Using Epidemiology Model To Predict Case Numbers for COVID-19

Wang, Caiwei Wang, Shuyuan

March 7, 2021

1 Introduction

1.1 Problem

The early state and late stage of a pandemic is very different. At early stage, the case number grows exponentially. Government agencies and institutions want to the ability to forecast the number of cases in order to allocated medical and other resources. Furthermore, knowing how protocols such as stay at home order can affect the case number is extremely useful to make predictions. However, in order to predict number of cases in the future, the growth factor is needed and can be generated from fitting previous data to an epidemiology model. The exponential factor is based on two factors that can be learned from data: the infection rate, β and number of days a patient stays infectious D.

1.2 Context

One of the pressing problems in epidemiology is long term prediction of the spreading of an infectious disease. Of particular interest is how mitigation measures (government policies) can affect the number of infected in the future. Numerous efforts have been tried around the world. Many cities and states in the U.S. have ordered stay-at-home policies. It is useful to see how administering these orders can affect the case numbers, how would the case numbers react if the government revoke the orders.[1]

2 Data

We use data from JHU's public COVID-19 GitHub repository [2], and mobility data provided by Descarted Lab [3].

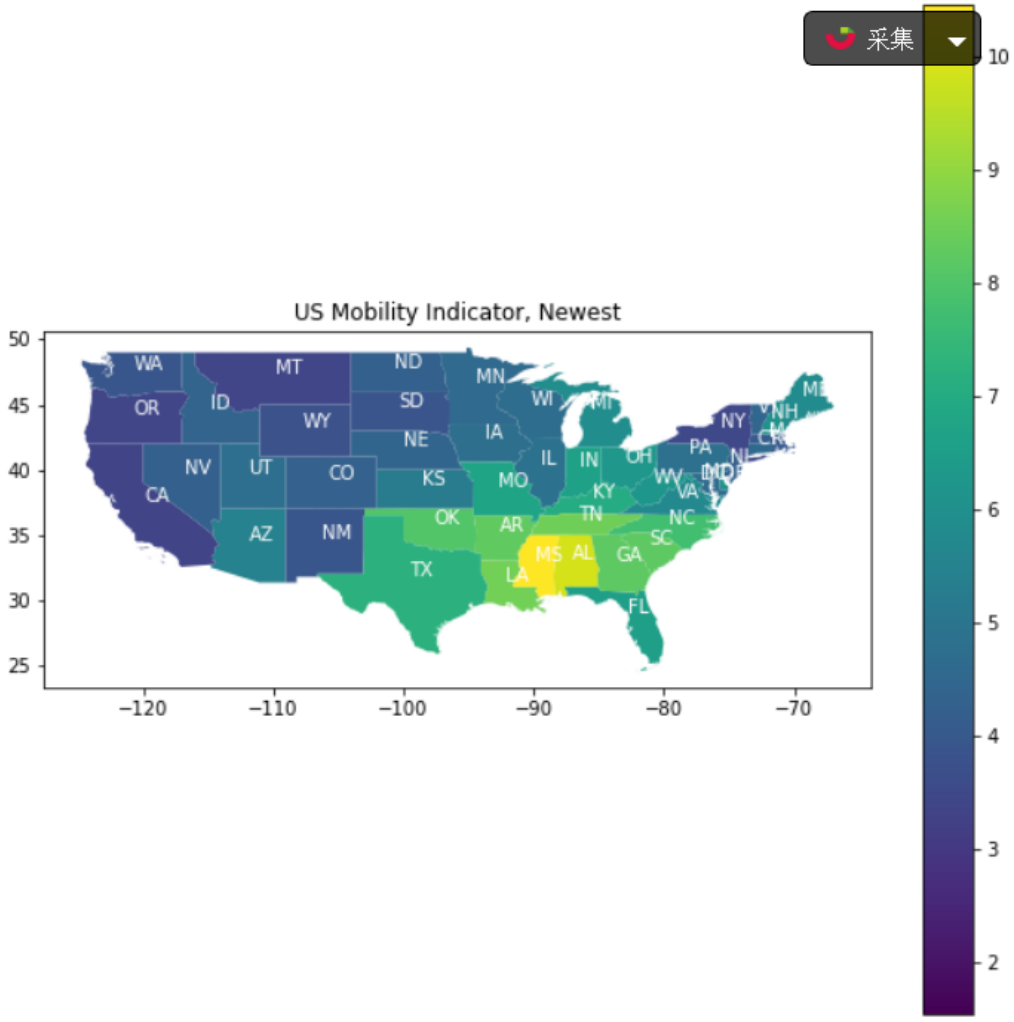
2.1 Collection

The U.S. confirmed, death and recovered case is updated daily by regional health departments of different jurisdictions, then collected by JHU, Because all the data is U.S. (unlike comparing data among different countries), the testing method and data reporting method are relatively uniform and reliable to make inferences on.
The mobility data is allocated from many sources. For example, Apple Map can estimate the distance a typical person travel in that region from their back-end data of users moving in that geological areas.

2.2 EDA

To perform gradient descent model fitting, we need to acquire data on the infected number, deaths, and recovered (also population) of a region. However, not everybody goes to hospital or gets tested.
Plus, due to the geographical closeness and transportation means, regions are hit differently, and transmission

between different regions can also led to the second wave or the third wave.
Our model is continually adjusted and improved based on a database that is updated in real time with COVID-19 data - not only to achieve patterns of change in the past, but also to predict future trends. We graph the cumulative number of infections, deaths, and recoveries in the United States since January 22, 2020.



