Forecasting Covid-19 Situation in Vietnam Using a Data-Driven Epidemiological Compartmental Model

Vo Le Tung
Department of Computer Science
Vietnamese - German University

Assessor: Assoc. Prof. Huynh Trung Hieu
Co-Assessor: Assoc. Prof. Nguyen Tuan Duc

October 9, 2021

Declaration

Acknowledgement

Abstract

Contents

List of Figures

List of Tables

1. Introduction

Since its emergence in 2019, the Covid-19 pandemic has evolved rapidly and is still affecting millions of lives around the globe. According to the World Health Organization (WHO), Covid-19 has caused over 230 million infections and over 4.7 million death worldwide [1]. Despite the high infection rate across the world, Vietnam had been fortunate during most of the pandemic's period, and the country was recognized for its effective government's interventions in reducing the number of cases [2], [3]. Until the first quarter of 2021, decisive actions and policies from the Vietnamese government have proven successful in containing the spread of Covid-19. But the situation has quickly changed since the end of April 2021, when a new wave of infection hit the country. The Delta variant of the SARS-CoV-2 virus, which is responsible for this infection wave, has a shorter incubation period and can spread quicker than the previous year [4], [5]. As a result, policies makers in Vietnam were caught off-guard, and the number of infections has been growing exponentially for the last few months (May 2021 - September 2021). In response to the escalating number of daily reported cases, the Vietnamese government has enacted similar strategies as in 2020 and more stringent policies in regions with a high infection rate. Unlike before, these aggressive responses to the virus spread where all citizens had to stay at home were not as effective in containing the spread of the virus. At the time of this writing (October 2021), the number of daily infections is on the decline but remains high, while prolonged restrictions on movement have negatively impacted the livelihood of lots of people and the entire economy [6], [7].

Like many other countries, one major factor that leads to the failure in swiftly containing the virus is the lack of knowledge in the dynamics of the pandemic under the effects of new variants and government interventions. Researchers have proposed diverse methods for modeling the disease to understand these dynamics. These methods use different techniques to investigate the strategies that can reduce the number of daily new infections. Rahimi, Chen, and Gandomi gave a systematic review of these methods [8]. Governments in different countries have employed these methods to aid in policy-making [9]. One popular method is an ensemble of multiple models submitted by researchers [10], used by the United States (US) Center for Disease Control and Prevention.

After almost two years of the pandemic, data has been collected and is available publicly in high quantity. One of the frequently used datasets is from the John Hopkins University, which aggregates the daily number of cases, recoveries, and deaths in many countries [11]. In addition, there is a worldwide effort in generating more datasets to help with analyzing the Covid-19 spread, such as mobility indices from companies with a massive amount of users like Apple [12], Google [12], and Facebook [13]. With an abundance of data, it has been shown that data-driven time-series forecasting techniques using deep learning can have high predicting power. Researchers have been experimenting with different architectures of Artificial Neural Networks (ANNs) for predicting the number of cases. Here we list some of the examples. Chimmula and Zhang used a Long Short Term Memory (LSTM) network for predicting the future transmission in Canada [14]. Ramchandani, Fan, and Mostafavi proposed a new ANN architecture that incorporates multivariate spatial time-series data to forecast the range of increase in COVID-19 infected cases in US counties [15]. Shahid, Zameer, and Muneeb performed a comprehensive comparison between the Autoregressive

Integrated Moving Average (ARIMA) statistical model and multiple different Recurrent Neural Networks (RNNs), including LSTM, Bidirectional Long Short Term Memory (Bi-LSTM), and Gated Recurrent Unit (GRU) [16]. While achieving high accuracy in predicting future cases, many of these models are black-box algorithms that are not interpretable, rendering it hard to quantify the causal effect of external factors on the progression of the pandemic. Having an explainable model is extremely important for healthcare and public authorities to derive meaningful analyses that aid in the planning process. Moreover, these black-box algorithms might not capture the underlying dynamics of the disease due to under-reporting, asymptomatic infections, or a general lack of data, especially in the early stages of the outbreak.

Compartmental models, such as the Susceptible-Infective-Removed (SIR) model and the Susceptible-Exposed-Infective-Removed (SEIR) model [17]–[20], are extensively used by researchers and public authorities. In the effort to understand the development of Covid-19, compartmental modeling is among the most popular approaches due to its simplicity and interpretability. This technique divides the population into compartments and models the transition of the number of people between these compartments using a system of differential equations. Because of its assumptions, compartmental models typically have many drawbacks: (1) low representational capability due to the low number of parameters, (2) the represented dynamics are stationary due to the constant parameters used in the model, (3) the population is assumed to be well-mixed, i.e., every individual is statistically indifferent, and (4) non-identifiability since a different set of parameters may result in the same dynamics [21]. Many studies have attempted to overcome these limitations by varying the model's parameters, typically the transmission rate, based on spatial and temporal factors. Schneider, Ngwa, Schwehm, *et al.* used a modified SEIR model where transitions between compartments are modeled as a stepwise process to simulate Covid-19 in Austria under different scenarios, eliminating the assumption that time-delay in those transitions is exponentially distributed [22]. IHME COVID-19 Forecasting Team used covariates to informed how the transmission rate changes over time to simulate different scenarios with Non-Pharmaceutical Interventions (NPIs) in the US [23]. Chang, Pierson, Koh, *et al.* used a fine-grained mobility network of the population's hourly movements, integrated it with a simple SEIR model having fixed parameters weighted over the mobility's data, and used that to model hourly infections in the US [24]. Chen, Lu, Chang, *et al.* derived the infection rate from historical data, used a regression method to extrapolate the current infection trends, and incorporated the extrapolated infection rate into an SEIR model for forecasting [25]. Li, Pei, Chen, *et al.* modeled the spreads of Covid-19 in China using city-to-city movement data with a SIR model [26].

Although countries have been using these modeling and forecasting techniques to analyze the Covid-19 situation, many are not applied to Vietnam. One of the reasons is because there was not a high number of recorded cases. Thus the specific dynamics of Covid-19 in Vietnam have not been well studied. Admit policies from other countries can be adopted; the effectiveness may vary due to socioeconomic, demographic, and cultural factors. As the number of reported cases is dropping, the Vietnamese government is slowly removing its restrictions in many places to help improve the current economic recession. Hence having a model for the distinct circumstances and data availability in Vietnam can be beneficial for assessing the pandemic's situation, especially when restrictions are lifted. The model could be of help in the following aspects: (1) informing policies makers about the effects of their decision, (2) informing health care facilities about a possible surge in the number of cases to ensure the supply of personnel and equipment, (3) informing business owners about possible policies to plan their supply and demand needs. Lastly, the recent surge in infections has generated data on the number of Covid-19 cases in Vietnam in a much higher quantity and quality that are publicly available and can be used for data-driven modeling.

Given that reasoning, this thesis focuses on implementing an interpretable disease model for Vietnam that can capture the current trends in the development of the pandemic. The implemented model will base on classical deterministic compartmental models where the mentioned issues are alleviated by using covariates to control the parameter(s). Instead of using a predefined multivariate function to adjust the parameters in the system of Ordinary Differential Equations (ODEs), we will be using an ANN to encode the covariates. The method takes advantage of the recent development in ANN architectures for solving forward-inverse problems with ODEs [27], [28]. Utilizing the capability of ANNs to approximate any arbitrary function [29]–[31], data-driven approaches can discover the underlying mechanics of Covid-19 without needing to define the governing multivariate function, which requires expert epidemiological knowledge.

The rest of the thesis is structured as follow:

- Chapter 2 gives an overview of the research backgrounds of the model. A list of related methods and techniques from other researchers is also presented.
- Chapter 3 discusses how the model was formulated, the used dataset, and the evaluation methods.
- Chapter 4 presents our findings and evaluations for the performance of the implemented model.
- Chapter 5 compares the results of the implemented model and its performance with related works produced by other researchers.This chapter includes a list of limitations of the model and further improvements that can be made to boost the effectiveness and performance of the model.
- Chapter 6 shows the overall achievement of this thesis.

2. Literature review

2.1 Compartmental modeling

One of the most transparent and most recognizable methods for modeling transmittable disease is through a compartmental model. This kind of modeling has been around for almost a century. The most basic was described by Kermack, McKendrick, and Walker in 1927, 1932, and 1933 [17]–[19]. In the subsection, we describe the basic concepts through two models. First, we go through the details of the basic SIR model, its assumptions, and the meaning of each term in the ODE system. Then, we illustrate an extension of the SIR model, the SEIR model. Because there have been countless extensions made to the basic SIR model, we are not going through each one of the extended versions for brevity. Having a basic understanding of how the most basic models work is sufficient for grasping the concepts of more complicated models. Compartmental models are flexible such that one easily inserts a new compartment into an existing model to accommodate for the disease that is being studied. We adopt the notations and formulations used in the subsections from Brauer [20].

