Step 1: Problem selection

[1] Wu, J. T., Leung, K., & Leung, G. M. (2020). Nowcasting and forecasting the potential domestic and international spread of the 2019-nCoV outbreak originating in Wuhan, China: a modelling study. *The Lancet*, *395*(10225), 689-697.

위 논문 [1] 은 중국 우한에서 발생한 코로나19 감염병 분석 논문 중 하나이다. 인용수가 약 4800회에 이르며 THE LANCET이라는 아주 유명한 저널에 투고된 논문이다. 이 논문은 (연립) 미분 방정식 모델인 SEIR을 이용하여 중국 우한의 코로나 19 확산도를 분석 및 예측했다. SEIR은 해석적으로 해를 표현할 수 없는 미분 방정식이기 때문에 수치적으로 근사하여 푸는 것이 일반적이다. 따라서, scML 수업에서 배운 PINN 방법을 이용하여 언급한 SEIR을 푸는 것은 아주 훌륭한 적용 예시가 될 것이다.

[2] Schiassi, E., De Florio, M., D'ambrosio, A., Mortari, D., & Furfaro, R. (2021). Physics-informed neural networks and functional interpolation for data-driven parameters discovery of epidemiological compartmental models. *Mathematics*, *9*(17), 2069.

[3] Kharazmi, E., Cai, M., Zheng, X., Zhang, Z., Lin, G., & Karniadakis, G. E. (2021). Identifiability and predictability of integer-and fractional-order epidemiological models using physics-informed neural networks. *Nature Computational Science*, *1*(11), 744-753.

[4] Grimm, V., Heinlein, A., Klawonn, A., Lanser, M., & Weber, J. (2022). Estimating the time-dependent contact rate of SIR and SEIR models in mathematical epidemiology using physics-informed neural networks. *Electronic Transactions on Numerical Analysis*, *56*, 1-27.

실제로, "physics informed neural networks epidemiology" 라고 검색하면 이미 많은 연구가 진행됐다. 첫째로, [2]는 Functional Connection이라는 것을 결합하여 SIR, SEIR, SEIRS를 연구했다. 다음으로, [3]은 다양한 compartment을 가정해서 PINN을 적용하여 성공적인 결과를 얻었다. (e.g. SEPOJDHR model). 마지막으로, [4]는 unknown constant를 time dependent parametrization을 통해 향상된 SIR, SEIR을 제시하였다. 이와 같이 PINN을 이용한 SIR 계열의 모델 분석은 활발하게 연구되고 있는 것으로 보인다.

*이후 나머지 reference는 마지막 장에 넣었습니다.

Step 2: Describe the problem.

코로나19 역학 데이터를 통해서 SIR 계열의 모델을 fitting하고 분석한다. 기본적으로 SIR 모델은 발생정도를 예측하고 미래에 발생할 수 있는 정도를 계산할 수 있다. 백신 혹은 자가격리 compartment를 도입한다면, 이것들의 effectiveness를 분석할 수 있다. 여건이 가능하다면, 면역자 혹은 사망자 compartment를 도입하여 더욱 다양한 요소를 분석할 수 있다.

(a) Why this problem is important (in science or engineering)?

코로나19는 지금까지도 유행하고 있는 거대 감염병 중 하나이다. 이를 모델링해야하는 이유는 다음과 같이 생각 할 수 있다.

1. 바이러스가 어떻게 퍼지고 미래에 어떻게 퍼질 것 같은지를 이해할 수 있다.
2. 고 위험 요소를 분석하여 superspreading 사건을 방지하는 것에 도움이 된다.
3. 정부가 미래를 계획하는 것에 도움이 될 수 있다. (e.g. 사회적 거리두기)

(b) What are the current challenges?

가장 중요한 문제는 (퀄리티가 높은) 데이터가 부족하다는 것이다. 시간이 흐를수록 바이러스에 대한 새로운 지식이 바뀌기 때문에 적절한 방법으로 이용가능한 데이터는 제한적이다. 또한, 코로나19의 증상은 일반적인 감기 증상과 비슷하여 컨트롤 타워에 보고하는 것이 늦거나 하지 않는 경우가 종종 발생한다. (사회적으로, 코로나19 감염을 밝히는 것이 어려운 측면도 고려할 수 있다.) 마지막으로 이상적으로 행동하지 않는 사람들의 패턴은 적합한 모델을 만드는 것에 악영향을 끼칠 수 있다. (e.g., 사회적 거리두기를 지키지 않는 사람)

(c) What is the state-of-the-arts in overcoming one of the challenges?

