Distribution Agreement

In presenting this thesis as a partial fulfillment of the requirements for a degree from Emory University, I hereby grant to Emory University and its agents the non-exclusive license to archive, make accessible, and display my thesis in whole or in part in all forms of media, now or hereafter now, including display on the World Wide Web. I understand that I may select some access restrictions as part of the online submission of this thesis. I retain all ownership rights to the copyright of the thesis. I also retain the right to use in future works (such as articles or books) all or part of this thesis.

Rui Yang March 29, 2021

Estimating and Predicting the Effect of Quarantine in the COVID-19 Pandemic:

Using a Modified SEIR Model

by

Rui Yang

Lars Ruthotto
Adviser

Department of Mathematics

Lars Ruthotto

Adviser

Le Chen

Committee Member

Davide Fossati

Committee Member

2021

Estimating and Predicting the Effect of Quarantine in the COVID-19 Pandemic: Using a Modified SEIR Model

By

Rui Yang

Lars Ruthotto

Adviser

An abstract of
a thesis submitted to the Faculty of Emory College of Arts and Sciences
of Emory University in partial fulfillment
of the requirements of the degree of
Bachelor of Science with Honors

Department of Mathematics

2021

Abstract

Estimating and Predicting the Effect of Quarantine in the COVID-19 Pandemic: Using a Modified SEIR Model

By Rui Yang

Governmental policies are one of the determining factors in slowing down the spread of the COVID-19 pandemic. It has been shown that effective government regulations, such as the mandate to wear masks, can control the development of the pandemic. This study aims to estimate the effectiveness of quarantine policies in the process of slowing down the pandemic.

For this study, we used the data from Wuhan, Italy, and South Korea after the cumulative confirmed cases in each region has exceeded 500 with a total period of 80 days. First, we replicated previous research on modeling the pandemic using a modified Susceptible-Infectious-Recovered-Quarantined epidemiological model. However, since this model does not consider the transformation from the "Quarantined" state to other states, we found the predicted quarantined population is too large to be realistic. Given this, we proposed a new model that estimates the quarantine strength directly from data and uses an LSTM network to make predictions for future periods. Several limitations with our model are its error sensitivity, model generalizability, and interpretability.

Through this study, we also realized that sometimes we might face the trade-off between the model interpretability and generalizability: a generic model like neural networks suffers from low interpretability, while a specific model may not apply to all situations. The best case is that we can recover the governing equations from the dataset directly. As an example to show this is possible, we used the Sparse Identification of Nonlinear Dynamics (SINDy) method to successfully recover the equations of the standard SIR model based on its trajectory.

Estimating and Predicting the Effect of Quarantine in the COVID-19 Pandemic: Using a Modified SEIR Model

By

Rui Yang

Lars Ruthotto

Adviser

A thesis submitted to the Faculty of Emory College of Arts and Sciences
of Emory University in partial fulfillment
of the requirements of the degree of
Bachelor of Science with Honors

Department of Mathematics

2021

Acknowledgements

I want to express my sincere thanks to my advisor Dr. Lars Ruthotto for his insightful comments at every stage of this project and for the assistance in writing this thesis. I am grateful that he would take time out of his busy schedule to meet with me weekly and guide me throughout the process. Besides, I would like to extend my special thanks to other committee members, Dr. Davide Fossati and Dr. Le Chen, for taking the time to attend my thesis defense and giving me helpful feedback on my thesis. Lastly, I would like to thank Dr. David Zureick-Brown for his patience in answering numerous questions I had before participating in the honors program.

Contents

List of Figures

Chapter 1

Introduction

The COVID-19 pandemic, caused by the SARS-CoV-2 virus, has become a global concern in the first half of 2020. According to the World Health Organization, as of March 14, 2021, more than 119 million cases have been confirmed globally, resulting in more than 2.6 million deaths [1]. To prevent further loss of human lives, many countries have enacted various measures, including quarantine, curfews, and closure of public places, which turned out to be effective in containing the spread of the pandemic. For example, just 45 days after Wuhan, the city with the first confirmed case in China, announced its lockdown on January 23, 2020, its daily confirmed cases have dropped from over 2000 at the peak to less than 100.

In academia, a diverse variety of mathematical models have also been devised to fit and predict the development of the pandemic on a regional scale. Viguerie et al., for example, presented a spatial-temporal Susceptible-Exposed-Infectious-Recovered-Deceased (SEIRD) model and used this model to give predictions on the number of infections in some Italian cities under different reopening scenarios [12]. Yang et al. employed a modified SEIR model with migration to predict the peak and the trend of the disease in China, while, at the same time, using a Long Short-term

Memory (LSTM) network to confirm the same conclusions [13]. Likewise, as an alternative to traditional SIR- or SEIR- based models, many studies have adopted machine learning models to predict the COVID-19 outbreak. Pinter et al. proposed a hybrid machine learning model of two algorithms, multi-layered perceptron-imperialist competitive algorithm (MLP-ICA) and adaptive network-based fuzzy inference system (ANFIS). They found that this model has promising results on data from Hungary [9]. Ardabili et al. continued their study by presenting a comparative analysis of various machine learning models, including the hybrid model, and testing them in more regions globally [2]. Their study suggests that machine learning is an effective tool to model the outbreak [2].

1.1 Contributions and Outline

While many models have been put forward to predict the development of the outbreak, we found very few researchers have considered the effects of quarantine within their model and tried to estimate and predict the quarantine strength over time. For example, Dandekar and Barbastathis proposed a modified SIR model with a new quarantine state in their paper, where they applied an artificial neural network as a non-linear function approximator to estimate the time-variant quarantine strength [6]. Given that various quarantine measures have played a non-negligible role in governmental policies to fight against the pandemic, we believe that it is essential to find models that can estimate the effectiveness of those measures to help governments make more prudent decisions regarding quarantine policies.

To fill this gap, this research introduces a preliminary model that recovers the quarantine strength from data and uses the Long Short-term Memory network to make predictions on future periods, based on Dandekar and Barbastathis's works [6]. This research has three parts. The first part replicates Dandekar and Barbastathis's research with some modifications in the data used and the neural network specifications. More details can be found at the beginning of Chapter 2. Section 2.3 discusses some possible limitations of their study. Given these problems, in the second part, we proposed our model in Chapter 3. Section 3.1 covers the derivations of our model, section 3.2 introduces how we used the dataset for prediction purpose, section 3.3 gives the mathematical specifications of the Long Short-term Memory (LSTM) network we used for the prediction, and section 3.4 lists the results after applying the network on an artificial dataset as well as the actual dataset. In section 3.5, we conducted an error analysis on our model to investigate its numerical stability in response to perturbations within data. Finally, in the last part Chapter 4, we provided an example of how we can use the Sparse Identification of Nonlinear Dynamics (SINDy) method to find the governing equations used to generate the specific dataset. Note the part is not directly related to modeling the pandemic but serves as an example to show that it is possible to recover the exact equations from data without knowing much about its actual dynamics.

Chapter 2

SIR-based Models under Consideration of Quarantine

Dandekar and Barbastathis tried to predict the spread of the pandemic using epidemiological models [6]. They proposed a modified Susceptible-Infectious-Recovered (SIR) model with a new compartment “Quarantined”, where the quarantine strength, which is the transformation rate between the infectious and the quarantined, was estimated with a neural network. Comparing with the simulation results from the standard Susceptible-Exposed-Infectious-Recovered (SEIR) model, they concluded that this SIRQ model could capture the approximate trend and the magnitude of the infectious population in most countries examined.

This chapter mainly replicates their study with some differences in the dataset and the neural network structure. More specifically, their paper used the number of accumulated confirmed cases as input, while in our study, we used the number of active cases as input. The reason is that the accumulated number will only increase over time, while active cases may either increase or decrease and is a better indicator of the effectiveness of quarantine. Besides, we used the Sigmoid function in the last layer to ensure the quarantine strength lies in $[0, 1]$.

In section 2.1, we gave some background information on the basic SIR-based models. Then in section 2.2, we replicated the same SIRQ model they proposed, using an artificial neural network to estimate the quarantine strength. In section 2.3, we fitted the data with the standard SEIR model, using grid search to find the optimal model parameters. Comparing the results from the two models, we confirmed their conclusion that the SIRQ model they proposed has a much better predicting ability than the SEIR model.

2.1 Introduction to Susceptible-Infectious-Recovered Epidemiological Models

In epidemiology, researchers usually employ a specific category of mathematical models, called compartmental models, to investigate the development of an infectious disease within a population over a certain period. The compartmental models assume the whole population can be divided into different compartments, and people may progress between those compartments over time. More information related to this topic can be found in the book [3].

