COVID-19 Forecast and Modelling - Italy

Chin Hang Wong
Big Data Technology
Hong Kong University of
Science and Technology
HKSAR, CHINA
chwongar@connect.ust.hk

Lai Chung Lau
Big Data Technology
Hong Kong University of
Science and Technology
HKSAR, CHINA
lclauab@connect.ust.hk

Vincent Fan
Big Data Technology
Hong Kong University of
Science and Technology
HKSAR, CHINA
vkcfan@connect.ust.hk

ABSTRACT

Coronavirus disease 2019 (COVID-19) is a contagious respiratory and vascular disease. It caused a global pandemic by spreading worldwide in 2020 with more than 70 million confirmed cases as of December 2020. The goal of this report is to perform modeling and prediction to the number of COVID-19 cases in Italy. Network models are employed in this project, including the SIR model, the SIRD model, SEIR model. On the other hand, we used machine learning in particular Graph Neural Network which resulted in very accurate results.

INTRODUCTION

Background:

The first known confirmed case in Italy appeared in Rome on January 31, 2020. The source was confirmed to be from two Chinese tourists. After that, Italy immediately suspended all flights to China, including Hong Kong, Macau, and Taiwan. Then, they declared a state of emergency for 6 months. It is the first country in the European Union to fully prevent congestion. However, the virus continues to spread across Italy until May 2020. On May 4, Italy lifted lockdown and released restrictions on social activity later that month as the number of cases were much lower than the initial peak.

Since the end of September 2020, the virus has revived and prevailed again. On October 14, the positive cases of COVID-19 exceeded the peak of infection in March. On 4 November 2020, Italy announced a new lockdown. Until late December 2020, the lockdown is still valid but the degree of severity is reduced.

Motivation:

Italy was one of the severest european countries that suffered from COVID earlier this year. Modeling and evaluating Italy cases is a good way to combat with COVID-19 as we can better understand the reasons behind such data movement and predict future trends.

Problem definition:

The main purpose of this paper is to use mathematical modelling and machine learning principles to better understand the trend of COVID-19 cases in Italy and predict the trends as well.

Dataset

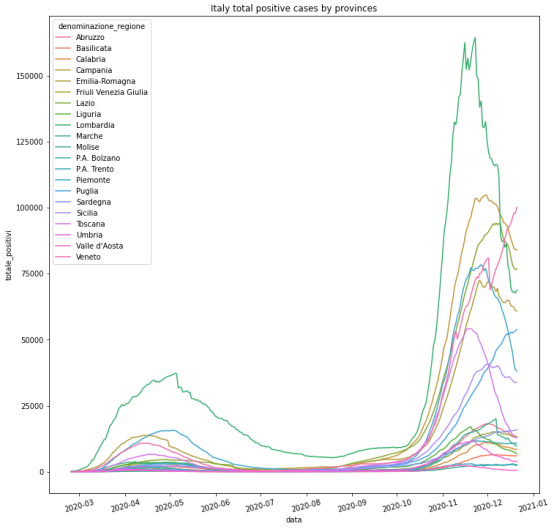
We retrieved the dataset from the Civil Protection Department of Italy [1]. The organization is a notional body in Italy that is responsible for prediction, prevention, and management of emergency events. The data is a csv file containing all the essential COVID related data like regions, daily new positive cases, daily deceased cases, daily recovered cases, etc. The dataset columns are all in italian so we are required to translate the columns back to English. The dataset is updated around 6pm every day since Feb 24, 2020. We stored the csv file in the Azure Container and retrieved the file using the container sharing URL. Once we set up the pipeline between our storage and coding environment. We can move on to the data pre-processing and data analysis part.

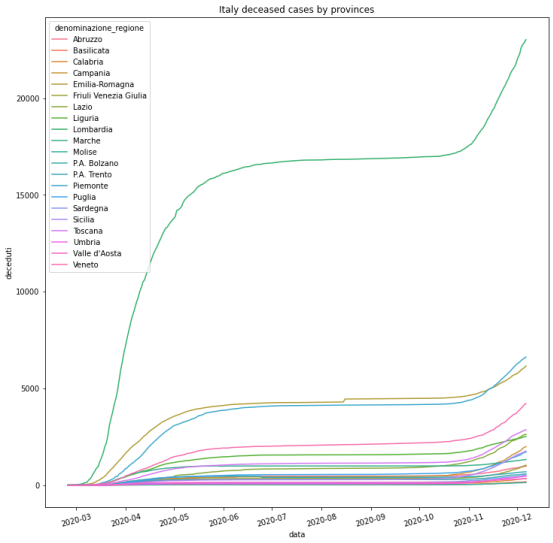
EDA

It is important to try to understand our data before performing any analysis. In order to proceed, we must first perform data preprocessing. Since the dataset comes from a reliable organization, the dataset is relatively clean and well formatted. We only needed to fill in 0 for a few NA positions, drop the notes columns, and reformat the date string for later processes.

The best way to get a better understanding of the data is by inspecting the distribution and trends of each individual column. This will allow us to check whether the data align with other public resources like Google. The two graphs below are the total positive cases for each region over time

and total deceased cases for each region over time respectively. Our Google Colab notebook will show the charts with more details. Please refer to our Colab link in the reference.





The trend in our dataset generally agrees with Google Covid-19 News trends. Lombardia and Veneto are both severe regions in both resources. Molise and Basilicata are less severe regions in the country and had lower impacts during those pandemic. As both graphs showed, Italy has undergone two big waves. The first wave ends around the beginning of August and the second wave starts around the beginning of September. Note that it is also very important that each province has its own contribution to the total cases while following the similar distribution but in different magnitudes.

SIR Model

Model background:

The SIR model is an infectious disease model. The model divides the population into three categories :

First category stands for Susceptible, whose number is denoted as S(t), which refers to those who are not infected, but lack immunity, and possible to be infected after contact with infected persons.

Second category refers to the Infective, whose number is denoted as I(t), which refers to people who are infected with diseases, which can be transmitted to susceptible.

Third category is referring to the Removal, whose number is denoted as R(t), which refers to a person who has immunity due to illness or is quarantined.

Suppose the total population is N(t), then N(t)=S(t)+I(t)+R(t).

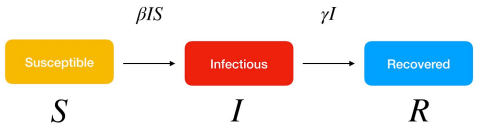
The SIR model is has made the following three assumptions:

1. There is no population dynamics. The population always maintains a constant. Birth rate, death rate, and mobility of the population are not considered.

2. Once a susceptible person comes into contact with an infected, she must be infectious. The model also assumes that the number of susceptible persons that a patient can infect per unit time is proportional to the total number of susceptible persons in the environment. So the number of people infected by all patients per unit time at time t is βS(t)I(t), where beta is the proportional coefficient.

3. The number of people removed from infected persons per unit time is proportional to the number of infected. The number of people removed per unit time is γi(t)., where gamma is the proportional coefficient.