위와 같은 문제들 때문에 일부 연구자들은 Deep Learning based approach가 주목 하고 있는 것 같다. 딥러닝 방법은 interpretability가 떨어지지만 복잡한 상황을 잘 다루며 정확한 예측을 해내고 있는 각광받는 방법론이다. 예를 들어, 이미지 인식과 같은 분야는 엄청난 퍼포먼스를 보여주고 있다. 같은 맥락에서, 딥러닝이 복잡한 감염병의 잠재적인 의미 분석과 사람들의 비상식적인 행동패턴을 수정할 것이라 기대할 수 있다.

(d) How the problem is handled traditionally (if there is any)?

딥러닝이 발달하기 이전 효과적인 방법 중 하나는 확률론적/통계학적 기반의 접근이다. 확률론적 접근 방법으로는 branching process를 이용한 방법이 있다. 이러한 process의 equilibrium condition을 분석하여 감염병 종료를 예측할 수 있다. 통계학적인 방법으로 individual productive number라는 것을 추론하는 방법이 있다. 이를 통해서 감염병의 위험정도와 superspreading event 발생도를 정량적으로 분석할 수 있다. [5]

Step 3: Problem Formulation in Mathematics.

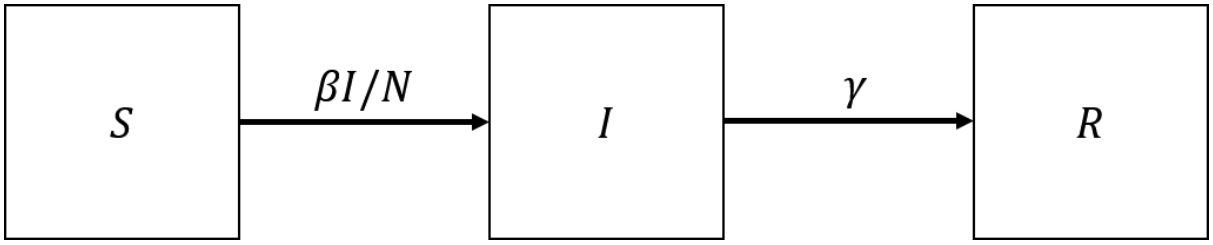

그림 1: SIR model

SIR (Susceptible-Infectious-Recovered) 모델은 3가지 compartment를 가진다. 시간 t에 대해서,

- S(t)는 Susceptible한 사람들의 모임으로서 감염병으로 위협을 받는 집단이다.
- I(t)는 Infectious의 약자로서 감염된 사람의 수이다.
- R(t)는 회복된 사람들의 수이다.

이 시나리오에 등장하는 모든 사람의 수를 N이라고 하자. 즉, $N = S + I + R$이다. SIR 모델은 다음과 같은 system of ODEs이다. [6]

$$\frac{dS}{dt} = -\beta S\frac{I}{N}$$

$$\frac{dI}{dt} = \beta S\frac{I}{N} - \gamma I$$

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

여기서 β, γ는 알려지지 않은 상수이며 각각 감염율과 회복율을 나타낸다. (위의 그림 1 참고).

이 system of ODEs를 풀기 위해서는 $S(0)$, $I(0)$, $R(0)$ (i.e., initial conditions)와 β, γ을 알아야한다. 게다가, 모든 값이 주어져도, 이 연립 미분 방정식의 닫힌 형태의 해는 존재하지 않는 것이 알려져 있다.

[Note] 통계학에서 관심 있는 unknown value중 하나는 basic reproductive number이다, R_0로 표기한다. 이는 한 사람이 감염시킬 수 있는 사람의 수의 기댓값이다. 전통적인 SIR 모델은 $R_0 = \beta/\gamma$로 놓는다.

Step 4: Describe Scientific Machine Learning Method.