The Susceptible-Infectious-Recovered (SIR) model is one of the simplest yet fundamental compartmental models. The standard SIR model assumes that the population is constant and consists of three compartments - “Susceptible”, “Infectious”, and “Recovered”. The susceptible are the healthy individuals that can be infected; the infectious are individuals capable of transmitting the pathogen to others; the recovered are those no longer infectious or died from the pathogen. The diagram below illustrates the transformation process among three compartments.

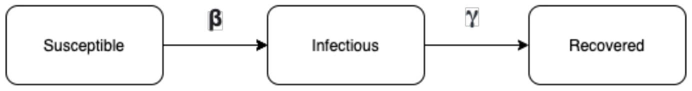


Figure 2.1: The standard SIR model

In Figure 2.1, β is the expected number of effective contacts that an infectious individual can make per unit of time. An effective contact refers to the one that infects a healthy person and transforms him/her from the Susceptible state to the Infectious state. Besides, γ is the inverse of the mean infection period, indicating the proportion of the infectious being transformed into the recovered. This process can be explained by the ordinary differential equations (ODE) below

$$\nu \equiv S(t) + I(t) + R(t) \tag{2.1}$$

$$S'(t) = -\beta \frac{S(t)I(t)}{\nu} \tag{2.2}$$

$$I'(t) = \beta \frac{S(t)I(t)}{\nu} - \gamma I(t) \tag{2.3}$$

$$R'(t) = \gamma I(t), \tag{2.4}$$

where ν is the constant population size, while $S(t)$, $I(t)$, and $R(t)$ are the size for each compartment with respect to time.

The equation (2.2) might not be evident at first sight. We know $\beta I(t)$ is the number of effective contacts that the whole infectious population can make. However, since not all these contacts are with susceptible individuals, we have to multiply $\beta I(t)$ with $\frac{S(t)}{\nu}$, the proportion of the susceptible population within the whole population, to derive the size of the newly infected population.

The Susceptible-Exposed-Infectious-Recovered (SEIR) model is a variant of the SIR model with a new compartment "Exposed", which are those already been infected yet still not infectious. The diagram below illustrates the new model.

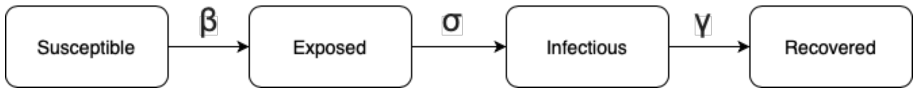


Figure 2.2: The standard SEIR model

The mathematical formulation of the standard SEIR model is slightly different from the SIR model, as shown below

$$\nu \equiv S(t) + E(t) + I(t) + R(t) \tag{2.5}$$

$$S'(t) = -\beta \frac{S(t)I(t)}{\nu} \tag{2.6}$$

$$E'(t) = \beta \frac{S(t)I(t)}{\nu} - \sigma E(t) \tag{2.7}$$

$$I'(t) = \sigma E(t) - \gamma I(t) \tag{2.8}$$

$$R'(t) = \gamma I(t), \tag{2.9}$$

where σ is the inverse of the mean incubation period, and other letters have the same meaning as in the standard SIR model.

2.2 Modeling with the SIRQ Model

The Susceptible-Infectious-Recovered-Quarantined (SIRQ) model in Dandekar and Barbastathis's paper is also a variant of the SIR model. A significant difference between the two is that the SIR model does not consider governments' actions to prevent the spread of the disease. As introduced in their paper, the effects of those actions can be quantified by introducing an additional time-dependent variable Q representing the quarantine strength [6]. Their model comes in the following form

$$\nu \equiv S(t) + E(t) + I(t) + R(t) \tag{2.10}$$

$$S'(t) = -\beta \frac{S(t)I(t)}{\nu} \tag{2.11}$$

$$I'(t) = \beta \frac{S(t)I(t)}{\nu} - (\gamma + Q(I(t), R(t), T(t))I(t) \tag{2.12}$$

$$R'(t) = \gamma I(t) \tag{2.13}$$

$$T'(t) = Q(I(t), R(t), T(t))I(t), \tag{2.14}$$

where the new compartment $T(t)$ represents the quarantined population, and the authors assumed that once an infectious individual is quarantined, the individual becomes no longer infectious.

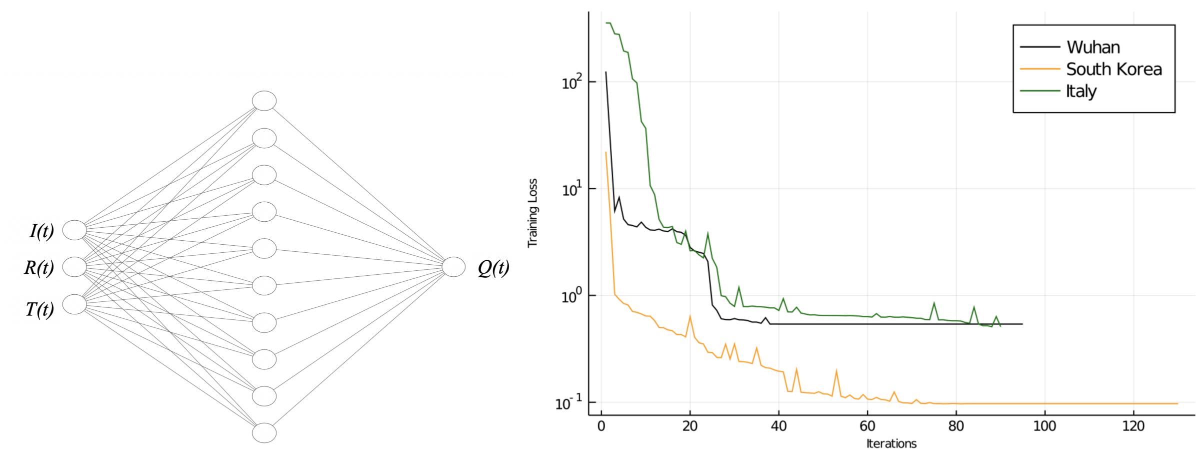
The quarantine strength $Q(I(t), R(t), T(t))$ is approximated using an n-layer neural network with weights W_1, W_2, ..., W_n, activation function r, and the input vector $U = (I(t), R(t), T(t))$

$$Q(I(t), R(t), T(t)) \equiv \mathrm{NN}(W, U) = r(W_n r(W_{n-1} ... r(W_1 U))). \tag{2.15}$$

As in the original paper, we implemented the artificial neural network with 3 densely connected layers and 10 nodes in the hidden layer, shown in Figure 2.3a. We applied the ReLU function as the activation function in the intermediate layer and the Sigmoid function in the output layer to ensure $Q(I(t), R(t), T(t))$ is between 0 and 1. The model contains 53 trainable parameters (β, γ and 51 weight parameters in the network). The optimal parameters are found to minimize the loss function

$$L(\beta, \sigma, \gamma, W, U) = w_1 \| \log(I(t)) - \log(I_{data}(t))\|^2 + w_2 \| \log(R(t)) - \log(R_{data}(t))\|^2, \tag{2.16}$$

where, for simplicity, we took equal importance for the infectious and recovered population ($w_1 = w_2$). We carried out the optimization procedures following a similar approach listed in [10], using the BFGS optimizer with a maximum of 300 iterations.



(a) **(b)**

Figure 2.3: Neural network and training loss
(a) Neural network architecture. (b) Training loss at each iteration.

We used the data from Wuhan in China, South Korea, and Italy with the initial condition $S(t = 0) = 11$ million, 52 million, and 60 million, respectively. The other initial conditions are $I(t = 0) = 500$, $R(t = 0) = 10$, and $T(t = 0) = 10$, which is the same for all countries. The data were obtained from the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University. For each country, only dates after the accumulated infected number greater than 500 were selected. We chose 80 days for each country, where the first 40 days were used for training and the remaining for testing. For Wuhan, data from January 24 to April 13 were used; for Italy, data from February 27 to May 17 were used; and for South Korea, data from February 23 to May 13 were used. The training loss per iteration for each region is shown in Figure 2.3b, while the simulated results and its corresponding estimated quarantine strength are shown in Figure 2.4. Note that the black dash lines in the figures are used to indicate the split for training and test set.

