Approaches to Optimizing Medical Treatment Policy using Temporal Causal Model-Based Simulation

Robert Horton
Microsoft
rhorton@microsoft.com

Maryam Tavakoli Hosseinabadi
Microsoft
matavako@microsoft.com

John-Mark Agosta
Microsoft
john-mark.agosta@microsoft.com

Abstract

It is notoriously difficult to draw conclusions about the effects of medical interventions from observational data, where statistical confounding is rampant. An important example is "confounding by severity" Salas et al. [1999] in which sicker patients receive more aggressive intervention, leading to a misleading positive correlation between stronger intervention and worsening outcome. This scenario is quite generally applicable because it represents negative feedback control, where some control mechanism responds to a change by affecting the change in the opposite direction. This leads to a causal loop: the change affects the feedback and the feedback affects the change. We employ the classic approach to breaking such loops by unrolling them in time, so that the disease severity before treatment is a separate node from the severity after treatment. Unrolling produces a dataset where the information about a patient is no longer contained on a single row of a dataframe, but is spread over a set of rows representing timeslices. We want to base treatment decisions on the final outcome, which is only found at the end of this set of rows. Since we are interested in outcomes that occur at a future timeslice, we borrow a term from reinforcement learning and describe our type of intervention as a "policy". Our challenge is to properly integrate temporal modeling with causal modeling on observational data so that we can deconstruct these causal loops and reach useful analytical conclusions. Here we demonstrate a suitable analytical approach with a simple toy problem, a drug dosing policy to treat the disorder arising from infection with the fictitious pathogen *Bogovirus*. We begin by writing a simple bespoke simulation program to match a given causal graph; this generates a simulated dataset where we know the ground-truth about causal interactions. Using the known correct influence graph, together with other aspects of "domain knowledge", we build causal model-based simulations of the simulated data ("simsim" models) that let us estimate the expected effects of various treatment policies on ultimate outcomes. We compare this approach to the closely-relate field of reinforcement learning, and show how they are complementary.

1 Introduction

1.1 Why apply causal models to synthetic data?

This paper demonstrates how temporal causal models can be used as the basis of simulations that make it possible to optimize treatment policies in virtual experiments. We show the approach on data

Preprint. Under review.

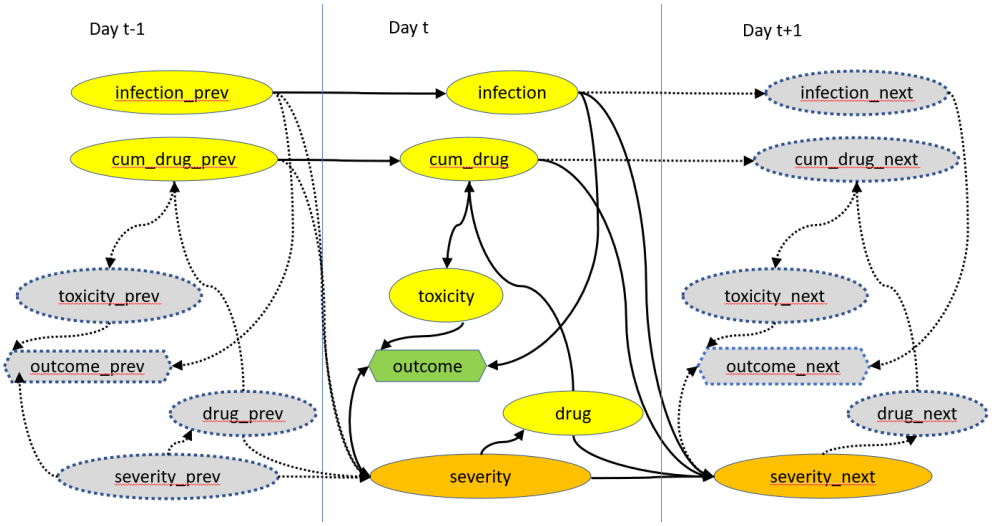


Figure 1: Unrolled causal model with sequential time steps. States explicitly included in the model are shown with solid outlines; dotted outlines and arrows show states and transitions that are captured in other timesteps.

that has itself been generated by a simulation, so that we know the 'ground truth' causal relationships. Our ultimate goal is to develop better treatments by applying this approach to historical patient data.

