Extracting Clinician’s Goals by What-if Interpretable Modeling

Chun-Hao Chang1,3, George Alexandru Adam1,3, Rich Caruana4, Anna Goldenberg1,3
1University of Toronto, 2Vector Institute, 3Hospital of Sickkids, 4Microsoft Research

Abstract

Although reinforcement learning (RL) has tremendous success in many fields, applying RL to real-world settings such as healthcare is challenging when the reward is hard to specify and no exploration is allowed. In this work, we focus on recovering clinicians’ rewards in treating patients. We incorporate the what-if reasoning to explain clinician’s actions based on future outcomes. We use generalized additive models (GAMs) - a class of accurate, interpretable models - to recover the reward. In both simulation and a real-world hospital dataset, we show our model outperforms baselines. Finally, our model’s explanations match several clinical guidelines when treating patients while we found the previously-used linear model often contradicts them.

1 INTRODUCTION

Reinforcement learning has achieved tremendous success in many fields including Go [28], autonomous driving [26], and data center cooling [1]. However, designing reward functions for real world problems is challenging since it is not clear how much more preferable some states are than others. For example, clinicians often administer vasopressors in order to increase blood pressure, but a too-high dose might cause vasopressor-induced shock. When robots are designed to navigate to a specific location, the reward function has to prefer not to break nearby items or hurt the people along the path [3]. Specifying all possible conditions in the reward is very hard, and also the magnitude of the reward is not easy to determine when multiple goals are needed (e.g. treating patients while reducing the side effects).

One way to avoid reward function tuning is to do imitation learning that directly mimics what experts do by their demonstrations. However, we only extract the rule of how experts make decisions (e.g. administer vasopressors when blood pressure is low) but not the reason why they do (e.g. maintain patient’s blood pressure above 65). Therefore, the rules extracted are not suitable for transferring when environments change or different actions are available, while goals recovered from inverse reinforcement learning (IRL) are more robust and allow the user to inspect and confirm if these are intended consequences.

In many settings such as medicine, users often behave based on what-if future outcomes: given the current information, what desirable outcomes would happen if I take certain actions? For example, doctors treat patients with vasopressors to increase their blood pressure or administer cancer drugs to reduce tumor volume. Compared to traditional IRL that is parameterized using the current states and actions, our approach of modeling expert goals based on what-if future outcomes by interpretable models allows us to rationalize expert behavior in terms of counterfactuals as a user would do. Importantly, the learned preference is transferable across different environments when actions are different (e.g. different hospitals may have different treatment protocols). Finally, it is of interest to clinicians to understand if their behavior matches the intended goals, and helps serve as a sanity check tool when designing the reward. Besides the clinical setting, it is also useful for discovering goals for agents who do not have the ability to explain themselves (e.g. animals).

Generalized Additive Models (GAMs) have been in popular use since the 80s serving as important tools to understand dataset patterns in many fields including healthcare, business and science [7]. GAMs are also used to audit black-box models [30] or discover fairness bias in the data [31]. As a white-box model by design it is surprisingly effective compared to black-box counterparts such as deep neural networks for tabular data. However, to the best of our knowledge it has not been used in IRL to understand experts’ goals.

In this work, we combine what-if reasoning i.e. coun-
Extracting Clinician’s Goals by What-if Interpretable Modeling

Figure 1: The overview of our work. We first learn a model that predicts the future counterfactuals. Then we model the reward by GAM and counterfactuals and interpret the GAM model by its shape graphs.

2 Related Work

We summarize prior works in Table 1.