SIR 모델의 솔루션 S(t), I(t), R(t)을 근사하는 식으로서 다음과 같은 neural network를 정의하자. (편의상 각각의 파라미터를 같은 기호로 적었다. 공유되는 값이 아니다.)

$$\mathrm{S^{NN}(t)} = \sum_{i=1}^{N} c_i \tanh(s_i(a_i t + b_i)) + \mathrm{d_i}$$

$$\mathrm{I^{NN}(t)} = \sum_{i=1}^{N} c_i \tanh(s_i(a_i t + b_i)) + \mathrm{d_i}$$

$$\mathrm{R^{NN}(t)} = \sum_{i=1}^{N} c_i \tanh(s_i(a_i t + b_i)) + \mathrm{d_i}$$

이는 width가 N이며, activation function을 $\tanh(x)$, adaptive activation이 정의된 two-layer two-layer neural network이다. 이들의 objective function은 다음과 같이 정의한다. 초기값 S(0), I(0), R(0)에 대해서,

$$\mathrm{Loss_S} = \left(\mathrm{S(0)} - \mathrm{S^{NN}(0)}\right)^2 + \frac{1}{M_{rd}} \sum_{j=1}^{M_{rd}} \left(\frac{dS^{NN}}{dt}(t_j) + \beta \frac{S^{NN}(t_j) I^{NN}(t_j)}{N}\right)\text{^2}$$

$$\mathrm{Loss_I} = \left(\mathrm{I(0)} - \mathrm{I^{NN}(0)}\right)^2 + \frac{1}{M_{rd}} \sum_{j=1}^{M_{rd}} \left(\frac{dI^{NN}}{dt}(t_j) - \beta \frac{S^{NN}(t_j) I^{NN}(t_j)}{N} + \gamma I^{NN}(t_j)\right)\text{^2}$$

$$\mathrm{Loss_R} = \left(\mathrm{R(0)} - \mathrm{R^{NN}(0)}\right)^2 + \frac{1}{M_{rd}} \sum_{j=1}^{M_{rd}} \left(\frac{dR^{NN}}{dt}(t_j) - \gamma I^{NN}(t_j)\right)\text{^2}$$

$$\mathrm{Loss_{final}} = \mathrm{Loss_S} + \mathrm{Loss_I} + \mathrm{Loss_R}$$

여기서, $\mathrm{M_{rd}}$는 sampling할 각 훈련 스텝마다 데이터의 수이다. 초기값과 β, γ는 데이터를 통해서 추론할 수 있다. 자세한 내용은 Step 5에서 기술한다.

전통적인 ODE solver들은 적당한 함수족을 가지고 ODE를 수치해석적 방법으로 푼다. (e.g., polynomials) 반면에 neural network은 어떠한 연속 함수도 근사 할 수 있기 때문에 (by universal approximation theorem) 복잡한 문제일수록 traditional method가 풀지 못하는 문제를 해결한다. 또한, PINN은 physical constraint 추가에 자유로우며, 병렬처리와 GPU 사용이 가능하다. [7]

Step 5: Data Generation or Acquirement.

Covid19datahub [8]에서 코로나 관련 통계량을 다운 받을 수 있다. 대한민국의 통계량 또한 제공되어 Kor.csv를 다운받았다. SIR 모델을 사용하기 때문에 사용할 데이터는 confirmed (infectious)와 recovered 정보이다. 사후 모델 평가를 위해서 결측치가 연속적으로 나타나지 않는 가장 긴 시간만을 사용했다. 즉, 62번째 행부터 550번째 행 사이의 값들이 사후 모델 평가에 어려움이 없다. (시간이 적혀 있지 않아서 행이라는 단어로 기간을 표현했다.)

우측 사진에 의하면 $I(0) = 9037, R(0) = 3507$이다. 대한민국 인구수를 5천만이라고 가정하면 $S(0) = 500000000 - I(0) - R(0) = 49987456$ 이다. 너무 숫자가 크면 연산량이 많기 때문에, 모두 1000으로 나누어 rescaling 작업을 하였다.

	confirmed	recovered
0	9037	3507.0
1	9137	3730.0
2	9241	4144.0
3	9332	4528.0
4	9478	4811.0
...	...	...
484	184103	163073.0
485	185733	164206.0
486	187362	165246.0
487	188848	166375.0
488	190166	167365.0

489 rows × 2 columns

그림 2: 489일간 대한민국 코로나 통계

나머지 β, γ를 추론하는 것은 굉장히 중요한 일이다. 이 값에 따라서 모델이 추론하는 의미가 굉장히 다르기 때문이다.