2.1.1 SIR model

A compartmental model divides the population that is being modeled into compartments, i.e., a subpopulation. Individuals can be moved from one compartment to another under certain assumptions about the nature and time rate of transfer. As its name, the SIR model partitions the population into three subpopulations: susceptible, infective, and removed.

Let $S(t)$ be the number of people susceptible to the disease at time t. These are the people that have not been infected with the disease at time t. $I(t)$ denotes the individuals who were infected with the disease and can spread it by coming into contact with the susceptible population. $R(t)$ denotes the individuals who were infected with the disease and can not be infected with the disease again. Note that individuals can enter the removed compartment either through isolation, immunization against infection, recovery with immunity against reinfection, or death caused by the disease. Note that the characterization of these compartments does not fit into the epidemiological perspective. But from the modeling standpoint, only the overall state of an individual is taken into account

The movement between one compartment to another is captured by a system of ODEs, having time t and the transfer rates between the compartments are independent variables. The derivatives in the ODEs system express the changes in the size of each compartment with respect to time. When modeling with a system of ODEs, the number of individuals in each compartment is assumed to be differentiable. The approximation is reasonable under the condition that there are many members in a compartment. We also assume that the course of the epidemic is deterministic when using a system of ODEs, i.e., the behavior of a population is completely based on its initial conditions and the rules which govern the system.

Now we will mathematically present the model and how it is formulated. The following

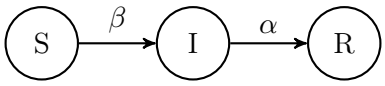

Figure 2.1: Graph of transitions between each compartment in the SIR model

ODEs system defines the SIR model

$$\begin{aligned} S' &= -\beta SI \\ I' &= \beta SI - \alpha I \\ R' &= \alpha I \end{aligned}$$

where interactions between the compartments can be visualized in Figure 2.1. The system is defined under the following assumptions

1. On average, a person makes contact sufficient to infect βN other people per unit time, where N is the total population (*mass action incidence*).

2. Population in the infective compartment transfers to the removed compartment at a rate of αI per unit time.

3. There is no new population entering into or leaving from the initial population, except deaths caused by the disease.

Because of the assumption (1) and the chance of an infective individual coming into contact with a susceptible individual is S/N, the number of new infections caused by a single infective individual per unit time is $(\beta N)(S/N)$. Consequently, the total number of newly added infective individuals per unit time is $(\beta N)(S/N)I = \beta SI$. On the assumption (2), the length of the infective period is exponentially distributed with a mean of $1/\alpha$. That is derived by considering $u(s)$ to be the number of individuals who are still infective s unit time after having been infected. If a portion α of these individuals leaves the infective class per unit of time then

$$u' = \alpha u,$$

and trivially we obtain

$$u(s) = u(0)e^{-\alpha s}.$$

As a result, the fraction of remaining infective individuals s unit time after being infected is $e^{-\alpha s}$, so the mean period is $\int_0^\infty e^{-\alpha s} = 1/\alpha$. Under assumption (3), the model considers that the time scale of the disease is much faster than the time scale of natural births and deaths. Hence demographic effects are negligible. Another alternative view for assumption (3) is that we only model the disease dynamics for a single outbreak.

We can not solve the system of ODEs analytically, but we can acquire its behavior through quantitative approaches. We know for a fact that the value $S(t)$ and $I(t)$ can not be negative. Thus the solving process can be terminated once $S(t) = 0$ or $I(t) = 0$. It can be observed that $S' < 0$ for all t and $I' > 0$ if and only if $S > \alpha/\beta$. Consequently, if $S(0) < \alpha/\beta$, I decreases to zero (no epidemic), whereas if $S(0) > \alpha/\beta$, I increases to a maximum achieved when $S = \alpha/\beta$ and then decreases to zero (epidemic). From the behavior, we can obtain a threshold value of $\beta S(0)/\alpha$, commonly called the *basic reproduction number*, denoted by $\mathcal{R}_0$. If $\mathcal{R}_0 < 1$, the number of infections dies out, whereas when $\mathcal{R}_0 > 1$, there will be an epidemic.

By definition, the value $\mathcal{R}_0$ is the number of secondary infections caused by a single infective individual placed within a fully susceptible population of size $K \approx S(0)$ over the

infective duration of this person. In this situation, the basic reproduction number becomes $\beta K/\alpha$. From here, we can deduce the final size of the epidemic by first consider the equation

$$\frac{I'}{S'} = \frac{dI}{dS} = \frac{(\beta S - \alpha)I}{-\beta SI} = -1 + \frac{\alpha}{\beta S}.$$

Integrating that gives the orbits (phase plane)

$$I = -S + \frac{\alpha}{\beta} \ln S + C, \tag{2.1}$$

where C is an arbitrary constant of integration. Alternatively, we can define a function that represents the orbits implicitly

$$V(S, I) = S + I - \frac{\alpha}{\beta} \ln S = C.$$

The constant C can be calculated given the initial conditions of S and I, since $C = V(S(0), I(0)) = S(0) + I(0) - \alpha \ln S(0)/\beta$. Knowing that $\lim_{t\to\infty} I(t) = 0$ and let $S_\infty = \lim_{t\to\infty} S(t)$ the relation $V(S(0), I(0)) = V(S_\infty, 0)$ is equivalent to

$$K - \frac{\alpha}{\beta} \ln S(0) = S_\infty - \frac{\alpha}{\beta} \ln S_\infty.$$

Rewriting the equation above in terms of $\mathcal{R}_0$ gives the *final size relation*

$$\ln S(0) - \ln S_\infty = \mathcal{R}_0 \left[1 - \frac{S_\infty}{K}\right]. \tag{2.2}$$

Equation 2.2 also shows that $S_\infty > 0$ because its right-hand size is finite.

Estimating the contact rate β is a challenging task because it depends not only on the disease itself but also on social and behavioral factors. In retrospection, one can obtain the values $S(0)$ and S_∞ and calculate $\mathcal{R}_0$ once an epidemic has ended. During the early phase of the epidemic, the number of infective individuals grows exponentially, where the equation for I can be approximated with

$$I' = (\beta K - \alpha)I.$$

The initial growth rate is given by

$$r = \beta K - \alpha = \alpha(\mathcal{R}_0 - 1).$$

Because both the values K and α can be measure, the contact rate can be calculated as

$$\beta = \frac{r + \alpha}{K}.$$

Note that early on in the outbreak, this estimation could experience high inaccuracy due to incomplete data or underreporting of the number of cases. The estimation accuracy might also suffer significantly with an outbreak of new unknown diseases, where early cases are more likely to be diagnosed.

By substituting $S = \alpha/\beta$ and $I = I_{max}$ into Equation 2.1 we can obtain the maximum number of infective individuals, which is attained when $I' = 0$

$$I_{max} = S(0) + I(0) - \frac{\alpha}{\beta} \ln S(0) - \frac{\alpha}{\beta} + \frac{\alpha}{\beta} \ln \frac{\alpha}{\beta}$$

Generalized SIR model

Assumption (1) of the basic SIR model is unrealistic. It is better to assume that the contact rate is a non-increasing function of the population size. A more generalized SIR model can be formulated by replacing assumption (1) with the assumption that each person in the population makes $C(N)$ contacts in unit time on average with $C'(N) \geq 0$. We can define

$$\beta(N) = \frac{C(N)}{N},$$

and make a reasonable assumption that $\beta'(N) \leq 0$ express the saturation in the number of contacts. With that assumption we have a new model

$$\begin{aligned} S' &= -\beta(N)SI \\ I' &= \beta(N)SI - \alpha I \\ R' &= f\alpha I \\ N' &= -(1-f)\alpha I \end{aligned}$$

In this model, we consider the rate of change in the total population N because now the contact rate depends on it. Therefore, we have to consider the distinction between the individuals who recover from the disease and the individuals who die of the disease. Hence, assuming that a fraction f of the αI members leaving the infective compartment at time t recover. The remaining fraction $(1-f)$ dies of the disease.

