CSE 8803 EPI: Data Science for Epidemiology, Fall 2022

Lecturer: B. Aditya Prakash — September 22, 2022
Scribe: Andy Chea, Swethasree Bhattaram, Ritu Sinha — Lecture # : 9 Inference II

1 Summary

As data collection will never be a perfect process, we continue to discuss ways that missing data can be inferred from what data we do have. In the first half of the class, we wrapped up the Inference I powerpoint by discussing the GT Wifi Mobility Project that was used during the pandemic to track spread. Then we discussed several possible ways to calibrate agent-based models in general.

Then we moved onto the Inference II powerpoint and discussed two main topic: how to find Patient Zero and how to infer missing infections. In the former, several methods are given to inituitively track down the center of an infection graph. Then in the latter, we discussed 4 methods in detail for finding the infections that have slipped through the crack. Using these methods in tandem, we can observe a more complete view of an epidemiological history and better understand how to contain current and future epidemics.

2 Lecture 8 Wrapup

We can consider the GT Wifi Mobility Study. Localized wifi connection points around campus were used to build mobility networks, and those were used with time series of GT postive test rates and Fulton county infection case rates to develop a model.

2.1 Parameters

They constructed a dynamic network using SEIR progression. Recall that this progression includes a noninfectious Exposed phase with its own probability of transition to Infected. As an agent-based model, the researchers used the mobility data to predict how people and the disease would move.
Parameters to estimate included, number of initial asymptotic cases, transmission probability β, and a scaling factor to determine spread due to cases outside of the Wifi network. Maximum Likelihood Estimation cannot be done since the likelihood is intractable.

To calibrate, first set up the objective function:

$$f(I_0, \alpha, \rho) = \sqrt{\frac{1}{W}[\sum_{w=1}^{W}(\frac{\sum_{i=1}^{N} S(I_0, \alpha, \rho)}{N} - R_w)^2]} \tag{1}$$

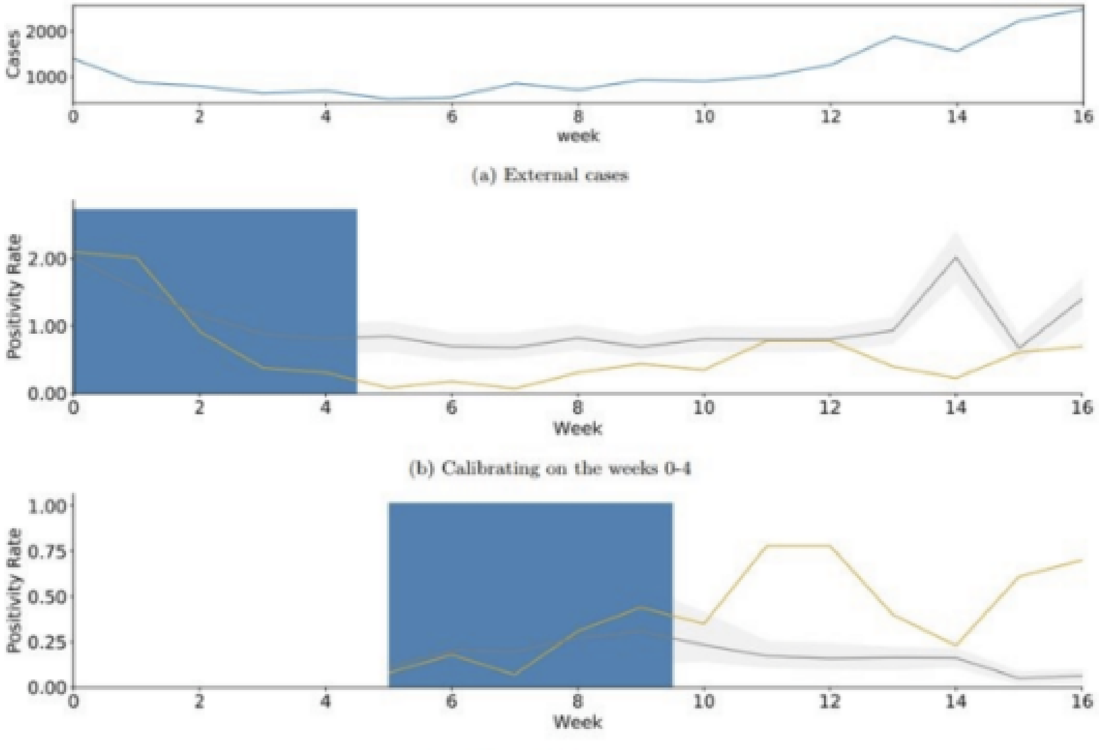
where $S = newasymptomaticinweekW$, $R =$surveillance testing aggregated result, and a key assumption is that each population gets tested at the same rate.

Then to optimize, find the arguments that minimize the function:

$$argminf((I_0, \alpha, \rho))$$

Lastly, to train and validate the set, they trained the model on the first five weeks of data and validated using the remaining weeks.