먼저 γ는 1/회복기간으로 정의할 수 있고, 회복기간은 original data에서 confirmed의 숫자와 recovered 숫자가 같을 때 기간차이를 계산했다. (자세한 코드는 후술) 이런 방식으로 얻은 값이 16, 14, 17, 16, 19, 13, 50, 47, 20 이었고 50과 47을 이상치로 간주하여 나머지 값들의 평균을 이용했다. 따라서 $\gamma = \frac{1}{16} = 0.06$을 얻었다.

[9]의 추정치에 의하면 유럽에서의 basic reproductive number는 2.2로 보고했다. 대한민국에서도 비슷한 양상을 가진다고 가정하면, $R_0 = \beta/\gamma$의 관계를 이용해서 $\beta = 2.2 \times 0.06 = 0.132$를 얻을 수 있다.

마지막으로, PINN의 훈련 시 필요한 내점은 $[0,489]$ 에서 무작위로 M_r 개 생성시켰다. Torch.rand(size=(first_value, second_value)) 코드를 통해서 얻을 수 있다.

Step 6: Training/Optimization related Techniques

최초의 세팅은 다음과 같이 설정하였다.

- Random seed = 42
- (The width of NNs) = 50
- β = 0.132
- γ= 0.06
- N = 50,000,000 / 1,000
- $I_0 = 9037/1000$
- $R_0 = 3507/1000$
- $S_0 = N - I_0 - R_0$
- Epochs = 500
- $R_d = 500$ & data are generated from $\mathrm{Uniform}(0,489)$
- NN의 구성은 Step 4 참고
- Optimizer: Adam (with default parameters)

사전 지식을 이용하여 다음과 같은 기술을 추가하였다.

- S, I, R은 양수 값을 추론한다. 따라서 각 네트워크는 $\log S, \log I, \log R$을 추정한다.
- N = S + I + R이라는 constraint을 만족하기 위해 각 네트워크는 상황에 맞는 최댓값을 가진다. S가 결정되면 I는 N-S 이하여야 하고, I가 결정되면 R은 자동으로 N-S-I가 되어야한다.

Implementation은 [10]의 구현 결과를 참고하여 만들었다. 이것은 SIR modeling과는 관련 없지만 PINN을 이해하기 좋은 Pytorch Implementation이다.

Step 7: Numerical Simulations and Results.

훈련 결과는 다음과 같다.

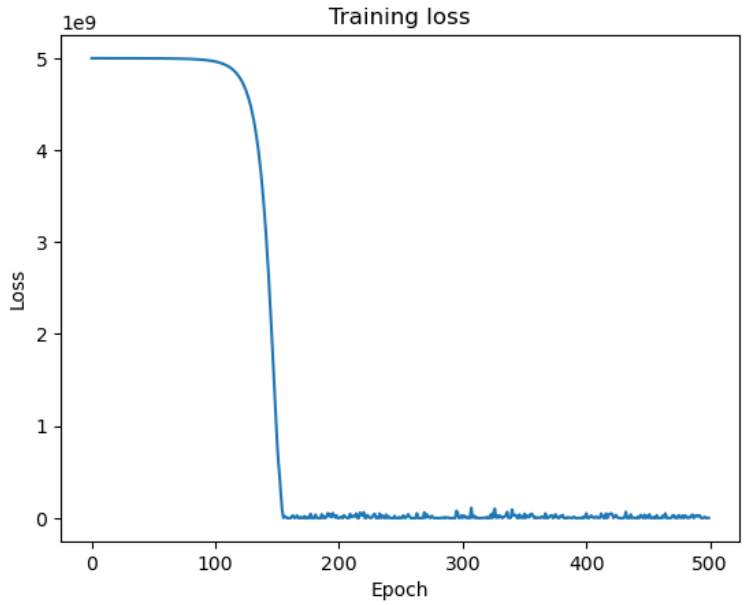

그림 3: The resulting training losses

위 그림은 epoch 마다 training loss를 그린 것이다. 생각보다 잘 수렴하였지만 training scheme을 조정하여 더욱 loss를 낮추는 것이 가능한지 시도해보아야 한다. 실제로, 마지막 Loss 값은 1017929.9375이다.