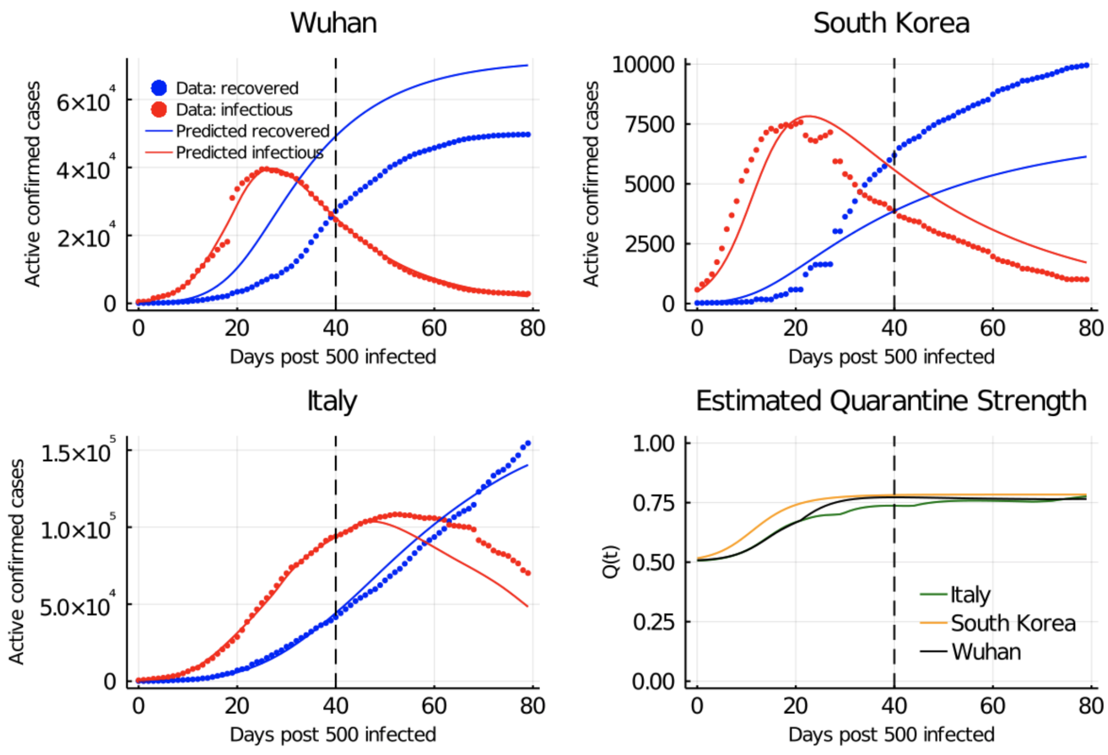


Figure 2.4: Simulation with quarantine control

For Wuhan in Figure 2.4, the predicted infected curve has successfully forecast the overall development of the infectious population, while the predicted recovered curve has significantly overestimated the recovered population. The case is similar for South Korea in Figure 2.4, as the model has correctly predicted the trend of the infectious population but at the same time remarkably underestimated the recovered population. However, the prediction on Italy is much more accurate: the predicted recovered population comes very close to the data, and the predicted infectious population has only slightly underestimated the actual population on the test set. After training, the parameters and the range of the quarantine strength $Q(I(t), R(t), T(t))$ are listed in Table 2.1.

Region	β	γ	Range of Q(I(t), R(t), T(t))
Wuhan	0.86	0.024	0.50-0.77
Italy	1.0	0.036	0.51-0.78
South Korea	0.71	0.03	0.52-0.78

Table 2.1: Estimated β, γ, and the range of $Q(I(t), R(t), T(t))$ for each region

2.3 Discussion

As a comparison, we fitted the standard SEIR model (equations 2.5 to 2.9) to the same dataset from Wuhan, South Korea, and Italy. The initial condition for the exposed population is $E(t = 0) = 10000$, while other states have the same initial condition as in section 2.2. We found the optimal parameters β, σ, γ that minimize the same loss function in equation 2.16 with $w_1 = w_2$ by using grid search with step size $d = 0.01$. After solving for the parameters, we then ran the ODE forward to find the number of predicted cases for the time interval. The results are shown below in Figure 2.5.

Comparing Figure 2.4 with 2.5, we observe that the SEIR model has a much worse predictive ability than the SIRQ model in Wuhan, as the predicted infectious and recovered population in Figure 2.5 grow exponentially for the Wuhan city. Similarly, comparing two figures for Italy and South Korea, we can conclude that, for all regions, the predicted curves based on the SIRQ model fit the actual data much better than those from the SEIR model.

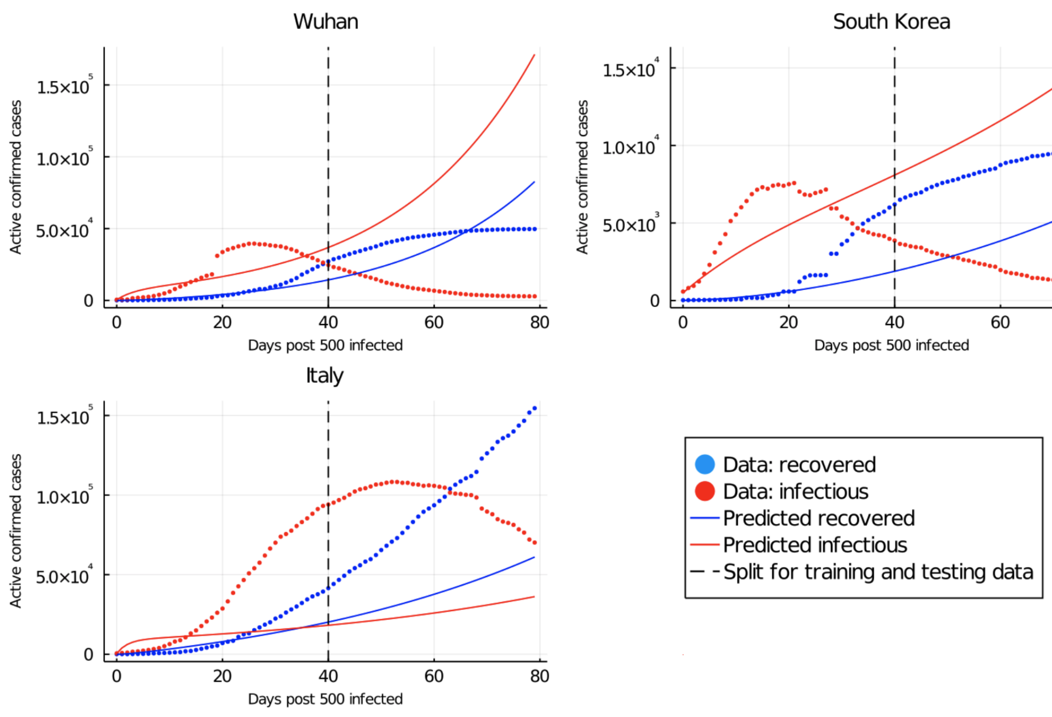


Figure 2.5: Simulation without quarantine control

However, we also found some limitations concerning the SIRQ model in section 2.2. While the model introduces a new state T(t), it does not have a transformation to other states. This is not possible, as it is known that, in the end, the majority of those quarantined will recover from the disease. For example, although the model makes minor errors with the infectious and recovered population in Wuhan, looking at the quarantined population trend (see Figure 2.6), it can be easily discovered that the size is too large to be reasonable. Adding a transformation from the quarantined to the recovered with even the same rate as the model recovery rate γ will remarkably ruin the results. The deviation suggests that there are other possible factors that are not yet reflected in the model but implicitly slow down the development of the pandemic. This inspired us to extend the definition of quarantine into a more broadly-defined situation. It does not directly decrease the number of the infectious population, but serves to offset β, which is the expected number of effective contacts that an infectious individual can make per unit of time. Some examples of the broadly-defined quarantine measure are wearing masks or social distancing. If, for example, the government mandates people to do so, the number of healthy people being infected per unit of time will undoubtedly drop.

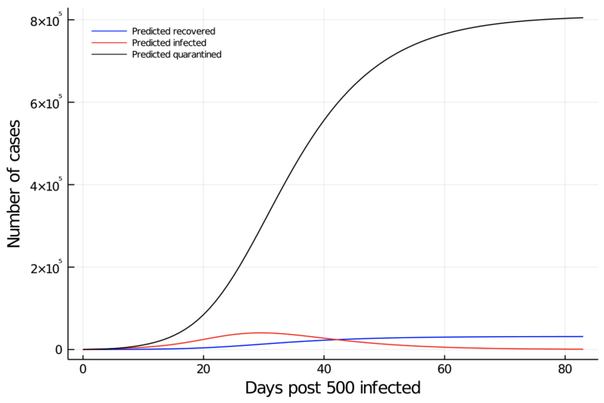


Figure 2.6: The predicted quarantined population (Wuhan)
recovered (blue), infectious (red), and quarantined (black) population

Besides, some improvements can be made to the SIRQ model. For example, there is no need to use the neural network, shown in Figure 2.3a, to estimate the function values of $Q(I(t), R(t), T(t))$. As the neural network itself represents a function $f : \mathbb{R}^3 \to \mathbb{R}$ whose values are defined on $[0, 1]$, we can discretize the function domain into small cubes, calculate the function values at each vertex, and approximate the entire function using methods such as trilinear interpolation.