Generally, this task can also be done through attempting different calibration time frames (use weeks 0-4 vs weeks 5-9) or through evaluating the model's effectiveness on other, similar datasets.



(a) External cases

(b) Calibrating on the weeks 0-4

(c) Calibrating on the weeks 5-9

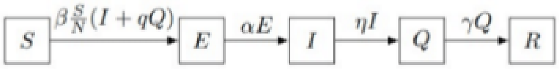
2.2 ABM Calibration

The Nelder Mead method of minimization is gradient-free, so gradient descent isn't used. Rather, 40 different parameter sets sampled from within 40% of the minimum root mean square error are taken and their means and standard deviations are calculated.

In general, it is computationally difficult to calibrate agent-based models. Other methods include

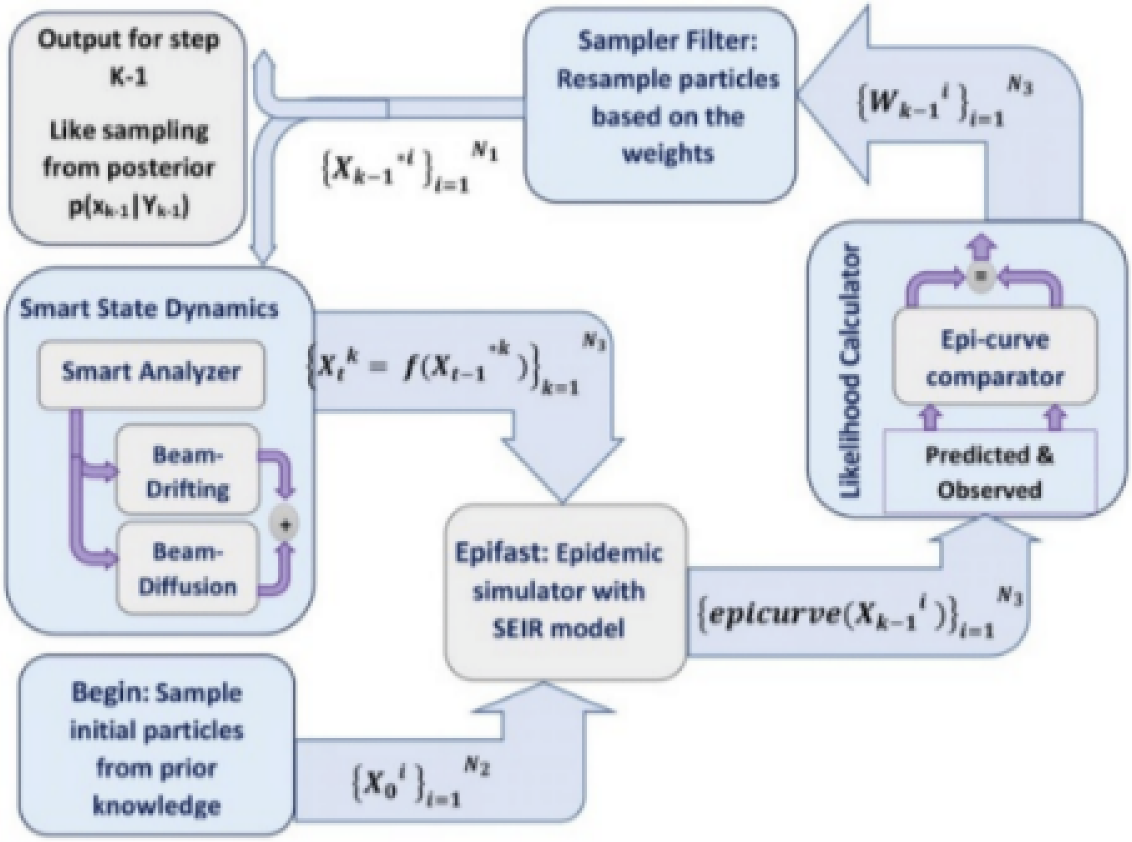
- Grid Search: Search ranges for each parameter and find where in the n-dimensional space RMSE is minimized. The multiple points are sampled evenly across the space.

- ODE-based: Observe time series data through a simple Ordinary Differential Equation model and transfer found parameters to ABM; computationally fast



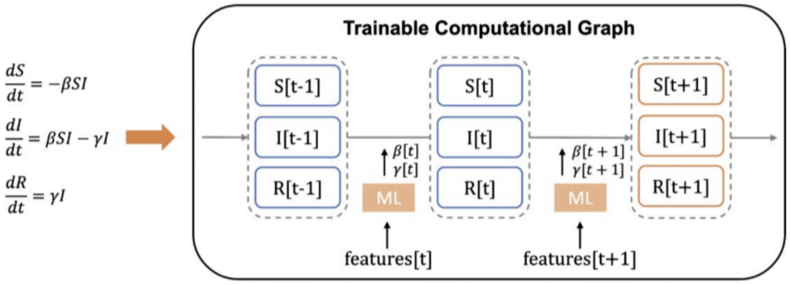
$$\theta,(r,\omega)\sim p(\theta,\phi);\quad R_{0:T}=M(\theta)$$
$$\hat{d}_{1:T}=\omega\left(R_{1:T}-R_{0:T-1}\right);\quad d_{1:T}\sim \text{NegBinomial}\left(r,\frac{\hat{d}_{1:T}}{\hat{d}_{1:T}+r}\right)$$

- Beam search particle filtering: Used to explore search space while avoiding local optima traps. Then we use particle filtering for the process of data assimilation where we sample and resample particles based on prior knowledge and learned weights. It is computationally slow. Can be used in models without gradients, where gradient descent isn't possible.


