
   The function inputs a string – the name of the data set, e.g., "PartB-C.csv" and returns a networx.Graph object will represent the nodes, the edges, and weights (when they exist) as they are at the data set.

Next, we wish to get a feel for the graph; visualizing thousands of nodes and edges is not always informative. Instead, we can visualize the histogram nodes degree, i.e., we count how many nodes have only one neighbor, how many nodes have two neighbors, and so on.

2. Complete the function calc_degree_histogram
   ```
   def calc_degree_histogram(graph: networkx.Graph) -> Dict:
   ```
   The function inputs a networkx.Graph object and returns a dictionary representation of the nodes' degree histogram, i.e, if histogram[1] = 10 then there are 10 nodes whose degree = 1.

3. Complete the function plot_degree_histogram
   ```
   def plot_degree_histogram(histogram: Dict)
   ```
   The function inputs a dictionary representation of the degree histogram object and plots it.

4. Use build graph and plot_degree_histogram to plot the data from PartA1.csv, PartA2.csv and, PartB-C.csv. Attach the plots to the PDF file of the submission (see submission instructions).

As briefly explained above, PartA1.csv, PartA2.csv are graphs from the same but separated social network. One network is for Portuguese and the other for British, so the question is which is which?

5. **In this question, you are not allowed to use the networkx clustering coefficient calculation function, you have to implement the following formula by yourselves**.
   Complete the function clustering_coefficient, use the definition from the "networks basics" lecture.
   ```
   def clustering_coefficient(graph: networkx.Graph) -> float:
   ```
   The function inputs a graph and returns the clustering coefficient as explained in the lecture:

$$CC(G) = \frac{3 \times |\{\text{triangles}\}|}{|\{\text{connected triplets}\}|}$$

$$= \frac{3 \times |\{\text{triangles}\}|}{3 \times |\{\text{triangles}\}| + |\{\text{open triangles}\}|}$$

6. Use the results from 3 and 4 to try to guess which of PartA1.csv/PartA2.csv is from Portugal, add your answer to PDF**. Any well-reasoned answer will get full score.**

Part B – Diffusion models (55%)

In this part, we will practice the Linear threshold and the Independent cascade models as taught in class, plus some minor additions.

Background story

More than 2.7 million deaths and 125 million cases worldwide were brought to the world by the Covid-19 pandemic. Many experts warn that the new globalized world is very vulnerable to such contagious diseases. Although Covid-19 is far from over, we must prepare for the next inevitable pandemic called: "pandemic X."

The Israeli minister of health appointed you to study and simulate different Diffusion models in order to understand the consequences of "pandemic X," taking into account the following variables:

Variables:

Contagion (static) – A real number that describes how contagious the disease is.
Lethality (static) – The probability for an infected person to die from the disease.
Concern (dynamic) – How worried is a particular person about the disease. The concern will cause a person to alter their behavior and reduce their chance of being infected.

To formalize our notation, we also define the following sets:

Sets:

V – The set of all people (**V**ertices in our graph)
E – The set of all undirected relations between two people (**E**dges in our graph)
I^0 – The set of initial patients $I^0 \subset V$
S^t – The set of **S**usceptible people at the end of iteration t (who are not infected), $S^0 = V \setminus I^0$
I^t – The set of **I**nfected and alive people, at the end of iteration t
R^t – The set of **R**emoved (infected and deceased) people at the end of iteration t

Pay attention that S^t, I^t, R^t are disjoint, that is, $S^t \cap I^t = S^t \cap R^t = I^t \cap R^t = \phi \; \forall t$

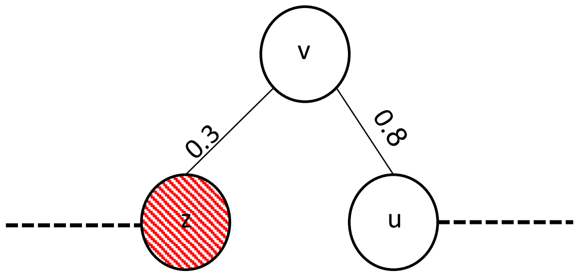
Linear threshold model (deterministic, without using Lethality and R)

- Each edge $(u,v) \in E$ has a weight $w_{uv} \in [0,1]$ that represents the relationship strength of persons u and v, which implies their exposure to each other
- Each person v has a concern variable $c_v^t = \frac{|\{u:(u,v)\in E\}\cap I^{t-1}|}{|\{u:(u,v)\in E\}|}$ *
- In every step t = 1,2,…:
    I. Every v $\in S^{t-1}$ watches all of her infected neighbors $cn_v = \{u:(u,\ v) \in E\} \cap I^{t-1}$
        - If $Contagion * \sum_{u:\in cn_v} w_{u,v} \geq 1 + c_v^{t-1}$ then v is in I^t
    II. $S^t = V \setminus I^t$
    III. Every v$\in S^t$ updates her concern c_v^t