2 BogoVirus: A Simulated Dataset

2.1 Components of a medical simulation

A common problem in causal modeling of medical disorders is the presence of feedback loops. A drug is given to manage a disorder, and the disorder responds to the drug. The traditional way to break such causal influence cycles is to unroll them in time; the severity of the disorder in the current timestep is the basis for administering the drug, and the response to the drug is reflected in the severity of the disorder in the next timestep.

Our model includes an underlying condition (infection with the imaginary organism *Bogovirus*); this condition will resolve over time as long as the patient does not die. Infection leads to a disorder (it could be something like a problem with blood clotting, breathing, or circulation) that must be managed because if it gets too severe it can lead to death. The disorder is managed by a drug which has a direct effect on the severity of the disorder, but becomes less effective as the level of the drug accumulates in the body. High cumulative levels of the drug can lead to fatal toxicity. The ultimate outcome is either recovery or death, but on most days neither of these things happens, and the patient just stays in the hospital for another day.

These are the variables in the simulation: **infection**: records how far the patient has passed through the course of the infection (this is basically a counter for percent progress; once it passes 100 and you have not died, you recover). **drug**: a dose of the treatment. This is an adjustable quantity. **cum_drug**: The accumulated dose of the treatment drug. This is an exponential average, and is subject to decay over time if treatment doses are not administered. **severity**: Quantified level of severity of the disorder caused by the infection. The worse this gets, the more likely the patient is to die. **outcome**: if severity gets high enough, the patient's chances of death increase. If the infection runs its course and the patient does not die, they recover. **toxicity**: a function of cum_drug that is much more pronounced at high cumulative dose. High toxicity leads to increased probability of death. Variables with suffixes '_prev' or '_next' hold lagging or leading values from adjacent timeslices.

Through trial and error, we adjusted the simulation parameters so that a simple scan of dose (see Figure 3) will show an optimum, but the peak performance is less than 100% survival, so it still leaves some room for improvement.

patient_id	cohort	day_number	infection_prev	severity	cum_drug_prev	infection	drug	cum_drug	severity_next	toxicity	outcome
4456	8	5	71	38.671956	0.512000	80	0.8	0.627200	40.706310	0.060875	none
4456	8	6	80	40.706310	0.627200	87	0.0	0.376320	53.476941	0.002840	none
4456	8	7	87	53.476941	0.376320	95	0.8	0.545792	57.973954	0.026434	none
4456	8	8	95	57.973954	0.545792	103	0.8	0.647475	64.359519	0.073678	recover

Figure 2: Simulated data for a single patient.

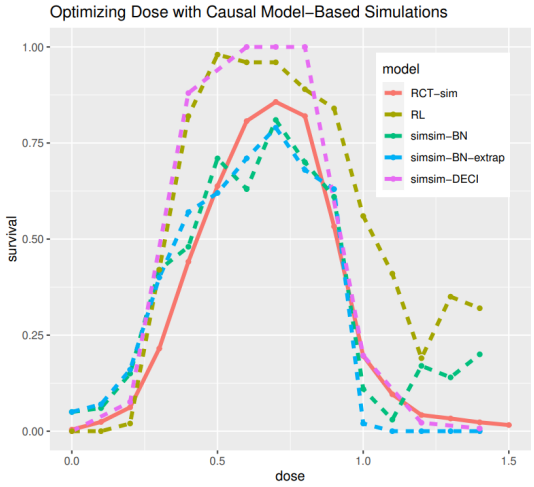


Figure 3: Finding the optimal dose to maximize survival. The solid red line shows the results of running a simulated Randomized Controlled Trial (RCT) in the primary simulation described in section 2; the dashed lines show the results obtained in analytical 'simsim' simulations, as described in subsequent sections. The disorder is almost uniformly fatal if untreated, drug toxicity is fatal at high cumulative doses, and there is a point where the optimal dose gives the best response. From this scan, the best outcome achieved is at a dose of 0.7 units, giving 87.1% survival.

2.2 The simulated dataset

Figure 2 shows simulated data for a single patient episode. There is one column for each node in the model, plus housekeeping attributes `patient_id`, and `day_number` (how many days that patient has been in the hospital). The `cohort` column indicates the group into which the patient was randomized in the simulated RCT; this determines the dose of drug the patient receives.