Chapter 3

SEIR-based Models for Quarantine Strength Estimation and Prediction

In light of the problems described in the section 2.3, we proposed a new SEIR-based model where we introduced a new time-dependent variable $P(t)$, representing the effects of the broadly-defined quarantine measures. Unlike the original model where the parameters are chosen first, and then ODE equations are solved forward, our proposed method goes backward where the model parameters and the value of $P(t)$ are solved directly from the data. Then we built a Long Short-term Memory (LSTM) network to make future period predictions on $P(t)$. For simplicity, from now on, we will refer to the broadly-defined quarantine rate as “policy rate”.

In section 3.1, we give the mathematical derivation of the new model. Then in section 3.2, we described how the data were used to train the neural network and make predictions for future periods. Section 3.3 provides detailed explanations about the structure of the LSTM network, and section 3.4 lists the training and prediction results on an artificial dataset and the actual dataset. Section 3.5 conducts an error analysis on the model we proposed to investigate the model’s sensitivity in response to perturbations in data. Finally, in section 3.6, we discussed some limitations of this study and pointed out several possible future directions that can further improve our model.

3.1 Model Setting

The new model takes the following form:

$$\nu \equiv S(t) + E(t) + I(t) + R(t) \tag{3.1}$$

$$S'(t) = -(\beta - P(t))\frac{S(t)I(t)}{\nu} \tag{3.2}$$

$$E'(t) = (\beta - P(t))\frac{S(t)I(t)}{\nu} - \sigma E(t) \tag{3.3}$$

$$I'(t) = \sigma E(t) - \Gamma(t)I(t) \tag{3.4}$$

$$R'(t) = \Gamma(t)I(t), \tag{3.5}$$

where $0 \leq t \leq T$ and $S(0) = S_0, E(0) = E_0, I(0) = I_0, R(0) = R_0$.

To solve backward for $P(t)$, in the ideal case, we are given $I(t)$ and $R(t)$ as continuous functions. Then, $I'(t)$ and $R'(t)$ can be obtained by differentiating $I(t)$ and $R(t)$ with respect to t. As a result, $P(t)$ can be solved as follows:

$$E(t) = \frac{I'(t) + R'(t)}{\sigma} \tag{3.6}$$

$$S'(t) = -(E'(t) + \sigma E(t)) \tag{3.7}$$

$$S(t) = S_0 + \int_0^t S'(x)dx \tag{3.8}$$

$$P(t) = \beta - \frac{\nu(E'(t) + \sigma E(t))}{S(t)I(t)}. \tag{3.9}$$

However, in reality, we are only given discrete measurements $I(t_j)$ and $R(t_j)$, where $0 \leq t_1 \leq t_2 \leq ... \leq t_N$. In order to find an expression for P_j, we can first estimate the values of $I'(t_j)$ and $R'(t_j)$ using the finite difference approximation

$$I'(t_j) \approx I'_j = \frac{I_{j+1} - I_j}{h_j} \tag{3.10}$$

$$R'(t_j) \approx R'_j = \frac{R_{j+1} - R_j}{h_j}, \tag{3.11}$$

where $h_j = t_{j+1} - t_j = 1$, as we are given daily measurements.

Then we are able to solve for P_j, the approximation of $P(t_j)$, in the following way:

$$E_j = \frac{I'_j + R'_j}{\sigma} \tag{3.12}$$

$$E'(t_j) \approx E'_j = E_{j+1} - E_j \tag{3.13}$$

$$S'(t_j) \approx S'_j = -(E'_j + \sigma E_j) \tag{3.14}$$

$$S_j = S_0 + \sum_{k=0}^{j-1} S'_k \tag{3.15}$$

$$P_j = \beta - \frac{\nu(E'_j + \sigma E_j)}{S_j I_j}. \tag{3.16}$$

Like previous notations, here E'_j and S'_j are the estimates of $E'(t)$ and $S'(t)$ at time t_j, while E_j and S_j are the approximations of $E(t)$ and $S(t)$ at t_j.

3.2 Training Process

We used the same dataset as in Chapter 2. The daily policy rate P_j was calculated based on the algorithm discussed in section 3.1. We only used the data from the day when 500 accumulated infected cases are recorded, with a total period of 80 days. The data was then partitioned into three parts, where the first part was used to train the model, the second part used for hyper-parameter tuning, and the last part to measure the model performance. The training set consists of data from the first 40 days, while the validation set has 14 days and the test set has the remaining 26-day data in order. Here, we intended to conduct single-step forecasting using the sliding window algorithm: the input is a window comprised of function values of 7 days, and the output is the function value on the next day. Since the whole testing period is greater than each prediction's length, we also employed the "walk-forward" approach to complete the prediction. Using this approach, the model will first make a prediction on the first day, and then the actual value for that day becomes available for the model to make another prediction on the subsequent day.

3.3 Neural Network Structure

Long Short-term Memory (LSTM) is one type of recurrent neural networks (RNNs) that is widely used for long-term sequence prediction. An introduction to the LSTM network can be found in its original paper by Hochreiter and Schmidhuber [8]. An RNN is a neural network where the output from one time-step is provided as an input for the subsequent time-step. This unique structure allows the neural network to have direct memory of the previous output and make decisions based on both the input of the current time step and output from the previous steps. Therefore, RNNs are popular in time-series sequence prediction where future time steps are related to previous time steps. For example, RNNs have been proven to be a powerful tool in missing value imputation [5] and predicting chaotic time series data [14]. However, sometimes it can be challenging to train a general RNN to learn long-term temporal relationships, as the long-term information has to sequentially pass through all previous cells to reach the current cell. For a deep RNN, repeatedly applying the activation function in each cell easily results in the gradient vanishing problem, meaning that the gradient of the loss function approaches zero exponentially in time in the back-propagation stage [7].

LSTM networks overcome this by having a series of internal gates within each cell that control when the memory is modified, outputted, and forgotten. In this way, the long-term information can go through the cells without being modified. An in-depth explanation of the LSTM nodes and how they have been designed to prevent gradient vanishing problems can also be found in its original paper [8].

Mathematically speaking, the LSTM neural network can be considered as a series of operations performed on an input matrix on dimension $\mathbb{R}^{n_t \times n_f}$, where n_t, n_f are the number of input time-steps and the number of features. Remember we are using the values of $P(t)$ from 7 days to predict its value on the next day. Every sample $\boldsymbol{p}_k$ is of size $(7, 1)(n_t = 7, n_f = 1)$ and the output sequence is of size $(1, 1)$. More specifically, given a vector $\boldsymbol{p}$ consisting of all consecutive values of $P(t)$, we know the k-th ($k \geq 1$) input vector is $\boldsymbol{p}_k = \boldsymbol{p}(k : k + 6)$ and the ground truth for prediction is $\boldsymbol{p}(k + 7)$. Also note here the parenthesis operation has the same meaning as in MATLAB to index and slice a vector.

The LSTM layer has n_l nodes, and so the output $\boldsymbol{a}^{[1]}$ is of dimension $\mathbb{R}^{n_l}$. Each LSTM node has three gates, an input gate, a forget gate, an output gate, and two states, a cell state and a hidden state. The hidden state is the result returned from the output gate, and changes when new data are passed into the node. The forget gate and the input gate take the previous hidden state and the current input data as the input. The forget gate decides which information in the input should be kept or discarded, while the input gate is used to discover which information should be used to update the cell state. The outputs from the forget gate and the input gate are then put together to calculate the new cell state. Finally, the updated cell state, the previous hidden state, and the current input data are fed into the output gate, which returns the new hidden state. A detailed list of operations performed within those gates can be found in its original paper [8]. In short,

$$\boldsymbol{a}^{[1]} = \text{LSTM}(\boldsymbol{p}_k, \boldsymbol{\theta}^{[1]}, \boldsymbol{b}^{[1]}), \tag{3.17}$$

where $\boldsymbol{\theta}^{[1]}, \boldsymbol{b}^{[1]}$ are the weights and bias that need to be learned. When $n_f = 1$, we know the weights vector $\boldsymbol{\theta}^{[1]}$ has dimension $\mathbb{R}^{8n_l+4n_l^2}$ based on the description from its original paper [8].