Briefly, the most similar work to ours is CIRL [4] which also uses what-if future outcomes to model rewards. However, the linear assumption in their work is too restrictive for many real-world problems including healthcare, where the goal usually is to maintain patients' vitals within normal ranges (e.g. temperature between 36-38). Linear models cannot assign high reward to a medium range of values, i.e. they learn that either extremely low or high values correspond to high reward making the recovered reward unrealistic. We use GAMS to avoid this problem. Also, CIRL adopts the apprenticeship learning framework that has been known to have no unique solution and might recover trivial solutions (e.g. learned weights are 0). Instead, we use Adversarial Imitation Reinforcement Learning framework (AIRL), which has unique solution and has achieved state-of-the-art performance [9].

iAIRL [29] also aims to recover clinician’s reward and uses an interpretable differential decision tree (DNDT) [32]. Due to the exponential feature combinations of DNDT, iAIRL only modeled 5 features, and their performance is much lower compared to using deep neural network (64% v.s. 71%) [29]. Using GAMS results in similar state-of-the-art performance while achieving interpretability. Besides, they did not consider what-if reasoning as we do.

3 Background

Markov Decision Process (MDP) We adopt the standard notations of MDP. An MDP consists of a tuple $(S, A, T, T_0, R, \gamma)$ where $s \in S$ states, $a \in A$ actions (discrete, in this work), $T(s'|s,a)$ the transition probabilities, and $T_0$ the initial state distribution, $R(s,a)$ the reward function, and $\gamma \in (0,1)$ the discount factor. A policy $\pi(a|s)$ gives the probability of taking an action $a$ in a state $s$. An optimal policy $\pi^*$ maximizes the cumulative reward $G$:

$$G_\pi = \sum_{t=0}^{T} E_{s_{t+1} \sim T(s_t,a_t),a_t \sim \pi(s_t)}[\gamma^t r(s_t,a_t)]$$

$$\pi^* = \arg\max_{\pi} G_\pi$$

Batch IRL In the Batch Inverse Reinforcement Learning (IRL) setting, an agent is given some trajectories $(s,a)$ from a policy which we are told is (near) optimal, and in turn, asked to determine what reward $R(s,a)$ must have been. Further we assume the "batch" setting which means the agent has no further interaction with the MDP, resembling high-stakes scenarios in real life such as healthcare. In the following, we introduce two prominent ways of doing IRL: apprenticeship learning and adversarial IRL.
Max-margin Apprenticeship Learning (MMA)

Abbeel and Ng [2] were one of the first to recover the reward function based on expert trajectories. They assumed the existence of an expert policy \( \pi_E \) that is optimal under some unknown linear reward function of the form \( R(s, a) = w \cdot \phi(s) \) for some weight vector \( w \in \mathbb{R}^d \) and some predefined feature map \( \phi(s, a) : S \times A \to \mathbb{R}^d \). By this linear assumption, the cumulative reward \( G \) can be written as an inner product between \( w \) and the feature expectation \( \mu^\pi \):

\[
G_\pi = \sum_{t=0}^{T} E_{\pi,T}[\gamma^t r(s_t, a_t)] = w \cdot \sum_{t=0}^{T} E_{\pi,T}[\gamma^t \phi(s_t, a_t)] = w \cdot \mu^\pi
\]

Conceptually, the feature expectation \( \mu^\pi \) represents the (expected, discounted) amount of the feature accumulated while acting under a policy \( \pi \). Max-margin IRL tries to find a \( w \) such that the cumulative reward of expert \( G^E \) is bigger than any other policy \( G^\pi \) that \( w \cdot \mu^E \geq w \cdot \mu^\pi \) for any \( \mu^\pi \) generated by any policy \( \pi \).

Max-Ent IRL One of the problems with Max-margin IRL is that it does not have unique solutions. Indeed, even \( w = 0 \) is an answer to the above optimization. Ziebart et al. [33] instead proposes Max-Entropy IRL that seeks to find a reward \( r \) that maximizes the likelihood of the trajectories sampled from the optimal policy \( \pi_E \). This formulation has a unique global solution and solves the ambiguity problem of MMA, but still assumes the reward is linear.

