PROJECT: VIRUS PROPAGATION

Overview: A fundamental problem in epidemiology is determining whether a virus will result in an epidemic (that is, if it will rapidly spread to many people) or if it will die quickly. Recent works on **virus propagation** have approached this problem from a network perspective. Given a network of individuals, where the edges represent who can potentially infect whom, the rate of propagation of a virus across the network will depend on the connectivity of the network and on the propagation model of the virus. More precisely, researchers have determined that the strength of the virus can be defined as a function of the largest eigenvalue of the network's adjacency matrix. These results have important implications for the design and evaluation of immunization policies, and can be applied to many problems, such as minimizing the spread of malware in computer networks or maximizing the spread of a marketing campaign across a social network.

Goal: Analyze the propagation of a virus in a network and prevent a network-wide epidemic. In order to do that, your team will need to:

1. Analyze and understand a virus propagation model.
2. Calculate the effective strength of a virus.
3. Simulate the propagation of a virus in a network.
4. Implement immunization policies to prevent a virus from spreading across a network.

Project Teams: For this project, you may work in teams of up to 3 students. You may choose your own teammates.

Input: You will be provided with the following materials in advance:

- Supplementary material on virus propagation.
- Parameter values for experiments:
  - Transmission probabilities $\beta_1 = 0.20$ and $\beta_2 = 0.01$.
  - Healing probabilities $\delta_1 = 0.70$ and $\delta_2 = 0.60$.
  - Number of available vaccines $k_1 = 200$.
- Static contact network (i.e., one undirected unweighted graph) for Option 1:
  - *static.network*
- Set of 2 alternating contact networks (i.e., two undirected unweighted graphs with the same nodes and different edges) for Option 2:
  - *alternating1.network*
  - *alternating2.network*

Output: Report and corresponding codes (see details below).

Project Details

You are only required to complete one of the two parts of the project (Option 1 or Option 2).

Option 1: Virus Propagation on Static Networks

1. For the SIS (susceptible, infected, susceptible) Virus Propagation Model (VPM), with transmission probability $\beta = \beta_1$, and healing probability $\delta = \delta_1$, calculate the effective strength (s) of the virus on the static contact network provided (*static.network*). See supplementary material provided for details on the SIS VPM and on how to calculate the effective strength of a virus. Answer the following questions:
   a. Will the infection spread across the network (i.e., result on an epidemic), or will it die quickly?

b. Keeping δ fixed, analyze how the value of β affects the effective strength of the virus (*suggestion*: plot your results). What is the minimum transmission probability (β) that results in a network-wide epidemic?
c. Keeping β fixed, analyze how the value of δ affects the effective strength of the virus (*suggestion*: plot your results). What is the maximum healing probability (δ) that results in a network-wide epidemic?
d. Repeat (1), (1a), (1b) and (1c) with $\beta = \beta_2$, and $\delta = \delta_2$.

2. Write a program that simulates the propagation of a virus with the SIS VPM, given a static contact network, a transmission probability (β), a healing probability (δ), a number of initially infected nodes (*c*), and a number of time steps to run the simulation (*t*). The initially infected nodes should be chosen from a random uniform probability distribution. At each time step, every susceptible (i.e., non-infected) node has a β probability of being infected by neighboring infected nodes, and every infected node has a δ probability of healing and becoming susceptible again. Your program should also calculate the fraction of infected nodes at each time step.
    a. Run the simulation program 10 times for the static contact network provided (*static.network*), with $\beta = \beta_1$, $\delta = \delta_1$, $c = n/10$ (*n* is the number of nodes in the network), and $t = 100$.
    b. Plot the average (over the 10 simulations) fraction of infected nodes at each time step. Did the infection spread across the network, or did it die quickly? Do the results of the simulation agree with your conclusions in (1a)?
    c. Repeat (2a) and (2b) with $\beta = \beta_2$, and $\delta = \delta_2$.

3. Write a program that implements an immunization policy to prevent the virus from spreading across the network. Given a number of available vaccines (*k*) and a contact network, your program should select *k* nodes to immunize. The immunized nodes (and their incident edges) are then removed from the contact network.
    a. What do you think would be the optimal immunization policy? What would be its time complexity? Would it be reasonable to implement this policy? Justify.

    For your program, use the following heuristic immunization policies:
    - Policy A: Select *k* random nodes for immunization.
    - Policy B: Select the *k* nodes with highest degree for immunization.
    - Policy C: Select the node with the highest degree for immunization. Remove this node (and its incident edges) from the contact network. Repeat until all vaccines are administered.
    - Policy D: Find the eigenvector corresponding to the largest eigenvalue of the contact network's adjacency matrix. Find the *k* largest (absolute) values in the eigenvector. Select the *k* nodes at the corresponding positions in the eigenvector.