Based on the above three assumptions, the SIR model is as follows:



$$\frac{dS}{dt} = -\frac{\beta SI}{N}$$
$$\frac{dI}{dt} = \frac{\beta SI}{N} - \gamma I$$
$$\frac{dR}{dt} = \gamma I$$

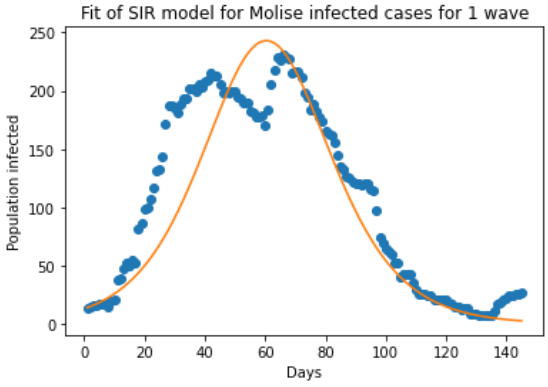
Since there are two peaks as previously observed, we need to separate the data into two periods in order to use the SIR model.

The first period is 14 days after the start of the data set to the last day of July 2020. The second period is from the start of August 2020 to 20th December 2020.

We use non-linear least squares to fit the SIR function.

Modeling Results:

For wave 1, the two best fitted regions based on Mean squared log error:

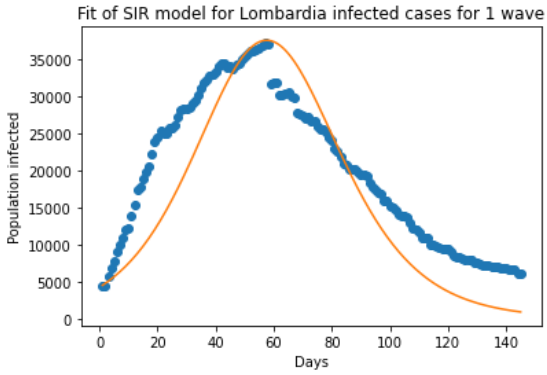


Optimal parameters:

beta = 1.787132305839738

gamma = 1.717952538683526

Mean squared log error is 0.27047525146162515



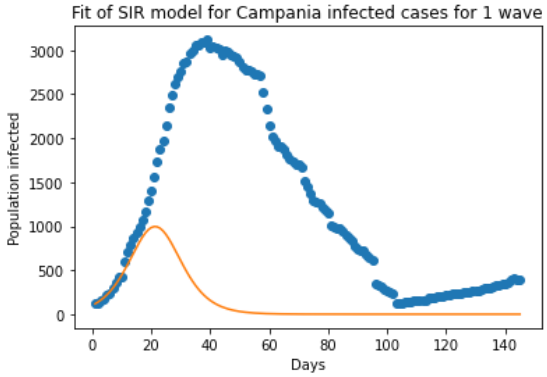
Optimal parameters:

beta = 0.7233481316025877

gamma = 0.6651605595058978

Mean squared log error is 0.4937207668574818

For wave 1, the two worst fitted regions based on Mean squared log error:

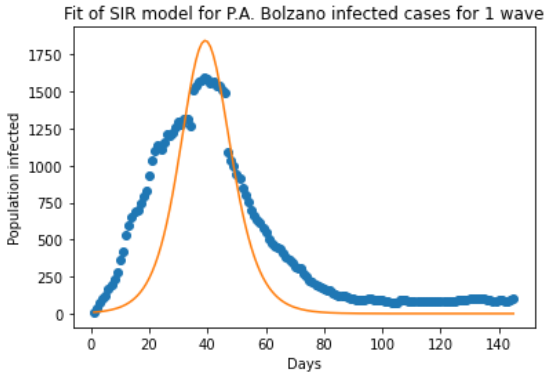


Optimal parameters:

beta = 9.190076075763859

gamma = 9.030265889782621

Mean squared log error is 24.236719381162935



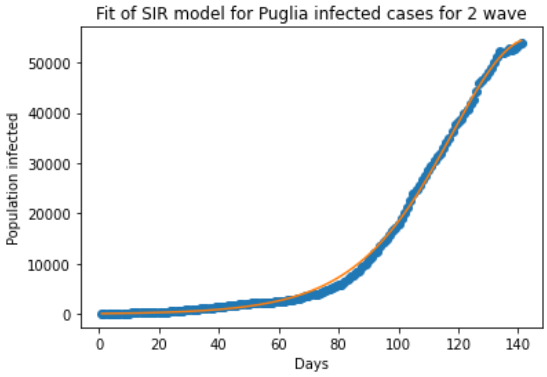
Optimal parameters:

beta = 2.128064491284612

gamma = 1.951840281952981

Mean squared log error is 9.250534882199236

For wave 2, the two best fitted regions based on Mean squared log error:

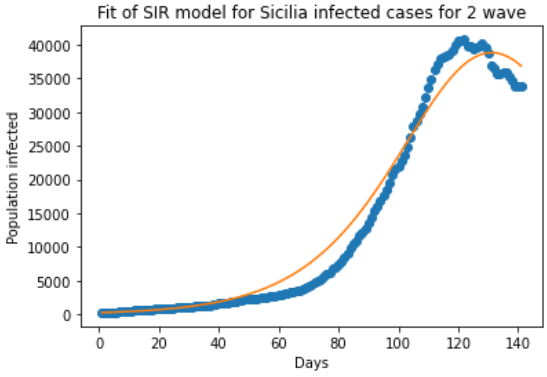


Optimal parameters:

beta = 0.3294455459228966

gamma = 0.2762211688223154

Mean squared log error is 0.04590677327574113



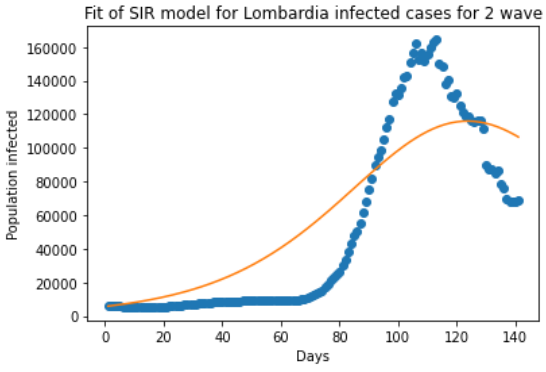
Optimal parameters:

beta = 0.3994050067989865

gamma = 0.35047413234114677

Mean squared log error is 0.07132699170682624

For wave 2, the two worst fitted regions based on Mean squared log error:

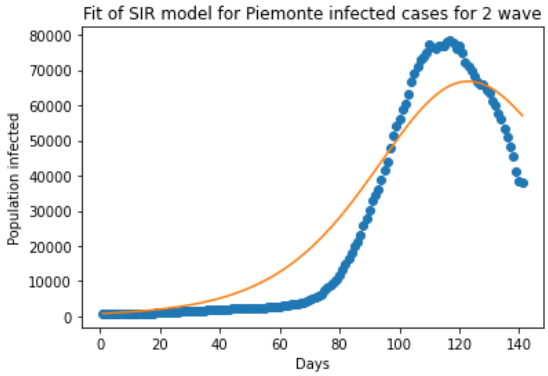


Optimal parameters:

beta = 0.24517858864611347

gamma = 0.20771817608206847

Mean squared log error is 0.671031029351356



Optimal parameters:

beta = 0.28411835605546953

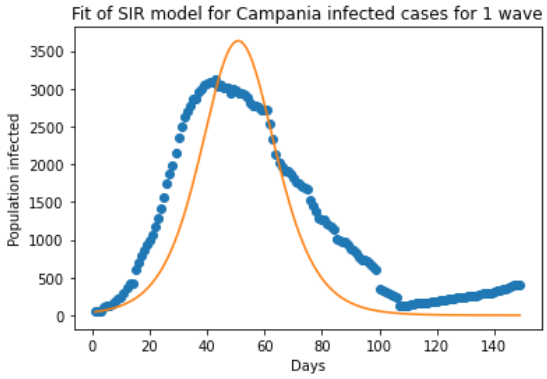
gamma = 0.23414210608983968

Mean squared log error is 0.634358955324747

Evaluation:

The model is not fitting well for some regions. We discovered that if the date of start is changed the result and performance will have dramatical differences.

For example, if we change the date of start four days earlier to the 10th day for Campania for 1st wave. The Mean squared log error reduced for 24.2 to 5.98.



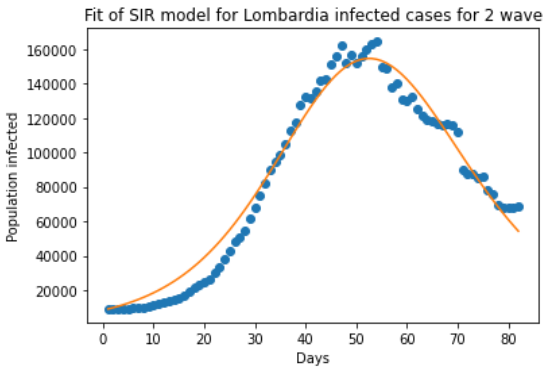
Optimal parameters:

beta = 3.295730608736182

gamma = 3.18031002646361

Mean squared log error is 5.983645595193276

The same techniques can also be applied to Lombardia and Piemonte for wave 2. If the starting date is 60 days later, the results are much better.

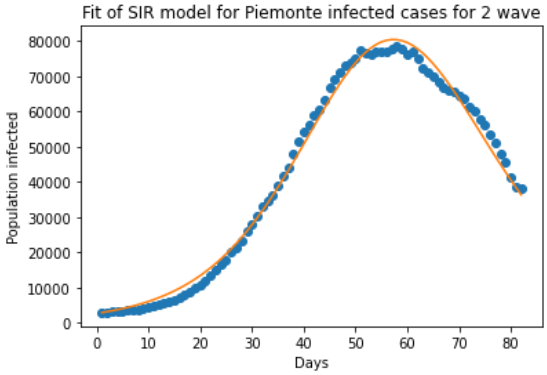


Optimal parameters:

beta = 0.49051356455625755

gamma = 0.4048928120269438

Mean squared log error is 0.05972923349553132



Optimal parameters:

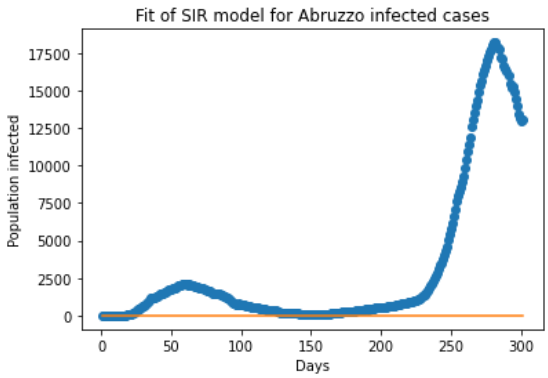
beta = 0.4593895949444321

gamma = 0.37194548086390056

Mean squared log error is 0.01903452803242127

This shows the SIR model is sensitive to the starting day. Choosing the right starting date will have a great impact on model performance.

Moreover, the model fails to deal with multi-peak problems. If we do not separate the date into two waves, the program will fail to generate results.



This shows the necessity of separation.

SIR Variants

Even though the SIR model provides a reasonable estimation of the epidemic situations, there are still lots of other features that need to be captured in order to present a realistic prediction. We chose two other SIR variants that can achieve better modelling results than original SIR models. The two variants are the SIRD model and the SEIR model.

SIRD Model

The original SIR model assumes no population dynamics and hence no death. In reality, however, death cases related to COVID19 are reported. To address this concern, we employed the SIRD model.

The model refines the assumption of the SIR model that an infected person has a chance of death at a constant rate mu for each day. The change in infected people is then also reduced by μI(t). The new system is shown below.

$$\frac{dS}{dt} = -\frac{\beta IS}{N},$$

$$\frac{dI}{dt} = \frac{\beta IS}{N} - \gamma I - \mu I,$$

$$\frac{dR}{dt} = \gamma I,$$

$$\frac{dD}{dt} = \mu I,$$

Since we have the data of infectious and death. We used least square regression to find the μ. Then, we constrain μ to our estimate and then solve the system using non-linear least squares. If we treat γi(t) in the SIR model as the γi(t)

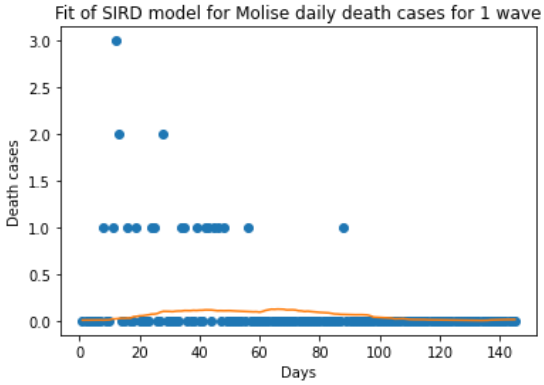
plus constant μi(t), the two systems are the same. Therefore, the estimates of infectious are the same of two models but γi(t) in the SIRD model is smaller.

Since the population infected graph will be the same as the SIR model, we will not show the result again in this section. Instead, we will highlight some spots about the death cases.

Results:

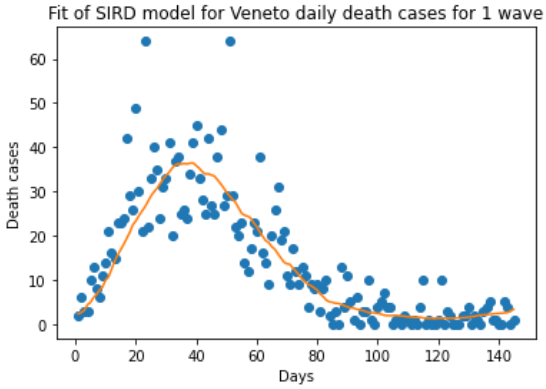
For wave 1:

Some regions have a few death cases.

