Outbreak of Covid-19 worldwide is on the decline

-----Recurrent Neural Reinforcement Learning and Health Interventions to Curb the Spread of Covid-19 in the world

Qiyang Ge[1], Zixin Hu[2,3], Kai Zhang[4], Shudi Li[4], Wei Lin[1], Li Jin[2,3] and Momiao Xiong[4,*]

Supplementary Materials

Supplementary Note A

Supplementary Figure Legend

Figure S1. RNRL algorithm flowchart.

Figure S2. The reproduction number and intervention measure curves as a function of time in the top fifteen most-affected countries where blue and red curve represented the reproduction number and intervention measure, respectively.

Tables S1-S5

Supplementary Note A

Recurrent Neural Reinforcement Learning for Action (Intervention) Selection and Evaluation

Architecture of Recurrent Reinforcement Learning Neural Networks

The basic structure of reinforcement learning (RL) is an agent and an environment [28]. The agent interacts with the environment by carrying out different actions under different environment (the state of the system). The RL has three components: state, action and reward. The agent gets a reward with each action. The RL attempts to generate a sequence of actions to make the best reward. The RL can be viewed as an open dynamical system with a correspondent reward function (or loss function). A typical dynamic system is a Markov Decision Process (MDP). The RL has two basic tasks: system identification and best (suitable) action selection. Therefore, the architecture of recurrent neural reinforcement learning (RNRL) consists of two RNNs. One RNN is designed for system identification to learn a model underlying the dynamics of Covid-19 from available historical data. The RNN for system identification is called an encoder (Figure 1). The second RNN is designed for action selection and evaluation to learn optimal intervention policy. The RNN for action selection and evaluation is called a decoder (Figure 2).

RNN Encoder

The RNN encoder was a mapping from a sequence space (or time series) to another sequence space (or time series space) and the current output depended on both current input and the whole observation history (whole time series). The RNN encoder extracted short-term local dependency patterns among variables and to discover long-term patterns for time series trends

[S1]. The RNN encoder had three layers: input, recurrent hidden and output layers (Figure 1). The input layer consisted of three types of variables: covariates $X_t = [X_t^1, \dots, X_t^n]^T$, a scaler intervention variable $A_t \in \mathcal{A}$, where $\mathcal{A}$ is an action space, and the numbers of cases (potential outcomes) Y_t at the time t. Similar to the reproduction number R in the epidemiological models which is often used to determine the dynamic behavior of epidemics, Intervention measure is a matric to quantify the degree of controlling infection. Interventions were measured by number in the interval [0, 1]. A value of 1 for intervention measure indicates that intervention is the strongest and reproducing number R is close to zero. A value of zero for intervention variables indicates that no restrictions on social-economic activities are imposed. The values between 0 and 1 indicated the various less strict interventions. The covariates can include the rate of virus test, Google mobility indexes, social distance index, weather, age, gender, race, . Define the input vector V_t as

$$C_t = \begin{bmatrix} y_t \\ \vdots \\ y_{t-l+1} \end{bmatrix}.$$

Let $h_t = [h_t^1, \dots, h_t^m]^T \in S$ be a m dimensional hidden state vector where m is set to be 100 in this study where S is a state space. The data C_t was inputted into the input layer. The linear transformation $W_{ch}C_t$ of the data C_t was then sent to the hidden layer, where W_{ch} is a $m \times l$ dimensional matrix. The input data also included the covariates X_t and action A_t. The hidden layer receives information from the input layer and hidden layer at the previous time point. The state h_t at the time t was determined by the following nonlinear system transition equation:

$$h_t = f_h(W_{hh}h_{t-1} + W_{ch}C_t + W_{ah}A_t + W_{xh}X_t + b_h)\,, \qquad \text{(A1)}$$

where W_{hh} was a $m \times m$ dimensional weight matrix that connected the previous state to the current state, W_{ah} was $m \times 1$ dimensional matrix that connected the action variable to the current hidden state, W_{xh} is a $m \times n$ dimensional matrix that connected the covariates to the current hidden state, and $b_h = [b_h^1, \dots, b_h^m]^T$ was a m dimensional bias vector that corrected the bias, and f_h was a element-wise nonlinear activation function and was often defined as the following "tanh" function:

$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$.

