Applying Deep Neural Networks To Population Genomic Data For Estimating Parameters In Process Based Evolutionary Models.

Abstract:

Computational models assist scientists in explaining real systems. These models quite frequently possess several parameters which must be 'learned' from data, and this is especially true the sciences. It is also possible to calculate values of these parameters with other traditional techniques, which has implications of slowness, since the computational methods are mostly done by hand, although it requires only fewer steps to get the calculations. Typically, the estimation of these parameters takes a long time but with neural differential equations, we present a new faster way of estimating these parameters with good accuracy. The approach we have adopted employs involved for the purpose of modelling systems of differential equations as well as a neural network useful in the estimation of the parameters generated from the data. Coupled together, this provides a fast method for estimating parameters and identifying the characteristics of a system with many constituent components (*Neural parameter calibration for large-scale multiagent ...*). We validated our method using synthetic data generated from an infection spreading model (SIR model).

Introduction:

These are busy times we are living in. The respective disciplines and the whole range of interdisciplinary fields related to social, economic, biological, epidemiological, etc., investigations can be described and analysed far more effectively as well as accurately with the help of powerful computational tools connected with significant amounts of data. It is the contemporary trend in the mathematical sciences to try to adjust the parameters of mathematical models by feeding in data. This leads to the creation of efficient, theoretically sound, and socially useful means of prediction. For instance, the models that have been used to estimate the transmission rates of diseases are necessary for executing governmental plans during the COVID-19 outbreak as they are refined and retrained with new data periodically. It is the same with the case of crime patterns, pedestrians' movements, social structures, populations, and economies, in computational models.

There are two main modelling approaches:

1. Agent-Based Models (ABMs): These represent time and space characterizing the dynamics of the acts of individual entities (Agents) in society.

2. Differential Equation Models: These are mathematical expressions of the processes occurring in time and space, and their analysis is done with the help of the methods of mathematical statistics.

MLEs for several parameters and data point-based Markov Chain Monte Carlo techniques account for several methods that can be cumbersome to estimate. In recent times, artificial neural networks have come to the foray as a possible solution. These networks, especially the deep ones, have proven their effectiveness in recognizing patterns and making predictions; however, the theoretical framework of these structures is not fully elaborated. Although, in this research we explore more on the theoretical framework of neural networks. Since, eural network does function like human brain, where it learns and modifies results based on this experience. For this research we used different datasets to help the model learn and understand the different patterns of the previous data so that it can use the trend to make a prediction on the future data. However, in many cases, they exhibit better computational qualities than traditional methods.

Our work thus fits into this rapidly expanding literature by investigating an alternative approach of establishing suitable calibration of mathematical models of real-world systems with neural networks. Here we outline straightforward yet highly effective approach of how to incorporate classical numerical models into the training and estimation of model parameters with the help of machine learning. We tested this method on various datasets, including:

- The SIR Model: We therefore estimated parameter densities from the synthetic data of infection spread.

Methods:

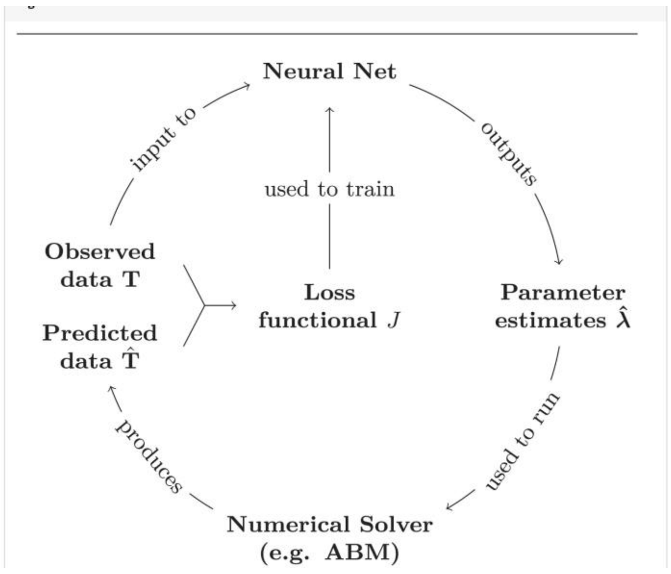
We have a way to calibrate them for systems with the help of the parameters of neural networks. The aim is to identify such parameters for which, given a model, the time series of data is observed. This way, the respective neural network takes as input the time series steps and produces the estimations of the parameters. These parameters are then used to run the model for a particular number of steps the batch size to estimate time series.