2.3 Finding the Optimal Dose

Figure 3 shows a scan of different doses of drug run in the primary simulation. This is a simulated experiment, where patients are assigned to cohorts, and each cohort receives a fixed dose of the drug. The y-axis shows the fraction of each cohort surviving the Bogovirus infection. The optimal dose (0.7 units) gives us a standard to compare with results we obtain from analytical simulations.

To simulate observational data, we use the primary simulator to generate another dataset (not the cohort-dose dataset used in Figure 3), in which a random dose of drug is given to each patient each day. This randomized policy dataset is input to the subsequent modelling tasks.

3 "SimSim-BN": An Analytical Simulation Built on a Causal BayesNet Model

A Bayes network whose parameters are learned from the randomized dosage dataset simulates building a model from observational data. This uses a different simulated dataset from that used to generate the optimal policy in Figure 3: For that run—shown by the solid line—a fixed dose was assigned to each patient, as in a randomized controlled trial. We use this to find the dosage policy that gives the best results by which we can compare the Bayes network model simulation, along with the "SimSim-DECI" model and the Offline Reinforcement model, as shown in Figure 3.

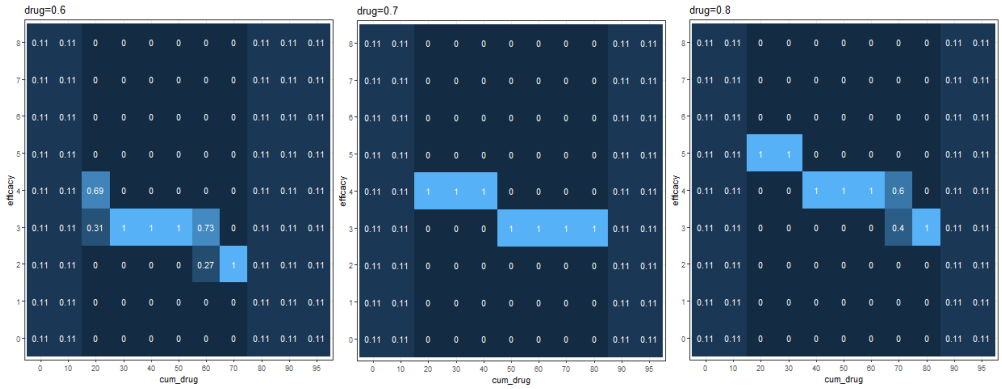


Figure 4: A portion of the Conditional Probability Table for efficacy by dose and cumulative dose. This CPT is a three-dimensional table where each cell holds a probability, and the coordinates of the cells are the categorical values of the inputs (`drug` and `cum_drug`) and the output (`efficacy`). This node is not shown in Figure 1; it was added to the causal graph to make it easier to fit the model, as explained in the text.

3.1 Modifying the graph to facilitate training

In construction of the Bayes network, multiple high-cardinality inputs to a node lead to a curse of dimensionality problem. In some cases we can apply domain knowledge to break these complicated nodes into a set of smaller, more easily learned nodes.

The astute reader may notice that the influence diagram in Figure 1 does not contain a node for `efficacy`; it has four values, `drug`, `cum_drug`, `infection` and `severity` flowing directly into `severity_next`. Though we were able implement the primary simulation using all four inputs in a single node, having this many inputs into a single node in the Bayes Net led to sparsity problems. If each variable is discretized into N levels, 4 inputs require N^4 cells in the CPT. So we once again drew upon our simulated domain knowledge; because `drug` and `cum_drug` interact with each other, but not directly with `severity` or `infection`, we were able to add a node to capture this interaction.

Figure 4 shows the CPT for `efficacy` for three selected discrete values of `drug`. We can see that these tables have captured the relationship that the efficacy of a given dose of the drug decreases at higher levels of `cum_drug`. However, the very high or very low values of `cum_drug` (the extremes of the x-axis in each panel) show a uniform probability across the levels of `efficacy`. This is because these values of `cum_drug` are not represented in the primary simulated data from which this Bayes Net was trained, and it defaults to using a uniform prior probability in these ranges. We therefore to advantage of our domain knowledge to modify the CPTs to extrapolate better to extreme values. This was done by simply filling out the extreme columns with copies of the closest non-uniform column.