Let θ_h denote the weight matrices and bias vector in equation (A1). Then, equation (A1) can be rewritten as

$$h_t = f_h(h_{t-1}, C_t, A_t, X_t, \theta_h) . \quad \text{(A2)}$$

The MDP is partially observed through output (observation) equation. The neurons in hidden layer were connected to the output layer via a m dimensional weight vector W_{hy}. The output $\hat{Y}_{t+1}$ was determined by

$$\hat{y}_{t+1} = f_y(W_{hy} h_t + b_y) , \quad \text{(A3)}$$

where f_y was an activation function, W_{hy} was $1 \times m$ dimensional weight matrix, and b_y was the bias of the output neuron. Let θ_y denote the weights and bias in equation (A3). Then, equation (A3) can be rewritten as

$$\hat{y}_{t+1} = f_y(h_t, \theta_y) . \quad \text{(A4)}$$

A deterministic policy maps a state h_t directly to an action via a feedforward neural network:

$$A_{t+1} = f_a(W_{ha} h_t + b_a), \quad \text{(A5)}$$

where W_{ha} was a weight matrix, b_a was a bias, and f_a was element-wise nonlinear activate function.

Again let θ_a denote the weights and bias in equation (A5). Then, equation (A5) can be rewritten as

$$A_{t+1} = g(h_t, \theta_a) \ . \tag{A6}$$

Reward r_t at the time step t was defined as

$$r_t = R(h_t, A_t) = \|y_{t+1} - \hat{y}_{t+1}\|^2 \ . \tag{A7}$$

At each time step, the agent took action A_{t+1} at the state h_t and received reward $R(h_t, A_t)$. The agent then transited into the next state h_{t+1}. The goal of the RL was to find a policy that minimizes the following reward

$$\min_A V(h_l) = \sum_{t=l}^{T-1} \|y_{t+1} - \hat{y}_{t+1}\|^2 \ , \tag{A8}$$

where $A = \{A_l, A_{l+1}, \dots, A_{T-1}\}$.

The popular methods for solving the RL problem was dynamic programming [S 2]. However, dynamic programming was suitable for discrete state space and action space and was not efficient for continuous state space and action space. A popular way to solve the above problem was the gradient method. All system transition, policy mapping and value functions were approximated by neural networks and parameterized [28]. Since the action variable was parameterized, the value function in equation (A8) should be augmented. Define

$$V_y\big(y, x, A, \theta_{h,}\theta_y\big) = \sum_{t=l+1}^{T-1} \|y_{t+1} - \hat{y}_{t+1}\|^2 \ , \tag{A9}$$

$$V_a(y, x, A, \theta_a) = \sum_{t=l+1}^{T-1} \left\|\hat{A}_{t+1}^{k} - \hat{A}_{t+1}^{k-1}\right\|^2 \ , \tag{A10}$$

and

$$V\left(y, x, A, \theta_{h,}\theta_y, \theta_a\right) = V_y\left(y, x, A, \theta_{h,}\theta_y\right) + V_a(y, x, A, \theta_a)\ , \tag{A11}$$

where $\theta_h = (W_{vh}, W_{ah}, W_{xh}, b_h)$, $\theta_y = \left(W_{hy}, b_y\right)$, and $\theta_a = (W_{ha}, b_a)$.

Define

$$\tilde{V}_y^k\left(y, x, \tilde{A}^k, \theta_h, \theta_y\right) = V_y\left(y, x, \tilde{A}^k, \theta_{h,}\theta_y\right) + V_a\ .$$

Define the following optimal value functions:

$$\bar{V}_y^k(y, x, \tilde{A}^k, \tilde{\theta}_h^{k+1}, \tilde{\theta}_y^{k+1}) = \min_{\theta_h, \theta_y} \tilde{V}_y^k(y, x, \tilde{A}^k, \theta_h, \theta_y)\ , \tag{A12}$$

where

$$\begin{bmatrix}\tilde{\theta}_h^{k+1} \\ \tilde{\theta}_y^{k+1}\end{bmatrix} = \arg\min_{\theta_h, \theta_y} \tilde{V}_y^k(y, x, \tilde{A}^k, \theta_h, \theta_y)\ , \tag{A13}$$