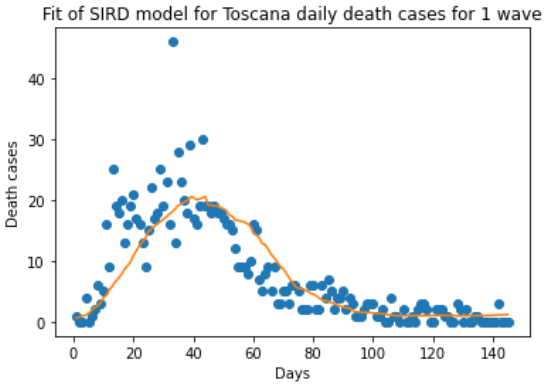


mu = 0.0005460057144485998

Some regions report more cases. But all regions show declining trends.



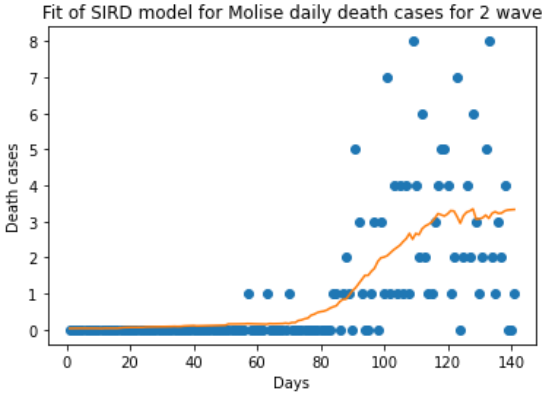
mu = 0.003374879264722325



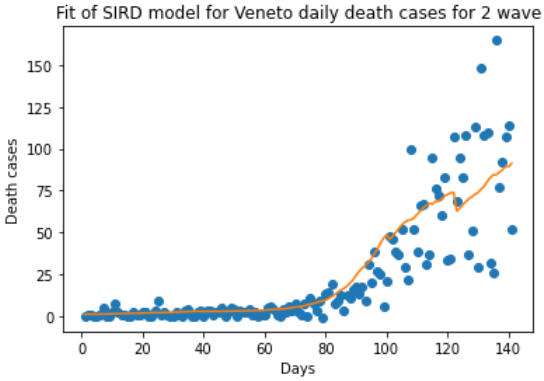
mu = 0.003103599104601055

For wave 2:

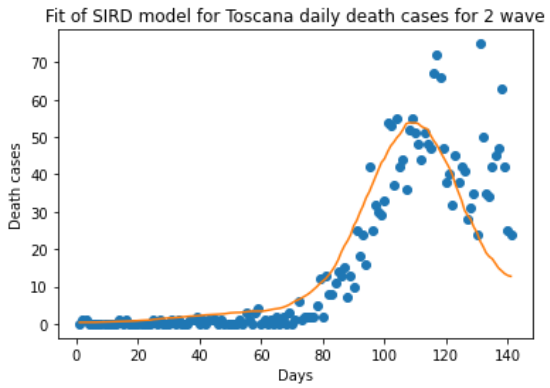
The death cases are more than wave 1 in general. We cannot see a declining trend for most of the regions. But mu is not showing a clear trend of moving up or down.



mu = 0.0012135961973601 54



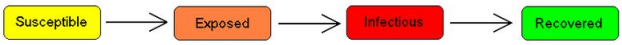
mu = 0.0009122607112815987



mu = 0.0009943914592349442

SEIR Model:

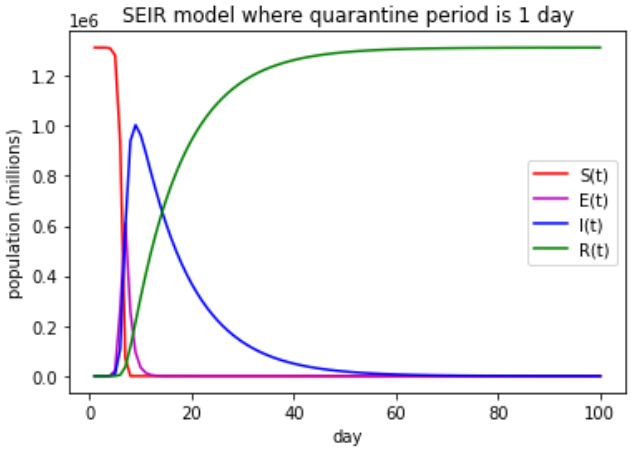
Compared to the SIR model, SEIR offers an additional state to represent the individuals under quarantine, where the population N = S+E+I+R and E is the number of people under quarantine.



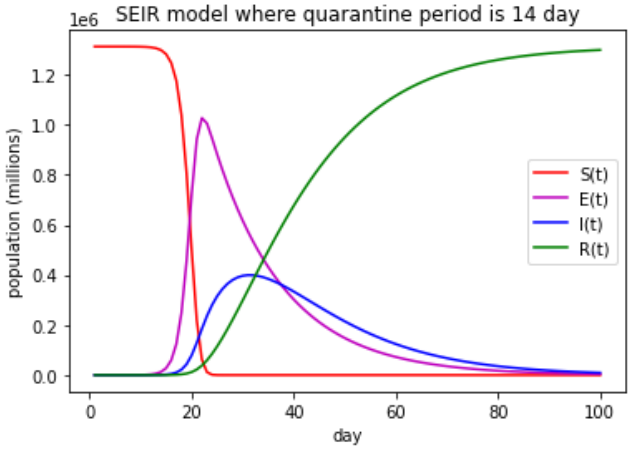
As the graph shown above, after an individual becomes infected, she will not be allowed to interact with the society so that she will not become infectious to other susceptible individuals. Here is the mathematical description of the model:

$$\frac{dS}{dt} = -\frac{\beta SI}{N}$$
$$\frac{dE}{dt} = \frac{\beta SI}{N} - \sigma E$$
$$\frac{dI}{dt} = \sigma E - \gamma I$$
$$\frac{dR}{dt} = \gamma I$$

Same as other models in our project, we will be using SEIR without vital dynamics which means we assume the total population is the same all the time, both population birth and death rates remain zero. Above formulas introduced a new parameter, sigma. It means how fast the individual can get out of quarantine. If the sigma decreases, the quarantine period will increase. Thus, we can expect the number of infectious individuals to decrease. Mathematically, sigma is equal to one over the quarantine period length. The figure shows that if we set the quarantine period to 1 day, the SEIR model would be very similar to the SIR model.



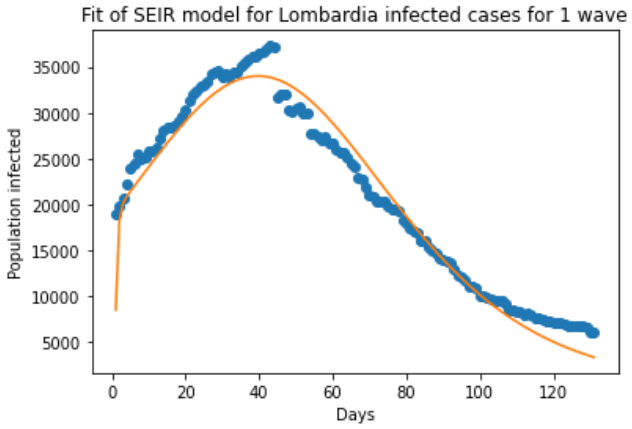
If we increase the quarantine period to 14 days, the number of infectious individuals would be significantly decreased.