We also simplified the input `infection`, which is basically a counter that keeps track of how far the patient has progressed through the course of infection, into the binary variable `infection_over` that is true if infection reaches 100 or more, and false otherwise. With these modifications, we feed only three inputs (`severity`, `efficacy`, and a binary variable for `infection_over`) into `severity_next`. This greatly simplifies the probability table.

3.2 Using the causal model to re-simulate the data

Rather than testing a treatment policy in the simulation (analogous to an experiment), we can use a causal model to build a simulation (since this is a simulation of a simulation we call this a 'simsim' model, of which we present three versions). We then use this causal simulation to perform the optimization scan. The simsim model is trained on a simulated dataset where each patient was given a random dose of the drug each day (as opposed to the dose-optimizing simulation where patients were assigned to cohorts that each received the same dose each day). This simulates the process of building a causal model from non-experimental data and using it to estimate an optimal treatment policy without actually running an experiment.

patient_id	day_number	severity	infection	drug	cum_drug	toxicity	die	recover	prev1_severity	prev1_infection	prev1_drug	prev1_cum_drug	prev1_toxicity	prev1_die	prev1_recover
3911	6	54.607584	55	0.8	0.479744	0.012192	0	0	47.696857	53	0.2	0.266240	0.000356	0.0	0.0
1326	10	46.535000	75	1.0	0.857256	0.396885	0	0	50.113805	73	1.4	0.762094	0.195908	0.0	0.0
451	0	23.000000	28	0.0	0.000000	0.000000	0	0	NaN	20	0.0	0.000000	0.000000	0.0	0.0

Figure 5: Dataset is re-framed to capture temporal dynamics on each row.

4 "SimSim-DECI": Expert-in-the-loop Causal Inference for Intervention Optimization

Deep End-to-end Causal Inference (DECI) Geffner et al. [2022] is an autoregressive-flow based non-linear additive noise structural equation model (SEM), which is designed to perform both causal discovery and inference, including average treatment effect (ATE) and conditional average treatment effect (CATE) estimation. DECI discovery takes prior knowledge of the graph structure as input, defined by a constraint matrix, and uses the data-driven causal calculation to estimate the graph structure, and ATE between nodes.

4.1 DECI for Timeseries

SimSim-DECI uses the DECI model to simulate temporal records for the patients under various drug-dose policies and finds the policy for the optimum dosage level. As DECI is primarily designed to digest generic types of tabular Dataset. In order to capture the temporal dynamics of our data, we designed our framework as follows:

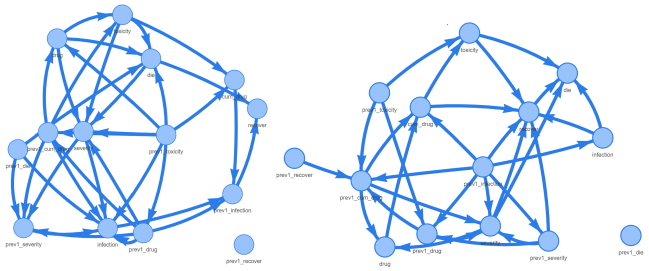
- *Temporal Sliding-Window featurization*: each row contains the information of the current time-step and the previous one as in (X_{t-1}, X_t). Figure 5 shows what re-framed dataset looks like.
- *Temporal constraints*: Features in the current steps cannot cause features in the previous step.
- *Temporal training*: The DECI model is trained on each row as an independent sample. Since each row holds the past and current features, the connection between features of the same day is learned as well as features of two consecutive days.
- *Temporal simulation*: We simulate the information of each step by conditioning the simulation on the information from the previous step. That means for data in Figure 5, we take the values of the seven right columns as the condition for the intervention and predict the next step by sampling the trained model.

4.2 Causal Discovery

DECI recovers the structure of the underlying causal graph that we can compare with the true underlying graph. Thus, we can be the expert-in-the-loop to modify the data-driven findings to improve its correspondence to the truth. We do this by re-training with additional constraints. Figure 6 shows the discovered unconstrained graph compared to the final graph with expert-in-the-loop constraints applied. Note that the domain expert prevented *confounding by indication*Kyriacou and Lewis [2016] by reversing the direction of drug-to-severity causality.

