Controlling Epidemic Diffusion with Temporal Graph and Graph Neural Network

Zora Che
Department of Computer Science
Boston University
zche@bu.edu

1 Introduction

Prompt and accurate testing and isolation of infected population is crucial to the containment of infectious diseases. Many currently enacted methods of testing rely on heuristics such as numbers of infection in one's neighborhood. The network nature of epidemic spread makes the problem an ideal setting for learning on graphs, which is project explores. Graph methods offer the advantage of higher accuracy as it considers long-range and beyond local information. This project implements the framework of SEIR epidemic spread as outlined in "Controlling Graph Dynamics with Reinforcement Learning and Graph Neural Networks" by Meirom et. al[1]. We also provide an implementation of heuristic methods, and a supervised GNN trained on the temporal graph. The GNN outperforms heuristic methods for initial infection rate of 0.05 and 0.01, even when no information of initial infect nodes is given.

2 Background

Graph neural networks solve problems on non-Euclidean datasets, where we may solve problems of node classification, network classification, edge prediction, and more. The basic approach of GNN relies on averaging (or other forms of aggregating) neighbor information and applying neural networks.

A temporal graph is a graph with edges that are time-stamped, such that at each time step the graph is dynamically changing. The epidemic spread can be modeled by a temporal graph, where at each time step a set of nodes interact. At the next time step, nodes may become latent nor infected based on prior interaction at the last time step.

The SEIR model describes the dynamic of an epidemic spread, and is the model underlying our simulation. Every node is in one of the following states: susceptible – a healthy person (S state), exposed/latent – infected but cannot infect others (L state), infectious – may infect other nodes (I state), or removed/in-isolation and isolated from the graph (R state) [2].

Common heuristics of selecting nodes for testing in an epidemic setting often relies on known information of infection, and check the possible infected neighborhoods therein. Such preprogrammed heuristic (no-learning) approaches include: ranking nodes based on known infected nodes in their 1-hop and 2-hop neighborhood[3]; ensuring the entire population is tested at a regular basis (For example, US Universities resume classes during Covid with requirements of regular testing. In December of

[1]Meirom, Eli, et al. "Controlling Graph Dynamics with Reinforcement Learning and Graph Neural Networks." *International Conference on Machine Learning. PMLR, 2021.*

[2]López, L., Rodó, X. The end of social confinement and COVID-19 re-emergence risk. Nat Hum Behav 4, 746–755 (2020). https://doi.org/10.1038/s41562-020-0908-8

[3]Meirom, Eli A., et al. "Localized epidemic detection in networks with overwhelming noise." ACM SIGMETRICS Performance Evaluation Review 43.1 (2015): 441-442.

Preprint. Under review.

2021, Boston University requires its undergraduate students to test once a week, approximately half of the expected incubation period for Covid).

3 Problem Setting

We design a setting of epidemic spread first by sampling from a graph to construct temporal graphs at each time step. We may interpret this as, the ground truth graph encode relationships of the nodes, and at each time step, a certain percentage of the nodes interact. Each edge is assigned a transmission probability between 0.5 and 1 inclusive, randomly We introduce the epidemic spread by randomly choosing a percentage of the nodes to be infected. Note that the initial infected node need not be included in the initial time step graph.

After each time step, if a susceptible node interacts with an infectious node, the probability of it remaining healthy is $\prod_{e \in E_v(t)}(1 - p_e(t))$. Namely, we consider all the exposure from other nodes that are infected that interacted with the healthy node in the last time step. If the node does not remain healthy, it becomes latent, with a latency period that is a random variable, dependent on the parameter to the setting. The node will become infectious after the incubation period is passed. If a node is tested and is either latent or infectious, it will go into isolation and be removed from the graph.

The goal of the problem is to keep the most percentage of nodes healthy at a later time step.

