writeup of ABC project

March 31, 2022

1 brief summary

$Hereisthepaperourworkisbasedon.$ Basically what we are attempting to do is the following:

we have some networks in which hate speech/misinfo have spread. G_T^* is the network G^*, the real network(s) we're studying, at timestamp T.

we have some function $F(G)$ that produces a vector of network characteristics ϕ.

our vector of "correct" network characteristics comes from a real network (or many real networks in aggregate) and is denoted as ϕ^*.

we have a markovian data generating process that occurs on randomly generated networks of actors. within each experiment, the network model will be the same (e.g. Erdos-Reyni, configuration model, hyperbolic graphs, etc).

during the data generating process $H(\theta, T)$, the network evolves; individual actors may change characteristics, share information, or influence other actors. after T timesteps, the simulation is considered to have concluded, and we have a graph G_T.

what is θ? θ is a parameter vector that governs the distributions from which the actors' characteristics and spreading parameters are drawn. for a toy example, let's say that our actors' susceptibility to misinfo μ is drawn from a beta distribution μ $\beta(s_1, s_2)$ and the likelihood of spreading misinfo from one actor to another is s_3. $\theta = [s_1, s_2, s_3]$.

so we have a parameter vector θ_i. we then run our data generation process (network simulation) $H(\theta_i, T)$ for T steps, producing a graph G_i. Next, we plug G_i into $F(G)$ to give us ϕ_i, our vector of network characteristics.

in order to tell how correct/incorrect our simulation was (compared to G_T^*), we compare ϕ_i and ϕ^* using a "goodness of fit" function / acceptance probability function $Z(\phi_i, \phi^*)$.

better θ_i vectors will lead to better values of ϕ_i; our end goal is to get a distribution over θ_i vectors. we aren't trying to solve for θ_i; instead, we'd like to know how it's distributed and how much uncertainty we're dealing with.

2 algorithm

here's a brief description of the algorithm in the paper, as we understand it.

first, we create a prior on θ. for simplicity's sake (the paper covers this case), we make a uniform prior on θ while constraining the values to reasonable ranges. for our purposes, we are primarily dealing with beta distributions, so we have a bunch of integer ranges from 1 to 10 that we draw from. future versions of this simulation will also try to infer the Markov steps for agent characteristic updates during the simulation.

next, we draw **ensemble P** from the prior distribution. **ensemble P** is a bunch of vectors θ_i^P that aren't trying to do well. their job is to give us a sense of what the distribution of the badness function $Z(\phi_i, \phi^*)$ is. if **ensemble P** is large enough, it should give us a good sense of how the badness function values are distributed. we smooth this distribution as well before continuing.

after that, we create **ensemble E**. the items in **ensemble E** are "good" compared to the "background" distribution given by the θ_i^P values in **ensemble P**. this can take a while because we reject a lot of samples, but is not computationally expensive compared to the rest of the simulation.

we also create a cooling rate parameter ϵ_t that varies as time goes on. this makes us less likely to make risky jumps in the distribution / make risky decisions when we are further into the process.

now the fun part: we choose a random parameter θ_i^E in **ensemble E**. then we generate another θ_{new}^E. we compare the results of $Z(\phi_i^E, \phi^*)$ and $Z(\phi_{new}^E, \phi^*)$ using a "temperature" function τ. $\tau(Z_1, Z_2)$ gives us the probability of keeping θ_{new}^E. then we do a uniform draw and decide whether or not we're keeping θ_{new}^E. if so, θ_{new}^E becomes the new i^{th} member of **ensemble E**.

we do a bunch of iterations of this step, stopping when the rolling acceptance probability of keeping a newly generated sample goes below a certain threshold.

when we finish our algorithm, **ensemble E** should give us a sense of how θ^*, the "correct" parameters to produce ϕ^*, is distributed. for example, here are some heatmaps over the beta distribution parameters for a couple of the agent characteristic distributions.