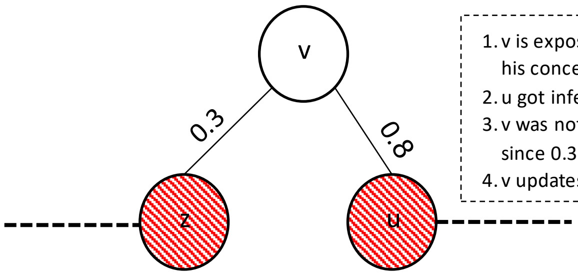
* Pay attention that c_v^t depends on I^{t-1} the reason for it is to simulate the case where a person is first exposed to her infected friend and only afterwards aware that her friend was infected.

Example Scenario:

$t = t^0\ , c_v^0 = 0, z \in NI^0$



$t = t^0 + 1\ , c_v^1 = 0.5, u \in NI^1$

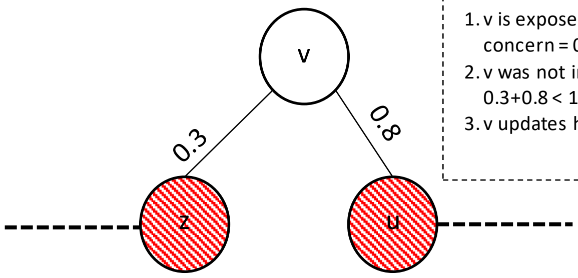


1. v is exposed to z while his concern = 0
2. u got infected
3. v was not infected since 0.3 < 1
4. v updates his concern

$$CONTAGION\ * w_{v,z}\ \leq 1 + c_v^0$$

$$(1\ * 0.3 \leq 1 + 0)$$

$t = t^0 + 2\ , c_v^2 = 1$

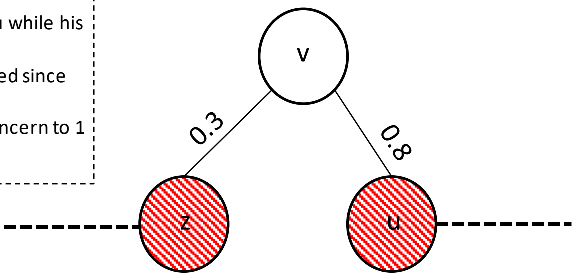


1. v is exposed to u while his concern = 0.5
2. v was not infected since 0.3+0.8 < 1 + 0.5
3. v updates his concern to 1

$$CONTAGION\ * w_{v,z}\ \leq 1 + c_v^1$$

$$1\ * (0.3 + 0.8) \leq 1 + 0.5)$$

$t = t^0 + 3\ , c_v^3 = 1$



1. Complete the function LTM (Linear threshold model),

```
def LTM(graph: networkx.Graph, patients_0: List, t: int) -> Set
```

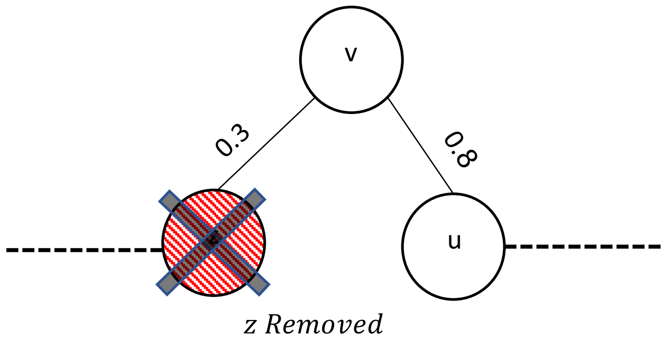
The function inputs a netwokx.Graph object, a set of infected persons, and the number of iterations. It returns the set of infected persons after t iterations of the Linear threshold model. Consider using node attributes.

Independent cascade model (stochastic)