Finally, we have one dense layer of dimension n_d, which serves to convert the shape of the output vector

$$\boldsymbol{y} = \sigma(\boldsymbol{a}^{[1]}\boldsymbol{\theta}^{[2]} + \boldsymbol{b}^{[2]}). \tag{3.18}$$

Here $\boldsymbol{y} \in \mathbb{R}^{n_d}$ is the output vector and the non-linear activation function $\sigma : \mathbb{R} \to \mathbb{R}$ is applied component-wise with $\sigma(x) = max\{x, 0\}$. $\boldsymbol{\theta}^{[2]}$ and $\boldsymbol{b}^{[2]}$ are the weights and bias vector in the fully-connected layer with $n_l \times n_d$ and n_d parameters, respectively.

The neural network $NN(\cdot)$ consists of two layers, an LSTM layer followed by a fully-connected dense layer, described above. The objective of the neural network $NN(\cdot)$ is to find a set of parameters $\boldsymbol{\Theta} = [\boldsymbol{\theta}^{[1]}, \boldsymbol{\theta}^{[2]}], \boldsymbol{B} = [\boldsymbol{b}^{[1]}, \boldsymbol{b}^{[2]}]$ such that the cost function over multiple examples is minimized

$$\min_{\boldsymbol{\Theta},\boldsymbol{B}} \frac{1}{n_s - 7} \sum_{k=1}^{n_s-7} \|NN(\boldsymbol{p}(k : k+6), \boldsymbol{\Theta}, \boldsymbol{B}) - \boldsymbol{p}(k+7)\|_2^2, \tag{3.19}$$

where $\boldsymbol{p}$ is the vector consisting of all consecutive values of $P(t)$ and n_s is the length of $\boldsymbol{p}$.

To summarize, in the neural network we adopt, each input example $\boldsymbol{p}_k$ is of size (7, 1). The first layer is an LSTM layer with 30 nodes ($n_l = 30$), and the next dense layer has only one node ($n_d = 1$). All layers use ReLU as the activation function. The output is a vector with a length of 1. The model is built in Python using the Keras platform with TensorFlow as the back-end. A diagram of the model is shown below.

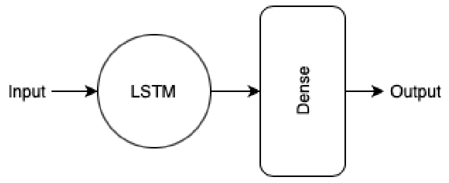
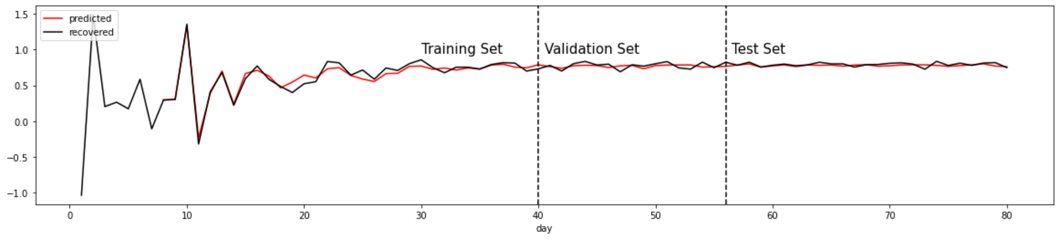


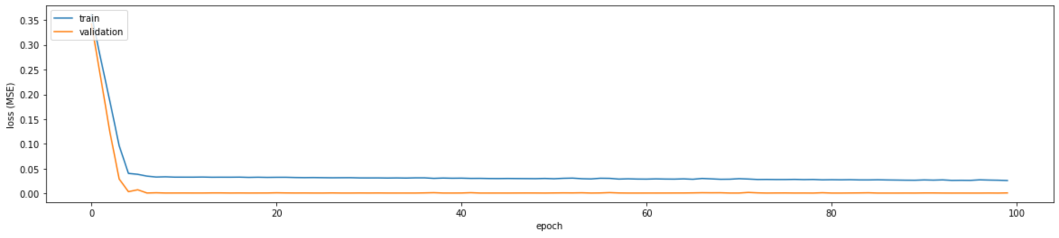
Figure 3.1: The LSTM network structure

3.4 Results

We used the model described in section 3.1 to derive the policy rate $P(t)$ from Italy and South Korea data. There is insufficient data for Wuhan city because the epidemic there ends quite shortly due to the local government's successful quarantine policies. As a result, we did not use the Wuhan dataset in this section. The Italy and South Korea datasets we used are the same as in Chapter 2, with the same initial conditions for each region. There are three constants in equation 3.16: ν is the constant population size, which is equal to $S(0)$ for each region; σ is the inverse of the average incubation period for an exposed individual with COVID-19 to become infectious; β is the constant transmission rate. According to the research by Rădulescu et al., σ is between $0.05 - 0.1$ and β is between $0.1 - 0.3$ for COVID-19 [11]. In this study, we set $\sigma = 0.07$, as [11] shows the mean incubation period is 14 days, and $\beta = 0.25$ for each region. After recovering the daily policy rate, we then trained the neural network in section 3.3 to make predictions for future periods. Figure 3.2 and Figure 3.3 are the training and testing results for each region.

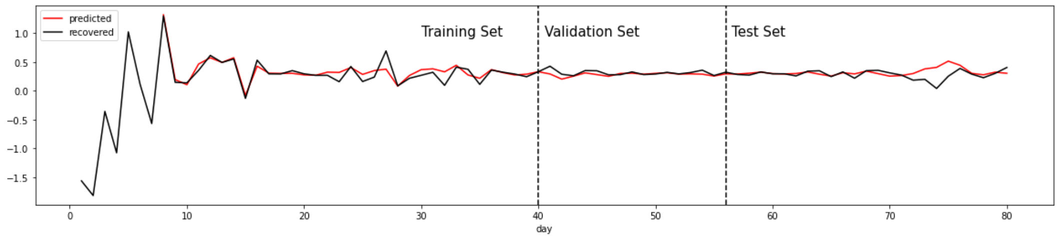


(a)

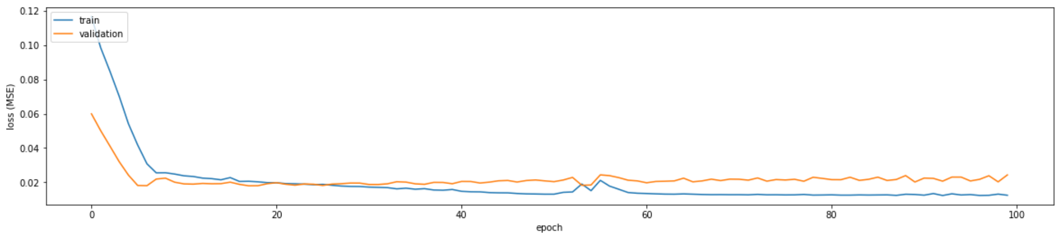


(b)

Figure 3.2: Prediction on policy rate (Italy)
(a) Prediction results. (b) Loss on the training and validation set.



(a)

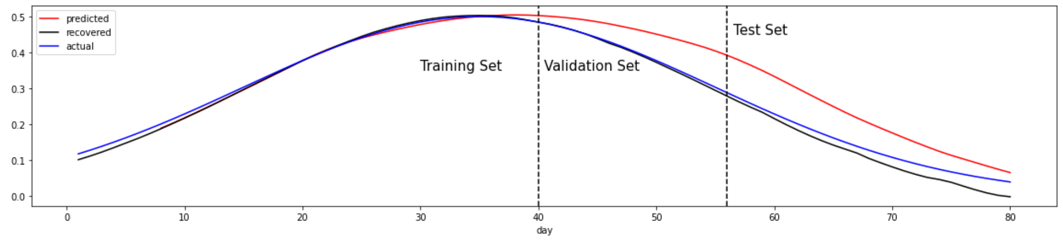


(b)

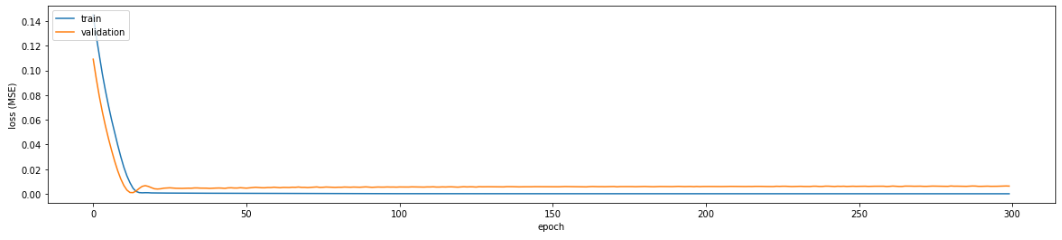
Figure 3.3: Prediction on policy rate (South Korea)
(a) Prediction results. (b) Loss on the training and validation set.