Mobility data is measured by Apple Maps routing volume. Indicated above, mobility data varies from state to state, county to county. We will do further research to find relationship between mobility routing volume and COVID-19 transmission rate.

2.3 Description

The pandemic hits different jurisdictions at a different time. Because the data set is collected by jurisdictions, we are able to calculate growth factor and make predictions at county or state level.

In order to fit data to the epidemiology model, we need to have time series data of three variables, estimated from confirmed, death, and recovered case numbers.

$$\text{Infected}, I = \text{Confirmed Cases}$$
$$\text{Susceptible}, S = \text{Population} - \text{Confirmed Cases}$$
$$\text{Removed}, R = \text{Recovered} + \text{Deaths}$$

The mobility data is representing the distance a typical member of a given population moves in a day. With this data set, we are able to see how stay-at-home orders by different states and the pandemic itself have effect on average mobility trends. The data set also provides additional information for our epidemiology model. [We haven't touched on this yet]

3 Methods

3.1 Overview

We are using gradient descent to find infection rate and infection duration. Then we want to check whether the parameters can predict useful information (accuracy).

However, in reality, the case numbers of the entire country/state is not evenly distributed among counties. Besides computing the individual infection rate for all the counties, each as a separate entity from its neighbors, we will predict case numbers based on the mobility data provided by Descartes Labs[3] (how fast is people moving inside each county and across county boundaries), and the case numbers of each county's neighboring counties. We will use a dt around 0.001 day instead of 1 day to better predict the dynamics. This process requires a fixed β and D predetermined for each county.

Furthermore, β and D are also dynamic. So in the future, we will replace the fixed β and D with dynamic, changing according to the data.

3.2 Inferring Parameters

See Appendix 5.1 for code. We are using gradient descent to solve for β and $\frac{1}{D}$, infection rate and 1 over days staying infected. Given:

$$\xi = \frac{1}{D}$$
$$f_s(I_n, N, S_n) = -\beta \left(\frac{I_n}{N}\right) S_n,$$
$$f_I(I_n, N, S_n) = -I\xi + \beta \left(\frac{I_n}{N}\right) S_n,$$
$$f_R(I_n) = I_n \xi,$$
$$h = 1$$

Loss function can be calculated by:

$\frac{1}{N}\sum_{n=1}^{N} \nabla_\theta \left(\left(\frac{s(n+1)-s(n)}{h} - f_s(s(n), I(n), R(n); \theta)\right)^2 + \left(\frac{I(n+1)-I(n)}{h} - f_I(s(n), I(n), R(n; \theta)\right)^2 + \dots \right)$

To calculate the above the term, we need to use the chain rule to differentiate with respect to β and ξ