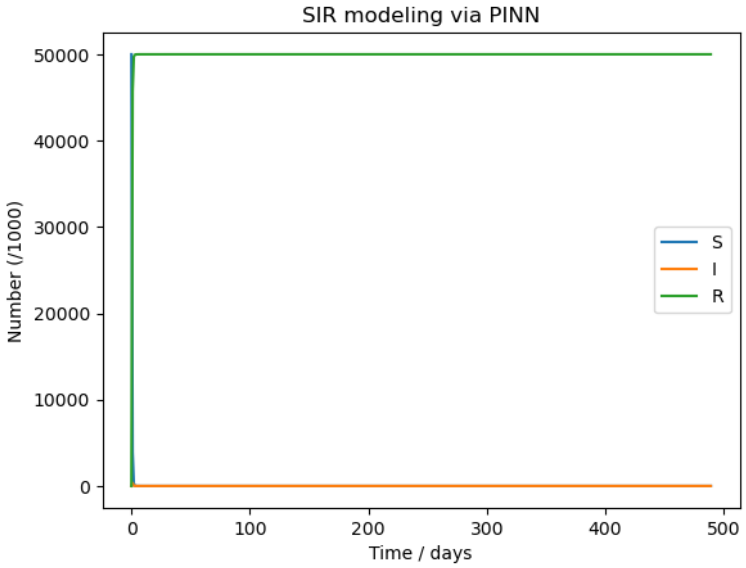

그림 4: The inferred solution to the SIR model.

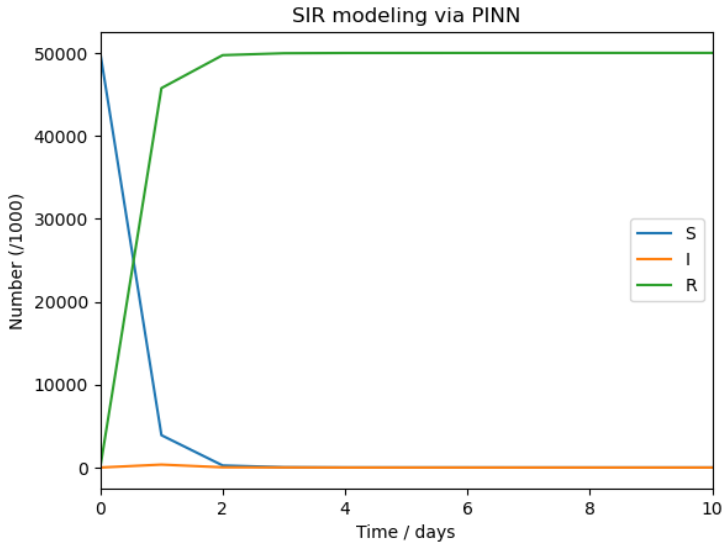

그림 5: The inferred solution to the SIR model, xlim=(0,10).

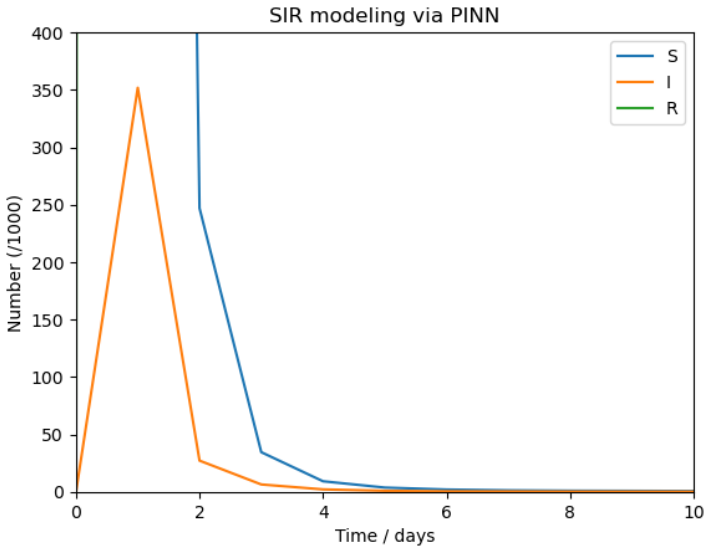

그림 6: The inferred solution to the SIR model, xlim=(0,10), ylim=(0,400).