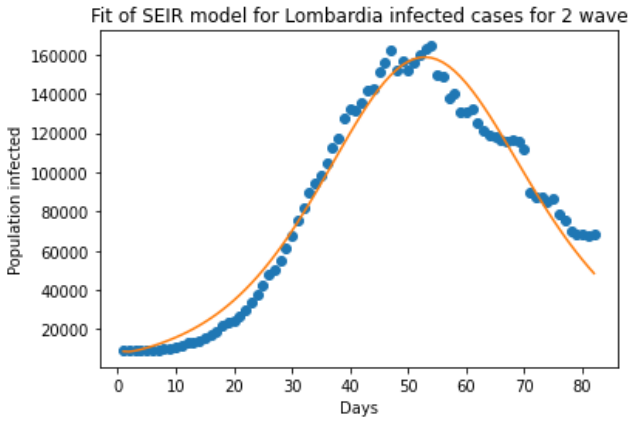


Unfortunately, the SEIR model does not fully capture the COVID-19 scenario. If a person is potentially infected, according to the model, they will eventually become infectious to the public after the latency period. In fact, patients can be cured during the latency period and move to R state directly. The latency period proposed by the SEIR model will only delay the outbreak of infectious individuals to the society.

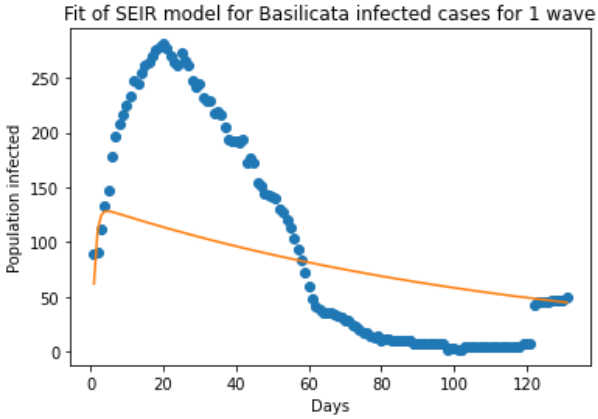
We tried to fit the SEIR model to the Italy COVID-19 data. We used a scipy package to solve the system of differential equations and try to fit the data into the model. Similar to other SIR model analysis, we broke down the data into 2 waves due to the SEIR model's limitation against multi-peak distribution. We cut off the data in Argust 1, 2020. And also, due to the accuracy issues, we discovered that skipping certain days in the beginning of each wave will make a curve fit closer to the actual data points. For this model, I skipped 28 days for the first wave and 60 days for the

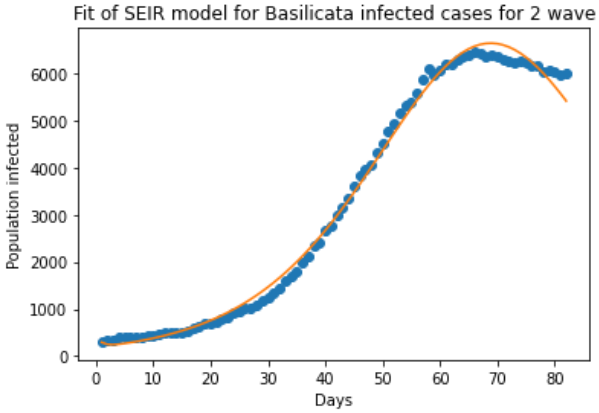
second wave. We grouped the dataset by regions and fitted the model against the grouped data. Our goal is to match the model prediction to the actual total positive case in the dataset. Once we fit the curve to the dataset, the scipy package will return us the optimal SEIR parameters to generate the curve. As a result, we were able to get some good curve fitting graphs but also some bad graphs. For the error measurement, we basically summed up the mean squared log error (MSLE) of wave 1 curve and MSLE of wave 2 curve. Province 'Lombardia' had the lowest MSLE which was around 0.0715.





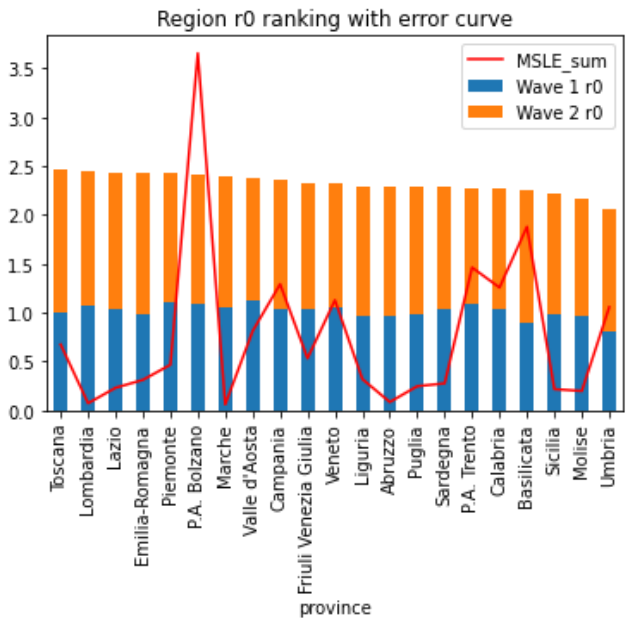
Basilicata had the lowest MSLE which is around 1.8772.





As we can see, some curves were not able to fit the data points very well and introduced a lot of errors which will also interfere with the parameters analysis.

Once we have the parameters estimation for all the regions, we calculated the virus length r0 by dividing beta by gamma. Here is the r0 trend of each region and each wave along with the MSLE of each region.



Despite there being some high error regions, in general we observed that the virus strength is much stronger in the second wave compared to the first wave. Potential causes are virus mutation, citizens got fatigued fighting against the virus or other unexpected causes.

Machine Learning

Motivation:

As neural networks have started gaining more traction due to its popularity and performance. We decided to explore the potential of applying neural networks to our problem. The reason why we believe neural network might be able to perform better than the SIR models is because neural networks are able to learn higher dimensional knowledge which the SIR models might have missed or are unable to perceive since it only used the Runge-Kutta methods instead of fitting the data in higher dimension.

On the other hand, after analysing the data and implementing the SIR models for each province, we notice a trend where neighbouring provinces have similar trends. This is a very logical deduction as neighbouring provinces means smaller travel distance between them. Smaller travel distance also means that there is a higher chance for the virus to travel from one province to its neighbouring provinces. Therefore, a logical deduction would be that there exists some underlying relationship between the province's coronavirus characteristics. For example, the number of hospitalized cases in one province might be reaching max capacity and must be redirecting the patients to the neighbouring hospital facilities. This shows that there can be multiple correlations between the province's data. Thus, we can potentially formulate this problem as a graph where the nodes are the provinces and edges are the connections between them. Then, our project's problem can be reduced to node value regression.