$\nabla_\beta = 2 \cdot \left(\frac{S_{n+1}-S_n}{h} - \left(-\beta S_n \frac{I_n}{N}\right)\right) \cdot \left(S_n \cdot \frac{I_n}{N}\right) + 2 \cdot \left(\frac{I_{n+1}-I_n}{h} - \left(-\xi_k I_n + \beta_k \frac{I_n}{N} S_n\right)\right) \cdot \left(-S_n \cdot \frac{I_n}{N}\right)$

$\nabla_\xi = 2 \cdot \left(\frac{I_{n+1}-I_n}{h} + \left(I_n \xi_k - \beta_k \frac{I_n}{N} S_n\right)\right) \cdot (I_n) + 2 \cdot \left(\frac{R_{n+1}-R_n}{h} - I_n \xi_k\right) \cdot (-I_n)$

First initialize θ at $\beta = 0.2$ and $\xi = 0.1$

Then, at each iteration update β and ξ according to the rules below

$\beta_{k+1} = \beta_k - h_G \partial_\beta L(\theta | s(1), \dots, s(N)),$

$\xi_{k+1} = \xi_k - h_G \partial_\xi L(\theta|s(1), \ldots, s(N))$. where the learning rate, $h_G = 1/N$. where N is the population

When β and D both converge (a.k.a is the same as the previous iteration), we stop the iterations and return the two value, in order to fit into the ODE model.

3.3 Determining Learning Rate

Lipschitz continuous gradient condition is essential to ensuring convergence of many gradient decent based algorithms[4].The step size should scale inversely with the Lipschitz contant. We can calculate the constant by taking the eigenvalue of the hessian matrix of θ See 5.1 to see code

The gradients are:

$$\nabla_\beta = 2 \cdot \left(\frac{S_{n+1} - S_n}{h} - \left(-\beta_k S_n \frac{I_n}{N} \right) \right) \cdot \left(S_n \cdot \frac{I_n}{N} \right) + 2 \cdot \left(\frac{I_{n+1} - I_n}{h} - \left(-\xi_k I_n + \beta_k \frac{I_n}{N} S_n \right) \right) \cdot \left(-S_n \cdot \frac{I_n}{N} \right)$$

$$\nabla_\xi = 2 \cdot \left(\frac{I_{n+1} - I_n}{h} + \left(I_n \xi_k - \beta_k \frac{I_n}{N} S_n \right) \right) \cdot (I_n) + 2 \cdot \left(\frac{R_{n+1} - R_n}{h} - I_n \xi_k \right) \cdot (-I_n)$$

The Hessian matrix will look like the following:

$$\begin{bmatrix} \frac{\partial \nabla_\beta}{\partial \beta} & \frac{\partial \nabla_\xi}{\partial_\beta} \\ \frac{\partial \nabla_\beta}{\partial_\xi} & \frac{\partial \nabla_\xi}{\partial \xi} \end{bmatrix}$$

We can calculate the hessian matrix given S_n, I_n and population N. n represents the timestamp, in our case, is the number of the day after the first day in our sequence. T stands for the total number of days in the sequence of data.

$$\begin{aligned}
\frac{\partial \nabla_\beta}{\partial \beta} &= \frac{1}{T} \sum_{n=1}^{T} \left(2 \cdot \left(S_n \cdot \frac{I_n}{N} \right)^2 + 2 \cdot \left(S_n \cdot \frac{I_n}{N} \right)^2 \right) \\
&= \frac{1}{T} \sum_{n=1}^{T} \left(4 \cdot \left(S_n \cdot \frac{I_n}{N} \right)^2 \right) \\
\frac{\partial \nabla_\xi}{\partial \xi} &= \frac{1}{T} \sum_{n=1}^{T} \left(2 \cdot I_n^2 + 2 \cdot I_n^2 \right) \\
&= \frac{1}{T} \sum_{n=1}^{T} 4 \cdot I_n^2 \\
\frac{\partial \nabla_\xi}{\partial_\beta} &= \frac{1}{T} \sum_{n=1}^{T} -2 \cdot S_n \cdot \frac{I_n^2}{N} \\
&= \frac{\partial \nabla_\beta}{\partial_\xi}
\end{aligned}$$