Output for step K-1
Like sampling from posterior $p(x_{k-1}|Y_{k-1})$
Sampler Filter: Resample particles based on the weights
$\{W_{k-1}{}^{i}\}_{i=1}^{N_3}$
$\{X_{k-1}{}^{*i}\}_{i=1}^{N_1}$
Smart State Dynamics
Smart Analyzer
Beam-Drifting
+
Beam-Diffusion
$\{X_t{}^{k} = f(X_{t-1}{}^{*k})\}_{k=1}^{N_3}$
Likelihood Calculator
=
Epi-curve comparator
Predicted & Observed
Epifast: Epidemic simulator with SEIR model
$\{epicurve(X_{k-1}{}^{i})\}_{i=1}^{N_3}$
Begin: Sample initial particles from prior knowledge
$\{X_0{}^{i}\}_{i=1}^{N_2}$

- Digital Source Seeding: Use another available database to seed the agent-based model. An example would be learning the initial conditions of the metapopulation model GLEAM by learning from tweets that are categorized by location.

- End-to-end learning with ODE/ABMs: As a more recent concept, they use differentiable modules that take advantage of gradient-based optimization. Setup involves continually adding modules between two polar opposites that are differentiable at either end: i.e. a cat vs dog prediction.



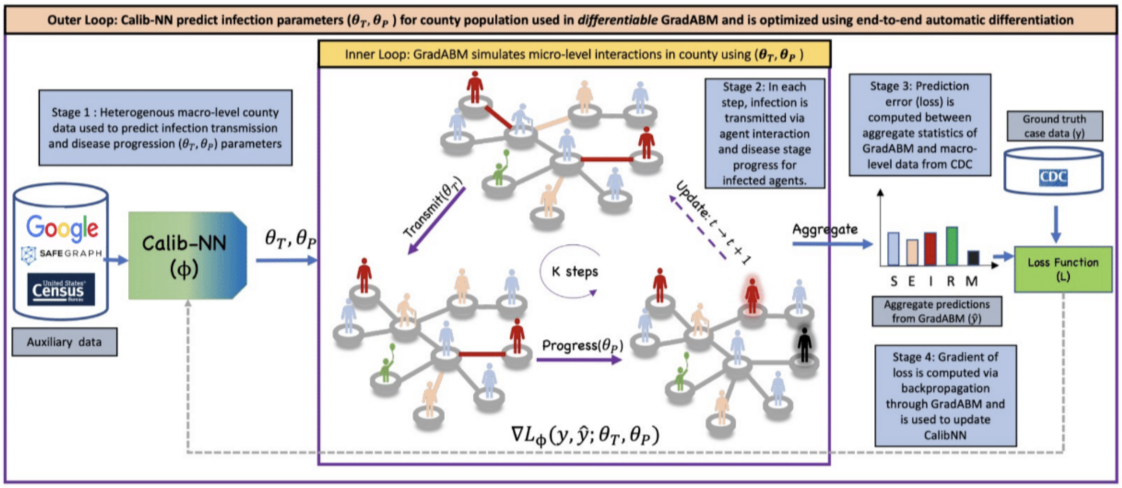
Rate variable	Covariates
β	Mobility, Interventions, Density, Past Counts
η	Census, Healthcare supply
γ	Census, Test count / pos. ratio, Past Counts
h, c, v, ϱ, κ	Census, Econometrics, Healthcare supply

$$v_i[t] = v_{i,L} + (v_{i,U} - v_{i,L}) \cdot \sigma\left(c + b_i + \mathbf{w}^\top \text{cov}(v_i, t)\right)$$

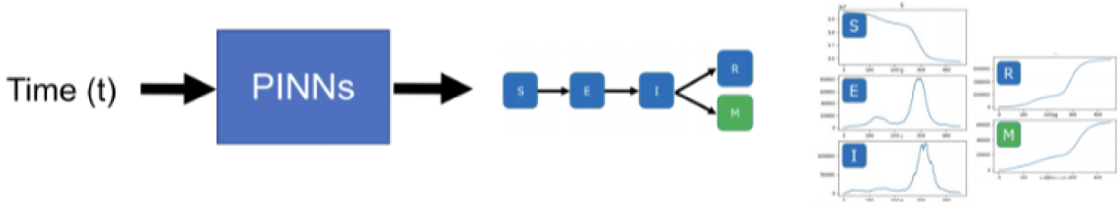
Update parameters via gradient-based optimization (RMSProp)