4 GNN Model

We implement a GNN ranking module to output the probability of a node being infected following the description in Meirom et al. The input for the GNN contains of three parts: the static node features, the dynamic node features, and the dynamic edge features. The static node feature is a vector of dimension 4 with calculated value per node of topological graph centralities (betweeness, closeness, eigenvector, and degree centralities). The dynamic node features contains one-hot vectors of dimension 4, corresponding to either untested, tested positive, tested negative, or tested positive in a step before the prior step. The dynamic edge feature is the multi-graph view of temporal change, where the edges are all the interactions for a window period prior and including the current time step, and the feature is a vector of dimension 2 with the time step difference of the edge compared to the current time, and the edge attribute of transmission probability. The model generates a latent epidemic node feature that captures local diffusion by passing the nodes and edges at the given time step through a 1 layer graph convolutional network. To capture long-term information, a GNN with two layers of edge conditioned convolutions is used on the multi-graph, containing interactions for the current time step and a window of time steps before that. The epidemic node feature and the information feature are then combined with a hidden state feature (initialized as random), to update the new hidden state of node representation. This serves for the network to retain long-term memory. Lastly, an MLP outputs scores for all nodes by inputting the previous hidden state, the updated hidden state, and the node features. The scores are probabilities of a node being infected at the current time step. The model is trained by back-propagating the binary cross entropy loss between the scores and the ground truth label of whether the node is infected at time t. Any node that is not healthy is infected. Based on the probability, k nodes are chosen to be tested, which then updates the dynamic node feature of its intervention status.

5 Experiments

We experiment on preferential attachment networks with mean degrees of 3 and 5 (the degrees exhibit a power-law), as well as the General Relativity and Quantum Cosmology collaboration network[4]. Both have been used in simulating epidemic spread.

We chose the k nodes in training and testing time using indices of the top output probability scores. Though we coded the normalized scores for sampling described in Meirom et al.'s paper. This would be a good sampling strategy if the training setting benefits from more exploration, and is especially

[4]Leskovec, Jure, Jon Kleinberg, and Christos Faloutsos. "Graph evolution: Densification and shrinking diameters." ACM transactions on Knowledge Discovery from Data (TKDD) 1.1 (2007): 2-es.

good for reinforcement learning. Since we decided to implement the project from scratch, we omit the RL component to focus on the GNN module.

We chose the same intervention allocation of k = 100 for PA networks of 1000 nodes, and the collaboration network with 5242 nodes. We kept the sampling for each time step as a percentage of the entire graph the same, at 30%. We also kept the same parameter of the incubation period's random variable, for the incubation to be between 1 to 4 time steps. We trained and tested on total time steps being 15. We varied on the level of initial infection percentage.

For the heuristic of 1-hop and 2-hop neighborhoods, we represented the graph using an adjacency list built from doubly linked linked list. This is especially helpful for large graphs or many time step simulation, since we can remove the isolated node easily with this graph representation format.

For the heuristic of testing nodes on a schedule of half-incubation time, inspired by University testing, each node is tested the half of the expected incubation period. Randomization is included by shuffling the nodes such that only random k tested (this reflects that some students may be behind schedule).

Experiment results are averaged across 5 trials.

We see that GNN outperforms compared to other methods for initial infection rate 5% and 10%. The success of heuristic-hop, checking 1-hop and 2-hop number of infected nodes and intervening those who has the most infected neighbors, for a low infection rate might be that it can contain most of the outbreak from the start, and don't need to take account of information beyond local information (Table 2; Figure 1,2). For a higher infection rate, just focusing on local does not produce the greatest result, as it does not take account of graph topology and past interactions. GNN is able to outperform for the higher infection rates.

Moreover, for infection rate=0.05 and 0.01, even when no initial infected node's index is given to the network, the network outperforms the heuristic methods (Table 3,4; Figure 1,2). Note that with GNN, we are less reliant on accurate early reporting of infected individuals, while the heuristic method of 1-hop and 2-hop neighborhoods needs to have infected nodes to have good performance.

Furthermore, the simple testing heuristic of testing nodes every half incubation time has good performance with less computation compared to the 1-hop, 2-hop heuristic method (Table 3,4; Figure 1,2).