3.4 Adding Mobility and Location Variable

After checking the accuracy of our model fitting process, we would calculate infection rates β for all the counties. Then we want to better predict the case numbers with more data: mobility and geographic information. Using Geographical Information And Mobility Data to Predict County Infection are necessary for improving the accuracy of our model. In order to find nearby counties, We used data provided by US Census to find out each counties' neighbors. In order to achieve a more accurate prediction, the direction of mobility is necessary. Therefore, we initially set out four closest counties in west-north-east-south direction. for two neighboring points: x^1, x^2.

$$\Delta_x I(x,t)|_{(x^1,x^2,x^3)} \approx \frac{\partial_{x_1} I(x,t)|_{(x^1,x^2)} - \partial_{x_1} I(x,t)|_{(x^2,x^3)}}{x_1^1 - x_1^3} + \frac{\partial_{x_2} I(x,t)|_{(x^1,x^2)} - \partial_{x_2} I(x,t)|_{(x^2,x^3)}}{x_2^1 - x_2^3} \quad (1)$$

$$+ \frac{\partial_{x_1} I(x,t)|_{(x^4,x^2)} - \partial_{x_1} I(x,t)|_{(x^2,x^5)}}{x_1^4 - x_1^5} + \frac{\partial_{x_2} I(x,t)|_{(x^4,x^2)} - \partial_{x_2} I(x,t)|_{(x^2,x^5)}}{x_2^4 - x_2^5} \quad (2)$$

	BETA	D
US Country	0.18328024349177005	101.42266296342494
California State	0.17522671120713582	399.8722628878253
San Diego County	0.1522507750708867	797.6176130395337

If x is a uniform mesh, then at $x_{(2,2)}$,

$$\Delta_x I = \frac{(I(x_{(1,2)},t) - I(x_{(2,2)},t) - (I(x_{2,2},t) - I(x_{(3,2)},t))}{\Delta x^2} + \frac{I(x_{(2,1)},t) + I(x_{(2,3)},t) - 2I(x_{2,2},t)}{\Delta x^2} \tag{3}$$

$$= \frac{I(x_{(1,2)},t) + I(x_{(3,2)},t) - 2I(x_{2,2},t)}{\Delta x^2} + \frac{I(x_{(2,1)},t) + I(x_{(2,3)},t) - 2I(x_{2,2},t)}{\Delta x^2}. \tag{4}$$

$$\frac{\partial I(x,t)}{\partial t}\Big|_{t=t_2} \approx \frac{I(x,t_1) - I(x,t_2)}{t_1 - t_2}$$

$$\frac{I(x_2,y_2,t_1) - I(x_2,y_2,t_2)}{t_1 - t_2} = \frac{I(x_1,y_2,t_2) + I(x_3,y_2,t_2) - 2I(x_2,y_2,t_2)}{\Delta x^2} + \frac{I(x_2,y_1,t_2) + I(x_2,y_3,t_2) - 2I(x_2,y_2,t_2)}{\Delta y^2}.$$

$\Delta x = \frac{x_3 - x_1}{2}$. $\Delta y = \frac{y_3 - y_1}{2}$.
Assume that $t_1 - t_2 = 1$, and that

$$I(x_2,y_2,t_1) = I(x_2,y_2,t_2) + I(x_1,y_2,t_2) + I(x_3,y_2,t_2) - 2I(x_2,y_2,t_2) + I(x_2,y_1,t_2) + I(x_2,y_3,t_2) - 2I(x_2,y_2,t_2).$$

What to do at (x_1, y_2)?

$$I(x_1,y_2,t_1) = I(x_1,y_2,t_2) + I(x_0,y_2,t_2) + I(x_2,y_2,t_2) - 2I(x_1,y_2,t_2) + I(x_1,y_1,t_2) + I(x_1,y_3,t_2) - 2I(x_1,y_2,t_2).$$

3.5 Accuracy

After getting the β and D, we want to plug in these two values into an ODE model to check whether this model can predicts infection numbers of that specific region. In theory, the curve of the ODE model should fit our training data.

4 Results

We applied our methods to US, California and San Diego County. We obtained the following results:

At US country level (Figure 1), the model is underestimating the number of cases. At state and county level (Figure 2 and Figure 3), the model performs well predicting the number of cases.

5 Discussion

Fitted Epidemiology Models' predictions are either overestimating or underestimating. The predictions are off because we are isolating the state from neighboring states, county from neighboring counties; there are constant transmissions between neighboring regions.