Stage 1: Param. prediction
Encoder-decoder GRU (Calib-NN)

Stage 2: Disease transmission + progression
Message passing in graph neural network (GNN) + reparametrization trick


Outer Loop: Calib-NN predict infection parameters (θ_T, θ_P) for county population used in *differentiable* GradABM and is optimized using end-to-end automatic differentiation
Inner Loop: GradABM simulates micro-level interactions in county using (θ_T, θ_P)
Stage 1 : Heterogenous macro-level county data used to predict infection transmission and disease progression (θ_T, θ_P) parameters
Google
SAFEGRAPH
United States Census Bureau
Auxiliary data
Calib-NN (ϕ)
θ_T, θ_P
Transmit(θ_T)
K steps
Progress(θ_P)
Update: $t \to t+1$
$\nabla L_\phi(y, \hat{y}; \theta_T, \theta_P)$
Stage 2: In each step, infection is transmitted via agent interaction and disease stage progress for infected agents.
Stage 3: Prediction error (loss) is computed between aggregate statistics of GradABM and macro-level data from CDC
Ground truth case data (y)
CDC
Aggregate
S E I R M
Aggregate predictions from GradABM ($\hat{y}$)
Loss Function (L)
Stage 4: Gradient of loss is computed via backpropagation through GradABM and is used to update CalibNN

- Physics Informed Neural Networks: They take time as input and match ODE gradients to learn the latent dynamics before outputting the SEIR fractions at each time step. They can be difficult to teach, even with a lot of data. They also require epidemiological knowledge such as time and SEIR transition probabilities to be trained and later discover the parameters. Lastly, they can be quite abstract, since they can be left with new data and figure out how to incorporate it.



- Other methods: Bayesian calibration, iterative methods, rejection ABC, Markov

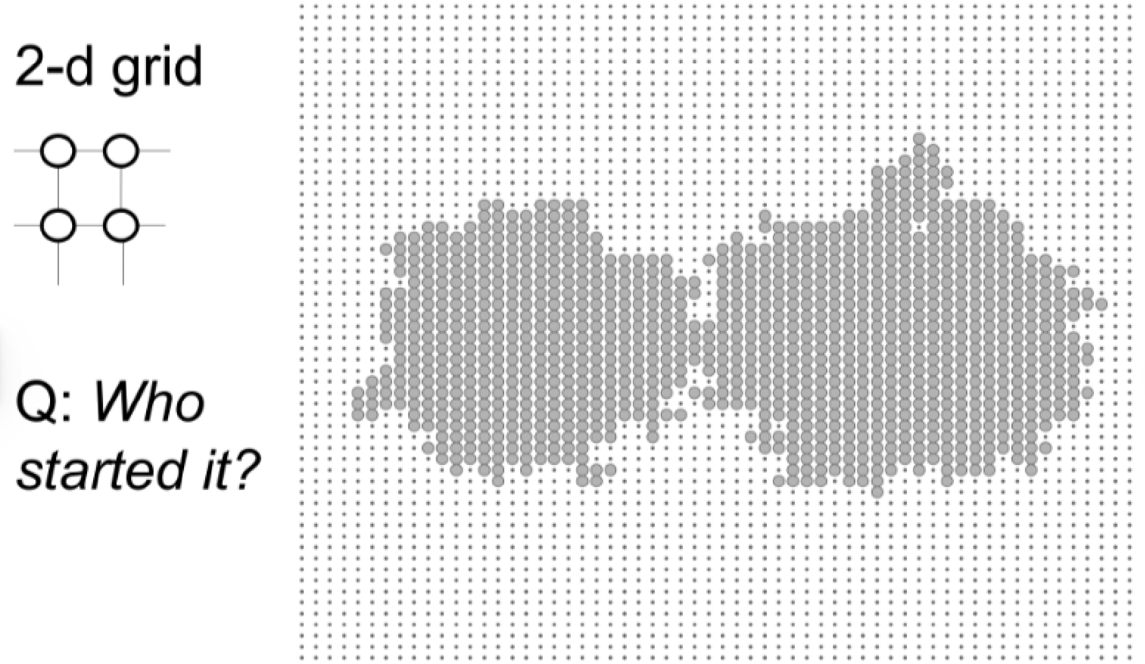
Chain Monte Carlo

Other methods: bayesian calibration, iterative methods, rejection ABC, Markov chain Monte Carlo

3 Finding Patient Zero

Data collection is not perfect and will miss some cases. In some real-world disease analyses, such as with Tuberculosis (CDC, 2007) or AIDS, the source was able to be predicted through reconstructing the likely transmission path to the first case, Patient Zero.

Humans can pick up on trends, such as figuring out from past experience where the likely origin of these two clusters are:



The best source is the one that maximizes the data likelihood in the SI model.

$$\hat{v} = argmaxP(G_N|v* = v)$$

where G_N is the infected subgraph and v is an observed node in G_N

For any given infection subgraph, there can be a multitude of starting positions. This means that calculating the probability of a given cascade is stochastic and non-trivial.