$$\tilde{V}_a^{k+1}\left(y, x, \tilde{A}^{k+1}, \tilde{\theta}_h^{k+1}, \tilde{\theta}_y^{k+1}, \tilde{\theta}_a^{k+1}\right) = \min_{A, \theta_a} \tilde{V}_a^k\ (y, x, A, \tilde{\theta}_h^{k+1}, \tilde{\theta}_y^{k+1}, \theta_a)\ \ , \tag{A14}$$

where

$$\begin{bmatrix}\tilde{A}^{k+1} \\ \tilde{\theta}_a^{k+1}\end{bmatrix} = \arg\min_{A, \theta_a} \tilde{V}_a^k\ (y, x, A, \tilde{\theta}_h^{k+1}, \tilde{\theta}_y^{k+1}, \theta_a)\ \ , \tag{A15}$$

$$\tilde{V}_y^k\left(y, x, \tilde{A}^{k+1}, \tilde{\theta}_h^{k+1}, \tilde{\theta}_y^{k+1}\right) = \bar{V}_y^k(y, x, \tilde{A}^{k+1}, \tilde{\theta}_h^{k+1}, \tilde{\theta}_y^{k+1})\ , \tag{A16}$$

and

$$\tilde{V}^{k+1}\left(y, x, \tilde{A}^{k+1}, \tilde{\theta}_h^{k+1}, \tilde{\theta}_y^{k+1}, \tilde{\theta}_a^{k+1}\ \right)\ = \tilde{V}_a^{k+1}\left(y, x, \tilde{A}^{k+1}, \tilde{\theta}_h^{k+1}, \tilde{\theta}_y^{k+1}, \tilde{\theta}_a^{k+1}\right) +$$

$$\tilde{V}_y^k\left(y, x, \tilde{A}^{k+1}, \tilde{\theta}_h^{k+1}, \tilde{\theta}_y^{k+1}\right)\ . \tag{A17}$$

Minimization problems in equations (A12) and (A14) are solved by the backpropagation algorithm.

Training Algorithm for RNN Encoder

Now we were ready to present training algorithm for RNN encoder.

Step 1: Initialization for action (intervention).

The initial intervention measure was calculated as follows. Set the intervention measure at the final time $A_{t_f} = 1$ for China, $A_{t_f} = 0.3$ for Korea South, Switzerland, United Kingdom, Spain, US, Italy, Germany, Iran, and France, and $A_{t_f} = 0$ for all other countries. Assume that the intervention measure curve was an exponential function starting at 0 and ends at A_{t_f}. The intervention measure A_t^0 is given by

$A_t^0 = \frac{p^t - 1}{p - 1} * \left(A_{t_f} - A_0\right) + A_0$, where $p > 0, p \neq 1$ was the curve shape factor and t took values in evenly sliced numbers of interval $[0, 1]$, A_0 was the intervention measure at the initial time t_0 . When $p = 1$, A_t^0 is a linear function $A_t^0 = t * \left(A_{t_f} - A_0\right) + A_0$. In this study, we set $p = 0.01$.

Set $k = 0$.

Step 2: Use forward dynamic programming and backpropagation to infer the best actions (interventions) for fitting the data.

While $\left\|\tilde{A}^{k+1} - \tilde{A}^k\right\|^2 > \boldsymbol{\varepsilon}$

Given $\tilde{A}^k$, the forward dynamic programming [S6] was used to find the best action value

$\tilde{A}^{k+1} = \{\tilde{A}_1^{k+1}, \dots, \tilde{A}_T^{k+1}\}$.

Assume that

$A_{t+1}^{k+1} = \rho j$,

where $0 \le \rho \le 0.01, \rho j \le 1$.

Define

$$R^t\left(\bar{Y}_t, X_t, A_t^{k+1} = \rho i, A_{t+1}^{k+1} = \rho j, \theta_h^*, \theta_y^*, \theta_a^*\right) = \min_{\theta_h, \theta_y, \theta_a} \left(\|y_{t+1} - \hat{y}_{t+1}\|^2 + \left\|A_{t+1}^k - \hat{A}_{t+1}^{k+1}\right\|^2\right),$$

where $\bar{Y}_t = \{y_t, \dots, y_{t-l+1}\}$ and $\hat{A}_{t+1}^{k+1} = g(h_t, \theta_a) = \rho j$.