This estimated time series is then used for training the parameters of the neural network including the weights and biases through a loss function and the actual time series. A popular form of this loss function is the least squares loss in which the distance metric is the l^2 norm of the difference between the estimated time series and the actual time series.

Given that we have the estimated parameters we then determine how to modify the neural network internal parameters to minimize the loss function. This is achieved with such tools as gradient descent, Nesterov schemes, or the Adam optimizer.

Following this, the real data are passed back to the neural network to generate a new set of parameters, and this forms the other part of the cycle. The full cycle over the specified number of examples is referred to as the epoch. Regarding the technical details of differentiation, there is an auto differentiation scheme that affects the random vectors as constants.

Below is a Diagram of how the neural network functions:



(*Neural parameter calibration for large-scale multiagent ...*)

This method aids in solving an optimization problem but does not give confidence intervals of a prediction. To obtain probability distributions we make use of the fact that as the model learns it samples parameter values and computes a loss for this parameterization of the model. It is useful in estimating a probability density and this loss value will later be used for the purpose.

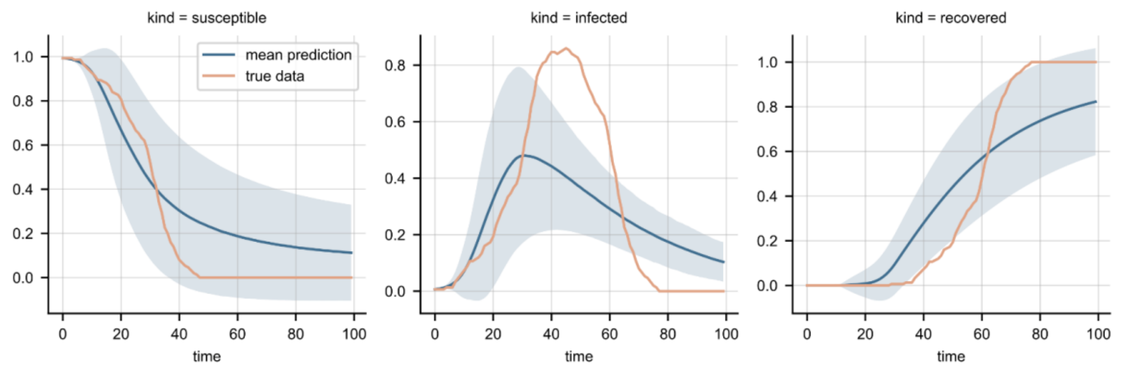
The likelihood of one's estimates given the data is then determined by a formula which includes both the loss and a prior probability. The marginals, or the probabilities of each parameter separately, is calculated by integrating this formula over all the other parameters. The formula is as follows:

$$\rho(\lambda_i) \sim \int \exp\,(-J)\mathrm{d}\widehat{\boldsymbol{\lambda}}_{-i},$$

(*Neural parameter calibration for large-scale multiagent ...*)