3.1 Rumor Centrality

This process sums up the likelihood of all ripples across different time snapshots of a network.

$$P(G_N|v* = v) = \sum_{allripples} R_i$$

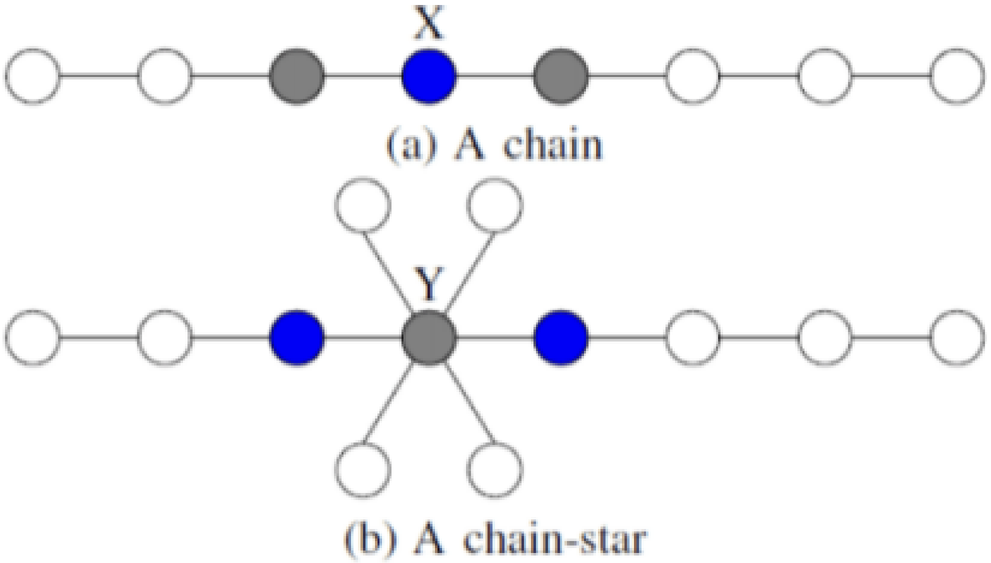
These are difficult to compute: P-hard! However, probabilities of different cascades can be computed efficiently for k-regular trees assuming it's an SI model.

To extend for general graphs, extract a tree from the graph such as a Minimum spanning tree that covers all nodes in the graph or use breadth-first search to generate a subgraph.

3.2 Finding Number and Identity of Sources

When infected nodes are surrounded by a lot of uninfected nodes, this reduces the chance that the former is the originator. This is called exoneration. Since the original infected has the longest time frame to infect neighbors, one that still has uninfected neighbors is less likely to be the first.

In the below example, Y is much less likely to be the source due to its uninfected edges.



(a) A chain

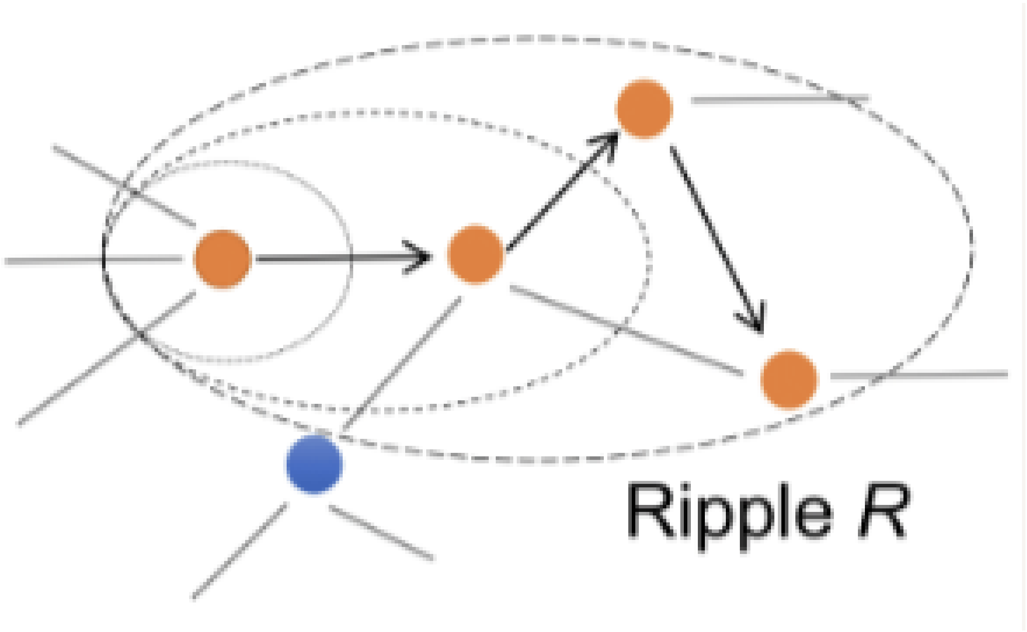
(b) A chain-star

We arrive at a two part solution: use minimum description length for various numbers of seeds, and for that seed number, calculate exoneration as the sum of centrality and penalty. The first part, MDL uses bit compression to figure out which source is the simplest, and therefore according to Occam's Razor, the most likely explanation of the spread. The running time of this method is linear and can be completed in NetSleuth. Seedset Scoring: $L(S) = L_N(|S|) + log\binom{N}{|S|}$ where —S— is the encoding integer and the log term is the number of possible —S—-sized sets. Big O Runtime: $O(k * (E_1 + E_F + V_I))$