It is interesting to observe that, except for the initial period, the recovered policy rate curves for both regions seem to converge to a specific value later on, which is 0.8 for Italy and 0.3 for South Korea.

To see if our model can make effective predictions on future policy rates, we repeated the procedures mentioned above on an artificial data set. The artificial data set has a total time-span of 80 days, produced by solving the standard SEIR model forward with $S_0 = 100000, E_0 = 1000, I_0 = 50, R_0 = 10$ and $\sigma = 0.14, \beta = 0.5, \gamma = 0.05$. The split of the training, validation, and test data sets on the artificial data set is the same as on the actual data set (i.e. training: 40 days, validation: 14 days, test: 26 days). The neural network was trained to learn from the training set, tune hyper-parameters on the validation set, and make predictions on the test set. The prediction results were then compared to the actual policy curve to check if the neural network can successfully learn the patterns of the curve. The comparison is shown in the figures below. It can be observed that the network can capture the patterns of the curve and therefore makes effective predictions on future trends. In the figure, the blue curve is the actual policy rate curve used in the dataset generation process; the black curve is the policy curve recovered from the dataset using the algorithm described in section 3.2.1; the red curve is the results predicted by the neural network.



(a)



(b)

Figure 3.4: Prediction on policy rate (the artificial dataset)
(a) Prediction results. (b) Loss on the training and validation set.

3.5 Error Analysis

Looking at the Figure 3.4a, we found a remarkable difference between the actual and recovered curves, which implies that significant error exists within our model. To further investigate this problem, we conducted the following error analysis to check our algorithm's numerical stability.

Given $\boldsymbol{i}, \boldsymbol{r} \in \mathbb{R}^n$, a vectorized version of equation 3.10 - 3.16 is listed below

$$\boldsymbol{i'} = \boldsymbol{D_t}\boldsymbol{i}, \boldsymbol{r'} = \boldsymbol{D_t}\boldsymbol{r} \tag{3.20}$$

$$\boldsymbol{e} = \frac{\boldsymbol{i'} + \boldsymbol{r'}}{\sigma} \tag{3.21}$$

$$\boldsymbol{e'} = \boldsymbol{D_t}\boldsymbol{e} \tag{3.22}$$

$$\boldsymbol{s'} = -(\boldsymbol{e'} + \sigma\boldsymbol{e}) \tag{3.23}$$

$$\boldsymbol{s} = \boldsymbol{D_i}\boldsymbol{s'} \tag{3.24}$$

$$\boldsymbol{p} = \beta - \nu(\boldsymbol{e'} + \sigma\boldsymbol{e}) \oslash (\boldsymbol{s} \odot \boldsymbol{i}) \tag{3.25}$$

$$= \beta + \nu\boldsymbol{s'} \oslash (\boldsymbol{s} \odot \boldsymbol{i}), \tag{3.26}$$

where $\odot$ is the component-wise product (Hadamard product), $\oslash$ is the component-wise division, $\boldsymbol{D_t}$ is the differential operator, and $\boldsymbol{D_i}$ is the integral operator.

Then, we are able to derive the Jacobian of $\boldsymbol{p}$ with respect to $\boldsymbol{i}$

$$\frac{\partial \boldsymbol{i'}}{\partial \boldsymbol{i}} = \boldsymbol{D_t}^T, \frac{\partial \boldsymbol{r'}}{\partial \boldsymbol{i}} = \boldsymbol{0} \tag{3.27}$$

$$\frac{\partial \boldsymbol{e}}{\partial \boldsymbol{i}} = \frac{\partial}{\partial \boldsymbol{i}}(\frac{\boldsymbol{i'} + \boldsymbol{r'}}{\sigma}) = \frac{\boldsymbol{D_t}^T}{\sigma} \tag{3.28}$$

$$\frac{\partial \boldsymbol{e'}}{\partial \boldsymbol{i}} = \frac{\partial}{\partial \boldsymbol{i}}(\boldsymbol{D_t}\boldsymbol{e}) = \frac{(\boldsymbol{D_t}^T)^2}{\sigma} \tag{3.29}$$

$$\frac{\partial \boldsymbol{s'}}{\partial \boldsymbol{i}} = -\frac{\partial}{\partial \boldsymbol{i}}(\boldsymbol{e'} + \sigma\boldsymbol{e}) = -(\frac{(\boldsymbol{D_t}^T)^2}{\sigma} + \boldsymbol{D_t}^T) \tag{3.30}$$

$$\frac{\partial \boldsymbol{s}}{\partial \boldsymbol{i}} = \frac{\partial}{\partial \boldsymbol{i}}(\boldsymbol{D_i}\boldsymbol{s'}) = -(\frac{(\boldsymbol{D_t}^T)^2\boldsymbol{D_i}^T}{\sigma} + \boldsymbol{D_t}^T\boldsymbol{D_i}^T) \tag{3.31}$$

$$\frac{\partial(\boldsymbol{s} \odot \boldsymbol{i})}{\partial \boldsymbol{i}} = \frac{\partial \boldsymbol{s}}{\partial \boldsymbol{i}}\text{diag}(\boldsymbol{i}) + \text{diag}(\boldsymbol{s})\frac{\partial \boldsymbol{i}}{\partial \boldsymbol{i}} = \frac{\partial \boldsymbol{s}}{\partial \boldsymbol{i}}\text{diag}(\boldsymbol{i}) + \text{diag}(\boldsymbol{s}) \tag{3.32}$$

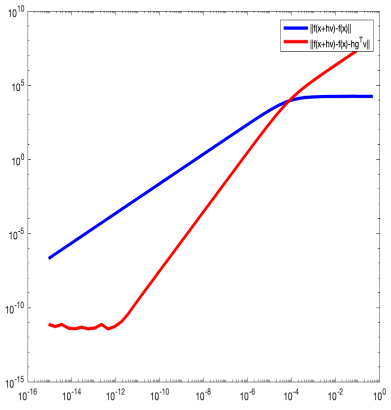
$$\frac{\partial \boldsymbol{p}}{\partial \boldsymbol{i}} = \nu\{\frac{\partial \boldsymbol{s'}}{\partial \boldsymbol{i}}\text{diag}(\boldsymbol{1} \oslash (\boldsymbol{s} \odot \boldsymbol{i})) + \text{diag}(\boldsymbol{s'})\frac{\partial}{\partial \boldsymbol{i}}[\boldsymbol{1} \oslash (\boldsymbol{s} \odot \boldsymbol{i})]\} \tag{3.33}$$

$$= \nu[\frac{\partial \boldsymbol{s'}}{\partial \boldsymbol{i}}\text{diag}(\boldsymbol{1} \oslash (\boldsymbol{s} \odot \boldsymbol{i})) - \frac{\partial(\boldsymbol{s} \odot \boldsymbol{i})}{\partial \boldsymbol{i}}\text{diag}(\boldsymbol{1} \oslash (\boldsymbol{s} \odot \boldsymbol{i}))^2\text{diag}(\boldsymbol{s'})]. \tag{3.34}$$

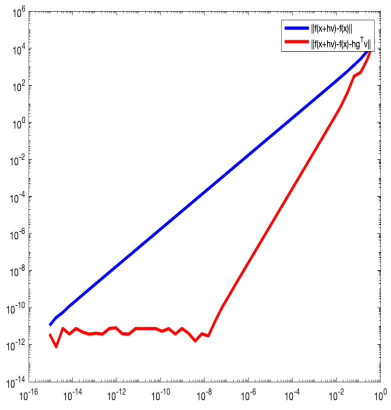
In a similar way, the Jacobian of $\boldsymbol{p}$ with respect to $\boldsymbol{r}$ is

$$\frac{\partial \boldsymbol{p}}{\partial \boldsymbol{r}} = \nu[\frac{\partial \boldsymbol{s}'}{\partial \boldsymbol{r}}\text{diag}(\mathbf{1} \oslash (\boldsymbol{s} \odot \boldsymbol{i})) - \frac{\partial(\boldsymbol{s} \odot \boldsymbol{r})}{\partial \boldsymbol{r}}\text{diag}(\mathbf{1} \oslash (\boldsymbol{s} \odot \boldsymbol{i}))^2\text{diag}(\boldsymbol{s}')]. \tag{3.35}$$

Taylor's theorem can be used to check whether the calculated gradient $\boldsymbol{g} \in \mathbb{R}^n$ is a gradient of $f : \mathbb{R}^n \to \mathbb{R}$ at any $\boldsymbol{x}$. For a random vector $\boldsymbol{v}$ and $h \to 0$, comparing $\|f(\boldsymbol{x} + h\boldsymbol{v}) - f(\boldsymbol{x})\|$ and $\|f(\boldsymbol{x} + h\boldsymbol{v}) - f(\boldsymbol{x}) - h\boldsymbol{g}^T\boldsymbol{v}\|$, by the Taylor's theorem, the former should converge linearly to zero, while the latter converges in a quadratic way. In a log-log plot, the slope for the latter should be nearly twice of the former. Below shows the results for $\frac{\partial \boldsymbol{p}}{\partial \boldsymbol{i}}$ and $\frac{\partial \boldsymbol{p}}{\partial \boldsymbol{r}}$.



