Parameter Inference in Epidemiology with Deep Learning

Violet Y. Zheng, supervised by Dr Alexander Zarebski

2023/24 Maths and Stats Vacation Scholarships Program, University of Melbourne 2nd year

Introduction

Birth-death models (BD) are often used to model the complicated dynamics of infectious diseases. These models contain important information regarding the spread of such diseases such as **birth rate (transmission rate), death rate (recovery rate), infectious period** etc, which are used to inform government response. Existing methods to estimate these parameters, such as Markov chain Monte Carlo and maximum likelihood struggle to accurately predict these parameters for larger and more complex models. Here, we attempt to **reduce the computation times [3]** in parameter inference with a **recursive neural network[6]** of simulated birth-death trees.

Methods

Parameter inferencing on wind speeds

As a demonstration, we begin by estimating the parameters of normal distributions (mean, standard deviation) and Weibull distributions (k, c) for wind speeds[2] with a feedforward neural network (specifically, the DeepSets framework[3]. We followed similar procedures for both models, but we will focus on describing the methods used for Weibull distribution.

- **Preprocess wind speeds data** We used 5 years of hourly wind speeds data from U.S. Department of Energy, and then generated summary statistics [mean, variance, min speed, max speed, standard deviation] as seen in Figure 1[3].

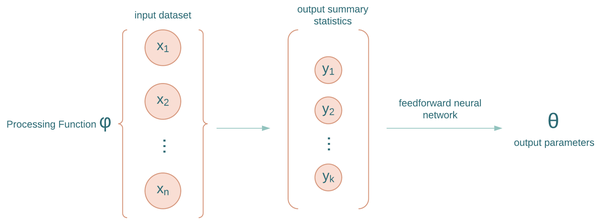

Figure 1. Generating the summary statistics from an input vector of data of length n to length k which is then passed through the network.

- **Prepare training and testing data** Estimate the actual k and c parameters using existing formulas and organise into pairs of summary statistics (which we will use as predictors) and the actual parameters (our target predictions).
- **Creating the neural network** Setup a model with 2 linear hidden layers. Then, setup a MSE (mean squared error) loss function and SGD optimizer.
- **Training the network** Experiment with different epochs, batch size and learning rates, try and minimise the resulting MSE loss.

Parameter inferencing on phylogenetic data

We then followed a similar procedure for estimating the birth and death rates of BD trees.

- **Generate BD trees** Generate 1000 Bio.Phylo.Basetree[5] objects with the Python Dendropy[4] package using uniformly generated death rate between 1 and 2, birth rate between death rate and 4, and a maximum time of 4.

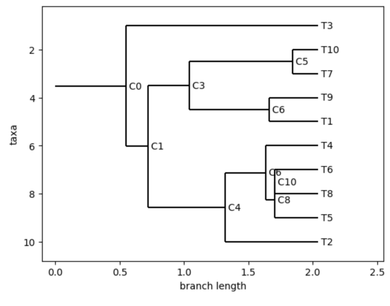

Figure 2. A randomly generated BD tree. Each branch represents a new infection event. Each leaf corresponds to a observed infected individual.

- **Creating the recursive neural network (RNN)** Set up the inner neural network for recursively collapsing all of the nodes' infectious period data into a vector of length 2 summary[1]: each node's summary data initially contains one entry for the branch start time, one for branch end time. The outer neural network is a sequence of layers that generate the final predicted parameters.
- **Training the network** Taking a random tree from the training data at each epoch, use backward propagation to train the network using SGD gradient descent.
- **Validation** The model was then tested with the remaining third of training data which involves a forward pass without any gradient descent nor optimization.

Neural Network Architecture

1. **Inner RNN tree processor [1]** As shown in Figure 3, the inner RNN collapses a BD tree into a vector recursively. For a non-leaf node (e.g. A, C in Figure 3), recursively concatenate the vector representation of its children nodes with its own branch length times into a 6-by-1 vector which is then passed through a linear (6 x 2) layer then a sigmoid activation function. For leaf nodes (e.g. B,D,E), simply return the nodes' branch length times.

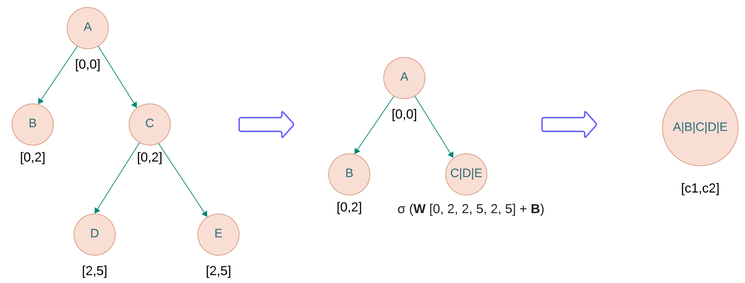

Figure 3. Recursively collapsing a tree's information into one set of summary data [$c1$,$c2$]. $\mathbf{W}$ refers to our weights matrix, $\mathbf{B}$ refers to the bias vector. Together, these make up the parameters of the inner RNN.