Adversarial IRL GAIL [12] is the first work to formulate Max-Ent IRL as an adversarial game between a policy learner (generator) and a reward model (discriminator) and show their policy can recover expert policies while allowing non-linear reward modeling using deep neural networks (DNNs). However, the reward model of GAIL might not recover the reward expert uses. AIRL further improves upon GAIL by avoiding the degradation of reward model (discriminator) in GAIL, and presents a practical implementation that scales well to various environments. Although AIRL recovers the reward, the adoption of DNNs in the reward modeling hinders the interpretability of understanding the reward expert optimizes. They also do not consider the "what-if" reasoning of counterfactuals and the difficult batch setting in this paper.

Generalized Additive Models (GAM) GAM are interpretable by design due to their simple functional forms. Given an input \( x \in \mathbb{R}^D \), a label \( y \), a link function \( g \) (e.g. \( g \) is identity function in regression), main effects \( f_j \) for each feature \( j \), GAM are expressed as:

\[
g(y) = f_0 + \sum_{j=1}^{D} f_j(x_j) \tag{1}
\]

Unlike full complexity models (e.g. DNNs) that have \( y = f(x_1, ..., x_j) \), GAMs are interpretable because the impact of each feature \( f_j \) can be visualized as a graph (i.e. for \( f_j \), x-axis shows \( x_j \) and y-axis shows \( f_j(x_j) \)). Humans can easily simulate how they work by reading \( f_j \)s from the graph and adding them together.

Node-GAM In this paper we adopt the deep-learning version of GAM: Node-GAM [6] which achieves state-of-the-art performance in tabular data. Unlike other non-differentiable GAMs such as Spline [11] or EBM [19], Node-GAM allows back-propagation and thus can be adopted in the AIRL framework.

4 Methods

First, we introduce how we train our counterfactual transition model that estimates the next state given expert batch data. Then we illustrate how we train our policy (generator) and the reward model (discriminator) based on both the expert batch data and the counterfactual data generated from the transition model.

Counterfactual Transition Model To explain the expert’s intent by what-if reasoning, we need to model the future outcome that defines the feature map \( \phi \). We adopt the potential outcomes framework [24]. Let \( Y[a] \) be the potential outcome for treatment \( a \in A \). Then

| Method     | Batch | What-if reasoning | Reward | Unique Soln | Interpretable | Modeled more than 5 features |
|------------|-------|-------------------|--------|-------------|---------------|-----------------------------|
| MMA [2]    | No    | No                | \( w \cdot \phi(s) \) | No          | Limited       | Yes                         |
| DSFN [17]  | No    | No                | \( w \cdot \phi(s, a) \) | No          | Limited       | Yes                         |
| CIRL [4]   | Yes   | Yes               | \( w \cdot E[Y_{t+1}|h] \) | No          | Limited       | Yes                         |
| AIRL [21]  | No    | No                | DNN(s) | Yes         | No            | Yes                         |
| iAIRL [29] | Yes   | No                | DNDT(s) | Yes         | Yes           | No                          |
| CAIRL (ours) | Yes   | Yes               | GAM(\( E[Y_{t+1}|h] \)) | Yes         | Yes           | Yes                         |

Table 1: Comparison to related work.
we learn the feature map $\phi$ as the potential outcome of the next state given the action $a_t$ and history $h_t$:

$$\phi = E[Y_{t+1}|a_t|h_t]$$

For factual action $a_t$ assigned under expert policy, the factual observed outcome $s_{t+1}$ is the same as the potential outcome $Y_{t+1}|a_t|h_t$. And the potential outcomes for other actions are the counterfactual ones and allows us to understand what might happen if patients receive different treatments. To identify the potential outcomes from the expert batch data, we have standard assumptions of consistency, positivity and no hidden confounders \[23, 22\], which are common across methods estimating counterfactual outcomes \[18\].

Learning potential outcomes from expert batch data necessitates counterfactual modeling, because the expert treatments which also consider the history of patients create a time-dependent confounding bias \[27\].

Specifically, we first estimate the expert policy $\pi_E$ by a GRU model. Then in each time step $t$ and given the marginal action probability $P(a_t)$, we set the sample weight $w_t$ as:

$$w_t = P(a_t)/\pi_E(a_t|h_t).$$

We train another GRU that predicts $s_{t+1}$ where the loss is weighted by $w_t$. The training details and hyper-parameters are in Supp. C.