	True initial value	The fitted value	MSE
S	49987.456	50000	157.32
I	9.037	0	81.66
R	3.507	0	12.29

그림 7: Initial values

그림 4를 보면 올바르게 훈련이 된 것처럼 보이지 않을 수 있다. 그러나 그림 5와 그림 6을 참고하면 처음 10일 이내에 모든 결과가 수렴하는 솔루션을 제시하였다. 그림 7은 initial value가 어느 정도 역할을 했는지 알 수 있는 표이다. 셋 모두 제대로 맞추지 못한 것처럼 보이지만 training loss가 1m 단위이므로 initial value의 optimization은 (상대적으로) 잘 됐다고 생각한다. 하지만 여전히 불만족스럽다.

관찰 1: training loss를 줄이는 것이 강렬한 변화를 주는가?

그렇지 않다. 한 예로서, 그림 3에서 loss 값들의 그래프를 분석하면 loss가 순간적으로 300까지 작아지는 것을 확인 할 수 있다. 이 때 training을 조기종료하고 evaluation을 진행하면 그림 4의 결과와 큰 차이가 없는 것을 확인하였다.

관찰 2: 신뢰할 수 있는 결과인가?

그렇지 않다. 하지만 최초의 시도가 부족하다는 그럴듯한 이론적인 이유를 설명할 수 있다. 각 솔루션의 개형을 결정하는 것은 그것의 도함수 부분이고 이것과 관련된 항은 다음과 같다.

$$\text{Loss}_1 = \sum_{j=1}^{M_{rd}} (\frac{dS^{NN}}{dt}(t_j) + \beta \frac{S^{NN}(t_j) I^{NN}(t_j)}{N})\text{^}2$$

$$\text{Loss}_2 = \sum_{j=1}^{M_{rd}} (\frac{dI^{NN}}{dt}(t_j) - \beta \frac{S^{NN}(t_j) I^{NN}(t_j)}{N} + \gamma I^{NN}(t_j))\text{^}2$$

$$\text{Loss}_3 = \sum_{j=1}^{M_{rd}} (\frac{dR^{NN}}{dt}(t_j) - \gamma I^{NN}(t_j))\text{^}2$$

예를 들면, 네트워크 입장에서 급수 내부항의 값을 줄일 수 있는 가장 쉬운 방법은 모든 항을 0으로 만들어버리는 것이다. 즉, 최대한 도함수가 0이되고 이하 나머지 항을 0으로 만들어 버릴 수 있냐는 것과 동치이다. SIR 관계식을 생각하면 가능하다. S, I를 굉장히 빠른 속도로 감소시켜 0으로 만들고 R은 N으로 만들어버리면 된다. 이런 관점에서 그림 4의 결과와 일관적이다.

이 문제를 해결하기 위한 가장 쉬운 방법은 추가적인 데이터를 이용해서 penalty를 주는 방법이 있다 (e.g., S(200)=25000). 하지만 그러한 데이터는 언제나 이용가능한 것이 아니고 방법론 자체가 좋은 아이디어 같지가 않다. 기본적인 시도로서, 여러가지 hyperparameter를 바꾸어 보았으나 효과는 없었다. 생각했던 것 중 가장 오래 고민했던 방법은 전역적으로 gradient 값을 키우는 것이다. 예를 들어 다음과 같은 regularization을 생각했었다.

$$\frac{-1}{M_{rd}} \sum_{j=1}^{M_{rd}} \left\| \frac{dS^{NN}}{dt}(t_j) \right\|$$

$$\frac{1}{M_{rd}} \sum_{j=1}^{M_{rd}} \left\| \frac{dS^{NN}}{dt}(t_j) \right\|^{-1}$$

이 값을 줄이는 것은 각 네트워크의 기울기를 늘리는 것과 같다. 하지만 간과하는 부분이 있는 것인지 구현의 문제였던 것인지 효과적이진 않았다.