Assume that

$A_t^{k+1} = \rho i, A_{t+1}^{k+1} = \rho j$.

Define

$r^t\left(A_t^{k+1} = \rho i, A_{t+1}^{k+1} = \rho j\right) = R^t\left(\bar{Y}_t, X_t, A_t^{k+1} = \rho i, A_{t+1}^{k+1} = \rho j, \theta_h^*, \theta_y^*, \theta_a^*\right)$ and

$$V_*\left(A_t^{k+1} = \rho i\right) = \min_{\bar{A}_t^i, \theta_h, \theta_a \theta_{y,}} \sum_{\tau=l}^{t} \left(\|y_{\tau+1} - \hat{y}_{\tau+1}\|^2 + \left\|A_{\tau+1}^k - \hat{A}_{\tau+1}^{k+1}\right\|^2\right),$$

where $\bar{A}_t^i = \{A_1, \dots, A_{t-1}, A_t^{k+1} = \rho i, \}$.

Using forward recursive:

$$V_*\left(A_t^{k+1} = \rho j\right) = \min_{all\ A_{t-1}^1 = \rho i} \left\{r^t\left(A_t^{k+1} = \rho i, A_{t+1}^{k+1} = \rho j\right) + V_*(A_{t-1}^{k+1} = \rho i\right\},$$

we obtain a sequence of actions $\tilde{A}^{k+1} = \{\tilde{A}_1^{k+1}, \dots, \tilde{A}_T^{k+1}\}$ that fit the data.

End

The standard back-propagation method and the Adam Optimizer were used for minimization in the parameter estimations. The initial learning rate in the updating parameters in the RNN encoder via backpropagation was 0.02 and learning rate decay was 0.0001.

Potential Outcome Framework for Evaluation of Actions (Public Health Interventions)

Before discussing the algorithm for training RNN decoder, we introduced the basic principles for action (intervention policy) selection and evaluation. Learning intervention policies was an extremely challenge problem. We employed the Counterfactually-Guided Policy Evaluation (CF-GPS) principle and RL to evaluate the effect of public health interventions on controlling the spread of Covid-19 [S3]. We viewed the public health intervention as treatment and the number of cases as the outcome. Counterfactual treatment outcome estimation was essentially a causal problem. Most methods for causal inference were designed for the statistic setting and cannot be applied to evaluating the effects of the sequence of public health interventions on (e.g. sequential application of intervention A followed by intervention B) the transmission dynamics of Covid-19 over time. Potential outcome framework was our basic model to evaluate the impact of the public health interventions on the spread of Covid-19. The potential outcome framework was often referred to the Neyman-Rubin model (Rubin 1974).

Potential outcomes consisted of actual (or observed) and counterfactual (hypothesized) outcomes. We were interested in number of cases of Covid-19 under some specific intervention. We observed the number of cases of Covid-19 (actual observation) without intervention or known intervention. However, we wanted to know what number of cases of Covid-19 (counterfactual, unobserved) would be if an alternative intervention was implemented. To evaluate the effect of intervention, we should compare the difference between the observed actual number of cases of Covid-19 and the counterfactual number of cases of Covid-19. Our

aim was to learn the counterfactual outcomes of Coid-19 under a sequence of public health intervention options and evaluate the impact of the intervention strategies on the spread of Covid-19.

The Recurrent decoder (Figure 2) was an architecture for estimating the effects of intervention on the spread of Covid-19 over time [S4]. The decoder network used the hidden state computed by the encoder to initialize the state of an RNN in the decoder which predicted the counterfactual outcomes for a sequence of hypothesized future interventions [S4]. The decoder attempts to propagate the encoder representation forwards in time, using only the planned interventions and avoiding the covariates.

Training Algorithm for RNN Decoder

The architecture of the RNN decoder was shown in Figure 2. The difference between the RNN encoder and decoder was that RNN decoder removed the covariates from the input since the future covariates were unobservable and difficult to forecast. The training algorithm for RNN decoder was almost the same as that for RNN encoder except that the final hidden state in the RNN encoder was used as the initial state of the RNN decoder and covariates X were removed in the RNN decoder. During decoder training, the outcomes $(Y_{t+2}, \dots, Y_{t+\tau})$ from the observational data that were batched into a shorter sequences of up to τ steps. The value function is defined as

$V(h_{t+\tau_b}) = \sum_{\tau=2}^{\min\{T-t,\tau_b\}} || y_{t+\tau} - \hat{y}_{t+\tau}||^2$.