In this model, the standard reproduction number is

$$\mathcal{R}_0 = \frac{K\beta(K)}{\alpha}. \tag{2.3}$$

A time-dependent reproduction number $\mathcal{R}^*$ can also be calculated. This value represents the number of secondary infections caused by an individual at time t. The equation for getting this value is

$$\mathcal{R}^* = \frac{S\beta(N)}{\alpha}.$$

The final size relation can also be quantified under the assumption that $\lim_{N\to 0}\beta(N)$ is finite, giving the inequalities

$$\mathcal{R}_0\left[1 - \frac{S_\infty}{K}\right] \leq \ln\frac{S(0)}{S_\infty} \leq \frac{\beta(0)(K - S_\infty)}{\alpha K}$$

2.1.2 SEIR model

With many infectious diseases, there is a period where an individual has been infected with the disease but can not transmit it. This could be modeled by adding an exposed compartment, labeled as E, where the exposed period is exponentially distributed with a mean of $1/\kappa$. The generalized version of the model with four compartments S, E, I, R, and $N = S + E + I + R$ is given as

$$\begin{aligned} S' &= -\beta(N)SI \\ E' &= \beta(N)SI - \kappa E \\ I' &= \kappa E - \alpha I \\ R' &= f\alpha I \\ N' &= -(1-f)\alpha I \end{aligned}$$

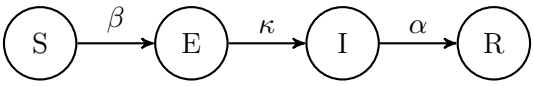

Figure 2.2: Graph of transitions between each compartment in the SEIR model

and the interactions between these compartments are illustrated in Figure 2.2. The calculation of the basic reproduction for this model is similar to the calculation for the generalized SIR model, which is given by Equation 2.3.

Following the same analysis in the previous subsections, the final size relation is expressed by the inequality

$$\ln \frac{S(0)}{S_\infty} \geq \mathcal{R}_0 \left[1 - \frac{S_\infty}{K}\right]$$

For diseases that have an asymptomatic phase rather than an exposed stage, we can introduce a factor ϵ. This factor represents a reduction in the infectiousness of those who are infected but not showing symptoms. A model that represents this situation is given by

$$\begin{aligned} S' &= -\beta(N)S(I + \epsilon E) \\ E' &= \beta(N)S(I + \epsilon E) - \kappa E \\ I' &= \kappa E - \alpha I \\ R' &= f\alpha I \\ N' &= -(1 - f)\alpha I \end{aligned}$$

where the basic reproduction number is calculated with

$$\mathcal{R}_0 = \frac{K\beta(K)}{\alpha} + \epsilon\frac{K\beta(K)}{\kappa},$$

and the final size relation is the following inequality

$$\ln \frac{S(0)}{S_\infty} \geq \mathcal{R}_0 \left[1 - \frac{S_\infty}{K}\right] - \frac{\epsilon\beta(K)}{\kappa} I_0.$$

2.2 Artificial Neural Networks

With the explosion of the amount of collected data and the rapid development of computer hardware in the twenty-first century, deep learning techniques have been used extensively for solving different classes of problems. Deep learning techniques are revolutionary in that they can automatically capture the relationship between the inputs and the outputs, allowing machines to perform complex tasks without having humans explicitly told them what to do. These capabilities are empowered by the underlying ANN architecture, allowing computing systems to mimic the functionality of biological neural networks, which comprise animal brains. An ANN simulates biological neural networks by having the ability to learn from experiences and examples. This process works by using a set of inputs with known outputs. The ANN is then used to guess the outputs from the inputs. The differences between the guess output values and the known output values indicate what changes need to be made to the ANN. These updates to the ANN are done multiple times until the guess error is minimized or some criteria are met. This process is now called *supervised learning*.

The idea for a computational model based on the brain was first proposed by McCulloch and Pitts in 1943 [32]. They observed that nervous activity, neural events, and their relations could be demonstrated using propositional logic. Later on, researchers had been

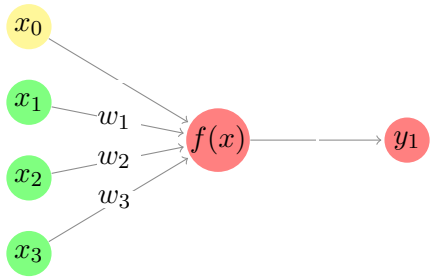

Figure 2.3: Graph representation of a perceptron described in Equation 2.4. x_0 is the bias b, and x_{1-3} are the input signals.

building upon that idea to find more methods for artificially representing the brain. In 1958, Rosenblatt introduced the concept of perceptrons which is considered the first ANN [33]. Mathematically, a perceptron is expressed as

$$f(x) = \begin{cases} 1 & \text{if } w \cdot x + b > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.4}$$

where w and x are vectors of real value, $w \cdot x = \sum_{i=1}^{m} w_i x_i$ is the dot product between the two vectors, and b is the bias for shifting the decision boundary. The perceptron is illustrated in Figure 2.3. This initial version of ANN only works with classification problems in which the classes are linearly separable. The training procedure for this type of ANN is given in Algorithm 1.

Algorithm 1 Perceptron training algorithm

$\mathcal{D} \leftarrow \{(x_t, y_t) \mid t \in [1, n]\}$ ▷ Get the training data with n samples
$t \leftarrow 1$ ▷ Initialize the time step to 1
$w^{(1)} \leftarrow 0$ ▷ Initialize all the weights to 1
for all $(x, y) \in \mathcal{D}$ **do** ▷ Go through all the training data
  $\hat{y} \leftarrow f(x)$ ▷ Compute the perceptron output
  if $\hat{y} = y$ **then**
    $w^{(t+1)} \leftarrow w^{(t)}$ ▷ Keep the weights on correct ouput
  else
    $w^{(t+1)} \leftarrow w^{(t)} + yx$ ▷ Update the weights on incorrect ouput
  end if
  $t \leftarrow t + 1$ ▷ Increment the time step
end for

Nowadays, neural networks are highly complex because of the considerably higher computing power we have and the developments of special-purpose hardware, such as the Graphics Processing Unit (GPU). Multiple different architectures of neural networks exist, a non-exhaustive list includes: Convolutional Neural Networks (CNNs) are typically used for processing images or other two-dimensional data [34]; LSTMs solve the vanishing gradient problem and have the ability to handle data with a mix of low and high frequencies [35]; Generative Adversarial Networks (GANs) are designed to have competing ANNs on tasks such as playing a game [36]. In this thesis, we are utilizing the Multi-Layer Perceptron (MLP) architecture. The architecture is illustrated in Figure 2.4.

MLP is a network that composes multiple nodes, called artificial neurons. These nodes in the network form a directed weighted graph, where each node can take input signals from nodes in the previous layer, performs some calculation and sends the results to nodes in the subsequent layer. In this configuration, an MLP can be thought of as multiple perceptrons

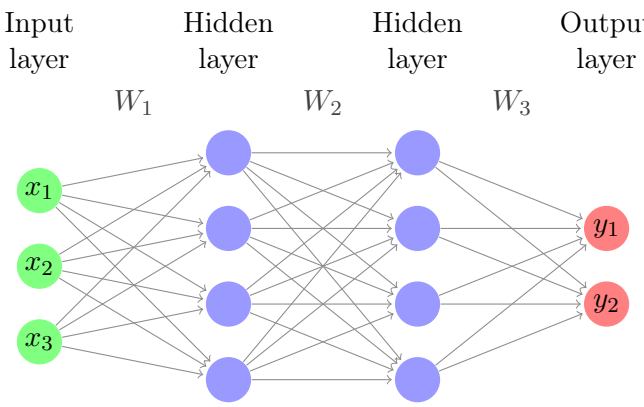

Figure 2.4: Graph representation of a multi-layer perceptron with four layers

that are organized into layers. The first layer in the network is called the input layer, and the last layer is called the output layer; any layers in-between are called hidden layers. Commonly, the term MLP is used to describe an ANN where each node in a layer connects to all of the nodes in the subsequent layer, and nodes can only connect from one layer to the immediately subsequent layer. Other terms for this are fully connected ANN or densely connected ANN. Additionally, an MLP is a type of feed-forward ANN, i.e., signals in this network are passed from one layer to another in one direction only. The counterpart of this is the feedback ANN. In a feedback ANN, signals can flow in any direction, as can be seen in RNN. Mathematically, each node in an MLP computes the following values

$$a_i^k = b_i^k + \sum_{j=1}^{r_{k-1}} w_{ij}^k z_j^{k-1}$$
$$z_i^k = \phi(a_i^k),$$

where ϕ is a nonlinear function that gets applied to the output a_i^k of layer k, w_{ij}^k is the weight that maps the node j in layer $k-1$ to node i in layer k, b_i is the bias term for node i in layer k, z_i^k is the product sum plus bias for node i in layer k, a_i^k is the activated output of node i in layer k, and r_k is the number of nodes in layer k. This computation is similar to perceptron, excepting the function ϕ, called the *activation function*. Let n be the depth of a network, i.e., the number of layers having adjustable weights, and X be the vectors of input signals. Let W_i, b_i, and ϕ_i be the vector of weights, vector of bias terms, and the activation function for each layer, where $i \in [1, n]$. An MLP is expressing the following function

$$g(X) = \phi_n(W_n\phi_{n-1}(\cdots(W_2\phi_1(W_1X + b_1) + b_2) + \cdots) + b_n),$$

which is simply a nesting of multiple simple vectors algebra and applications of the activation functions.

