SimSearch

John M. Aronis and Gregory F. Cooper

1 Introduction

This report describes a method we developed that automatically tunes a simulator model to predict data well. The approach is general and could be applied using any stochastic simulator. As an illustration, we apply the method to learn a simulator model that predicts data from a real COVID-19 outbreak.

In the example, we use data that consists of daily counts of COVID-19 patients who sought care at healthcare facilities during the pandemic. Assume the outbreak proceeds according to a stochastic SEIR model with compartments for *susceptible*, *exposed*, *infectious*, and *recovered*. We wish to estimate the parameters of the underlying population-level model that generated the data. Those parameters include the *initial susceptible*, *initial exposed*, *initial infectious*, *initial recovered*, R_o, *latent period*, *infectious period*, and *start day*. Because the number of affected individuals in the general population is unknown, realistic modeling must be based on the observed number of patients who seek treatment.

A simulator is a generative model of data. Traditionally, simulator models are largely manually developed and either tuned manually or semi-manually to better fit the available, real data. This report introduces a method for automatically searching and tuning simulators. Central to this approach is a method for scoring a simulator S_i given real data D. We use $P(D, S_i) = P(D|S_i)P(S_i)$ as a score, because $P(D, S_i)$ is proportional to $P(S_i|D)$, which is the posterior probability of simulator S_i given real data D. We perform search over the simulator parameters to generate a set of n simulators (models), $S_1, \ldots, S_n$, and derive $P(S_i, D)$ for each $i = 1, \ldots, n$. We are interested in those S_i that have high posterior probabilities, because they are our best approximation of the real world. We can also derive $P(S_i|D)/P(S_j|D)$, which is the relative probability of two simulators, without assuming the set $S_1, \ldots, S_n$ is exhaustive.

The remainder of this report describes the *SimSearch 2.1* algorithm that searches for a model that best explains a set of data. The algorithm is presented using the example introduced above; however, the principles are general and apply to a much wider set of simulators and problems.

2 The SEIR Simulator

We model COVID-19 with SEIR models that divide the population into four compartments:

Susceptible: Individuals who could become infected.

Exposed: Individuals who have the disease but cannot yet transmit it.

Infectious: Individuals who have the disease and can transmit it.

Recovered: Individuals who are immune to contracting the disease.

Because COVID-19 is a novel virus all or nearly all of the population is initially susceptible, few are initially exposed or infectious, and none are initially recovered.

After the entire population has been partitioned into these four compartments, they move through the compartments each day d according to the difference rules:

$$S_{d+1} = S_d - \lambda_d S_d$$

$$E_{d+1} = E_d + \lambda_d S_d - f E_d$$

$$I_{d+1} = I_d + f E_d - r I_d$$

$$R_{d+1} = R_d + r I_d$$

where:

S_0, E_0, I_0, R_0 are the initial number of susceptible, exposed, infectious, and recovered.

$\lambda_d = \beta I_d$ where $\beta = R_0/(ND)$, N is the total population, and D is the duration of infectiousness.

R_0 is the expected number of individuals that an infectious person would infect if he or she were introduced into an entirely susceptible population.

$f = 1/(\textit{Latent Period})$ where *Latent Period* is the average amount of time before an exposed individual becomes infectious.

$r = 1/(Infectious\ Period)$ where *Infectious Period* is the average amount of time an individual is infectious.

We implement these rules in PRAM with one modification: each day the quantity $\lambda_d S_d$ is determined stochastically. That is, rather than select exactly λ_d fraction of S_d to move to E_d, each individual in S_d moves independently to E_d with probability λ_d. As mentioned, a SEIR model is a population-level model that determines the number of individuals in the general population that are susceptible, exposed, infectious, or recovered each day. However, our evidence is from hospitals and other healthcare facilities. We assume that each infectious individual with COVID-19 in the population will independently seek care with probability θ. We say that our SEIR/PRAM models are *doubly stochastic* because both the population-level compartments and the number of hospitalized cases coming from the population are determined probabilistically.

3 Scoring Simulators

Let D be a dataset of real data. In the example, D is a vector in which each element is a count of disease cases on a given day. Let $D(d)$ be the number of cases on day d. If we assume that all simulators are equally likely, we can score a simulator S (defined by a set of parameters as above) using $P(D|S)$. We compute it as follows:

1. Use S to generate a set of simulated outbreaks $\{O_i\}$. Because S is stochastic, these outbreaks will be somewhat different from each other. For each outbreak O_i let $O_i(d)$ be the number of cases on day d as output by simulation i. Let $O(d)$ be a vector that denotes the number of cases of the simulated outbreaks on day d.

