Using Reinforcement Learning to Find Near-Optimal Policies for Controlling Covid-19 Pandemic

Daniel Ahn
UID: 504817439
dahn@g.ucla.edu

Dawei Huang
UID: 304792166
daweihuang@g.ucla.edu

Yuting Miao
UID: 504760254
ytmiao@g.ucla.edu

Zhengqi Wu
UID: 005230430
zhengqiwu@g.ucla.edu

Abstract

Throughout the history, epidemic of infectious diseases led to spikes in human illness and mortality, and caused great burden on economy locally or globally. Especially for now, COVID-19 outbreaks are taking place around the world, and human infection and death cases are continuing to accumulate. Currently, for administrative groups, one major challenge is to identify a collaborative intervention strategy in order to control the spread of COVID-19 and reduce its damage to economy. In this work, we use a Gillespie Simple Contagion model to simulate the transmission and infection within and between neighborhoods. To develop an intervention strategy, a deep reinforcement learning (RL) model with a Deep Q-Network (DQN) is constructed to evaluate the effectiveness of potential interventions. We show that a combination of interventions like local / global quarantine and social distancing for a specific duration would greatly decrease human infection and death as well as prevent the economic regression.

Introduction

Computational models of epidemic spread and control have been widely studied for better prediction of disease spreading as well as better containment to halt a pandemic in its earliest stages (Kiss et al. [2017]). Especially for now, due to the outburst of COVID-19, optimal policies dependent on the developmental stage of the disease are in great need worldwide. Large benefits are expected to be achieved from any interventions that are able to contain the spread of this pandemic and eliminate from the human populations. However, due to the rapid rate of spread of COVID-19, now we are at a stage where a customized and realistic control strategy need to be developed for both individual countries and internationally. Different policies need to be put forward taking into account all the information received and projecting the current situation into the future spread of the disease.

Apart from mathematical models, reinforcement learning (RL) optimization strategies have been proposed recently to automatically find optimal policies to support decision makers (Probert [2019]). A major advantage of using RL is that the environment can be modified accordingly after choosing each intervention, and a sequence of intervention combinations can be learned independently without human bias. RL can help to develop a more time-sensitive policy that is insightful and effective for the long run (Khadilkar et al. [2020]). Here, we adopt a deep Q-network (DQN) into our RL model, and learn the optimal policy on the simulated graph from the Gillespie Simple Contagion model (Miller and Ting [2019]). We simulate a community structure with multiple neighborhoods spread around a center (e.g. a department store), and learn Q values with the RL model. A series of interventions are selected from a pre-defined action set, and the model aims to minimize the number of people infected and dead as well as to keep the strong economy performance. In the end, we show that a combination of interventions like local / global quarantine and social distancing for a specific duration would greatly decrease human infection and death as well as prevent the economic regression.

Methods

Simulation Settings

For the simulation part, we design a graph to represent the pandemic. Define graph G = <V, E>, Where V is the set of nodes. Each node represent a person in the community. And E is the edge between nodes, which represent the connectivity among people. We also have afflicted sets node states and edge states. For each node, it has an attribute representing the likelihood of this person getting infected. We call this attribute node state. For each edge, it has an attribute representing the weight of its connectivity, and it is called edge state.

Our network is designed as a small society with neighbourhoods. We have a central node connected to k neighbourhoods, where each neighbourhood is associated to a adjacency matrix. In our experiment, we choose $k = 9$ and each neighbourhood is a complete graph with 9 nodes. The network is shown in figure 1.

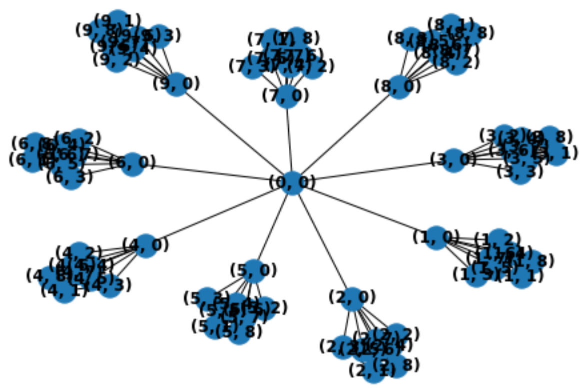


Figure 1: Simulation Network. Each node has a label (a,b) where a is the index of neighbourhood and b is its index within the neighbourhood.