2.2.1 Training Artificial Neural Networks

MLPs and other ANNs can be used for various tasks because they are universal approximators [29]–[31]. That means a sufficiently large ANN with an arbitrary number of nodes can reasonably approximate any function $f : \mathbb{R}^M \mapsto \mathbb{R}^N$, given that appropriate weights can be found. However, the exact method for finding suitable weights is not defined. Commonly, the fitting weights are learned by ANNs through the back-propagation algorithm [37] and the gradient descent optimization technique [38]. A simple training procedure for most ANNs is given in Algorithm 2. Back-propagation is the algorithm for finding the gradients of a loss function with respect to the network's weights. A loss function measures how large the

error is between the ANN's guess and the true value. One choice is the Mean Squared Error (MSE) $\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}-y)^2$ The algorithm calculates the gradients starting from the output layer and going backward to the input layer. To find the gradient of the loss function $\mathcal{L}$ with respect to the weight w_{ij}^L, where L is the number of layers, we must calculate

$$\frac{\partial \mathcal{L}}{\partial w_{ij}^L} = \frac{\partial \mathcal{L}}{\partial a_i^L}\frac{\partial a_i^L}{\partial z_i^L}\frac{\partial z_i^L}{\partial w_{ij}^L}. \tag{2.5}$$

Similarly we can compute the gradient with respect to the bias term

$$\frac{\partial \mathcal{L}}{\partial b_i^L} = \frac{\partial \mathcal{L}}{\partial a_i^L}\frac{\partial a_i^L}{\partial z_i^L}\frac{\partial z_i^L}{\partial b_i^L}. \tag{2.6}$$

If we want to calculate the gradient of $\mathcal{L}$ with respect to w_{jk}^{L-1}, we first consider the gradient of $\mathcal{L}$ with respect to a_j^{L-1}

$$\frac{\partial \mathcal{L}}{\partial a_j^{L-1}} = \sum_{i=1}^{r_{L-1}}\frac{\partial \mathcal{L}}{\partial a_i^L}\frac{\partial a_i^L}{\partial z_i^L}\frac{\partial z_i^L}{\partial a_j^{L-1}}. \tag{2.7}$$

Then we can compute the gradients

$$\frac{\partial \mathcal{L}}{\partial w_{jk}^{L-1}} = \frac{\partial \mathcal{L}}{\partial a_j^{L-1}}\frac{a_j^{L-1}}{z_j^{L-1}}\frac{z_j^{L-1}}{w_{jk}^{L-1}},$$

and

$$\frac{\partial \mathcal{L}}{\partial b_j^{L-1}} = \frac{\partial \mathcal{L}}{\partial a_j^{L-1}}\frac{a_j^{L-1}}{z_j^{L-1}}\frac{z_j^{L-1}}{b_j^{L-1}},$$

This same approach can be applied for any layer in the ANN. By iterating backward and utilizing the chain rule, recalculations of derivatives can be avoided. As can be seen from Equation 2.5, Equation 2.6, and Equation 2.7, the term $\frac{\partial \mathcal{L}}{\partial a_i^L}\frac{\partial a_i^L}{\partial z_i^L}$ is used multiple times.

Algorithm 2 Batch gradient descent for training ANN. The learning rate η influences how much the weights and bias terms are updated in each iteration.

$\mathcal{D} \leftarrow \{(x_t, y_t) \mid t \in [1, n]\}$ ▷ Get training data with n samples
$\eta \leftarrow$ learning rate ▷ Choose a learning rate
$W \leftarrow$ randomly values ▷ Randomly initailize the weights
$b \leftarrow$ randomly values ▷ Randomly initailize the bias terms
while criteria are not met **do**
  $\Delta W \leftarrow 0$ ▷ Set the accumulated gradients w.r.t to each weight to zero
  $\Delta b \leftarrow 0$ ▷ Set the accumulated gradients w.r.t to each bias terms to zero
  for all $(x, y) \in \mathcal{D}$ **do** ▷ Go through all training data
    $\hat{y} \leftarrow \mathcal{NN}_{W,b}(x)$ ▷ Compute the neural network output
    $E \leftarrow \mathcal{L}(\hat{y}, y)$ ▷ Compute the loss value
    $\Delta W \leftarrow \Delta W + \eta\frac{\partial E}{\partial W}$ ▷ Accumulate the loss gradients w.r.t to each weight
    $\Delta b \leftarrow \Delta b + \eta\frac{\partial E}{\partial b}$ ▷ Accumulate the loss gradients w.r.t to each bias term
  end for
  $W \leftarrow W - \Delta W$ ▷ Update all the weights with the accumulated gradients
  $b \leftarrow b - \Delta b$ ▷ Update all the bias terms with the accumulated gradients
end while

2.2.2 Activation functions

An activation function introduces non-linearity into the ANN without it, the ANN becomes a simple linear transformation and does not have the power to express complex relations between the input and the output. In the case of perceptron, the output of the node is either one or zero based on some threshold applied to the product sum. The activation function that gets applied in that case is called the Heaviside activation. The derivative of the Heaviside activation function is undefined at zero and is zero at every other point, making it unsuitable in deep ANN, which is trained using gradient-based optimization methods.

Some classic activation functions used in deep ANN include the sigmoid function

$$\phi(z) = \frac{1}{1 + e^{-z}},$$

and the hyperbolic tangent (tanh) function

$$\phi(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}.$$

Both of these activation functions suffer from the vanishing gradient problem. Recall from Section 2.2.1, training ANNs involves computing the gradients of the loss function with respect to its parameters. This process is done by applying the chain rule using the back-propagation algorithm. The chain rule is fundamentally a sequence of multiplications of intermediate derivatives. Because the derivatives of both sigmoid function and tanh function tend to be closer to zero, in deep ANNs with large numbers of layers, the gradients of the loss function become extremely close to zero. As a result, the ANN can not learn due to insignificant changes in the parameters caused by the infinitesimal gradients. One advantage the tanh function has over the sigmoid function is that its outputs are centered around zero, so its input can be highly negative or highly positive without affecting the ANN's performance.

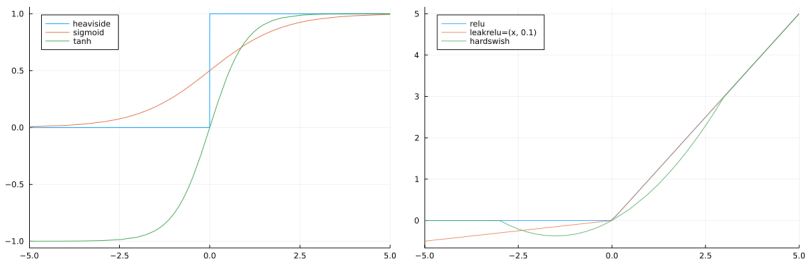

Figure 2.5: A visual comparison between the outputs of Heaviside, sigmoid, tanh, glsReLU, Leaky ReLU, and hard swish activation functions

To solve the vanishing gradients problem, the Rectified Linear Unit (ReLU) activation function is used

$$\phi(z) = max(0, z).$$

One disadvantage of this activation function is the dying ReLU problem, which is when the node's output gets close to zero or is negative. When that happens, the gradients of the loss function with respect to the parameter for that node are zero, causing the node to stop learning. A function that addresses this issue is the Leaky ReLU activation

$$\phi(z) = max(\alpha z, z),$$

where α is a hyperparameter that can be chosen to define negative output values of the function. Or the hard swish activation function [39]

$$\phi(z) = \begin{cases} 0 & \text{if } z \leq -3 \\ z & \text{if } z \geq 3 \\ z * (z+3)/6 & \text{otherwise} \end{cases}$$

2.3 Physics Informed Neural Networks

Applications of deep learning methods have been achieving multiple breakthroughs, especially in computer vision and natural language processing. The performance of these models is enabled by utilizing a vast amount of data that is readily available. Nonetheless, the application of deep learning in many domains of science has not been gaining success. The cost of data collection for analyzing complex physical, biological, or engineering systems is prohibitive. Without big data, state-of-the-art deep learning techniques lack robustness and are not guaranteed to converge. Specifically, in the case of Covid-19 or any other disease, we would like to have an accurate model that can fairly estimate the trajectories of the disease at an early stage when data is scarce. Raissi, Perdikaris, and Karniadakis observed that prior knowledge about dynamical systems, which is captured by mechanistic models, had not been utilized for training ANNs [40]. By utilizing this information, we can limit the solutions space that an ANN can take and enrich the data seen by models to help them converge faster. Physics-Informed Neural Networks (PINNs) work by using the knowledge given by mechanistic models in the form of ODEs or Partial Differential Equations (PDEs) as a regularization term for the loss function [40], [41]. With physical constraint incorporated within the loss function, we can guide the ANNs to learn the parameters that give rise to a solution that obeys physical laws. Figure 2.6 shows the general schematic of PINNs.