관찰 3: Data preprocessing이 어려움을 일부 해소한다. (혹은 NN initialization)

이 문제는 Data scale 혹은 network의 initialization과 관계가 있음을 알아냈다. 각 네트워크를 생성하고 어떠한 훈련도 없이 inference를 진행하면 대부분의 값이 0~2 사이에 존재한다는 것을 알았다. (그 결과 R은 50000에 가까운 값으로 생성된다.) 기본적으로, R은 거의 0이어야하므로 초기화가 매우 좋지 않다는 것을 알 수 있다. 또한, NN이 기본적으로 생성하는 초기화가 추론할 함수의 scale과 맞지 않았다고 생각했고 N=5로 정규화를 데이터에 적용해보았다. (한편, 정규화를 진행하지 않고 네트워크에 다양한 초기화를 방법을 적용해볼 수 있지만 정규화가 더 쉬울 것이라 판단했다.)

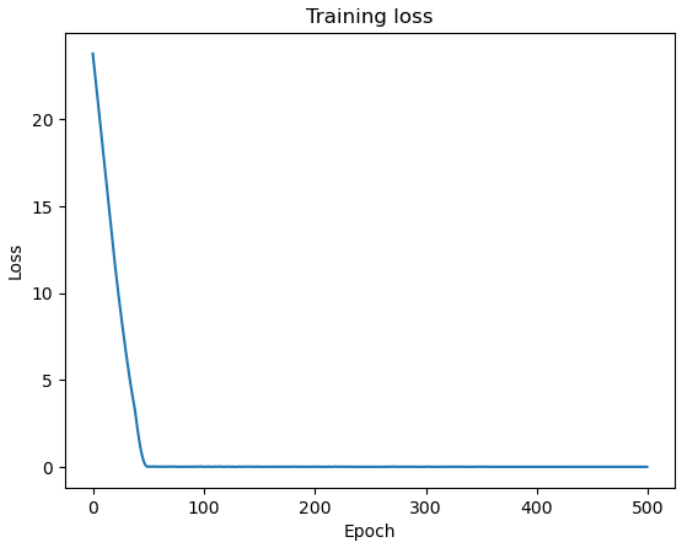

그림 8: The resulting training losses (Normalization)

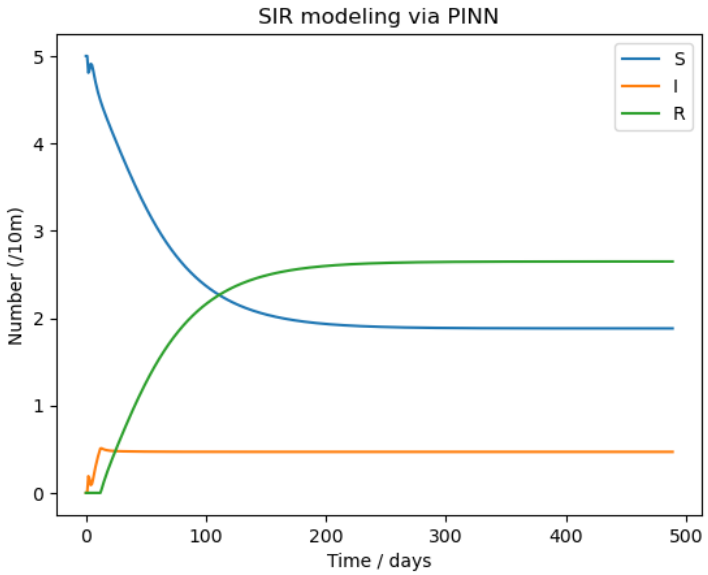

그림 9: The inferred solution to the SIR model (Normalization)

그림 9는 새로운 정규화로부터 얻은 추론 결과이다. 이전 그림 4와 다르게 S와 R의 기울기가 상당히 개선되었다. 더불어 I는 초반에 증가하는 경향을 보였으나 이후에는 느리지만 서서히 감소하는 경향을 보였다.