    For each heuristic immunization policy (A, B, C, and D) and for the static contact network provided (*static.network*), answer the following questions:
    b. What do you think is the intuition behind this heuristic?
    c. Write a pseudocode for this heuristic immunization policy. What is its time complexity?
    d. Given $k = k_1$, $\beta = \beta_1$, and $\delta = \delta_1$, calculate the effective strength (*s*) of the virus on the immunized contact network (i.e., contact network without immunized nodes). Did the immunization policy prevented a network-wide epidemic?
    e. Keeping β and δ fixed, analyze how the value of *k* affects the effective strength of the virus on the immunized contact network (*suggestion*: plot your results). Estimate the minimum number of vaccines necessary to prevent a network-wide epidemic.

f. Given $k = k_1$, $\beta = \beta_1$, $\delta = \delta_1$, $c = n/10$, and $t = 100$, run the simulation from problem (2) for the immunized contact network 10 times. Plot the average fraction of infected nodes at each time step. Do the results of the simulation agree with your conclusions in (3d)?

Recommended Reading: B. Aditya Prakash, Deepayan Chakrabarti, Michalis Faloutsos, Nicholas Valler, and Christos Faloutsos. Got the Flu (or Mumps)? Check the Eigenvalue! *arXiv:1004.0060 [physics.soc-ph]*, 2010.

Option 2: Virus Propagation on Time-Varying Networks

4. Repeat problem (1) using the set of 2 alternating contact networks provided (*alternating1.network* and *alternating2.network*). Here, instead of calculating the effective strength of the virus, you should calculate the largest eigenvalue of the system-matrix (see supplementary materials for details on the system-matrix). Answer questions (1), (1a), (1b), (1c), and (1d).

5. Extend your virus propagation simulation program from problem (2) to allow time-varying graphs. Your modified program will be given a set of T alternating contact networks, a transmission probability (β), a healing probability (δ), a number of initially infected nodes (c), and a number of time steps to run the simulation (t). Your program should alternate between contact networks at each time step. That is, at a given time step t_i, for $i \in [0, t)$, your program should simulate the propagation of the virus on alternating contact network $(t_i \bmod T) + 1$.
   For the modified virus propagation simulation program, and the set of 2 alternating contact networks provided (*alternating1.network* and *alternating2.network*), answer questions (2a), (2b), and (2c).

6. Extend your immunization policy program from problem (3) to allow time-varying graphs. Your modified program will be given a set of T alternating contact networks and a number of available vaccines (k). Answer question (3a).
   For your program, use the following heuristic immunization policies:
   - Policy A: Select k random nodes for immunization.
   - Policy B: Select the k nodes with highest average degree across all alternating contact networks for immunization.
   - Policy C: Select the node with the highest degree out of all the alternating contact networks for immunization. Remove this node (and its incident edges) from all the alternating contact networks. Repeat until all vaccines are administered.
   - Policy D: Find the eigenvector corresponding to the largest eigenvalue of the system-matrix (see supplementary materials for details on the system-matrix). Find the k largest (absolute) values in the eigenvector. Select the k nodes at the corresponding positions in the eigenvector.

   For each heuristic immunization policy (A, B, C, and D) and for the set of 2 alternating contact networks provided (*alternating1.network* and *alternating2.network*), answer questions (3b), (3c), (3d), and (3e).

Recommended Reading: B. Aditya Prakash, Hanghang Tong, Nicholas Valler, Michalis Faloutsos, and Christos Faloutsos. Virus Propagation on Time-Varying Networks: Theory and Immunization Algorithms. In José L. Balcázar, Francesco Bonchi, Aristides Gionis, and Michèle Sebag, editors, *ECML/PKDD (3)*, volume 6323 of *Lecture Notes in Computer Science*, pages 99-114. Springer, 2010.

Submission Requirements

i. Code for implementation of virus propagation simulation (see problems (2) and (5)).
ii. Code for implementation of immunization policies (see problems (3) and (6))
iii. README file with detailed instructions for codes (i) and (ii). It should include the following information:
    a. Packages that need to be installed (if any), and instructions for installing.
    b. Environment variable settings (if any) and OS it should/could run on.
    c. Instructions on how to run the codes.
    d. Instructions on how to interpret the results.
    e. Sample input and output files.
    f. Citations to any software you may have used.

    In short, the TA should be able to install any required software, set up the environment, execute your codes, and obtain results without any prior knowledge about your project.
iv. Project Report. The report should include your answers for the problems (1)-(3) and/or problems (4)-(6). **Please include all sections of the report in a single file.**

Project was created by Gonzalo Bello