To derive the loss function for ODEs in this framework, we first consider

$$\frac{du}{dt} = f(u,t),\ t \in [0,1],\ u(0) = u_0$$

An ANN is then used to approximate the solution to this problem

$$\mathcal{NN}(t) \approx u(t).$$

Because the ANN is differentiable, we know that if $\mathcal{NN}(t)$ is the solution of our problem, then $d\mathcal{NN}(t)/dt = f((NN)(t), t)$ for all t. This condition can then be incorporated into the loss function

$$MSE = \frac{1}{N}\sum_{i}^{N}\left(\frac{d\mathcal{NN}(t_i)}{dt} - f(\mathcal{NN}(t_i), t_i)\right)^2.$$

Note that the initial condition $u(0) = u_0$ has to be satisfied. Thus we define a new function that encodes the initial condition within itself. This function will trivially satisfies the initial condition for any possible set of parameters

$$g(t) = u_0 + t\mathcal{NN}(t).$$

The function above inherits the property of ANNs as universal approximators for any continuous function while satisfies the condition $g(0) = u_0$. At this point, the loss function turns into

$$MSE = \frac{1}{N}\sum_{i}^{N}\left(\frac{dg(t_i)}{dt} - f(g(t_i), t_i)\right)^2.$$

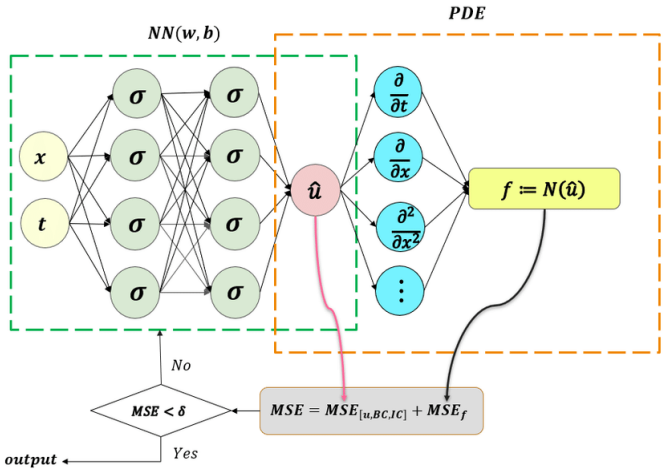

Figure 2.6: The schematic of PINNs for solving PDEs. Figure is taken from Guo, Cao, Bainian, *et al.* [42]

To derive the loss function for PDEs in this framework, we first consider parameterized nonlinear PDEs of the general form

$$u_t + \mathcal{N}[u;\lambda] = 0, \ x \in \Omega, \ t \in [0,T], \tag{2.8}$$

where $u(t,x)$ denotes the hidden solution, $\mathcal{N}[\cdot;\lambda]$ is a nonlinear differential operator parameterized by λ, and Ω is a subset of $\mathbb{R}^D$. We define the function $f(t,x)$ to be given by the left-hand side of Equation 2.8

$$f := u_t + \mathcal{N}[u]. \tag{2.9}$$

The hidden solution $u(t,x)$ is then approximated by an ANN, denoted as $\hat{u}(t,x) = \mathcal{NN}(t,x)$. From the approximation $\hat{u}(t,x)$ and Equation 2.9, we can define the function $\hat{f}(t,x)$ that describes the PDEs in terms of the approximation

$$\hat{f} := \hat{u}_t + \mathcal{N}[\hat{u}].$$

To train the parameters of the ANN, we employ the following loss function

$$MSE = MSE_u + MSE_f,$$

where

$$MSE_u = \frac{1}{N_u}\sum_{i=1}^{N_u} |\hat{u}(t_u^i, x_u^i) - u^i|^2$$

and

$$MSE_f = \frac{1}{N_f}\sum_{i=1}^{N_f} |\hat{f}(t_f^i, x_f^i)|^2.$$

We have $\{t_u^i, x_u^i, u^i\}_{i=1}^{N_u}$ are the initial and boundary training data, and $\{t_f^i, x_f^i\}_{i=1}^{N_u}$ are the collocation points. The loss value MSE_u enforces the initial and boundary conditions on the ANN, while the loss value MSE_f enforces the dynamics specified by Equation 2.8.

The ANN under this framework can be training using well-known techniques in deep learning. These techniques include automatic differentiation, back-propagation, and gradient-based optimization. In the research, Raissi, Perdikaris, and Karniadakis showed empirical evidence that the ANN will converge to a global minimum under the condition that the differential equations are well-posed and their solution is unique. They also showed that the model achieved high prediction accuracy with out-of-sampled data when given a sufficiently expressive ANN and a sufficient number of collocation points N_f.

2.4 Neural Ordinary Differential Equations

When modeling dynamical systems, it is generally challenging to create a model by hand because real-world data are often hard to interpret or are sampled at an irregular interval. Using Neural Ordinary Differential Equations (NeuralODEs), the system's dynamics can be learned through data-driven approaches without having explicitly specified ODEs or PDEs. NeuralODEs can also be used to replace a stack of residual blocks because they perform the same functionality while reducing the memory cost of training the model. Chen, Rubanova, Bettencourt, *et al.* made an observation that models such as residual networks (see Figure 2.7), RNN decoders, or normalizing flows build complex transformations by composing a sequence of modifications to a hidden state [27]

$$h_{t+1} = h_t + f(h_t, \theta_t)$$

where $t \in \{0 \cdots T\}$ and $h_t \in \mathbb{R}^D$. These iterative updates can be seen as an Euler discretization of a continuous transformation. In NeuralODE, the continuous dynamics of hidden units are parameterized using an ODE specified by an ANN

$$\frac{dh(t)}{dt} = f(h(t), t, \theta).$$

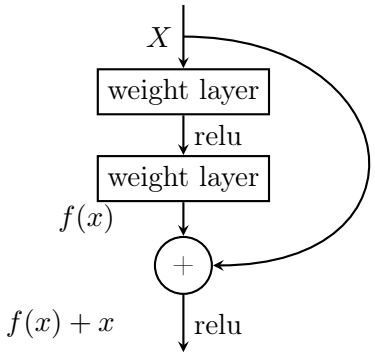

Figure 2.7: Example of a skip connection in residual network

That is starting from the input layer $h(0)$, the output layer $h(T)$ can be defined as the solution to the ODE initial value problem at time step T. Typical ANNs create a mapping from the inputs to the outputs, whereas NeuralODEs define the dynamics that can turn the inputs into the outputs. There are several benefits when doing this, such as:

- **Memory efficiency** Computing the gradients of the loss function with respect to all the inputs of the ODE solver does not require back-propagation. Thus, intermediate values of the forward pass do not need to be stored. This allows for training with constant memory cost.
- **Adaptive computation** Modern ODE solver guarantees the growth of approximation error. They keep track of errors and adapt the evaluation strategy to provide the requested level of accuracy. The cost of evaluating the model can scale with the problem's complexity.
- **Scalable and invertible normalizing flows** Continuous transformations allow the change of variables formula to become easier to compute.
- **Continuous time-series models** Continuously defined dynamics can incorporate data that arrive at arbitrary times. Network models such as RNNs require the discretization of observation and emission intervals

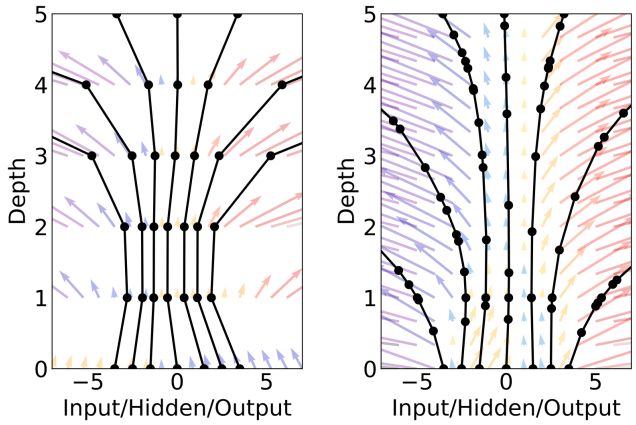

Figure 2.8: *Left*: A residual network defines discrete sequence of transformations. *Right*: A ODE network defines a vector field, which continuously transform the state. *Both*: Circles represent evaluation locations. Figure is taken from Chen, Rubanova, Bettencourt, *et al.* [27])