The initial learning rate in the updating parameters in the decoder training was 0.02 and learning rate decay was 0.0001.

Evaluation of the Intervention Strategies and Forecasting

After completion of the training, the decoder can be used to evaluate interventions. The potential outcome framework for treatment effect estimation which accounts for the time varying treatments [S4] was extended to evaluation of the effects of the interventions on the spread of Covid-19. Let $\widetilde{H}_t = (\tilde{Y}_t,\ \tilde{A}_t)$ be the history of the outcomes (number of cases) $\tilde{Y}_t = (\bar{Y}_l, \dots, \bar{Y}_t), \bar{Y}_t = (Y_t, \dots, Y_{t-l+1})$, and interventions $\tilde{A}_t = (A_1, \dots, A_t)$. Let $Y[\tilde{A}]$ be the potential outcomes that can be either observed or counterfactual, under each possible sequence of intervention $\tilde{A}$. The potential outcome framework assumed the existence of the hypothetical outcome with some interventions which was not observed in the data. The hypothetical outcome under hypothetical intervention was called counterfactual outcome. Given the history $\widetilde{H}_t$ and a sequence of planned interventions $\tilde{A}(t, t+\tau-1) = (A_t, \dots, A_{t+\tau-1})$, the counterfactual outcome $Y_{t+\tau}[\tilde{A}(t, t+\tau-1)]$ can be predicted by

$E(Y_{t+\tau}[\tilde{A}(t, t+\tau-1)]|\widetilde{H}_t)$,

which defined the future dynamic trajectory of the Covid-19 under the planned sequence of interventions, given the previous history of Covid-19 and its environments. To make the prediction of the dynamic trajectory of the Covid-19 under the potential outcome framework to be identifiable, we need to make the following assumptions [S4].

Assumptions

We introduced assumptions in the Neyman-Rubin model [S4].

Assumption 1. (Consistency). If a nation received an intervention $A_t = a_t$, then the potential outcome for the intervention a_t which can be counterfactual was equal to the observed (factual) outcome $Y_{t+1}(a_t) = Y_{t+1}$.

Assumption 2. (Overlap). For all $(a_1, \dots, a_{t-1})$, we had

$0 < P(A_t = a_t | A_1 = a_1, \dots, A_{t-1} = a_{t-1}) < 1$. In other words, at each time step, each intervention had non-zero probability of being implemented.

Assumption 3. Sequential strong ignorability. Conditional on $A_1 = a_1, \dots, A_{t-1} = a_{t-1}$, the potential outcomes Y_{t+1} were independent of A_t,

$Y_{t+1}(a_t) \perp\!\!\!\perp A_t | A_1 = a_1, \dots, A_{t-1} = a_{t-1}$.

Assumption 3 implied that there was no confounders which affected both outcomes and interventions.

During evaluation, we did not have access to ground-truth outcomes. Therefore, we used the decoder to make one step ahead forecasting. The outcomes forecasted by the decoder $(\hat{Y}_{t+1}, \dots, \hat{Y}_{t+\tau-1})$ were recursively used as inputs. For each country, by running the trained decoder, with a sequence of planned interventions and recursively forecasted outcomes, we forecasted the response dynamics, i.e., the number of new cases of Covid-19 of the country over time under a sequence of interventions and evaluated the effects of various intervention strategies on controlling the spread of Covid-19. By running the decoder, we could select starting and ending time of different interventions and the optimal or appropriate interventions to give over time to obtain the best outcomes of controlling the spread of Covid-19 for each country.

References

S1. Francois-Lavet V, Henderson P, Islam R, Bellemare MG, Pineau J. (2018). An Introduction to Deep Reinforcement Learning. arXiv:1811.12560.

S2. Malik AA. (2020). Robots and COVID-19: Challenges in integrating robots for collaborative automation. arXiv:2006.15975.