4.3 Simulation-based Policy Optimization

The trained model is a structural causal model (SCM) that captures the functional relations and error distributions. Hence, it is capable of estimating the expected updates following an intervention. We use this capability to synthesize the current day, having information about the previous day. We add the desired drug dose based on the candidate policies, explained in previous sections, to the interventions. To make reduce estimation error, we take average 50 samples for the continuous variables and take mod for the binary ones. The simulation for each patient will stop if “die” or “recove” returns True. Figure 3 demonstrates the results of different dose policies. The model captures the optimum drug dose correctly, although is less accurate at extremes where the observational data is sparse.



(a) No constraint (b) Expert-defined Constraints

Figure 6: Discovered graph by DECI with no constraint (left) vs. with prior constraints defined by Expert including temporal (right).

5 Offline Reinforcement Learning: The Third Simulation Reconstruction

For purposes of comparison we took the same randomized policy simulated observation dataset and used a comparable offline reinforcement learning (RL) method to derive a constant-dose policy comparable to the previous simulations. Since we used a constant policy, the method is degenerate, since the policy function does not depend on the current state, and might equivalently be considered just a simple Markov process.

For the RL environment, we wrapped the simulation dataset with `OpenAI Gym` [Brockman et al. [2016]], to simulate the stage transition probability distribution. Unlike the previous "sim-sim" reconstructions, state variables are continuous-valued, so to generate a random state value given the current state, we used a k-nearest neighbor regressor built on the `faiss` [Johnson et al. [2019]] library for similarity search. Since, even with the simulation dataset of $1M$ rows, there is no likelihood that a lookup will find a record within some epsilon of the current state, so instead we average the predicted next-stage values of the $k = 3$ nearest neighbours. Equivalently the prediction function could have been implemented as a neural-network trained on the simulation dataset. Either way, predictions are confined to the space that has been explored in the simulation data, with all the limitations this implies for an offline-algorithm.

We ran a constant policy in dose increments of 0.1 from 0 to 1.4, for 100 episodes, which in almost all cases terminated in less than 20 stages, then averaged the termination counts to estimate survival rates, as shown in Figure 3. Admittedly using a constant policy is a trivial algorithm, and cannot do better than an adaptive policy. But for purposes of comparison with the previous "sim-sims", it illustrates the consequences of the lack causal analysis. As the figure shows, the offline-RL finds the same optimum dose, but over-estimates the survival rates at extremes. Most notably it believes high doses that should accumulate toxic cumulative effects are not deleterious. Inspection of predictions as revealed in the episode traces show that, at extreme values, the offline predictor is wildly off with predicting accumulation of the drug over time. This highlights a known limitation of offline policy optimization, where even in modest dimension state-spaces the ability to explore new policy areas is severely limited by the existing data. Levine et al. [2020]

6 Discussion: Where do we go from here?

We plan to collaborate with medical researchers in our next iteration of this simulation and modeling process to focus on an actual disease and therapies, to reproduce the analytical logic in a more realistic setting. We have experience implementing a similar forward-inference Bayes Net in the Synthea electronic medical record simulator [HHS2021]; porting our next version to that platform would let us develop shareable feature engineering exercises to conduct the kind of simulation-based analysis described here on data in a realistic schema.

Causal graphs provide an opportunity to capture domain expertise; this is a modeling process that, as we have shown, blends gracefully into simulation modeling.

References

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

Tomas Geffner, Javier Antoran, Adam Foster, Wenbo Gong, Chao Ma, Emre Kiciman, Amit Sharma, Angus Lamb, Martin Kukla, Nick Pawlowski, et al. Deep end-to-end causal inference. *arXiv preprint arXiv:2202.02195*, 2022.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2019.

Demetrios N Kyriacou and Roger J Lewis. Confounding by indication in clinical research. *Jama*, 316(17):1818–1819, 2016.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Maribel Salas, Albert Hotman, and Bruno H Stricker. Confounding by indication: an example of variation in the use of epidemiologic terminology. *American journal of epidemiology*, 149(11): 981–983, 1999.