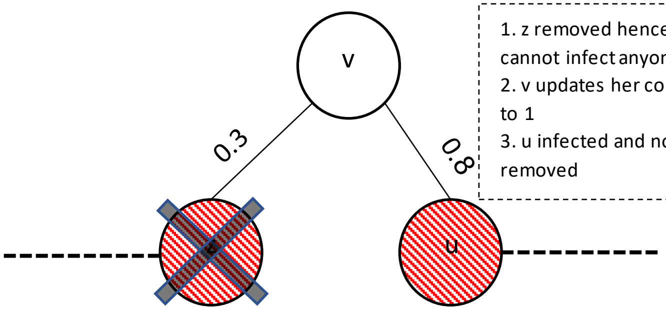
- Each edge $(u,v) \in E$ has a weight $w_{uv} \in [0,1]$ such that if person u is infected, **then $\boldsymbol{w_{uv}}$ will affect $\boldsymbol{P_{u,v}^{t}}$** Which is defined below.
- If a person is infected, they are removed with probability Lethality **without infecting anyone else**.
- Each person v has a concern variable, $c_v^t = \min\left(\frac{|\{u:\,(u,v)\in E\}\cap I^{t-1}| + 3*|\{u:\,(u,v)\in E\}\cap R^{t-1}|}{|\{u:\,(u,v)\in E\}|}, 1\right)$ **
- Denote NI^t are the **new** nodes who were infected during iteration t
- Denote $P_{u,v}^t$ as the probability for person u to pass the virus to person v during iteration t, then: $P_{u,v}^t = \min\left(1, Contagion * w_{u,v} * \left(1 - \boldsymbol{c_v^{t-1}}\right)\right)$
- **On $\boldsymbol{t_0}$ patients also die w.p Lethality**
- In every step t = 1,2,…:
  - I. For each $v \in S^{t-1}$ and $u \in NI^{t-1}$ u infects v w.p $P_{u,v}^t$
    - ▪ If infection occurs, v is added to NI^t
  - II. Every v $\in NI^t$ is added to R^t with probability Lethality
  - III. $R^t = R^t \cup R^{t-1}$
  - IV. $I^t = (I^{t-1} \cup NI^t) \setminus R^t$
  - V. $S^t = V \setminus (I^t \cup R^t)$
  - VI. Each $v \in S^t$ updates concern c_v^t

Example Scenario:

$t = t^0\,, c_v^0 = 0$

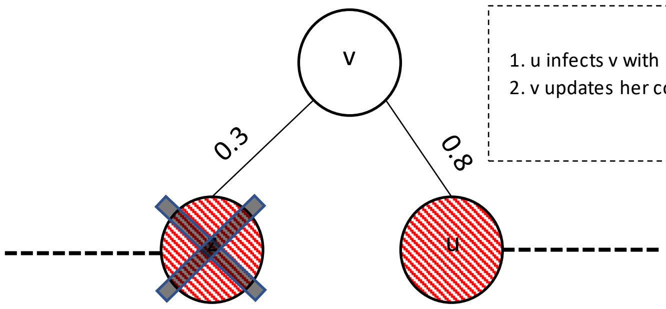


$t = t^0 + 1\,, c_v^1 = 1, u \in I^t$



1. z removed hence cannot infect anyone else
2. v updates her concern to 1
3. u infected and not removed

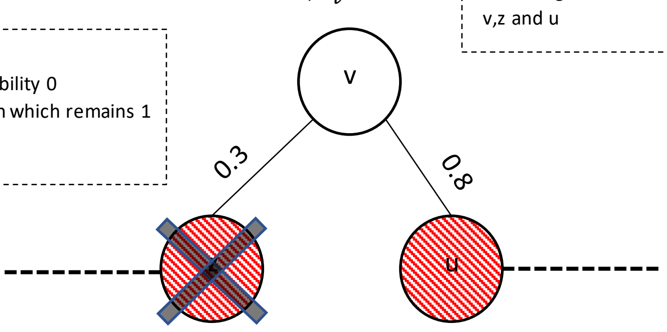
$t = t^0 + 2\,, c_v^2 = 1$



1. u infects v with probability 0
2. v updates her concern which remains 1

$P_{u,v}^1 = min\left(1,\ CONTAGION * w_{z,v} * \left(1 - c_v^1\right)\right) = 0$

$t = t^0 + 3\,, c_v^3 = 1$



No changes can occur to v,z and u

** Pay attention that c_v^t depends on $\boldsymbol{I^{t-1}}, \boldsymbol{R^{t-1}}$ the reason for it is to simulate the case where a person is first exposed to her infected friend or removed and only afterwards aware that her friend was infected or removed.

2. Complete the function ICM (Independent cascade model),

   ```
   def ICM(graph: networkx.Graph, patients_0: List, t: int) -> [Set, Set]
   ```

   The function inputs a networkx.Graph object, a set of infected persons, and the number of iterations. It returns the set of infected persons **and the deceased (Removed)** after t iterations of the Independent cascade model.

Next, we wish to see how a change in Lethality affects the mean number of infected and deceased (Removed).

3. Complete the function compute_lethality_effect

   ```
   def compute_lethality_effect(graph: networkx.Graph, t: int) -> [Dict, Dict]
   ```

   The function inputs a networkx.Graph and runs the following:

   ```
   for each Lethality value in [0.05, 0.15, 0.3, 0.5, 0.7]:
       do 30 times:
           • Choose 50 random patients_0
           • Run ICM for the chosen random patients_0 for t iterations
   ```

   The function returns two dictionaries, one for the mean infected number and the second, for the mean number of deceased.