2. Over the outbreaks, compute the mean and variance of $\{O(d)\}$ for each day d. Let $\mu(d)$ be the mean number of cases on day d over the set of simulated outbreaks on that day, and let μ be the vector of these means over all days.

3. Compute the covariance matrix Σ among the $O(d)$. This matrix represents how the number of cases on each day are statistically associated with the number of cases on each other day.

The *score* of S is the value of the multivariate Gaussian probability distribution of vector D, which represents the data:

$$P(D|S) = \frac{1}{\sqrt{(2\pi)^m|\Sigma|}} e^{-\frac{1}{2}((D-\mu)^\top \Sigma^{-1}(D-\mu))} \tag{1}$$

where m is the number of days being monitored.

For practical reasons, we simulate about 100 to $1,000$ outbreaks using the simulator S. Because S is a stochastic simulator, it produces different outputs each time it is run. Using fewer outbreaks leads to having too much variance and gives poor results. Using more simulated outbreaks may lead to long runtimes.

4 Simulator Search

In the example, we perform hill-climbing parameter search with restarts within the parameter ranges shown in Table 1. The initial location (state) is randomly and uniformly selected from the parameter ranges. Moves are generated by randomly selecting a dimension and direction to move. The number of initial susceptible is moved $10,000$ up or down, R_0 is moved 0.1 up or down, *latent period* and *infectious period* are each moved one day up or down, and *start day* is moved one day up or down. For each setting of all the parameters, we score the defined simulator model using the method described in Section 3. If after 25 iterations of search none of the simulators has a score greater than the best found simulator so far, the search is restarted at a random location. The search continues indefinitely or until the user stops it.

5 Preliminary Experiment

Ideally, we would like to know the parameters of the outbreak we are trying to model. Thus, in this preliminary experiment, we selected a set of parameters and generated synthetic outbreak data D using a SEIR model. We then searched for a simulator S with the highest score $P(D|S)$.

Parameter	Minimum	Maximum
initial susceptible	100000	900000
R_0	1.0	5.0
latent period	1	5
infectious period	1	5
start day	1	20

Table 1: Parameter ranges for hill-climbing search.

We used the parameters in the *Actual Value* column of Table 2 to generate D. The *Estimated Value* column in Table 2 shows the parameter values of the best state found by the conclusion of the search. (That is, when we simply decided to halt the computer after about six hours.) A dash means we did not search over that dimension.

To improve efficiency, vector μ included the means of the simulated outbreaks at days 10, 20, 30, ..., 100, rather than at every day.

Parameter	Actual Value	Estimated Value
population	1,000,000	-
initial susceptible	500000	525,599
initial infectious	100	-
R_0	3.0	1.8
latent period	3	3
infectious period	3	3
baseline	100	-
startday	10	13
theta	0.1	-

Table 2: Hidden *Nature* parameters and estimated values after search.

To understand the search, we created several *glide paths* through the space. Each glide path starts at a corner of the space with all parameters set to extreme values, then moves toward the true values used to generate a dataset by incrementally adjusting each parameter. Figure 1 shows the scores along several *glide paths* of the search space.

6 Experiment Using Real COVID-19 Data

Table 3 shows the results of running SimSearch on data from New York State for several weeks starting on January 1, 2020. The data consists of daily counts of confirmed COVID-19 cases. The peak number of daily confirmed cases occurred on April 15 (day 105). The table shows the predicted peak day 4, 3, 2, and 1 weeks ahead of the actual peak for three scoring methods:

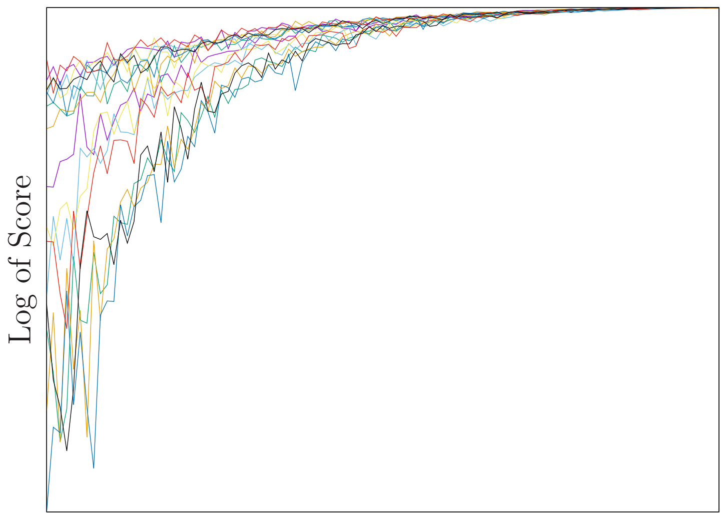

Figure 1: Scores of glide paths through search space where each color denotes a single path.

- *Covariance* is the method described above.
- *Least Squares* uses the sum of the squares of differences between predicted and actual cases as the score.
- *No Covariance* assumes no correlation between the number of cases day-to-day, and thus, assumes independence of the probabilities of the counts on each day.

The upper-left entry "89 (-16)" means that four weeks before the actual peak the *Covariance* method predicted the peak would be on day 89 which is 16 days before the actual peak. The *Least Squares* and *No Covariance* methods used all of the data but the *Covariance* method used only a sample of data (every tenth day) due to computational issues.

	4 weeks	3 weeks	2 weeks	1 week
Covariance	89 (-16)	84 (-21)	73 (-32)	92 (-13)
Least Squares	91 (-14)	92 (-13)	98 (-7)	94 (-11)
No Covariance	96 (-9)	99 (-6)	111 (+6)	115 (+10)

Table 3: SimSearch results 1, 2, 3, and 4 weeks before peak of COVID-19 in New York State.

We note that one reason the peak predictions in Table 3 are not closer to the peak in the data-generating model may be because the number of confirmed COVID-19 cases on each day is a lower bound on the actual number of cases of COVID-19. Indeed, the number of actual cases may be 10 times the number confirmed cases.

7 Conclusions

Often, all that we know about nature comes from the data we collect. Regularities in these data lead us to believe that nature generally works in ways that we can describe mathematically. If so, one way to understand nature is to build simulators: if the simulators mimic the data then that provides support that they model how nature works. We have designed, implemented, and demonstrated a system—SimSearch—that infers simulators from data. We search for simulators that produce data with the same set of correlations between data points that are seen in the data from nature. The simulators that are inferred can be used for a variety of purposes, including explanation and forecasting.

In the work described here, we used a covariance matrix to summarize essential aspects of a simulators behavior. To evaluate how well a given simulator (with particular settings of parameter values) predicts real data, we predict the probability of that data given a covariance matrix learned from the simulator.

The process of approximating a simulator's behavior (e.g., a SEIR model) with a simpler model (e.g., a covariance matrix), as we have done, puts our work in the realm of emulator research.[1] A promising application of emulators is to perform highly efficient parameter search. To do so, we include in the simpler model (the emulator) a set of variables that represent the parameters of the simulator, as well as include the domain variables. In our example above, we would include in the covariance matrix a set of variables representing the SEIR parameters of be tuned, as well as domain variables representing the counts of cases on each day. We then learn the covariance relationships among all these variables from data generated by multiple runs of the simulator. Finally, we adjust (tune) the parameters in that covariance matrix (the emulator) to predict the real data D as well as possible. Parameter tuning using an emulator is generally much more efficient than tuning a simulator directly. We have done some preliminary work along these lines, and our results show that we need to model non-linear relationships between the parameters and the domain variables. Such non-linear relationships are not captured by a covariance matrix. Recent results by other researchers have shown that deep neural networks can work well in developing non-linear emulators that produce up to a billion-fold increase in parameter-search efficiency.[2] We plan to explore this direction in future research.

[1]Hutson M. *AI shortcuts speed up simulations by billions of times*, Science, 14 February 2020, Vol. 367, Issue 6479, pp. 728, DOI: 10.1126/science.367.6479.728.

[2]Kasim MF, Watson-Parris D, Deaconu L, Oliver S, Hatfield P, Froula DH, Gregori G, Jarvis M, Khatiwala S, Korenaga J, Topp-Mugglestone J. *Up to two billion times acceleration of scientific simulations with deep neural architecture search*, arXiv preprint arXiv:2001.08055. 2020 Jan 17.