Environment, EoN and Reward

To model the environment that the DQN will interact with, we used the Gillespie Simple Contagion model provided in the Epidemics on Networks (EoN) Library by Miller and Ting [2019]. The Gillespie Simple Contagion model allows us define transitions from one node state to another, e.g. Infected to Recovered, which is particularly useful for modeling the effects of arbitrary diseases. In addition to the heavy lifting provided by the EoN library, we have defined a way for actions to be translated into tangible effects in the following way

$$f : A \times G \times R \to G' \times R' \tag{1}$$

where A is an action, G is the graph representing the environment, R is the set of all transition rates defined in the environment. For reinforcement learning, state set can be viewed as $\mathcal{S} = \{G \times R\}$. Using an action space consisting of the following values

$$\begin{aligned}\mathcal{A} = \{&\texttt{Nothing},\ \texttt{Social Distancing},\ \texttt{Start Global Quarantine},\\ &\texttt{End Global Quarantine}, \texttt{Start Local Quarantine (Neighborhood 1)},\\ &\texttt{End Local Quarantine (Neighborhood 1)}, \cdots\}\end{aligned} \tag{2}$$

and based on a review of the realistic effects of each action, we can modify G and R in between time steps of the iteration. This effectively models the changes a government agency can take during the course of a pandemic.

In order to prevent a policy from naively suggesting the most extreme action, `Quarantine` as soon as possible, we have defined a reward function that takes into consideration the economic output of the people under quarantine.

$$\texttt{Reward} = f(I, D, \texttt{Economy}) \tag{3}$$

In this reward function, the reward experienced by the agent is a function of I, the number of infected people, D, the number of dead people, and `Economy`, an indicator of the performance of the economy

$$\texttt{Economy} = g(QT, SD) \tag{4}$$

which is itself a function of QT, the number of quarantined people, and SD, social distancing people. The addition of this economic factor would incur a significant penalty for closing the economy due to quarantine and thus should only happen when it can find a balance between the safety of its people and the health of its economy. This would also force the policy to consider ending the quarantine before the virus has been completely eliminated. This may give us insight into when it may be acceptable to risk potential second waves weighed against other factors.

We can consider (I, D, QT, SD) as a vector. The most intuitive way is to use norms on the vector to generate the reward function. The most commonly used are 1-norm and 2-norm. 2-norm can better consider the balance among factors, restricting any of the factors becoming too large, while 1-norm is equally contributed by all factors. Thus we define our reward function as squared sum of the factors. Due to the nature of 2-norm to balance all factors, our reward function performance will depend less on hyperparameter design, which is a quite difficult problem.

$$\texttt{Reward} = -\alpha I^2 - \beta D^2 - \gamma QT^2 - \delta SD^2 \tag{5}$$

α, β, γ and δ are hyperparameters that adjust the weights of the factors. In our experiment we choose $\alpha = 1$, $\beta = 2$, $\gamma = 0.2$ and $\delta = 0.1$.The reward function represents safety loss plus economy loss in an nonlinear way.

Deep Q Networks (DQN)

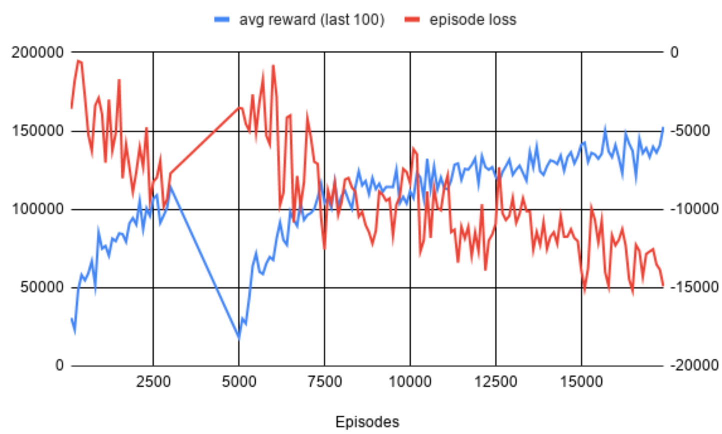
The DQN uses a deep neural network whose architecture is 4 hidden layers of 1000 units in each layer. The model target, $\hat{Q}(s, a; \theta_{target})$, approximates the optimal state-action-value function $Q^*(s, a)$ and back-propagates the loss gradient in each iteration of each episode. The loss function is defined by: $\frac{1}{|Batch|} \cdot \sum_{(s,a,r,s') \in Batch} (Q^*(s, a) - \hat{Q}(s, a; \theta_{train}))^2$, which is the mean square error (MSE) of the Q-function and the target Q-function. But how can we find the ground-truths if we don't know what Q^* is? We find the ground truths using the Bellman Equation $Q^*(s, a) = \max_x r + \hat{Q}(s', a; \theta_{target})$. Note that $\hat{Q}(s', a; \theta_{target})$ is 0 when s' is a terminal state. Why do we have a target model θ_{target} and a training model θ_{train}? The reason is because of the concept of fixed target. Fixed target trains with θ_{train} and every once in a while updates θ_{target}. This, though increasing the cost of memory, improves the stability of the target weights. In addition, why do we have $Batch$ in the MSE up above? It's because every time the DQN trains, it takes a batch sample from its experience replay buffer and trains upon this subset of experiences. This means that the samples that the DQN is sampling from is not correlated with respect to time and because the subset is a random selection of previous experiences, the DQN model can generalize to a greater extent. Furthermore, because DQN is ϵ-greedy, the DQN can initially generate a variety of different experiences (s, a, r, s') by occasionally exploring random action spaces. Combining the fixed target and experience replay buffer, the DQN can improve its approximation.