Table 1: Dataset Summary

Method \Graph	avg_{deg}	nodes	edges
PA$_2(Generated)$	2	1000	—
PA$_5(Generated)$	5	1000	—
GR-QC	—	5242	28980

Table 2: Percentage of healthy nodes at t=14, initial infection rate =0.01

Method \Graph	PA$_3$	PA$_5$	GR-QC
No intervention	0	0	0.403
Heuristic-hop	**0.36**	0.186	**0.553**
Heuristic-half-incubation	0.008	0.009	0.42
GNN	0.271	**0.253**	0.496

Table 3: Percentage of healthy nodes at t=14, initial infection rate =0.05

Method \Graph	PA$_3$	PA$_5$	GR-QC
No intervention	0	0	0.255
Heuristic-hop	0.031	0.023	0.284
Heuristic-half-incubation	0.01	0.009	0.275
GNN	**0.195**	**0.146**	**0.310**
GNN, no init given	0.092	0.13	0.287

Table 4: Percentage of healthy nodes at t=14, initial infection rate =0.1

Method \Graph	PA_3	PA_5	GR-QC
No intervention	0	0	0.190
Heuristic-hop	0.005	0.011	0.194
Heuristic-half-incubation	0.006	0.005	0.228
GNN	**0.141**	**0.113**	**0.282**
GNN, no init given	0.078	0.103	0.228

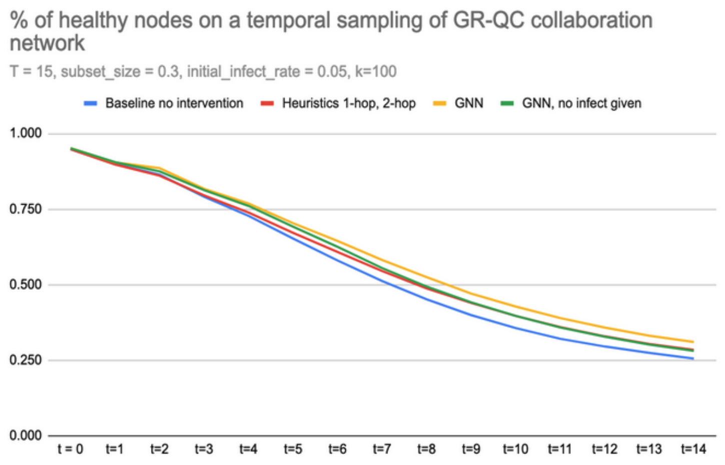

Figure 1: Initial infection rate = 0.05

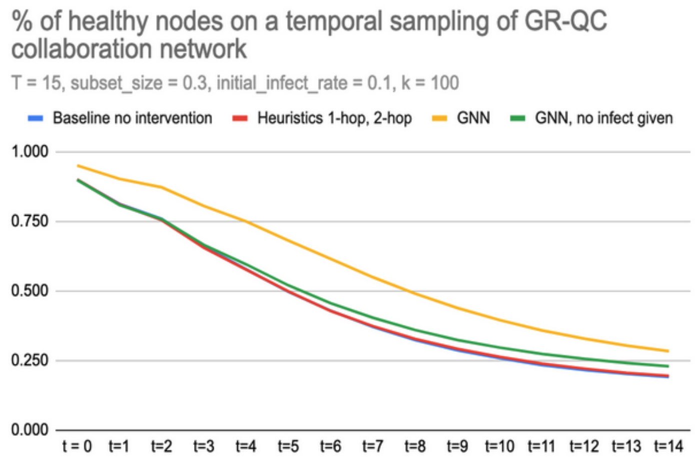

Figure 2: Initial infection rate = 0.1

6 Discussion

In this project we provided an implementation of the SEIR model of epidemic spread, 2-hop heuristic methods and a GNN network based on work by Meirom et. al. This project experiments on preferential attachment networks, the General Relativity and Quantum Cosmology Collaboration Network (Leskovec et al. 2007), and includes another heuristic method of testing every half incubation time. The code can be found at https://github.com/zorache/Controlling-Graph-Dynamics/.

We have provided the first partial codebase for the ICML 2021 paper by Meirom et. al, with emphasis on the GNN module. We have experimented on different graphs and different levels of population infection, with insights towards when the GNN ranking module may be most useful in selecting nodes for intervention. We note GNN's ability to not rely on accurate initial reporting of infection status to achieve good performance for later time steps.

This project may be expanded to include all baselines and the RL training procedure in the paper, as well as training the GNN on a mixture of graphs to encourage generalization.

To improve this project and for its development beyond a class project, more compute may be needed to train a larger network (for example, replacing the hidden states network with a GRU), and train on more epochs across different graphs.