For a NeuralODE to be trainable with gradient descent, we must have a way to compute the gradients of the loss with respect to the parameters of the NeuralODE. The regular method for taking the loss gradients in ANNs (back-propagation) can not be applied for NeuralODEs because it incurs high memory cost and introduces additional numerical errors. Hence, the authors of the method use *adjoint sensitivity analysis* to compute the gradients. The gradients are computed by solving a second, augmented ODE backward in time. This method applies to all ODE solvers. The approach scales linearly with the problem size, and the numerical error can be controlled explicitly. Algorithm 3 shows the overall steps that need to be taken to compute the derivatives. We consider the loss function L, which takes the result of an ODE solver

$$L(z(t_1)) = L\left(z(t_0) + \int_{t_0}^{t_1} f(z(t), t, \theta)dt\right) = L(ODESolve(z(t_0), f, t_0, t_1, \theta)).$$

First, we consider the gradient of the loss with respect to the hidden state $z(t)$, called the adjoint $a(t) = \partial L/\partial z(t)$. The dynamics of the adjoint are then given by another ODE

$$\frac{da(t)}{dt} = -a(t)^T \frac{\partial f(z(t), t, \theta)}{\partial z}.$$

The gradient of the loss with respect to the hidden state at the initial time step can be calculated by integrating the adjoint dynamics backward in time starting from the value $\partial L/\partial z(t_1)$

$$\frac{dL}{dz(t_0)} = \frac{\partial L}{\partial z(t_1)} + \int_{t_1}^{t_0} -a(t)^T \frac{\partial f(z(t), t, \theta)}{\partial z} dt. \tag{2.10}$$

Computing $dL/dz(t_0)$ requires the hidden state $z(t)$ at each evaluated time step which is calculated by integrating the dynamics of our system backward in time starting from the value $z(t_1)$

$$z(t) = z(t_1) + \int_{t_1}^{t} f(z(t), t, \theta)dt. \tag{2.11}$$

Finally, we can calculate the gradients of the loss function with respect to the parameters θ using

$$\frac{dL}{d\theta} = \int_{t_1}^{t_0} -a(t)^T \frac{\partial f(z(t), t, \theta)}{\partial \theta} dt. \tag{2.12}$$

The vector-Jacobian products $a(t)^T \frac{\partial f}{\partial z}$ and $a(t)^T \frac{\partial f}{\partial \theta}$ from Equation 2.10 and Equation 2.12 can be computed efficiently with automatic differentiation. The integrals in Equation 2.10, Equation 2.11, and Equation 2.12 can be computed in a single call to an ODE solver.

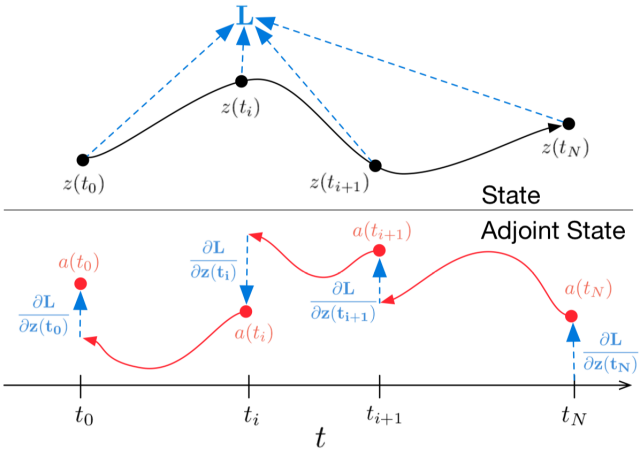

Figure 2.9: Reverse-mode differentiation of an ODE solution. An augment system contains the original state and the gradients of the loss with respect to the state is solved backwards in time. If the loss depends on the state at multiple observation times, the adjoin state is updated in the direction of the gradients of the loss with respect to each observation. Figure is taken from Chen, Rubanova, Bettencourt, *et al.* [27]

Algorithm 3 Reverse-mode derivative of an ODE initial value problem. Figure is taken from Chen, Rubanova, Bettencourt, *et al.* [27])

function ODEDERIVATIVE($\theta, t_0, t_1, z(t_1), \frac{\partial L}{\partial z(t_1)}$)
  $s_0 \leftarrow \left[z(t_1), \frac{\partial L}{\partial z(t_1)}, 0_{|\theta|}\right]$ ▷ Define initial augmented state
  function AUGDYNAMICS($[z(t), a(t), \cdot], t, \theta$) ▷ Define dynamics on augmented state
    return $\left[f(z(t), t, \theta), -a(t)^T \frac{\partial f}{\partial z}, -a(t)^T \frac{\partial f}{\partial \theta}\right]$ ▷ Compute vector-Jacobian products
  end function
  $\left[z(t_0), \frac{\partial L}{\partial z(t_0)}, \frac{\partial L}{\partial \theta}\right] \leftarrow$ ODESOLVE(s_0, AUGDYNAMICS, t_1, t_0, θ) ▷ Solve ODE backward
  return $\left[\frac{\partial L}{\partial z(t_0)}, \frac{\partial L}{\partial \theta}\right]$ ▷ Return the gradients
end function

2.5 Universal Differential Equations

Because of the scarcity of data in many scientific domains, mechanistic models are frequently employed instead of deep learning. These models often come in the form of ODEs or PDEs that describe the system's behavior over time, referred to as the system's dynamics. Mechanistic models are explainable, based on prior structural knowledge, and backed by many years of rigorous studies, but they lack the flexibility of deep learning models and usually make unrealistic assumptions about real-world phenomenons. In contrast, deep learning models are highly flexible, but they require a massive amount of data to be effective. Based on those observations, many researchers have recently presented with different methods for merging machine learning and mechanistic models.

As introduced in Section 2.3, PINNs can enforce physical constraints on ANNs through the use of PDEs/ODEs as a regularization term in the loss function. PINNs were shown to

have high performance for some scientific applications with limited data. However, it still lacks the interpretability of mechanistic models even though we know that the ANN has learned some legitimate physical characteristics. Instead of incorporating scientific knowledge into the model, NeuralODEs take a different approach by using the structure of scientific models as a basis for machine learning. While NeuralODEs can learn the continuous dynamics data, as shown in Section 2.4, the resulting models do not represent any known mechanisms. To address these issues, Rackauckas, Ma, Martensen, *et al.* proposed Universal Differential Equations (UDEs) that extend the approach taken by NeuralODEs [28]. UDEs apply mechanistic modeling in conjunction with universal approximators where part of the differential equation embeds a universal approximator, e.g., a neural network, a random forest, or a Chebyshev expansion. This combination helps create a robust model, where well-proven terms in the mechanistic model can stay unchanged, while the terms with complex unknown interactions can be discovered by the universal approximator.

In general, a UDE model will be in the form of

$$u' = f(u, t, U_\theta(u, t)),$$

where f denotes a known mechanistic model whose missing terms are defined by some universal approximator U_θ. The process for training UDEs is similar to training typical ANNs where we minimize a loss function $\mathcal{L}(\theta)$ defined for the approximated solution $u_\theta(t)$ with respect to the current choice of parameters θ. Common choices for a loss function used to train ANNs are applicable in this case, such as the MSE $\mathcal{L}(\theta) = \sum_{i=1}^{N}(u_\theta(t_i) - d_i)^2$ at discrete data points $\{(t_i, d_i)\}_{i=i}^{N}$. Then the gradients of the loss with respect to the parameters $\frac{\partial \mathcal{L}}{\partial \theta}$ are calculated for applying local gradient-based methods such as gradient descent. Methods for computing the gradients of UDEs of different types of differential equations are implemented by Rackauckas, Ma, Martensen, *et al.* [28] as a library in the Julia programming language [43], which will be utilized in this thesis.

2.6 Sparse Identification of Nonlinear Dynamical Systems

Extracting the governing equations from data has been a great challenge in the fields of science and engineering. While recent rapid advances in machine learning and data science have enabled the ability to analyze and understand complex static data, methods for capturing physical models of dynamical systems from data have slowly progressed. To ease this problem, Brunton, Proctor, and Kutz proposed the Sparse Identification of Nonlinear Dynamical System (SINDy) method for discovering the underlying dynamics of systems using sparse regression and compressed sensing [44]. Suppose that we are working with dynamical systems of the form

$$\frac{d}{dt}x(t) = f(x(t)),$$

where $x(t)$ is a vector of the state of the system at time t, and the function $f(x(t))$ describes the dynamic constraints that define the behaviors of the system. One key observation made by the author is that in many dynamical systems, the governing function f is sparse in the space of possible functions. Under that observation, methods in compressed sensing and sparse regression can be used to determine which terms are most likely to describe f without performing a brute-force search.