We want to add another variable into our Epidemiology model, the mobility, how fast are people in an area changing locations, which can be obtained from the mobility data-set at county level. This part will be incorporated in Winter quarter.

Later we tried on a auto supervised learning model to see what we could learn from the performance from the such model. However, because of the limitation of the time series data set, it failed to predict the data of the later time after training on the earlier data set .

From the map we generated (attached below) , we can see that the counties we have really "inaccurate" predictions are San Luis Obispo, Imperial and San Diego counties, which are all in the corners of all the counties we took into consideration. The missing data for neighboring counties for the counties on edges/corners can explain away some of the inaccuracy in predictions for these counties.

6 Appendix

[1] Professor Ma

[2] JHU public GitHub Repository `https://github.com/CSSEGISandData/COVID-19/tree/master/csse_covid_19_data`

[3] Descarted Lab Mobility Data `https://github.com/descarteslabs/DL-COVID-19`

[4] Lipschitz continuous gradient `https://xingyuzhou.org/blog/notes/Lipschitz-gradient`

6.1 Code

```
from numpy import linalg as LA
def get_country(start_days,duration,country = "US"):
    s = [332865671, 332865671, 332865671, 332865671, 332865671, 332865671,
       332865670, 332865670, 332865662, 332865655, 332865632, 332865613,
       332865580, 332865503, 332865450, 332865284, 332865168, 332865093,
       332864910, 332864537, 332864133, 332863571, 332862804, 332862457,
       332861006, 332859167, 332856512, 332852521, 332846735, 332840254,
       332831566, 332820683, 332810066, 332798351, 332781931, 332763443,
       332744382, 332726361, 332705522, 332678896]
    i = [    16,     16,     16,     16,     16,     16,     17,     17,
           25,     32,     55,     74,    107,    184,    237,    403,
          519,    594,    777,   1150,   1554,   2116,   2883,   3230,
         4681,   6520,   9175,  13166,  18952,  25433,  34121,  45004,
        55621,  67336,  83756, 102244, 121305, 139326, 160165, 186791]
    r = [    5,     5,     5,     5,     6,     6,     6,     7,     8,
           8,    13,    14,    18,    19,    21,    24,    28,    29,
          36,    41,    55,    63,    69,    81,   112,   143,   285,
         358,   480,   606,   699,   859,  1219,  1516,  2165,  2791,
        3581,  5720,  9260, 11426]
    p = 332865687
    return s,i,r,p

def calculate_gradient(s,i,r,population,beta,epsilon):
    result1 = 0 #continue adding to solve for beta
    result2 = 0 #continue adding to solve for 1/D aka epsilon
    for n in range(len(s)-1):
        result1 += 2*(s[n+1]-s[n]+beta*s[n]*(i[n]/population))*(s[n]*i[n]/population)
        result1 += 2*(i[n+1]-i[n]-beta*s[n]*(i[n]/population) + i[n]*epsilon)*(-s[n]*i[n]/population)

        result2 += 2*(i[n+1]-i[n]+i[n]*epsilon-beta*i[n]*s[n]/population)*(i[n])
        result2 += 2*(r[n+1]-r[n]-i[n]*epsilon)*(-i[n])

    return result1,result2
```

```
def calculate(s,i,r,population,learning_rate1,learning_rate2):
    beta = 0.2
    epsilon = 1/14

    loss = 0
    length = len(s)
    betas = []
    ds = []

    for itera in range(1000): # do it for 1000 iterations.

        loss1,loss2 = calculate_gradient(s,i,r,population,beta,epsilon)
        beta_new = beta - learning_rate1* loss1/length #0.001 is the learning rate
        epsilon_new = epsilon - learning_rate2 * loss2/length
        if (beta_new == beta) & (epsilon_new == epsilon):
            print(beta_new)
            print(1/epsilon_new)
            break
        beta = beta_new
        epsilon = epsilon_new
        betas.append(beta)
        ds.append(1/epsilon)

    return betas,ds
def calculate_hessian(s,i,r,population):
    result_beta_second = 0
    result_epsilon_second = 0
    result_both_second = 0
    for n in range(len(s)-1):
        result_beta_second += 4*(s[n] * i[n]/population) **2
        result_epsilon_second += 4*i[n]
        result_both_second += -2*s[n]*i[n]**2/population
    return result_beta_second/len(s),result_epsilon_second/len(s),result_both_second/len(s)

if __name__ == "__main__":
    #Get 40 days of US data starting on the 30th day since the first case of coronavirus in Wuhan
    s,i,r,p = get_country(30,40)
    top_left,bottom_right,the_other_two = calculate_hessian(s,i,r,p)

    w, v = LA.eigh(np.array([[top_left, the_other_two], [the_other_two, bottom_right]]))
    lip_constant = w[w>0][0]
    learning_rate = 0.1/lip_constant

    iterations = 1000
    betas,ds = calculate(s,i,r,p,learning_rate,iterations)
    plt.plot(betas)
    plt.show()
    plt.plot(ds)
    plt.show()
```