The idea of combining a graph problem and neural network leads to the idea of Graph Neural Network. Essentially it is a neural network that directly operates on a graph-like object. This also allows the neural network to have shared weights among the edges that creates a relationship between the nodes. Thus, enabling the network to infer knowledge on edges and neighbouring nodes. So, in theory, it should be better than typical neural network architecture since we have observations regarding the relationship between the neighboring provinces. Although node classification is the majority usage in other problems such as protein folding, we wanted to experiment with node regression and see how we can further improve the modeling of the SIR model and variants to create better results.

In this section, we will first explore the typical fully connected neural networks architectures, then we will move onto the graph neural networks and go deeper as well. Finally, we will combine the SIR models and GNN to mimic a model stacking behaviour.

Feed Forward Neural Network:

First, we decided to explore the potential of a simple neural network at predicting the total positive cases. This can also serve as a baseline model for the GNN performance comparison. Naturally, if we use a fully connected neural network to train on each province data separately like SIR, we can easily achieve high performance since it is only a linear regression problem. However, since our goal is to infer higher dimensional knowledge from the interrelationship of the nodes. Moreover, we would not be able to predict another province's data from another province's data because there are no weights shared among these individual regression models. Therefore, We have to train the feedforward neural network on all provinces data. Thus, the shape of the training data should be as follows:

$$N \times M \times K$$

Where N is the number of training examples or the date, M is the number of provinces, and K is the number of features. Note that we dropped some unnecessary columns such as string and location data. Moreover, we calculated the correlation matrix to drop some undesirable column members. For

example, we decided to drop the daily number of home isolation because it was shown to be 1 to 1 with our target column (total positive cases). Then, naturally, our target ground truth should have the shape of:

$$N \times M \times 1$$

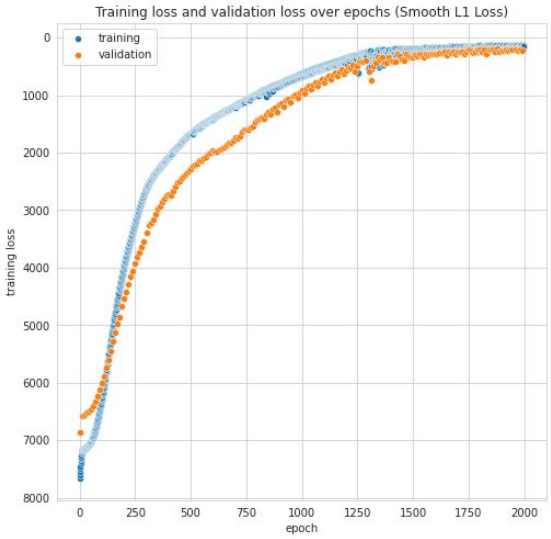
This will give us the regression result of each province for all training examples. Then, we chose to explore two kinds of popular neural network architectures. A wide but shallow neural network and a deep and narrow neural network.

Before we train the networks, we must also create node masks for validation purposes to measure overfitting and the generalization power. We used 80/20 training/validation splitting where 80% of the nodes (provinces) are used as training targets, whereas the 20% are not used for training but only evaluation.

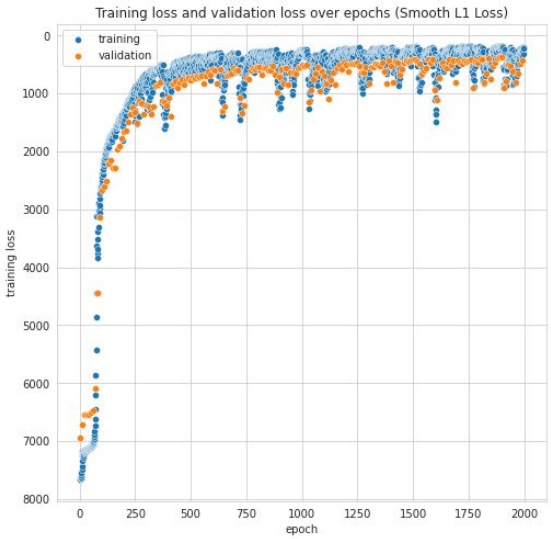
Then, we also need to have a suitable loss function to measure our network's performance so that it can be optimized. Since our target value is the number of total positive cases for each province, the scale of the target number is in the 105to 106scale, thus we would need a loss that does not have exploding gradients in this scale. A perfect candidate would be the L1 loss which is also known as L1 norm or mean absolute error. In particularly, the smooth L1 loss which excels for regression tasks when the target is very large. It can be described as a combination of L1 and L2 loss which earned its name of smooth as it creates a differentiable bridge between L1 and L2.

After the above definition, we can continue to the neural networks.

As the universal approximation theorem suggests, one hidden layer with arbitrary neurons is capable of capturing the data to an arbitrary performance. Thus, we created a wide neural network with over 500 hidden neurons in 1 single fully connected layer. Note that there are only 20 features so 500 hidden neurons are much wider than the feature space. The training curve is as follow:



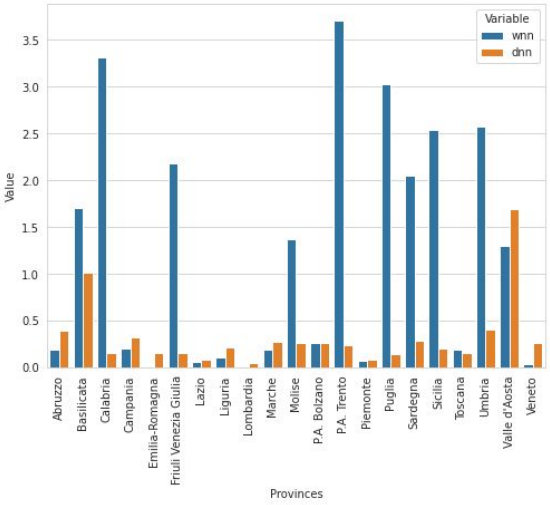
On the other hand, for the deep neural network, we choose to have 6 hidden layers where each layer has 50 neurons. This adds up to 300 neurons in the hidden layers which is less than the amount of hidden neurons in the previous wide neural network. This training curve is as follow:



As we can see from the both models' training curve, there is a difference between the wide neural network and the deep neural network. Although the number of neurons in the deep network is about 60% of the number of neurons in the wide network, we can see that the deep network outperformed the wide network in terms of loss and convergence rate.

From the graphs, we can observe that the initial training and validation loss are similar in both networks. However, the deep neural network converges much faster than the wide neural network. In order to better examine the difference in

performance, we also choose to find the mean squared log error from the true distribution. Here are the results:

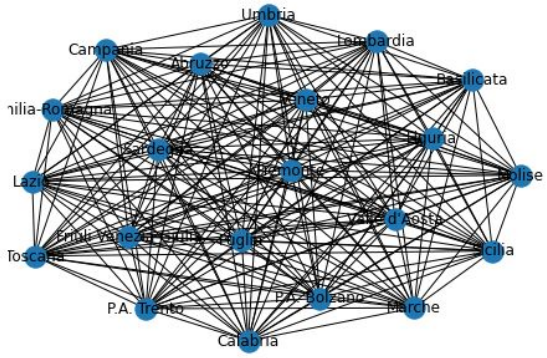


From the barchart, we can notice that while the losses are similar, the MSLE of the deep neural network is much better than the wide network in most cases. Therefore, we can conclude from our initial observation from the training curve that deep neural networks are slightly better than wide but shallow networks. Moreover, we can say that both networks are somewhat capable of inferring knowledge from neighboring nodes, while the deep neural network seems to be able to do a better job. This experiment has confirmed other scholar findings regarding deep neural networks. Therefore, we would also expect deep graph neural networks to perform even better than fully connected neural networks.