In addition, DQN implements an Adam optimizer, which improves the speed at which the DQN's deep neural network converges. The Adam optimizer updates the weights accordingly:

$$m_t = \beta_1 \cdot m_{t-1} + (1-\beta_1)g_t$$
$$v_t = \beta_2 \cdot v_{t-1} + (1-\beta_2)g_t^2$$
$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$$
$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}$$
$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \hat{\epsilon}} \cdot \hat{m}_t$$

m_t and v_t represents the running average and second moment of the gradient g_t. $\hat{m}_t$ and $\hat{v}_t$ represents fixing the "inertia" which favors m_{t-1} and v_{t-1} when β_1 and β_2 are close to 1. Lastly, η is the learning rate hyperparameter.

Results



We trained the DQN agent until the loss and rewards function begins to converge, which in our case is 17,400 episodes. The loss function is the mean square error of the optimal value Q^* and the approximate value $\hat{Q}_\theta$. The red line represents the mean loss over the course of the episodes. The blue line represents the mean reward of the last 100 episodes of each point. The sharp increase in episodic loss and decrease in average reward is a technical error due to the fact that Google Colab crashed, preventing us to record the relevant details. Fortunately, because we have saved the model checkpoints, the weights of the target and training models were saved and we can continuing training the model until convergence. Then afterwards, we compare the policy of the DQN to two other policies: Lenient and Paranoid. The Lenient policy simply chooses the action that "does nothing" amongst all available action in the action space. Paranoid, similarly, chooses the action that instigates "global quarantine" and "global quarantine" only throughout all time steps of the episodes. The below table represents the average rewards of each policy. We can see that DQN does much better reward-wise compared to the other policies.

Policy	avg rewards (last 100)
DQN	-4735.24
Paranoid	-23549.7
Lenient	-22799.3

To see the specific outcomes generated by the policies, we will explore their real effects and corresponding rewards.

Although visually similar, there are several major benefits the DQN policy has over the Lenient policy. First we can see the DQN policy has a less drastic spike in number of infected compared to

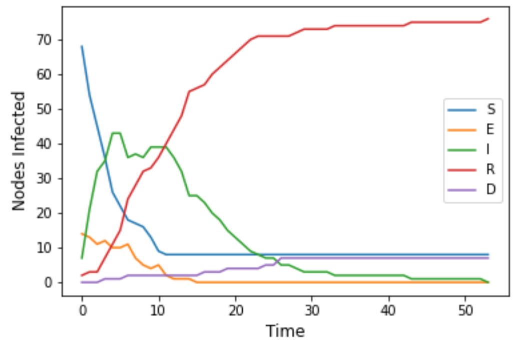


Figure 2: Effects of DQN policy

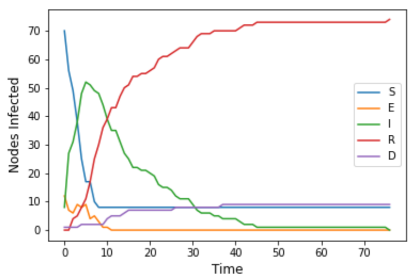


Figure 3: Effects of Lenient policy

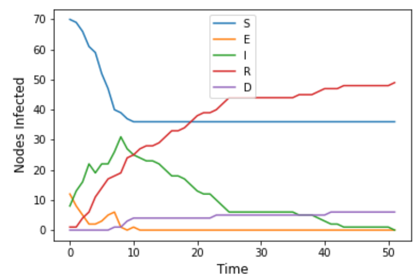


Figure 4: Effects of Paranoid policy

the Lenient policy. This shows that the policy learns the value of quarantining neighborhoods and enforcing social distancing measures. This is much like real life where such measures are intended to flatten the curve of infected people.