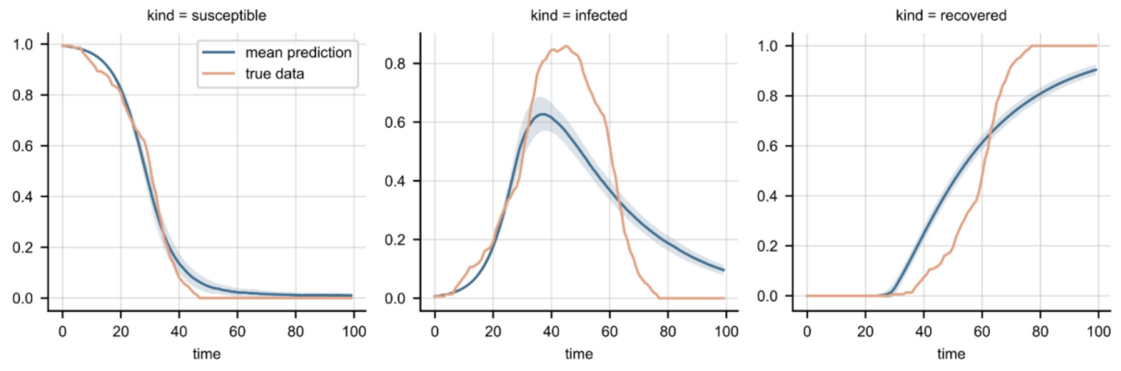
Thus, we begin with the network parameters being uniformly distributed, which enhances the accuracy. Due to nonconvexity that may be exhibited by the optimization landscape, multiple runs of the neural network are performed where the network is initialized with different weights. This increases the number of parameters that result. Although this uses more computation, running multiple training sessions in parallel can increase performance.

SIR model is nothing much a model that can be used to understand the rate at which a certain disease is spreading within a community or population. The many goals for using this model were to understand how neural networks functions and how it can be used to make predictions in this example it would be to estimate the rate of the spread and how long it takes for the individuals to recover from the disease. When running the neural network for the first time for the model SIR it was evident that the model was not well trained for predictions using the true data. This was understood through the graph, which was included in the evaluation file. As for the graph it clearly explained that for the model to predict the outcomes, the training loss was rather high, meaning that the points of predictions are more way off from the real data plots. This figure shows the picture of the graph plotted during the first run of the model for the features selection.



The orange graph represents the true data, and the blue graph represents the predictions made by the model. The blue colour shade represents the training error or the training loss.

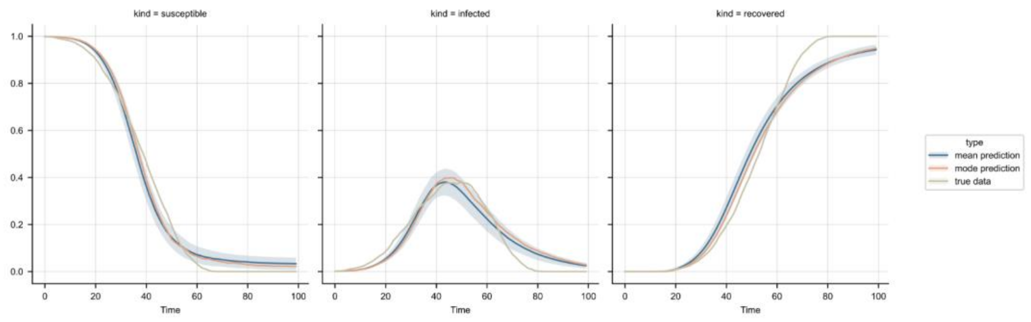
In the second run of the model the focus was on changing the goal or say focus which was set for training the model to lower the training loss or the error so that the proposed model can predict better. To do this we extended the time spent in training of the model while at the same time, increasing the batch size of the model. The batch size is the data volume, here, more data values are fed to the model to train the model enough for the validation. When executing the above-defined model, the training error was observed to have decreased and the model improved than the first time it was trained. The second graph of the model is provided in the screenshot below after the execution of the code for the second time while some changes to the code lines have been made.