The method to optimize the score is first we identify the high-quality node set given k, and then given the nodes we must optimize the ripple R. The ripple cost of each additional one added is:

$$L(R|S) = L_N(T) + \sum_{t}^{T} L(F^t)$$

where $L_N(T)$ represents how long a ripple is and the summed term describes how the "frontier" advances.



Overall, the total MDL cost is

$$L(G_I, S, R) = L(S) + L(R|S)$$

Some extensions that have been published include:

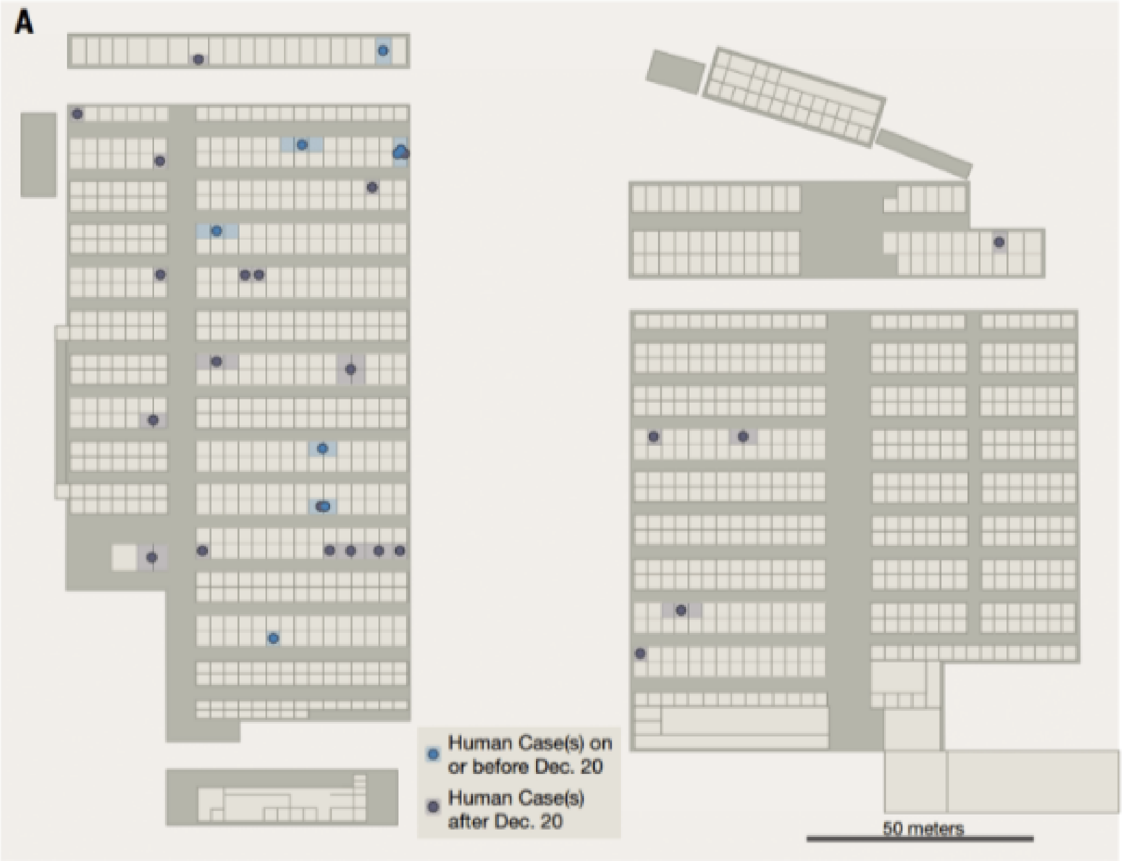
- Different models
- Temporal networks
- More rigorous results on specific graphs
- Noisy input
- Using graph neural networks

A final method to detect patient zero is using graph neural networks. We can use a deep generative model to estimate the distribution of diffusion sources. We can use the graph neural networks to approximate the forward direction of the epidemic and discover the diffusion probabilities. It can both learn the model parameters by iterating forward and guess the source through iterating in reverse.

3.3 Reconstructing the COVID-19 Pandemic Epicenter

How was Hunan Seafood Wholesale Market in Wuhan guessed as COVID ground zero? Researchers used spatial relative risk analysis done by compiling data from many sources: Weibo cases, CCDC sequencing PCR report, population density data, animal sales records, and mobility data.

Investigators were able to pinpoint the source to vendors selling live mammals.