6.2 Figures

'us.png'

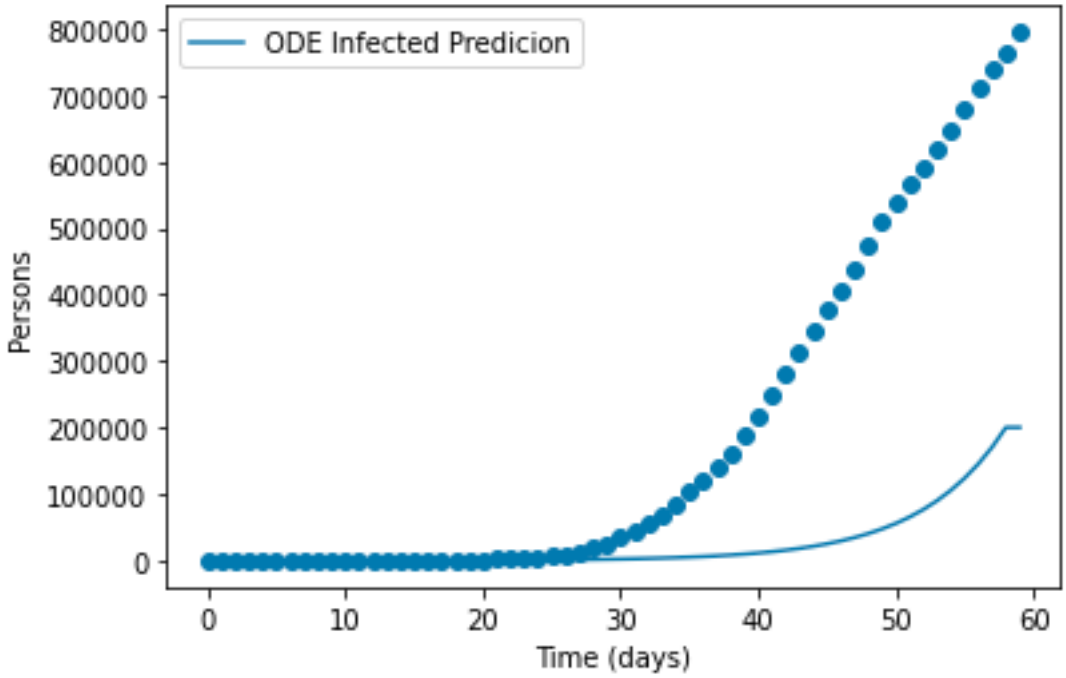


Figure 1: US Country Level, Model Predictions VS Actual Case Numbers

'california.png'