Graph Neural Network:

After implementing and training the neural networks, it is evident that there exists a higher dimension connection between the provinces which we can utilize the strength of graph neural networks and further extrapolate features and achieve better generalization results.

For this section, we used Pytorch and Deep Graph Library (DGL) libraries to create our graph neural network. First, we need to define our graph. As mentioned before, the nodes are defined as the provinces, the edges are the connection between each node, and the edge weight is the distance between each province. Then, the resulting graph is a fully connected graph with edge weights equals to the travelling distances between the nodes.



Similar to our fully connected neural network setup, we will be using smooth L1 loss as our loss function. We will also explore the difference between a wide but shallow and a deep but narrow graph neural network.

The major distinction between a graph neural network and typical neural network is that it operates on a graph object. In order to do so, we chose to use DGL's GraphConv module which stands for graph convolution. In particular, we chose to use GraphConv layer which is the introduced graph convolution operation published in the paper by Thomas Kipf etl. We also defined our fully connected layers within this convolution layer.

Before we get to training, it is important to note that the input shape requirement of a graph neural network using dgl is that the number of nodes must be in the first dimension. So, in order to maximize parallelization potential to speed up the training process. Our training data has the following shape:

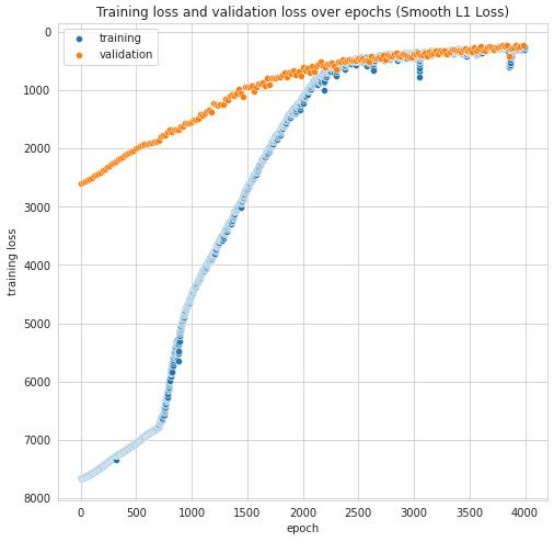
$$M \times N \times K$$

Similarly, the targe ground truth will be:

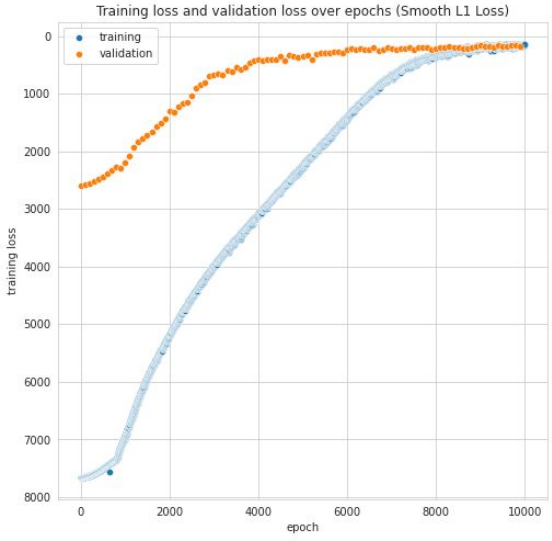
$$M \times N \times 1$$

Where N is the number of training data and K is the number of features. Note that we utilize the fact that Pytorch's Linear layer will only operate on the last dimension of the data, meaning that the hidden layer will transform the input data fromM N K to M N H where H is the number of hidden features.

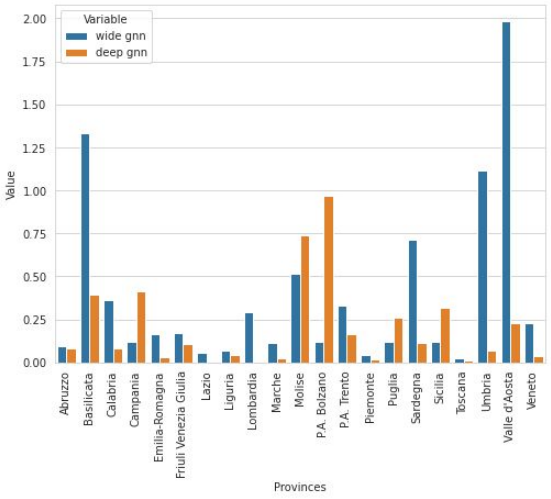
For the first wider model, we used 2 GINConv layers with 2 linear layers inside along with batch normalization and LeakyReLU activation layers, we also chose to have 500 neurons in each hidden layer, resulting in 2000 hidden neurons in total. The training curve is as follow:



On the other hand, the deeper graph neural network has 3 GINConv layers with 4 linear layers in each convolution layer, resulting in 12 hidden layers in total. We chose to have 50 hidden neurons in each hidden layer, which resulted in 600 hidden neurons in total. The training curve is as follow:

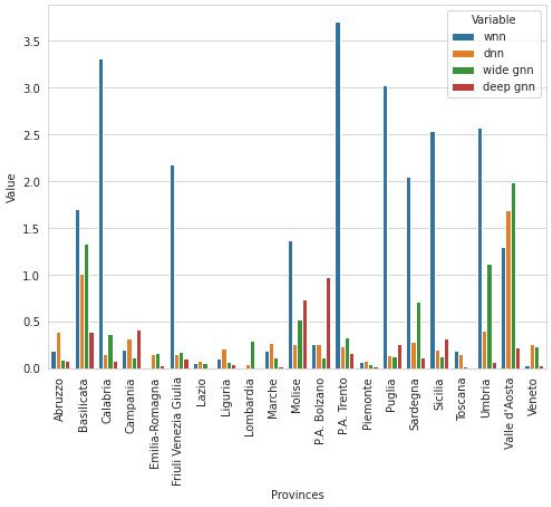


As we can observe from the training curve, we can see that the wide graph network was able to converge much faster than the deep graph network, this can be due to the fact that gradients are much harder to backpropagate in deeper neural networks since we are using LeakyReLU as our activation function. On the other hand, one interesting note is that the starting position of the validation loss is much higher than the neural networks. Note that we are using smooth L1 Loss which is why the validation loss can be lower than the training loss. Next, we should also examine the number of epochs needed to reach convergence. For the wide model, we only need about 4000 epochs to reach convergence while the deep model needs 10000 epochs to reach convergence. The final loss of both neural networks are similar, but we must again use mean square log error to examine the differences between the 2 models.



From the bar graph, we can see that on average the deep graph neural network performs better than the wide neural network where all errors are below 1. This confirms with our previous hypothesis that deep neural networks are more beneficial to generalize.