   For example, if the function returns the dictionaries mean_deaths and mean_infected then, mean_infected[0.15] = 4500 means that for a Lethality value of 0.15, the mean number of infected people after t iterations, starting with random 50 patients_0 is 4500.

4. Complete the function plot_lethality_effect

   ```
   def plot_lethality_effect(mean_deaths: Dict, mean_infected: Dict)
   ```

   The function's input is the dictionaries from B.3 and plots two lines such that the x-axis is the Lethality value and the y-axis are the mean number of deaths or the mean number of infected.

5. Use compute_lethality_effect and plot_lethality_effect to plot the mean_deaths, mean_infected with t=6, and explain the result. Attach the plot and the explanation to the submission PDF.

Part C – Competition (10%)

In this part, the goal is to investigate the graph's structure and use it to change the mean outcome using constrained resources.

Background story

Just before "pandemic X" erupted in Israel, in a secret underground lab inside the Carmel Tunnels, 50 doses of 100% successful single-shot vaccine are ready; unfortunately, due to heavy rain, the lab is flooded, which stops the vaccine production for six days.

The Israeli minister of health appointed you to choose the first lucky 50 people to get the vaccine in order to reduce the total infected and deaths for the next six days.

The information arrived from scientists around the world indicates that for "pandemic X":

Contagion = 0.8

Lethality = 0.15

1. Choose 50 people (nodes) whose removal from the graph would minimize the mean total infected and deaths after six iterations of the ICM model.
    a. Complete the function choose_who_to_vaccinate
    ```
    choose_who_to_vaccinate(graph: networkx.Graph) -> List
    ```

    The function's input is a networkx.Graph, and returns the 50 people whose removal would minimize the mean total infected and deaths after six iterations of the ICM model.

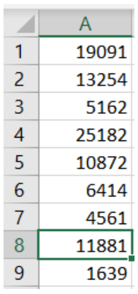
    To test your selection, we will run it repeatedly with different 50 random patients 0. ***Each submission will be tested on different large subsets of the graph.***

2. Briefly explain the criteria of your choice, add them to the submission PDF.
3. **The maximum allowed run time for six iterations is 1 min. Surpassing it will cause automatic failure.**

Submission instructions:

A zip file: StudentID1_StudentID2_HW1.zip which contains:

1. main.py
2. HW1.pdf for A.3, A.5, B.5, C2
3. vaccinated.csv as the answer to C.1, the table should be a single column with the chosen patients' ids. For example, for a choice of nine vaccinations, the fie will look like this:



	A
1	19091
2	13254
3	5162
4	25182
5	10872
6	6414
7	4561
8	11881
9	1639

For any questions regarding exercise 1, please post in the designated forum on the moodle.

Appendix A – Inputs and outputs for LTM and ICM of Part B

- Datafiles: PartB-C.csv as the graph data and patients0.csv as the initial patients for tests.
- To simplify the input/output tests, we will only present the size of the resulted sets.

LTM

Notation: LTM(patients0=patients0[:25], t=6) -> 465, 10sec means:

I. Input: the first 25 patients0 from patients0.csv and running 6 iterations.
II. Result: in $|I^6|$ =465, and runtime should take about 10 seconds.

Contagion = 1

LTM(patients0=patients0[:50], t=6) -> 856, 10sec

LTM(patients0=patients0[:48], t=6) -> 854, 7sec

LTM(patients0=patients0[:30], t=6) -> 30, 5sec

Contagion = 1.05

LTM(patients0=patients0[:30], t=6) -> 4549, 50sec

LTM(patients0=patients0[:20], t=6) -> 2035, 40sec

ICM

Notation: ICM(patients0=patients0[:25], t=6) -> 465, 40, 10sec means:

I. Input: the first 25 patients0 from patients0.csv and running 6 iterations.
II. Result: in $\mathrm{mean}(|I^6|) = 465, mean(|R^6|) = 40$, **mean is calculated over 30 runs,** and run time of each run should take about 10 seconds.

Contagion = 0.8, Lethality = .2

ICM(patients0=patients0[:50], t=6) -> 11918, 1971, 4sec

ICM(patients0=patients0[:20], t=4) -> 2835, 202, 3sec

Appendix B – Toy example for Part C

The following heuristic for Part C is simply taking the top 50 friendly people; that is, it returns the top 50 nodes in the graph with the highest degree.

```
def choose_who_to_vaccinate_example(graph: networkx.Graph) -> List:
    node2degree = dict(graph.degree)
    sorted_nodes = sorted(node2degree.items(), key=lambda item: item[1],
reverse=True)[:50]
    people_to_vaccinate = [node[0] for node in sorted_nodes]
    return people_to_vaccinate
```