**Generator: deriving optimal policy under a given reward** To derive a policy under a given reward generated by discriminator, we adopt the state-of-the-art offline RL method: Soft-Q learning \[10\] that allows optimization on both the expert batch data and the counterfactual data generated by the transition model. Specifically, given a network $Q$, an experience of $(s, a, r, s')$, and entropy coefficient $\alpha$, we minimize the Huber loss $L_H$:

$$\min_{Q} L_H((Q(s, a), r(s') + \sum_{a'} \pi(a'|s') (Q(s', a') - \log \pi(a'|s')))$$

where $\pi(a|s) = \text{Softmax}(Q(s, a)/\alpha)$

Since our transition model may not predict the future states perfectly in the real-world data, we use three ways to alleviate this bias when updating the generator. First, instead of simulating the full trajectories from our transition model acted under policy $\pi$ like MMA does, we instead only do one-step future predictions from the sampled expert demonstrations as our counterfactual data; this reduces the extrapolation error in multiple timesteps. Second, when the sampled action matches the expert action in the history, we directly use the logged next state $s_{t+1}$ instead of the prediction from the transition model. Finally, we put a smaller penalty on the loss of these counterfactual data. We search $\delta$ over $[0, 0.5, 1]$ and find $\delta = 0.5$ produces the best result. Specifically, given the expert batch data $D_E$ and the transition model $T$:

$$L = E_{D_E}[L_H(s, a, s')] + \delta E_{s\sim D_E, a\sim \pi(h(s), s'|T(s,a))}[L_H(s, a, s')]$$

**Behavior cloning regularization** To stabilize the generator optimization, we find that it’s crucial to regularize the first part of optimization to be close to the policy derived from behavior cloning (i.e. using the supervised learning to predict actions). We first train a GRU model to directly predict $a_t$ by history $h_t$ that learns $\pi_{bc}(a_t|h_t)$: then when updating our Q-network, we add an additional KL divergence loss between the current policy $\pi$ and $\pi_{bc}$:

$$L_{bc} = \lambda_{bc} KL(\pi_Q, \pi_{bc})$$

And we linearly decay the $\lambda_{bc}$ to 0 in the first half of the optimization.

**Discriminator: the reward model** We follow the similar design from AirlR that trains a binary classifier to predict whether $\phi$ comes from the expert or the generator, but here $\phi$ is the next state $s_{t+1}$ instead of $s_t$. Specifically, given $g$ as the reward model, $h$ as the shaping term modeling, $\pi$ the generator’s policy, the discriminator logit $D$ is:

$$D(s, a, s') = g(\phi) + h(s') - h(s) - \log \pi(s, a)$$

And we set feature map $\phi$ as $s'$ while AirlR sets $\phi$ as $s$. We set both $g$ and $h$ as the Node-GAM. Then we set the class $y$ of expert batch as 1 and generated data as 0, and optimize the binary cross entropy loss (BCE):

$$L_D = E_{s, a, s'\sim D_E} [\text{BCE}(D(s, a, s'), 1)] + E_{s\sim D_E, a\sim \pi(h(s), s'|T(s,a))} [\text{BCE}(D(s, a', s'), 0)].$$

**Discriminator Stabilizing Tricks** When optimizing discriminator, we use both one-sided label smoothing \[25\] and add add a small Gaussian noise \[13\] to the inputs which has been shown useful to stabilize GAN adversarial optimization. See Supp. D.1 for details.