To determine the function f from data, we collect the history of the state $x(t)$ and its

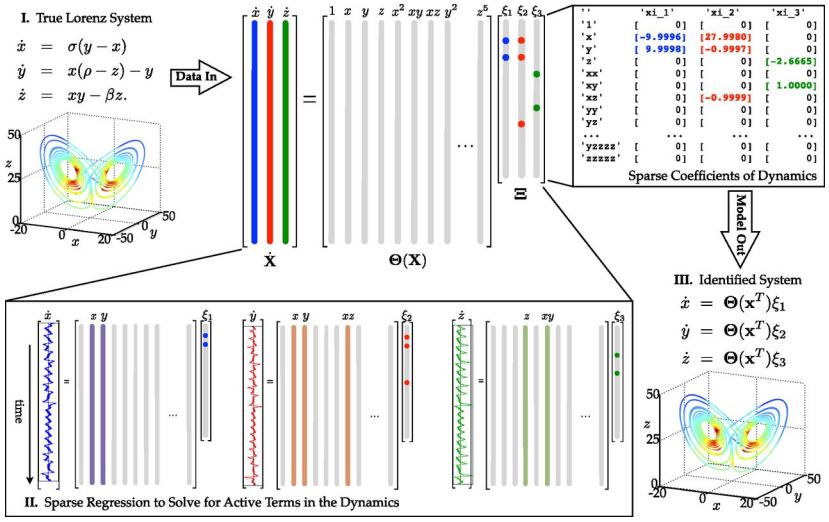

Figure 2.10: Schematic of the SINDy algorithm, demonstrated on the Lorenz equations. The few entries in the vectors of Ξ, solved for by sparse regression, denote the relevant terms in the right-hand side of the dynamics. Parameter values are $\sigma = 10$, $\beta = 8/3$, $\rho = 28$, $(x_0, y_0, z_0)^T = (-8, 7, 27)^T$. The trajectory on the Lorenz attractor is colored by the adaptive time step required, with red indicating a smaller time step. Figure is taken from Brunton, Proctor, and Kutz [44]

derivative $\dot{x}(t)$ at time step $t_1, t_2, \cdots, t_m$

$$X = \begin{bmatrix} X^T(t_1) \\ X^T(t_2) \\ \vdots \\ X^T(t_m) \end{bmatrix} = \begin{bmatrix} x_1(t_1) & x_2(t_1) & \cdots & x_n(t_1) \\ x_1(t_2) & x_2(t_2) & \cdots & x_n(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ x_1(t_m) & x_2(t_m) & \cdots & x_n(t_m) \end{bmatrix},$$

$$\dot{X} = \begin{bmatrix} \dot{X}^T(t_1) \\ \dot{X}^T(t_2) \\ \vdots \\ \dot{X}^T(t_m) \end{bmatrix} = \begin{bmatrix} \dot{x}_1(t_1) & \dot{x}_2(t_1) & \cdots & \dot{x}_n(t_1) \\ \dot{x}_1(t_2) & \dot{x}_2(t_2) & \cdots & \dot{x}_n(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ \dot{x}_1(t_m) & \dot{x}_2(t_m) & \cdots & \dot{x}_n(t_m) \end{bmatrix}.$$

Next, a library $\Theta(X)$ of candidate nonlinear functions of the columns of X is constructed. An example of a library that consists of constant, polynomial, and trigonometric terms is given

$$\Theta(X) = \begin{bmatrix} | & | & | & | & & | & | & \\ 1 & X & X^{P_2} & X^{P_3} & \cdots & sin(X) & cos(X) & \cdots \\ | & | & | & | & & | & | & \end{bmatrix},$$

where X^{P_n} denotes higher polynomials, i.e., X^{P_2} denotes quadratic nonlinearities in the state x

$$X^{P_2} = \begin{bmatrix} x_1^2(t_1) & x_1(t_1)x_2(t_1) & \cdots & x_2^2(t_1) & \cdots & x_n^2(t_1) \\ x_1^2(t_2) & x_1(t_2)x_2(t_2) & \cdots & x_2^2(t_2) & \cdots & x_n^2(t_2) \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ x_1^2(t_n) & x_1(t_n)x_2(t_n) & \cdots & x_2^2(t_n) & \cdots & x_n^2(t_n) \end{bmatrix}.$$

Each column of $\Theta(X)$ represents a candidate function for the dynamics $f(x(t))$. Based on the observation on the sparsity of terms, i.e., only a few of the nonlinearities are active, a sparse regression problem can be set up to determine the vector of coefficients $\Xi = [\xi_1 \ \xi_2 \ \cdots \ \xi_n]$ such that

$$\dot{X} = \Theta(X)\Xi.$$

Once Ξ has been found, the governing equations can be constructed as

$$\dot{x} = f(x) = \Xi^T(\Theta(X^T))^T,$$

where $\Theta(X^T)$ is a vector of symbolic functions of x, unlike $\Theta(X)$ which is a data matrix.

The effectiveness of the SINDy method depends on the choice of measurements, data quality, and sparsity of the function basis. In some problems, the columns of $\Theta(X)$ might need to be normalized, particularly when entries in X are small. One might need to test with different bases and use the sparsity and accuracy as evaluation metrics for the resulting model. In many areas, the data for the derivative $\dot{X}$ of X can not be measured, and $\dot{X}$ has to be numerically approximated which introduces noise into the data. These noises make it hard to find the true governing equations of the system. The SINDy method is also ill-suited for problems with a high state dimension, where dimensionality reduction techniques can not be applied. In general, the right coordinates and function basis are needed to yield correct sparse dynamics. Requirements on the system and solutions for overcoming these shortcomings are discussed in detail in the original paper [44].

2.7 Related works

Ever since Covid-19 emerged, countless researches have tried to model the dynamics of the disease with varying successes. With an on-going epidemic, it is generally challenging to model its dynamics under partially observations. As briefly introduced in Chapter 1, many researches had been conducted to overcome the issue of partial observation and unknown interactions in the ongoing Covid-19 pandemic. In this section, we are presenting in depth a non-exhaustive list of prior works and researches in Covid-19 modeling that share a close resemblance to the method that will be introduced in Chapter 3. This section aims to give the readers a clear view into how these models were created and which factors were considered to create these models.

[45] modified the classic SIR model and introduced a new compartment $T(t)$ to represent the number of individuals that are infect but currently in quarantine at time t. In addition, they introduced a time-varying term $Q(t)$ that represents the strength of quarantine which governs the number of individuals entering the quarantine compartment at each time step. An ANN is then used to learn $Q(t)$ from data, creating an UDE. The ANN consists of two densely connected hidden layers with ten nodes in each layer and the ReLU activation function. The system of ODEs for the model is

$$\begin{aligned}
\frac{dS(t)}{dt} &= -\frac{\beta S(t)I(t)}{N} \\
\frac{dI(t)}{dt} &= \frac{\beta S(t)I(t)}{N} - (\gamma + Q(t))I(t) \\
\frac{dR(t)}{dt} &= \gamma I(t) + \delta T(t) \\
\frac{dT(t)}{dt} &= Q(i)I(t) - \delta T(t) \\
Q(t) &= \mathcal{NN}(S(t), I(t), R(t); \theta)
\end{aligned}$$

3. Methodologies

4. Results

5. Discussion

6. Conclusion

References

[1] "WHO Coronavirus (COVID-19) Dashboard." (), [Online]. Available: https://covid19.who.int (visited on 09/30/2021).

[2] "Emerging COVID-19 success story: Vietnam's commitment to containment," Our World in Data. (), [Online]. Available: https://ourworldindata.org/covid-exemplar-vietnam (visited on 09/30/2021).

[3] B. T. T. Ha, L. Ngoc Quang, T. Mirzoev, N. T. Tai, P. Q. Thai, and P. C. Dinh, "Combating the COVID-19 Epidemic: Experiences from Vietnam," *International Journal of Environmental Research and Public Health*, vol. 17, no. 9, p. 3125, Apr. 30, 2020.

[4] "Vietnam's 'new COVID variant' part of existing Indian strain: WHO," Nikkei Asia. (), [Online]. Available: https://asia.nikkei.com/Editor-s-Picks/Interview/Vietnam-s-new-COVID-variant-part-of-existing-Indian-strain-WHO (visited on 09/30/2021).

[5] E. Mahase, "Delta variant: What is happening with transmission, hospital admissions, and restrictions?" *BMJ*, vol. 373, n1513, Jun. 15, 2021.

[6] "Rapid assessment of design and implementation of Government's 2nd support package for the affected by Covid-19 | UNDP in Viet Nam," UNDP. (), [Online]. Available: https://www.vn.undp.org/content/vietnam/en/home/library/Assessment2package.html (visited on 09/30/2021).

[7] "Vietnam poised to miss GDP target as COVID squeezes economy," Nikkei Asia. (), [Online]. Available: https://asia.nikkei.com/Spotlight/Coronavirus/Vietnam-poised-to-miss-GDP-target-as-COVID-squeezes-economy (visited on 09/30/2021).

[8] I. Rahimi, F. Chen, and A. H. Gandomi, "A review on COVID-19 forecasting models," *Neural Computing & Applications*, pp. 1–11, Feb. 4, 2021.