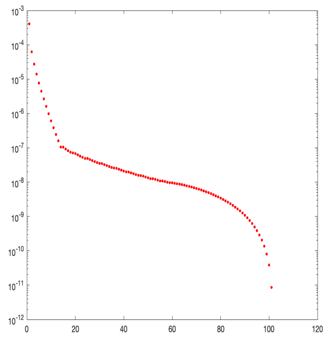
(a)



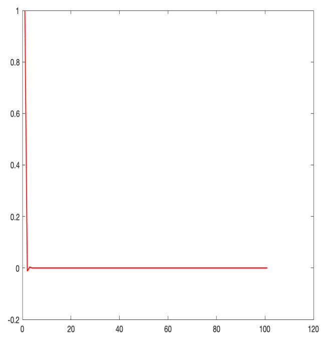
(b)

Figure 3.5: The gradients of $\boldsymbol{p}$
(a) The gradient of $\boldsymbol{p}$ with respect to $\boldsymbol{i}$. (b) The gradient of $\boldsymbol{p}$ with respect to $\boldsymbol{r}$.

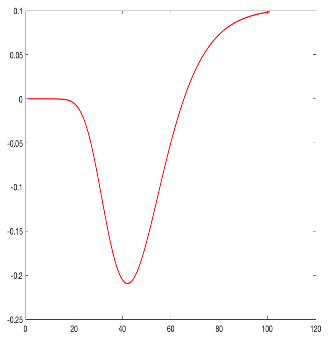
To analyze the sensitivity of the algorithm to the perturbations of the data, we used the same dataset as in section 3.2.4 and calculated the Jacobian of $\boldsymbol{p}$ with respect to $\boldsymbol{i}$ and $\boldsymbol{r}$. We then combined the two Jacobian matrices and did a singular vector decomposition (SVD) on the merged Jacobian matrix. The singular values and the maximum/minimum left and right singular vectors are plotted below.



(a)

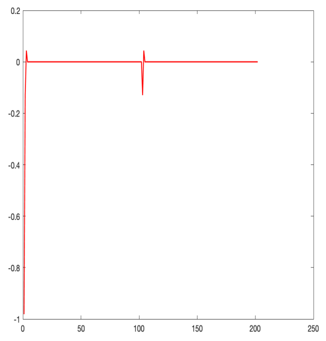


(b)

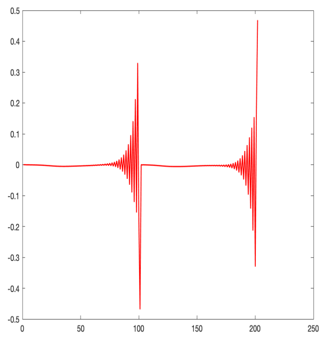


(c)

Figure 3.6: Singular value decomposition results (I)
(a) Singular values. (b) The maximum left singular vector. (c) The minimum left singular vector.



(d)



(e)

Figure 3.6: Singular value decomposition results (II)
(d) The maximum right singular vector. (e) The minimum right singular vector.

We can see the singular values in Figure 3.6a decay gradually to zero with no noticeable gaps in the spectrum, while the singular vectors in Figure 3.6 oscillate around zero more frequently as the corresponding singular values decrease. Therefore, we conclude that this is a discrete ill-posed problem, and the solutions to this problem, which is $\boldsymbol{p}$, are very sensitive to the perturbations in the data. This conclusion further corroborates our finding in Figure 3.4a that significant error exists between the actual curve and the recovered curve.

3.6 Discussion

Based on results from section 3.4, our proposed model can recover the approximate relationship from data and make meaningful predictions on future periods. However, it also has many limitations, including but not limited to error sensitivity, generalizability, and interpretability. As we have discussed in section 3.5, the approach we took to recover $P(t)$ from data is ill-posed, and thus the solutions we found are sensitive to the errors within data. For real-world datasets, as the error is inevitable during the data-collection process, the model solution may deviate significantly from the actual solution. Various regularization techniques could help to remediate this problem. Regularization can be applied either when pre-processing the dataset to remove excessive fluctuations within data (e.g. total variation regularization), or during the model training process by adding a penalty term (e.g. L2-norm) to prevent overfitting in model solutions. For future studies, both schemes can be tested to see which one works better for this problem.

Another problem with this model is the generalizability. The data for this study was collected in June 2020, and the majority of work was done in July and August. Till then, all the regions we examined have demonstrated a single peak in their distribution of the active confirmed cases. However, we also observed that beginning from October 2020, Italy and South Korea had experienced the second wave of the pandemic, resulting in multiple and higher peaks in their daily confirmed cases. As our model has not been tested on datasets with multiple peaks, we can say nothing about its generalizability, and it is likely that our model does not generalize well to new situations. So for future directions, when evaluating the model performance, we should further enlarge the datasets to include the most recent pandemic situation.

In addition, our model also suffers from the problem of low interpretability. We employed a vanilla LSTM neural network when making the prediction for future periods. In each LSTM node, there are multiple addition, multiplication, and concatenation operations involved with many non-linear activation functions applied. As a result, a single prediction can involve thousands of mathematical operations, and together the neural network becomes so complicated that no human is able to follow precisely. It is like a black box: people feed data and get the output, but they have no clue what is happening within the box. The complexity of neural networks shapes their ability to approximate any function in the world, but at the same time also leads to their low interpretability.

As a result, researchers usually face the trade-off between model expressivity and model interpretability and are forced to make a hard decision between a generic model and an expressive model. For example, neural networks are a generic model that can simulate any complex relationship as long as the network is deep enough. On the other hand, an SEIR model is an expressive model that only explains a particular relationship. Adopting an expressive model has the advantage of being simple, transparent, and interpretable, but may fit poorly if this model does not apply to the specific problem that we are investigating.

Therefore, the most favorable situation is that we know the exact mathematical equations underlying the problem. To this end, many methods have been proposed to discover the governing equations directly from data, among which is the Sparse Identification of Nonlinear Dynamics (SINDy) method. The following chapter gives a brief work-through of the SINDy method and provides an example of using this method to recover the underlying equations from an artificially generated dataset.

Chapter 4

Recovering the Governing Equations Using Sparse Identification of Nonlinear Dynamics (SINDy) Method

The Sparse Identification of Nonlinear Dynamics (SINDy) method has been recently proposed by Brunton et al. to recover the governing equations from data [4]. The model takes the assumption that most systems has only a few terms in their governing equations and thus they are sparse in a high-dimensional function space. Instead of performing a brute-force search to find them, the recent advance in machine learning makes it possible to discover the governing equations in much less time. More specifically, given a trajectory of the state $\mathbf{x}(t)$, we can construct a data matrix $\mathbf{X} = \left[\mathbf{x}^T(t_1)\ \mathbf{x}^T(t_2)\ \dots\ \mathbf{x}^T(t_n)\right]^T$ and a time derivative matrix $\dot{\mathbf{X}} = \left[\dot{\mathbf{x}}^T(t_1)\ \dot{\mathbf{x}}^T(t_2)\ \dots\ \dot{\mathbf{x}}^T(t_n)\right]^T$ by either measuring or numerically approximating each derivative of $\mathbf{x}(t)$ from $\mathbf{X}$.