S3. Grigorescu S, Trasnea B, Cocias T, Macesanu G. (2019). A Survey of deep learning techniques for autonomous driving. arXiv:1910.07738.

S4. Charpentier A, Elie R, Remlinger C. (2020). Reinforcement Learning in Economics and Finance. arXiv:2003.10014.

S5. Yu C, Liu J, Nemati S. (2019). Reinforcement Learning in Healthcare: A Survey. arXiv:1908.08796.

S6. Seinfeld JH and Lapidus L. (1968). Aspects of forward dynamic programming algorithm. Ind. Eng. Chem. Process Des. Dev. 7:(3) 475–478.

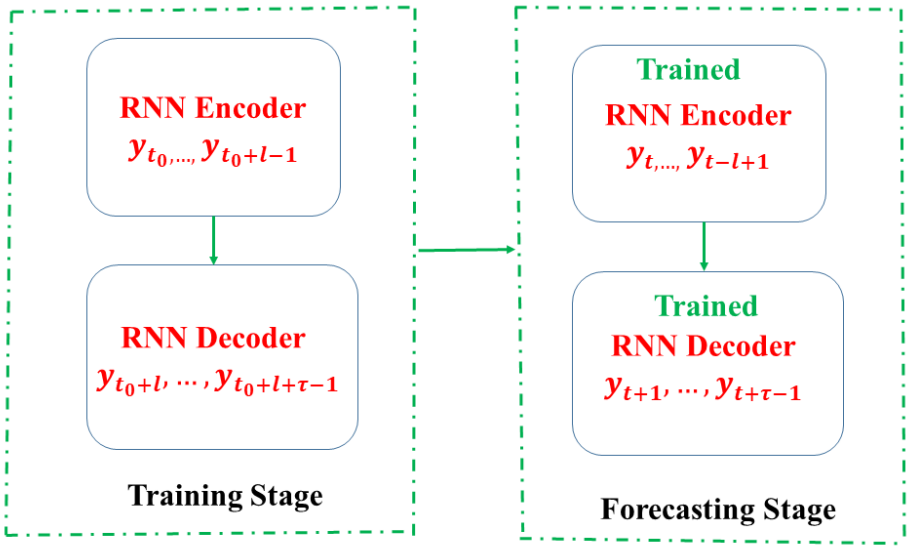

Figure S1. RNRL algorithm flowchart.

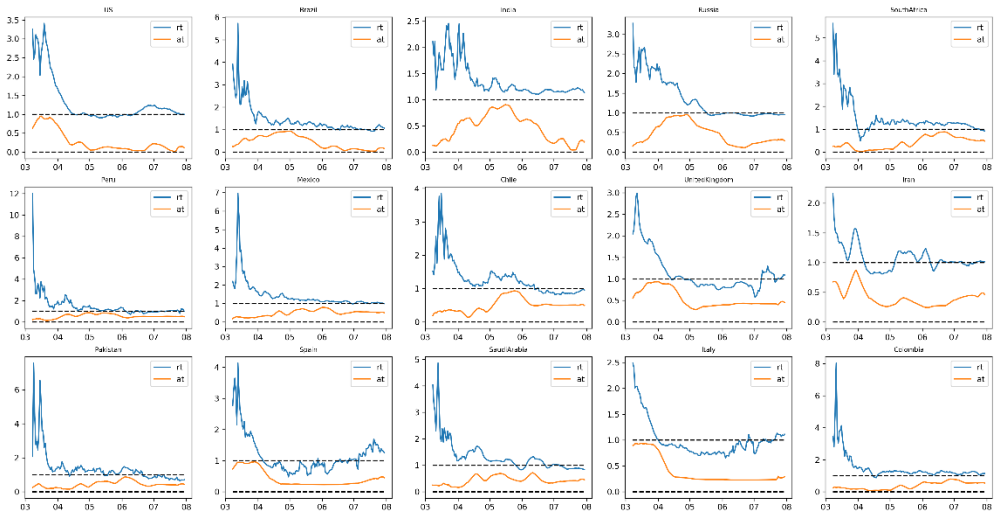

Figure S2. The reproduction number and intervention measure curves as a function of time in the top fifteen most-affected countries where blue and red curve represented the reproduction number and intervention measure, respectively.