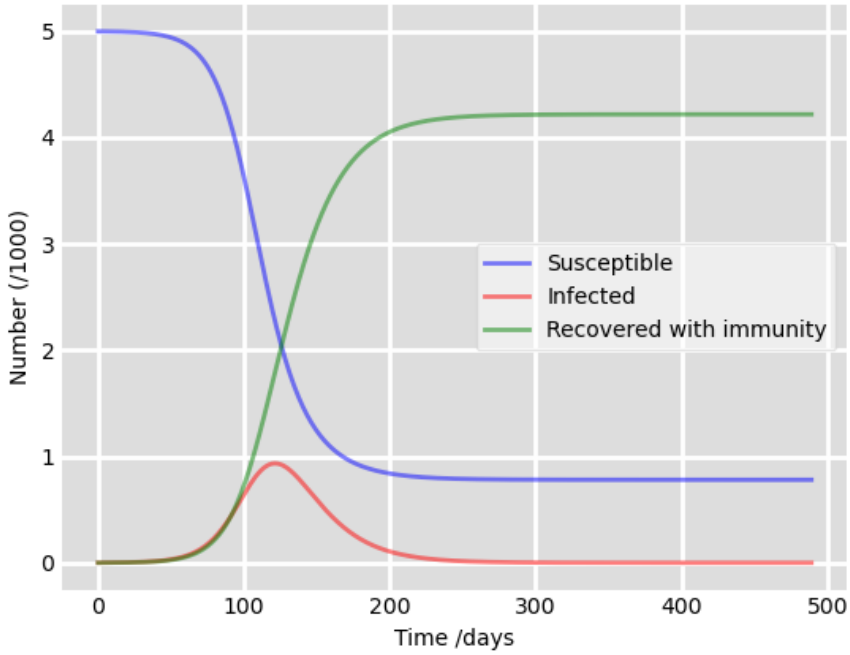

그림 10: The solutions from ODEsovler

그림 9의 결과를 비교하기 위해서 전통적으로 사용했던 ODEsolver를 이용해서 연립 미분방정식을 수치적으로 계산해보았다. 이는 scipy의 odeint라는 class로 쉽게 계산할 수 있으며, 그 결과는 그림 10과 같다. 그림 9와 10을 비교하면 S, R의 경향은 비슷한 것으로 보인다. 그러나 I는 그림 9에서 주어진 시간안에 0 되지 않는 반면 그림 10은 약 300일즈음에 0으로 소멸한다. 이 모델이 어느 정도 좋은 성능을 보인다고 가정할 때 PINN으로부터 만들어진 inference도 부족하지만 잘 근사 된 것으로 보인다.

Conclusion

코로나 19의 확산정도를 분석하기 위해서 SIR 계열의 모델은 흔하게 사용되어왔다. 이러한 연립 미분 방정식을 푸는 것에 PINN을 적용하여 근사 해를 구할 수 있다. 아직 ODEsovler 만큼의 정확도를 가진 모델을 구현하지 못했지만 몇몇 regularization을 추가하여 발전시킨다면 ODEsolver 만큼의 정확도를 유지하면서 neural network 만의 강점을 확보할 수 있는 강력한 모델이 될 것으로 예상된다.

References

[5] Lloyd-Smith, J. O., Schreiber, S. J., Kopp, P. E., & Getz, W. M. (2005). Superspreading and the effect of individual variation on disease emergence. *Nature*, *438*(7066), 355-359.

[6] Chitnis, N. (2017, May). Introduction to SEIR models. In *Workshop on mathematical models of climate variability, environmental change and infectious diseases, Trieste, Italy*.

[7] Christopher Rackauckas, A Comparison Between Differential Equation Solver Suites In MATLAB, R, Julia, Python, C, Mathematica, Maple, and Fortran, The Winnower 6:e153459.98975 (2018). DOI:10.15200/winn.153459.98975

[8] Guidotti, E., Ardia, D., (2020), "COVID-19 Data Hub", Journal of Open Source Software 5(51):2376, doi: 10.21105/joss.02376.

[9] Locatelli I, Trächsel B, Rousson V (2021) Estimating the basic reproduction number for COVID-19 in Western Europe. PLOS ONE 16(3): e0248731.

[10] Raissi, Maziar, Paris Perdikaris, and George E. Karniadakis. "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations." Journal of Computational Physics 378 (2019): 686-707.