[9] D. Adam, "Special report: The simulations driving the world's response to COVID-19," *Nature*, vol. 580, no. 7803, pp. 316–318, 7803 Apr. 2, 2020.

[10] E. L. Ray, N. Wattanachit, J. Niemi, A. H. Kanji, K. House, E. Y. Cramer, J. Bracher, A. Zheng, T. K. Yamana, X. Xiong, S. Woody, Y. Wang, L. Wang, R. L. Walraven, V. Tomar, K. Sherratt, D. Sheldon, R. C. Reiner, B. A. Prakash, D. Osthus, M. L. Li, E. C. Lee, U. Koyluoglu, P. Keskinocak, Y. Gu, Q. Gu, G. E. George, G. España, S. Corsetti, J. Chhatwal, S. Cavany, H. Biegel, M. Ben-Nun, J. Walker, R. Slayton, V. Lopez, M. Biggerstaff, M. A. Johansson, N. G. Reich, and o. b. o. t. C.-1. F. H. Consortium, "Ensemble Forecasts of Coronavirus Disease 2019 (COVID-19) in the U.S.," p. 2020.08.19.20177493, Aug. 22, 2020.

[11] E. Dong, H. Du, and L. Gardner, "An interactive web-based dashboard to track COVID-19 in real time," *The Lancet Infectious Diseases*, vol. 20, no. 5, pp. 533–534, May 1, 2020.

[12] "COVID-19 - Mobility Trends Reports," Apple. (), [Online]. Available: https://www.apple.com/covid19/mobility (visited on 10/01/2021).

[13] "Data For Good Tools and Data." (), [Online]. Available: https://dataforgood.facebook.com/dfg/tools (visited on 10/01/2021).

[14] V. K. R. Chimmula and L. Zhang, "Time series forecasting of COVID-19 transmission in Canada using LSTM networks," *Chaos, Solitons, and Fractals*, vol. 135, p. 109 864, Jun. 2020.

[15] A. Ramchandani, C. Fan, and A. Mostafavi, "DeepCOVIDNet: An Interpretable Deep Learning Model for Predictive Surveillance of COVID-19 Using Heterogeneous Features and Their Interactions," *IEEE Access*, vol. 8, pp. 159 915–159 930, 2020.

[16] F. Shahid, A. Zameer, and M. Muneeb, "Predictions for COVID-19 with deep learning models of LSTM, GRU and Bi-LSTM," *Chaos, Solitons, and Fractals*, vol. 140, p. 110 212, Nov. 2020.

[17] W. O. Kermack, A. G. McKendrick, and G. T. Walker, "A contribution to the mathematical theory of epidemics," *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, vol. 115, no. 772, pp. 700–721, Aug. 1, 1927.

[18] ——, "Contributions to the mathematical theory of epidemics. II. —The problem of endemicity," *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, vol. 138, no. 834, pp. 55–83, Oct. 1, 1932.

[19] ——, "Contributions to the mathematical theory of epidemics. III.—Further studies of the problem of endemicity," *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, vol. 141, no. 843, pp. 94–122, Jul. 3, 1933.

[20] F. Brauer, "Compartmental Models in Epidemiology," in *Mathematical Epidemiology*, ser. Lecture Notes in Mathematics, F. Brauer, P. van den Driessche, and J. Wu, Eds., red. by J. .-M. Morel, F. Takens, and B. Teissier, vol. 1945, Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 19–79.

[21] K. Roosa and G. Chowell, "Assessing parameter identifiability in compartmental dynamic models using a computational approach: Application to infectious disease transmission models," *Theoretical Biology and Medical Modelling*, vol. 16, no. 1, p. 1, Jan. 14, 2019.

[22] K. A. Schneider, G. A. Ngwa, M. Schwehm, L. Eichner, and M. Eichner, "The COVID-19 pandemic preparedness simulation tool: CovidSIM," *BMC Infectious Diseases*, vol. 20, p. 859, Nov. 19, 2020.

[23] IHME COVID-19 Forecasting Team, "Modeling COVID-19 scenarios for the United States," *Nature Medicine*, vol. 27, no. 1, pp. 94–105, Jan. 2021.

[24] S. Chang, E. Pierson, P. W. Koh, J. Gerardin, B. Redbird, D. Grusky, and J. Leskovec, "Mobility network models of COVID-19 explain inequities and inform reopening," *Nature*, vol. 589, no. 7840, pp. 82–87, 7840 Jan. 2021.

[25] Y.-C. Chen, P.-E. Lu, C.-S. Chang, and T.-H. Liu, "A Time-Dependent SIR Model for COVID-19 With Undetectable Infected Persons," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 4, pp. 3279–3294, Oct. 2020.

[26] R. Li, S. Pei, B. Chen, Y. Song, T. Zhang, W. Yang, and J. Shaman, "Substantial undocumented infection facilitates the rapid dissemination of novel coronavirus (SARS-CoV-2)," *Science*, vol. 368, no. 6490, pp. 489–493, May 1, 2020.

[27] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud. "Neural Ordinary Differential Equations." (Dec. 13, 2019), [Online]. Available: http://arxiv.org/abs/1806.07366 (visited on 09/26/2021).

[28] C. Rackauckas, Y. Ma, J. Martensen, C. Warner, K. Zubov, R. Supekar, D. Skinner, A. Ramadhan, and A. Edelman. "Universal Differential Equations for Scientific Machine Learning." (Aug. 6, 2020), [Online]. Available: http://arxiv.org/abs/2001.04385 (visited on 09/11/2021).

[29] G. Cybenkot, "Approximation by superpositions of a sigmoidal function," p. 12,

[30] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, no. 2, pp. 251–257, 1991.

[31] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, Jan. 1989.

[32] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[33] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain.," *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958.

[34] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation Applied to Handwritten Zip Code Recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, Dec. 1, 1989.

[35] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[36] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis. "Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm." (Dec. 5, 2017), [Online]. Available: http://arxiv.org/abs/1712.01815 (visited on 10/04/2021).

[37] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 6088 Oct. 1986.

[38] S. Ruder. "An overview of gradient descent optimization algorithms." (Jun. 15, 2017), [Online]. Available: http://arxiv.org/abs/1609.04747 (visited on 10/05/2021).

[39] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam. "Searching for MobileNetV3." (Nov. 20, 2019), [Online]. Available: http://arxiv.org/abs/1905.02244 (visited on 10/09/2021).

[40] M. Raissi, P. Perdikaris, and G. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, vol. 378, pp. 686–707, Feb. 2019.

[41] I. Lagaris, A. Likas, and D. Fotiadis, "Artificial neural networks for solving ordinary and partial differential equations," *IEEE Transactions on Neural Networks*, vol. 9, no. 5, pp. 987–1000, Sep. 1998.

[42] Y. Guo, X. Cao, L. Bainian, and M. Gao, "Solving Partial Differential Equations Using Deep Learning and Physical Constraints," *Applied Sciences*, vol. 10, p. 5917, Aug. 26, 2020.

[43] J. Bezanson, S. Karpinski, V. B. Shah, and A. Edelman. "Julia: A fast dynamic language for technical computing." (2012).

[44] 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*, vol. 113, no. 15, pp. 3932–3937, Apr. 12, 2016.

[45] R. Dandekar, C. Rackauckas, and G. Barbastathis, "A Machine Learning-Aided Global Diagnostic and Comparative Tool to Assess Effect of Quarantine Control in COVID-19 Spread," *Patterns*, vol. 1, no. 9, p. 100 145, Dec. 11, 2020.

Glossary

ANN Artificial Neural Network. 1, 3, 8–18, 20

ARIMA Autoregressive Integrated Moving Average. 1

Bi-LSTM Bidirectional Long Short Term Memory. 1, 2

CNN Convolutional Neural Network. 1, 9

GAN Generative Adversarial Network. 1, 9

GPU Graphics Processing Unit. 1, 9

GRU Gated Recurrent Unit. 1, 2

LSTM Long Short Term Memory. 1, 2, 9

MLP Multi-Layer Perceptron. 1, 9, 10

MSE Mean Squared Error. 1, 11, 18

NeuralODE Neural Ordinary Differential Equation. 1, 15, 16, 18

NPI Non-Pharmaceutical Intervention. 1, 2

ODE Ordinary Differential Equation. 1, 3–5, 13, 15–17, 20

PDE Partial Differential Equation. 1, 13–15, 17

PINN Physics-Informed Neural Network. 1, 13, 17

ReLU Rectified Linear Unit. 1, 12, 20

RNN Recurrent Neural Network. 1, 2, 10, 15

SEIR Susceptible-Exposed-Infective-Removed. 1, 2, 4

SINDy Sparse Identification of Nonlinear Dynamical System. 1, 18, 20

SIR Susceptible-Infective-Removed. 1, 2, 4, 5, 8, 20

UDE Universal Differential Equation. 1, 18, 20

US United States. 1, 2