A
Human Case(s) on or before Dec. 20
Human Case(s) after Dec. 20
50 meters

4 Finding Missing Infections

A large number of COVID-19 infections went and still are continuing to go unreported. When there was only 23 reported infections in 5 major American cities in March 1st, there may have been greater than 28,000 in actuality.

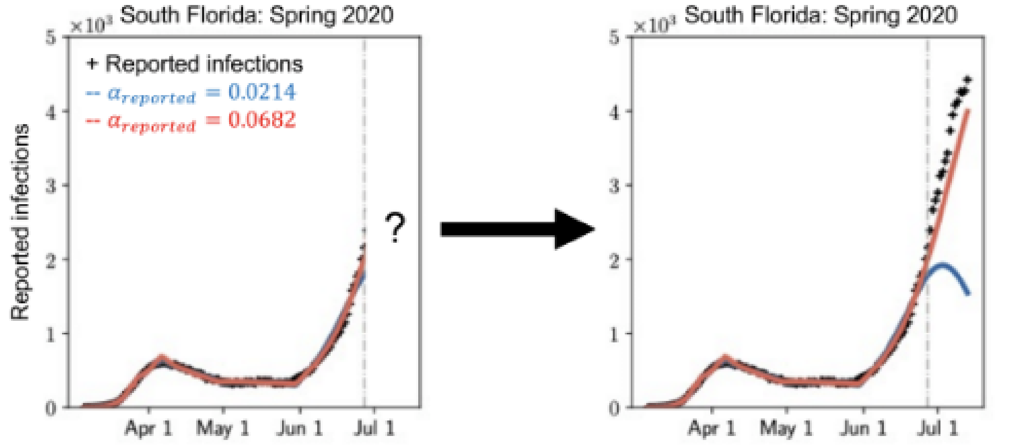
The difficulty in estimating the unreported infections allowed it to spread more quickly in the US and worldwide, and when severity is slowed so too is the remedial action that follows.

To identify the true reported rate, consider a serological methodology. This is considered the gold standard for disease reporting, It tests if the person's body has the required antibodies for the disease, indicating that they contracted the disease. Hence, it can give a very accurate picture of the people who contracted the infection. However, this is an expensive method and is also a delayed process and we might not obtain the information for analysis in time.

4.1 Approach 1: Real-World Calibration

Calibrate an epidemiological model using reported data and infer missing infections. Then fit I_r to reported infections to estimate the unknown parameters such as the rate of reporting α. However, they suffer greatly from ad-hoc modeling assumptions. This means altering α

slightly can drastically alter forecasts.



Seems similar here

But much more difference for forecast!

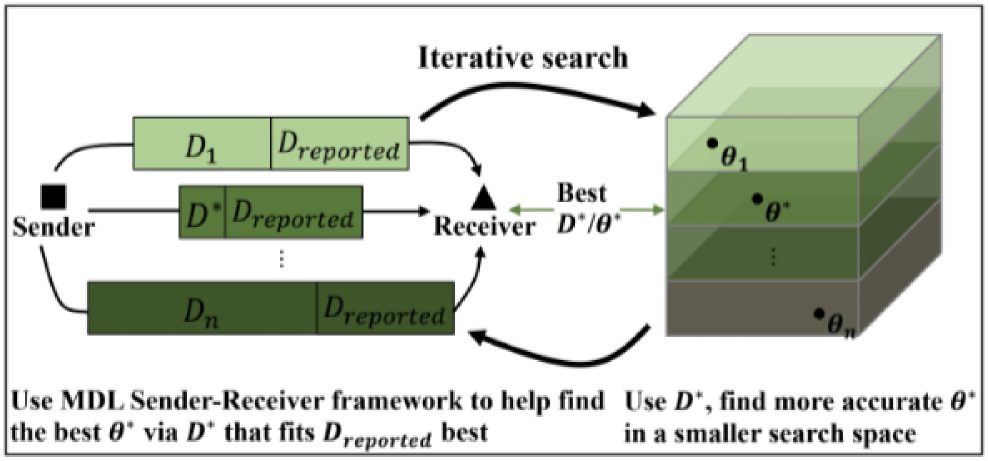
4.2 Approach 2: MDLInfer

In this approach the number of reported infections is known: $D_reported$. If we are then given the correct count of total infections D, the model can be calibrated with both to find a better fit of $D_reported$. Put simply, we want to find the $D*$ that fits the $D_reported$ curve the best.

- Problem formulation:

$$\boxed{D^* = \underset{D}{\operatorname{argmin}}\, L\left(D_{reported} \middle| D\right) + L(D)}$$

- Here, L(·) denotes the number of bits for encoding.