The forward method is:

$$f_{\text{tree}}(x) = \sigma(L_{\text{tree}}([x_{\text{summary}}, f_{\text{tree}}(x.left), f_{\text{tree}}(x.right)]))$$

where σ represents the sigmoid function, L represents a linear layer, $x.summary$ is a vector with its branch's start and end times.

2. **Outer RNN** The outer RNN consists of two sets of linear then sigmoid layers. After trialling with different hidden dimensions of the linear layers, we settled on size 10 with a 2-dim input and 2-dim output.
The forward method is:

$$f_{\text{outer}}(x) = \sigma(L_{\text{outer}}(\sigma(L_{\text{outer}}(f_{\text{tree}}(x)))))$$

Results

Parameter inferencing on normal and Weibull distributions

The regression neural network worked well for estimating parameters in normal and Weibull distributions. Both trained with a 0.1 learning rate due the problems' simplicity, the model resulted in MSE loss of 0.1752 for Weibull and 0.00008 for normal distributions. Since these losses are for unscaled data, so relative to their context, these losses are very optimistic.

Valuable knowledge is also gained through constructing these networks, the main being that having sufficient summary statistics directly impacts the final loss. As seen in Figure 4, the model was able to predict the parameters almost perfectly. Of course, this could be improved with higher epochs or more training data.

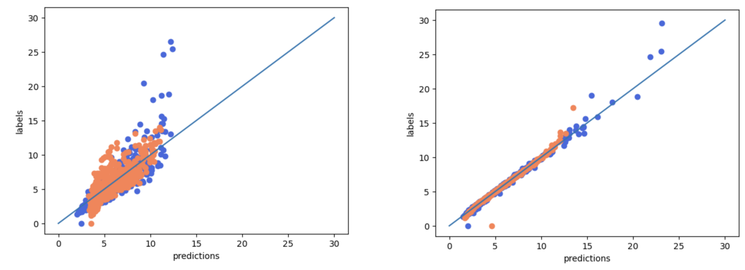

Figure 4. Predicted k (in blue) and c (in pink) parameters vs the "gold standard" estimates of the k and c parameters. As seen on the left, when only the mean and variance were provided as summary statistics, the model had trouble predicting the parameters. However, when the model was given 5 summary stats as described in Methodology, it was able to accurately predict the k parameter.

Parameter inferencing on BD trees

Due to the increased complexity of the BD trees, more time was devoted to tuning the parameters of the model (e.g. learning rate, epochs, hidden layer size). After experimenting with higher learning rates of 0.1 which resulted in the loss diverging to infinite, we settled on a learning rate of 0.001. So far, with 200 epochs, the final MSE loss was 0.1417. Of course, this can likely be improved with higher epochs, but due to the long time taken to train the model, there was insufficient time to conduct further testing. Figure 5 (left) shows the training curve for the MSE of the model across the 500 epochs of training. While the loss has largely converged, it seems plausible that with further training a smaller loss could be achieved. Figure 5 (right) shows the predictions and the true values for the data. There is a strong agreement between the predictions and the true values.

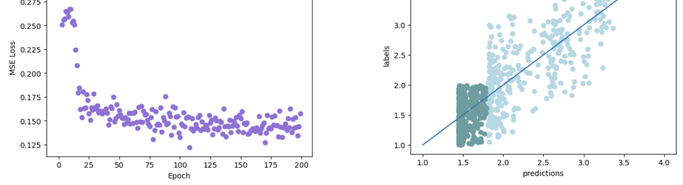

Figure 5. The left figure shows the improvements in loss as epoch number increases. As seen on the right, the plot is roughly around the $y = x$ diagonal, demonstrating that the model is able to give good estimates of the birth (light blue) and death (dark blue) rates.

Future Directions

- Trialing a similar RNN framework to estimate parameters in more complex BD models such as BD model with exposed and infectious diseases or BD with superspreading.
- Extending the model to quantify uncertainties during parameter prediction.
- Further reduce the final MSE loss through experimenting more with epochs, learning rate, different s, and larger scale training data.
- Testing this model on existing trees generated from real world infectious disease outbreaks.

References

[1] Gori M.- Sperduti A. Frasconi, P. A general framework for adaptive processing of data structures. *IEEE transactions on neural networks*, 9(5):768–786, 1998.

[2] T.W. Lambert J.V. Seguro. Modern estimation of the parameters of the weibull wind speed distribution for wind energy analysis. *Journal of Wind Engineering and Industrial Aerodynamics*, 85(1):75–84, 2000.

[3] Zammit-Mangion A. Huser R. Sainsbury-Dale, M. Likelihood-free parameter estimation with neural bayes estimators. *The American Statistician*, 2023.

[4] J. Sukumaran and Mark T. Holder. DendroPy: A python library for phylogenetic computing. *Bioinformics*, 26:1569–1571, 2010. Version 4.6.1.

[5] Caravas J.-Hartmann K. Jensen M. Vos, R. and C. Miller. Bio::Phylo: phyloinformatic analysis using perl. *BMC Bioinformatics*, 12(62):12–63, 2011.

[6] Zhukova A.-Boskova V. Saulnier E. Lemoine F. Moslonka-Lefebvre M. Gascuel O. Voznica, J. Deep learning from phylogenies to uncover the epidemiological dynamics of outbreaks. *Natural communications*, 13(1):3896, 2022.