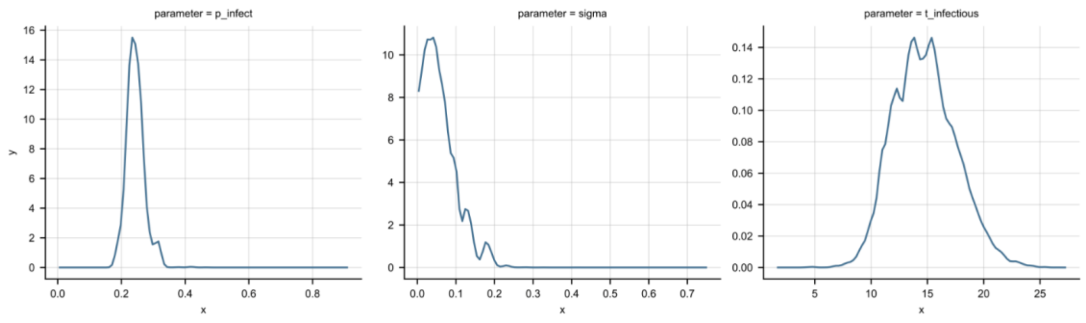
Although the training error has reduced much more than the previous time, we still cannot use the model for prediction, because there are still some huge differences between the graphs. We see this clearly in the second and third graphs. Therefore, we had to rerun the model again, and this time round the main goal was to make the difference between the graphs less. To do this we introduced a random generated value, and seed value. The seed value is value that the neural network model uses as the initialization value with different conditions. In this run the model runs through a range of different parameters with different conditions then the neural network model makes a range of predictions. Switching through different parameters is done through parameter sweeping. Sample example code of the parameter sweeping is shown below:

```
perform_sweep: True
parameter_space:
  seed: !sweep
    default: 1
    range: [60]
```

In the python code above, perform sweep variable is set to “True”, ensuring that the model goes through different parameters with 60 different initialization conditions, before making a final prediction. However, if it is set to false the neural network goes through the default condition and updates the model. Upon the running of the neural network model the graph gets even better, and we believe that this model would be the best one to be used for the prediction. The model is better than the previous ones because when we compare the graph with the previous graphs, we notice that the graph below have a lower training error. The prediction of the model is much closer to the true data. We can clearly see that there is not much deviation in the predictions or outcomes. The model predictions are closer to the true data. Below is a screenshot of the graph displayed from the 3rd run of the model.



Since the model seems to much better than the other versions of the run, we investigated the predicted density plot graph of the parameters to get the range of the parameters to determine how long it takes an individual to recover from the disease, and infection rate at which the disease is spreading within the sample population. From the density plot graph, we see that the infection rate is about 0.21 and the period of the recovery is about 15 days. Shown below is the predicted density plot graph:



In future work, this sample based trained model for SIR analysis will be further developed for prediction analysis on different categories of data like genomics data and COVID-19 data. In case this becomes necessary, we will adjust some of the parameters to retrain the model under certain circumstances so that its reliability and heuristic ability to provide certain kinds of predictions can be further improved. Further, the next model aimed at the prediction of the outcomes utilizing a logistic regression equation is going to be built.

To these ends, the existing code from the Kuramoto model repository stored in the GitHub will be modified and reused. The neural network model that will be designed will be in a way that the certain predictor variables used in the logistic regression equation shall be predicted and these predictions shall then be used in mathematical computations. Subsequently, the above-mentioned model is going to be employed on a variety of datasets to enhance the efficiency and preciseness of the outcomes.

References:

"Compartmental Models in Epidemiology." *Wikipedia*, Wikimedia Foundation, 9 July 2024, en.wikipedia.org/wiki/Compartmental_models_in_epidemiology.

ThGaskin. "Thgaskin/NeuralABM: Neural Parameter Calibration for Multi-Agent Models. Uses Neural Networks to Estimate Marginal Densities on Parameters and Networks." *GitHub,* github.com/ThGaskin/NeuralABM?tab=readme-ov-file#how-to-run-a-model. Accessed 7 Aug. 2024.

Neural Parameter Calibration for Large-Scale Multiagent ..., www.pnas.org/doi/10.1073/pnas.2216415120. Accessed 7 Aug. 2024.