To summarize both the fully connected neural networks and graph convolutional neural networks, we have also plot the MSLE score against each other:

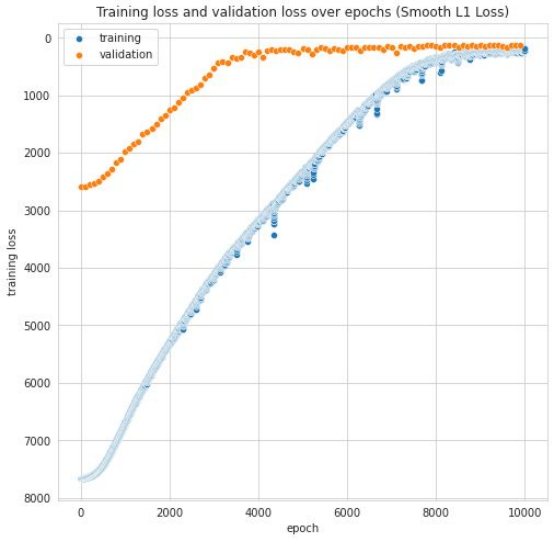


We can see that overall that deep graph neural network (red) is better than the other models in general. Note that the wide fully connected neural network performed the worst as we have predicted since other scholarly findings

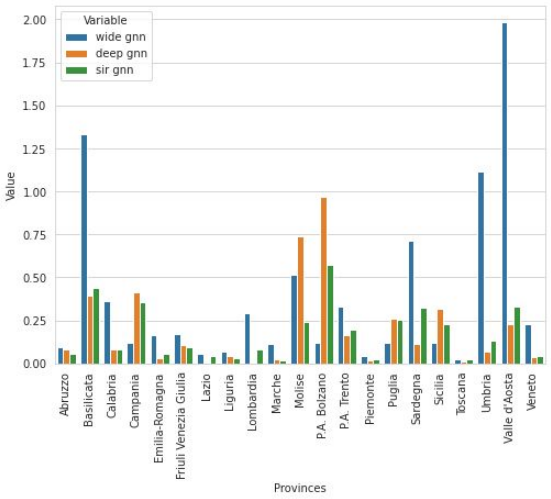
also concluded that deep learning has much more potential than wide learning.

Finally, after experimenting with both wide and deep graph neural networks, we are confident that graph neural network structure is beneficial in increasing our performance. Therefore, we have decided to use the SIR model prediction as an extra feature to our graph neural network. This can also be seen as a model stacking technique.

Model stacking is a technique that combines the learning of multiple models into one, which resembles the idea of bagging. By stacking these models, we will be able to infer more accurate learning as we have one SIR model and prediction for each individual province. Therefore, we can also see this graph neural network as a weighted sum regressor where it decides the weight between the SIR model prediction and its own regression results. Next, we choose to add the SIR prediction as a feature to the deep graph neural network as it was shown to be the best candidate to further improve. The training curve is as follow:



As we can see from the curve, it is very similar to the original deep graph neural network's training curve. However, if we also calculate the MSLE and compare with all other graph neural networks:



We can again observe that the performance of the SIR deep graph neural network outperforms or matches the previous deep graph neural network. We also calculated the total MSLE of all provinces where the SIR GNN received a total of 3.60 while the deep GNN received a total of 4.09. For the reference, the MSLE sum for the wide neural network is 28.8, deep neural network is 25.0, and wide graph neural network is 8.09. Although it seems to be insignificant, note that the error is logarithmic which means that it was a considerable amount of improvements. Therefore, we can conclude that our SIR deep graph neural network was a success.

Evaluation

This section aims to evaluate the performance of all models used in this paper. Firstly, the original SIR model was shown to be somewhat accurate in the sense that it was able to capture the infection wave meaning that COVID-19's infectious behaviour is still somewhat within the SIR model's assumption. Therefore, this gave us the confirmation that COVID-19 in Italy can still be represented by mathematical models.

Next, the performance of the 2 SIR variants, SIRD and SEIR, were also shown to be very promising in the sense that they were both able to capture more traits of COVID-19. To be more specific, we gain more understanding in the death rate and exposed rate of COVID-19 which showed us further insights and are definitely constructive knowledge to our report.

In terms of the MSLE score of the 3 SIR models, for some provinces, it was shown to be very accurate while it can also be very off from the target data.

On the other hand, we were able to achieve a much lower average MSLE using machine learning, in particular deep graph neural networks with stacks SIR model predictions.

In conclusion, the graph neural networks were much more accurate with the modelling and prediction. However, the insights from the SIR models were also very important for our understanding of the nature of COVID-19 in each individual province which gave us the intuition of utilizing graph neural networks.

Future Works & Conclusion

COVID-19 is an ongoing challenge for each of us. Virus is still mutating over time and policies of different countries keep changing from time to time. It is impossible to conclude a single model that can precisely predict the future trends. However, we can definitely make our model better by using more complex SIR variants.

According to our SIR model analysis, the number of infectious individuals would only increase rapidly if we have a high beta rate which is also the human interaction frequency. Regardless of the strength of the virus, if we can keep the human interaction rate as low as possible, the pandemic situation should be under control.

REFERENCES

[1] Presidenza del Consiglio dei Ministri - Dipartimento della Protezione Civile. (n.d.). Presidenza del Consiglio dei Ministri - Dipartimento della Protezione Civile/COVID-19. Retrieved December 22, 2020, from https://github.com/pcm-dpc/COVID-19

[2] Wong, C., Lau, L., & Fan, V. (n.d.). Google Colaboratory. Retrieved December 22, 2020, from https://colab.research.google.com/drive/18Md7NeEpC25nn4Jt96hMAWYufZUCpFSR?usp=sharing

[3] SmoothL1Loss. (n.d.). Retrieved December 22, 2020, from https://pytorch.org/docs/stable/generated/torch.nn.SmoothL1Loss.html

[4] Universal approximation theorem. (2020, December 13). Retrieved December 22, 2020, from https://en.wikipedia.org/wiki/Universal_approximation_theorem

[5] Kipf, T., & Welling, M. (2017, February 22). Semi-Supervised Classification with Graph Convolutional Networks. Retrieved December 22, 2020, from https://arxiv.org/abs/1609.02907

[6] Protezione Civile. (2020, November 06). Retrieved December 22, 2020, from https://en.wikipedia.org/wiki/Protezione_Civile

[7] (n.d.). Retrieved December 22, 2020, from https://sites.me.ucsb.edu/~moehlis/APC514/tutorials/tutorial_seasonal/node4.html

[8] SEIR and SEIRS models¶. (n.d.). Retrieved December 22, 2020, from https://docs.idmod.org/projects/emod-hiv/en/latest/model-seir.html

[9] Department, P., & 13, N. (2020, November 13). Italy: Resident population by region 2020. Retrieved December 22, 2020, from https://www.statista.com/statistics/617497/resident-population-italy-by-region/

[10] COVID-19 pandemic in Italy. (2020, December 21). Retrieved December 22, 2020, from https://en.wikipedia.org/wiki/COVID-19_pandemic_in_Italy