Another benefit we see is the duration of the pandemic. Without any measures taken to counteract the pandemic, a simulation by the Lenient policy lasts nearly 80 time steps. On the other hand, the DQN policy is able to recover from the pandemic in just over 50 time steps.

The Paranoid policy clearly performs best because a quarantine is the most effective response to the pandemic. It has the smallest spike in infected, lowest number of deaths, and highest number of uninfected. However this ignores the reality of the situation we face. Often there are real life factors that prevent everyone from adhering to the rules of a quarantine. This is why we introduce an economic factor which penalizes quarantines.

Here we see the reward accrued by each policy taking into account not only deaths but also the economic output of each node. Since node's economic output decreases when it is infected, quarantined, or dead, this reward captures a better representation of the true state of our environment.

By this metric, the DQN outperforms other policies by finding a balance between the acceptable number of infections and the loss in economic productivity.

Difficulties

Although we were able to successfully learn a policy that outperforms the simple policies explored in this paper, there were several difficulties we faced that may have decreased the DQN's potential performance.

As mentioned previously, the reward function consisted of economic impact which the DQN learned to balance with the deaths and infections in the environment. The ideal weighting of each term inside the reward function is an open question and the policy learned by the DQN is entirely dependent on it. This could lead to wildly different results if another set of weights are used. However this does not

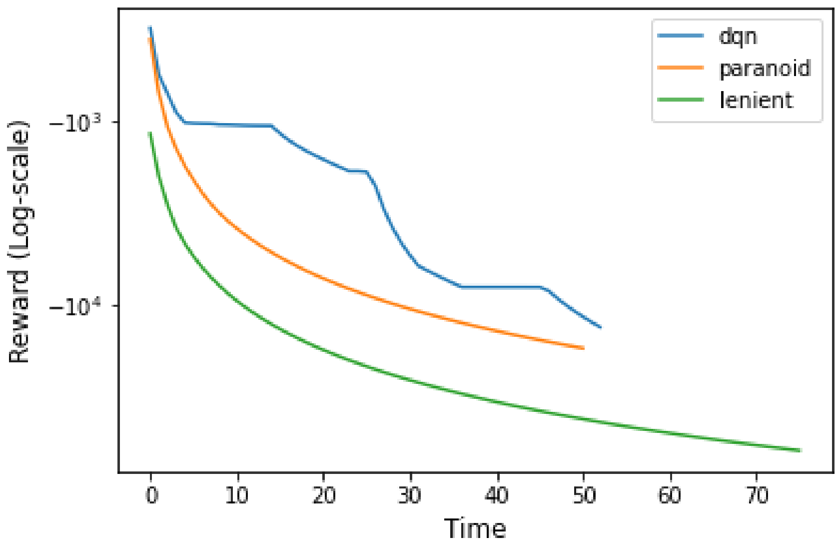


Figure 5: Reward of each policy for an episode

necessarily invalidate our results since we have showed that the DQN policy still outperforms the Lenient and Paranoid policies, representing the opposite ends of the weighting spectrum.

Contributions

- Daniel Ahn: primarily responsible for designing gym-covid environment. Also responsible for generating and interpreting results.
- Dawei Huang: responsible for the DQN and partial construction of the custom gym-covid environment. Also responsible for generating and interpreting results.
- Yuting Miao: responsible for designing the action set and the effect of each to simulation networks, and implementing it into the training.
- Zhengqi Wu: contributes to the simulation network structures. Also designed and implemented the loss/reward function.

References

Lakkur S. Fonnesbeck C. J. Shea K. Runge M. C. Tildesley M. J. & Ferrari M. J. Probert, W. J. Context matters: using reinforcement learning to develop human-readable, state-dependent outbreak response policies. *Philosophical Transactions of the Royal Society B*, 374(1776), 2019. doi: https://doi.org/10.1098/rstb.2018.0277.

Harshad Khadilkar, Tanuja Ganu, and Deva P Seetharam. Optimising lockdown policies for epidemic control using reinforcement learning, 2020.

Hayes C. & Glavin F. Yanez, A. Towards the control of epidemic spread: Designing reinforcement learning environments., 2019.

István Z Kiss, Joel C Miller, Péter L Simon, et al. Mathematics of epidemics on networks. *Cham: Springer*, 598, 2017.

Joel Miller and Tony Ting. Eon (epidemics on networks): a fast, flexible python package for simulation, analytic approximation, and analysis of epidemics on networks. *Journal of Open Source Software*, 4(44):1731, Dec 2019. ISSN 2475-9066. doi: 10.21105/joss.01731. URL `http://dx.doi.org/10.21105/joss.01731`.