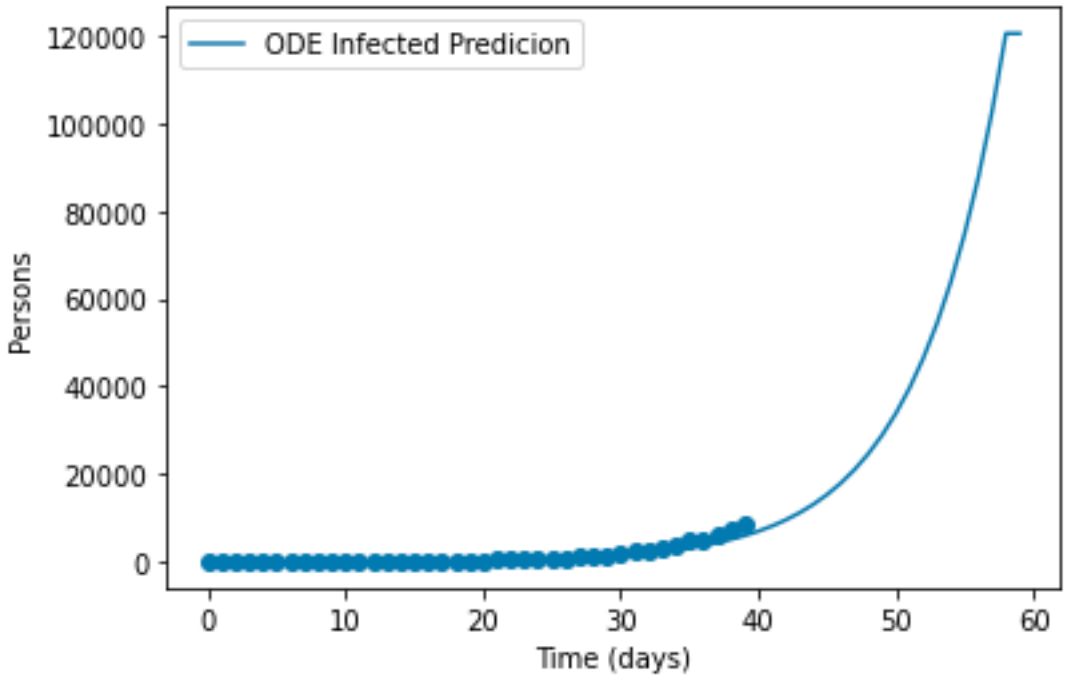


Figure 2: CA State Level 40-60 Days into the Pandemic

'sd.png'

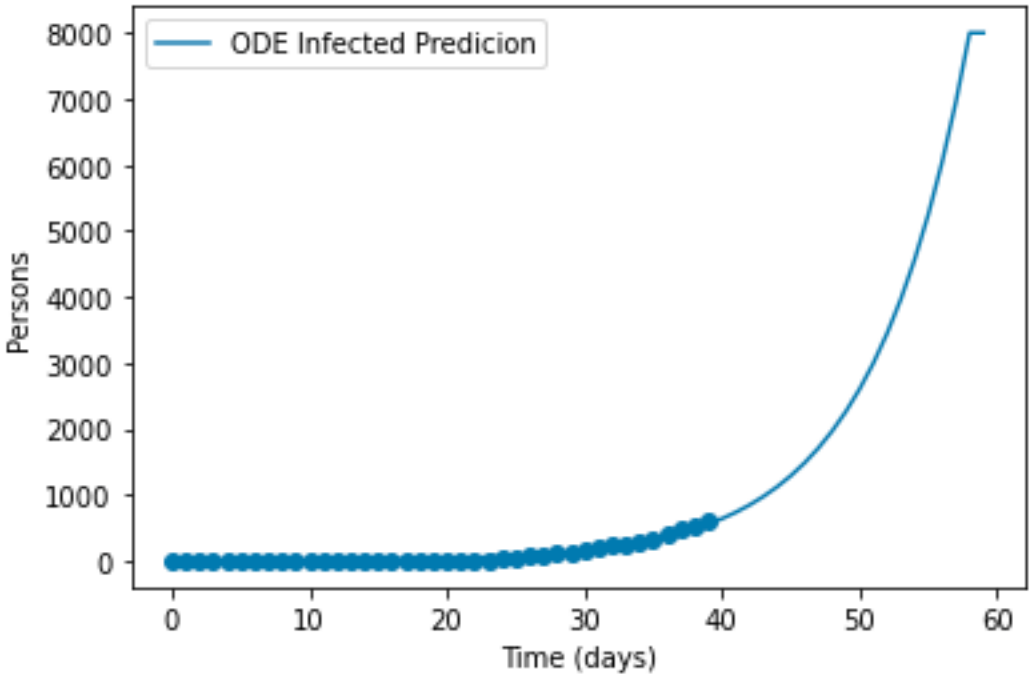


Figure 3: SD CountyLevel 40-60 Days into the Pandemic