Use MDL Sender-Receiver framework to help find the best θ^* via D^* that fits $D_{reported}$ best

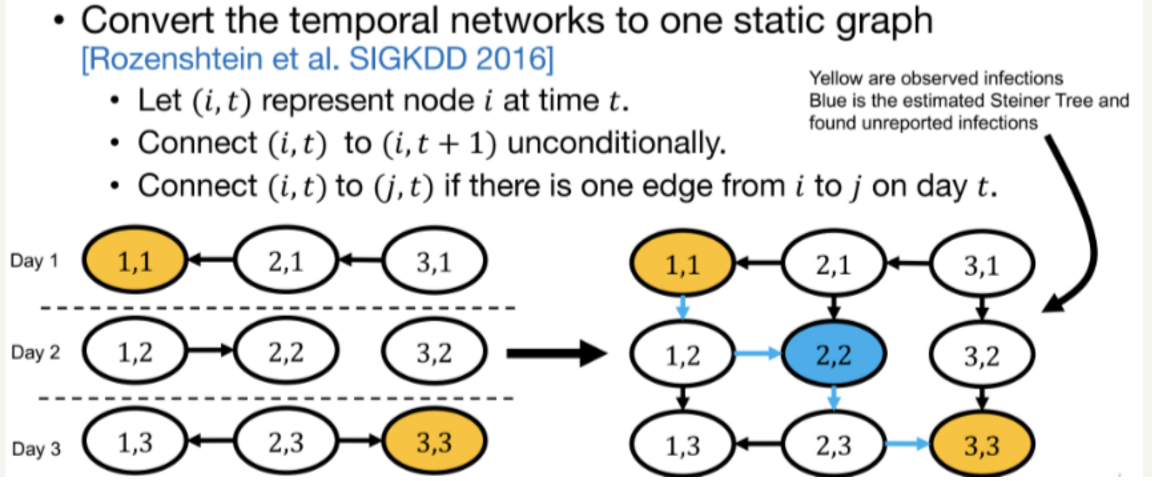
Use D^*, find more accurate θ^* in a smaller search space

MDLInfer tends to better match the total infected curve than other approaches.

4.3 Approach 3: Steiner Trees

This is a datamining method that learns the tree with minimum weight in the graph that connects all of the given terminals. In our case, the terminals would be known cases and the minimum tree built to include each of those would implicate many other nodes that may be the unreported infections. These are determined with dynamic solvers.

The Steiner tree method can be extended to temporal networks by converting it to one static graph. It can also be used to learn node weights from features, thus allowing us to estimate high-danger patients, for instance.



- Convert the temporal networks to one static graph [Rozenshtein et al. SIGKDD 2016]
  - Let (i, t) represent node i at time t.
  - Connect (i, t) to $(i, t+1)$ unconditionally.
  - Connect (i, t) to (j, t) if there is one edge from i to j on day t.

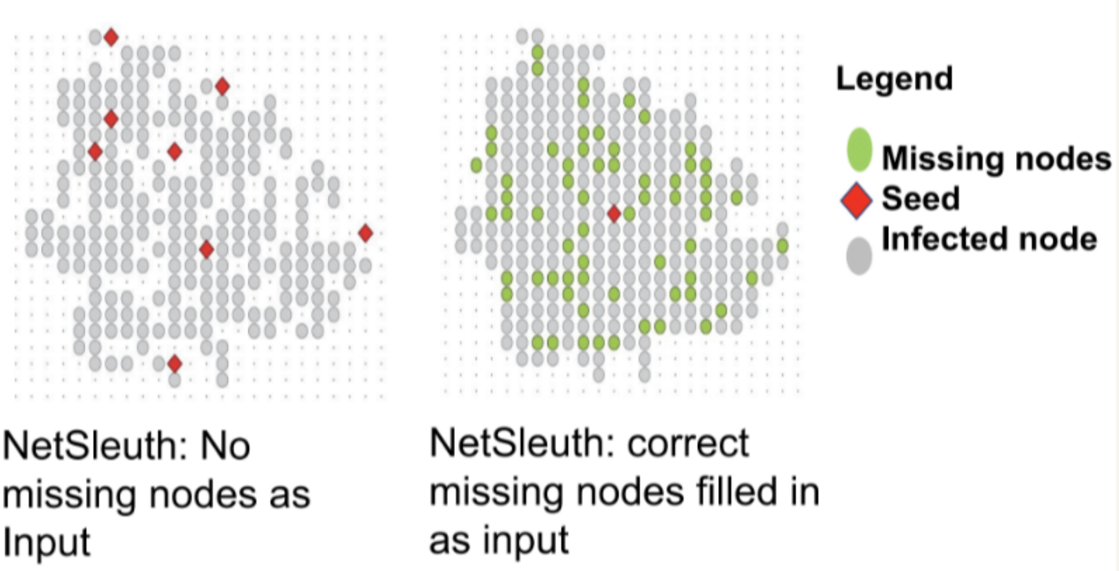
Yellow are observed infections
Blue is the estimated Steiner Tree and found unreported infections

4.4 Approach 4: Netfill

Steps

1. Find starting points given missing nodes
2. Find missing nodes given starting points
3. Iterate above steps until convergence

The two ideas directly feed into one another and can be done in NetSleuth.



NetSleuth: No missing nodes as Input

NetSleuth: correct missing nodes filled in as input

4.5 Other Approaches

Researchers can also consider graphical approaches or using contact tracing. Privacy must be maintained in the latter case.