Then we can construct another matrix $\Theta(\mathbf{X})$ which consists of all candidate functions of the columns of $\dot{\mathbf{X}}$. An example of $\Theta(\mathbf{X})$ is shown below. Note here I only give an example where all candidate functions are polynomials of the data matrix **X**, and it can certainly contain other basis, such as sine and cosine functions.

$$\Theta(\mathbf{X}) = \begin{bmatrix} | & | & | & \cdots \\ 1 & X & X^2 & \cdots \\ | & | & | & \cdots \end{bmatrix}$$

A sparse regression problem can then be set up to determine Ξ, the coefficients of the candidate functions, such that

$$\dot{\mathbf{X}} = \Theta(\mathbf{X})\Xi, \tag{4.1}$$

where $\Xi = [\xi_1\ \xi_2\ ...\ \xi_n]$ determines which candidate functions are active. Specifically, Ξ can be found by solving the optimization problem

$$\Xi = \min ||\dot{\mathbf{X}} - \Theta(\mathbf{X})\Xi||_2 + \lambda||\Xi||_1, \tag{4.2}$$

where λ is the strength of regularization. We can thus determine the governing equations underlying the data matrix **X** based on the candidate functions matrix $\Theta(\mathbf{X})$ and coefficient matrix Ξ.

Below we gave an example that the parameters of Susceptible-Infectious-Recovered (SIR) models are recoverable using the SINDy method. This process is done in Julia with "DataDriven-Diffeq" package. The data matrix **X** and the derivative matrix $\dot{\mathbf{X}}$ are constructed by solving the standard SIR model, shown in section 2.1, forward with parameters $\beta = 0.2$ and $\gamma = 0.1$ and the initial condition $S(0) = 100, I(0) = 10, R(0) = 10$. The candidate functions consist of all monomials of S, I, R up to the third degree.

Usually, with a large regularization parameter λ, we have fewer terms in the recovered equations (high bias), but with a smaller regularization parameter, the recovered equations may contain redundant terms (high variance). To find the optimal regularization parameter λ, we begin with a large value and gradually decrease it by a factor of 0.8. Figure 4.1 was created to help us choose the trade-off between the bias and the variance in the recovered equations. The x-axis is $||\dot{\mathbf{X}} - \Theta(\mathbf{X})\Xi||_2$, which is the L_2-error between the recovered results and the derivative matrix, while the y-axis is $||\Xi||_1$, which is the L_1-norm of the coefficient matrix Ξ.

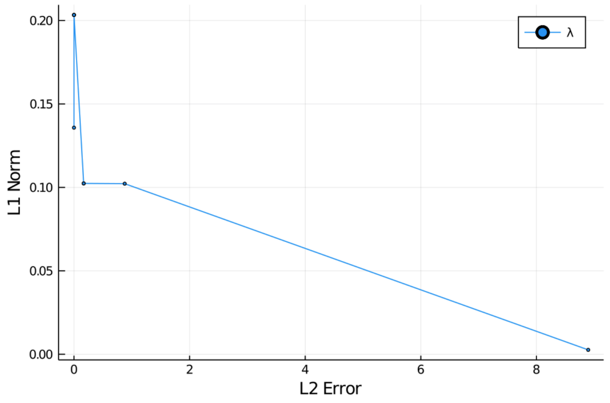


Figure 4.1: The bias-variance trade-off with various λs

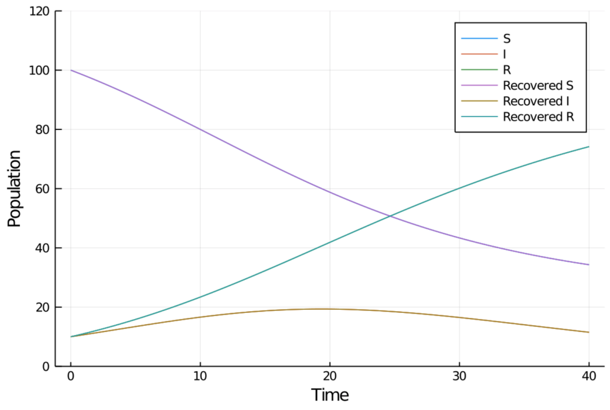


Figure 4.2: Actual vs. recovered data trajectory

We noticed that when $\lambda = 2.5$, which is near the elbow position in the plot, we could recover the original ODE model with the same variable combinations and the parameters with very low error: L_2-error $= 8.4e^{-14}$. Moreover, in Figure 4.2, we verified that the data matrix generated using the recovered equations overlap the actual data matrix $\mathbf{X}$ with no noticeable difference. This means we have successfully discovered the governing equations from the data matrix for this problem. However, SINDy is not panacea for all problems. In our experiments, we also found that after adding some noise (e.g White Noise) to the data matrix, the regularization parameter has to be much larger to recover the governing equations. Also, when noise is introduced, it becomes much more likely that the recovered equations contain redundant terms, and the recovered trajectory deviates significantly from the actual trajectory.

Chapter 5

Conclusion and Outlook

In this research, we analyzed several SEIR-based epidemiological models' performance on the development of the COVID-19 pandemic. In addition, we proposed an approach that recovers the strength of quarantine from the infectious and the recovered population data, and, with that approach, we applied a Long Short-term Memory (LSTM) network to make predictions on quarantine strength for future periods. Finally, we presented an example of using the Sparse Identification of Nonlinear Dynamics (SINDy) method to discover the governing equations directly from data to achieve maximal model interpretability without affecting its generalizability. But when the data contains noise, the ability of SINDy to recover the governing equations becomes much more limited.

For future research, in addition to those discussed in section 3.6, an interesting direction would be to investigate the interplay among model parameters. As an example, in equations 2.10-2.14, there are two model parameters (β, γ) and 51 neural network parameters and those parameters interact in a very subtle way. We found that even a tiny change in the first two constants' initial condition, such as decreasing γ from 0.05 to 0.049, will result in a completely different neural network parameter set. On the other hand, assigning a different initial parameter set to the neural network will only result in a minimal difference in the first two constants after training. Thus it can be inferred that those parameters weigh very differently in the optimization process, and it is worth conducting an in-depth analysis of the interplay between the model constants and the neural network parameters.

Bibliography

[1] Who coronavirus disease (covid-19) dashboard. 1

[2] S. F. Ardabili, A. Mosavi, P. Ghamisi, F. Ferdinand, A. R. Varkonyi-Koczy, U. Reuter, T. Rabczuk, and P. M. Atkinson. Covid-19 outbreak prediction with machine learning. *Algorithms*, 13(10), 2020. 2

[3] F. Brauer, C. Castillo-Chavez, and Z. Feng. *Simple Compartmental Models for Disease Transmission*, pages 21–61. Springer New York, New York, NY, 2019. 5

[4] S. L. Brunton, J. L. Proctor, and J. N. Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the national academy of sciences*, 113(15):3932–3937, 2016. 29

[5] Z. Che, S. Purushotham, K. Cho, D. Sontag, and Y. Liu. Recurrent neural networks for multivariate time series with missing values. *Scientific reports*, 8(1):1–12, 2018. 17

[6] R. Dandekar and G. Barbastathis. Quantifying the effect of quarantine control in covid-19 infectious spread using machine learning. *medRxiv*, 2020. 2, 4, 7

[7] S. Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998. 18

[8] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. 17, 18, 19

[9] G. Pinter, I. Felde, A. Mosavi, P. Ghamisi, and R. Gloaguen. Covid-19 pandemic prediction for hungary; a hybrid machine learning approach. *Mathematics*, 8(6), 2020. 2

[10] C. Rackauckas, Y. Ma, J. Martensen, C. Warner, K. Zubov, R. Supekar, D. Skinner, and A. Ramadhan. Universal differential equations for scientific machine learning. *arXiv preprint arXiv:2001.04385*, 2020. 9

[11] A. Rădulescu, C. Williams, and K. Cavanagh. Management strategies in a seir-type model of covid 19 community spread. *Scientific reports*, 10(1):1–16, 2020. 20

[12] A. Viguerie, G. Lorenzo, F. Auricchio, D. Baroli, T. J. Hughes, A. Patton, A. Reali, T. E. Yankeelov, and A. Veneziani. Simulating the spread of covid-19 via a spatially-resolved susceptible–exposed–infected–recovered–deceased (seird) model with heterogeneous diffusion. *Applied Mathematics Letters*, 111:106617, 2021. 1

[13] Z. Yang, Z. Zeng, K. Wang, S.-S. Wong, W. Liang, M. Zanin, P. Liu, X. Cao, Z. Gao, Z. Mai, et al. Modified seir and ai prediction of the epidemics trend of covid-19 in china under public health interventions. *Journal of Thoracic Disease*, 12(3):165, 2020. 2

[14] J.-S. Zhang and X.-C. Xiao. Predicting chaotic time series using recurrent neural network. *Chinese Physics Letters*, 17(2):88, 2000. 18