**Reward scaling** Since the reward can be arbitrarily shifted and scaled without changing the resulting optimal policy, we need to set the scale for each model to compare them meaningfully. Therefore, in the simulation, we set the scaling of each model to have the smallest $\ell_1$ distance to the ground truth reward under the state distribution of the expert batch data. In real-world data where there is no ground truth, we choose
Table 2: The performance for 7 IRL models in sepsis simulation environment. Here we show both the reward and the L1 distance (Dist) to the ground truth reward in the GAM shape graph.

|                      | \( \gamma = 0.9 \) |                      | \( \gamma = 0.5 \) |
|----------------------|---------------------|---------------------|---------------------|
|                      | GAM MDP             | Linear MDP          | GAM MDP             | Linear MDP          |
|                      | Reward Dist         | Reward Dist         | Reward Dist         | Reward Dist         |
| MMA                  | -6.112 (0.027)      | 1.631 (0.004)       | -1.081 (0.010)      | 0.316 (0.001)       |
| CIRL                 | -7.637 (0.040)      | 1.629 (0.013)       | -1.111 (0.001)      | 0.341 (0.003)       |
| Linear-AIRL         | -0.919 (0.012)      | 1.664 (0.010)       | -0.362 (0.003)      | 0.344 (0.003)       |
| FCNN-AIRL           | -0.931 (0.012)      | 1.670 (0.013)       | -0.357 (0.003)      | 0.342 (0.003)       |
| GAM-AIRL            | -6.449 (0.008)      | 1.690 (0.011)       | -0.357 (0.003)      | 0.345 (0.004)       |
| Linear-CAIRL        | -0.947 (0.010)      | 1.687 (0.009)       | -0.357 (0.004)      | 0.345 (0.004)       |
| FCNN-CAIRL          | -0.894 (0.013)      | 1.682 (0.022)       | -0.357 (0.001)      | 0.345 (0.004)       |
| GAM-CAIRL           | -0.894 (0.013)      | 1.682 (0.022)       | -0.357 (0.001)      | 0.345 (0.004)       |
| Expert              | -0.883 (0.002)      | 1.708 (0.008)       | -0.356 (0.009)      | 0.345 (0.005)       |

the scale to minimize the difference of max and min value in each feature between models to make them display in the similar range. See Supp. E for details.

5 Results

We evaluated our model on two tasks: a simulated sepsis task and a real-world clinical treatment task.

Baselines We compare with the widely-used Max-Margin Apprenticeship learning (MMA) that follows the linear design of the reward. We also compare with the counterfactual version of MMA, CIRL, in our simulations. In addition, we also compare with the AIRL framework that uses the current state \( s_t \) instead of our what-if reasoning of the future outcome \( s_{t+1} \). Within both AIRL and CAIRL frameworks, we compare 3 reward models: (1) Linear, (2) Node-GAM (GAM), and (3) Fully-Connected Neural Network (FCNN).

5.1 Sepsis simulator: a clinically-motivated environment with known ground truth reward and dynamics

We first experiment on a challenging sepsis simulation environment from Oberst and Sontag [20]. This is a coarse physiological model for sepsis with 4 time-varying vitals (Systolic BP, Percentage of Oxygen, ...) that’s discretized (e.g. “low”/“normal”/“high”). Combined with 3 different binary treatments (total 8 actions) and 1 static variable (diabetes), our resulting MDP consists of 1440 possible discrete states. Trajectories are at most 20 timesteps. In addition to 4 vitals, in our feature space we include a uniform noise feature to make it harder. Since this simulator is a discrete environment, we can solve the exact optimal policy via value iteration to generate expert data. We also use the underlying MDP as our transition model that resembles a perfectly trained counterfactual model. We generate 5000 trajectories with optimal policy for both training and test data.

To test if our model can recover the ground truth reward, we design the reward function in two forms. (1) GAM MDP: we modify the reward as an additive function of \( s_{t+1} \), i.e. \( r = \sum_j f_j(s_{t+1}) \). (2) Linear MDP: we modify the reward as a linearly additive function of \( s_{t+1} \) i.e. \( r = w \cdot s_{t+1} \). Its specific functional form can be found in Supp. E.

In Table 2 we show the reward and the distance to ground truth reward of all 8 models under \( \gamma = 0.9 \) and 0.5. First, in GAM MDP, since Linear model would not work in this environment, we see MMA, CIRL and Linear models perform poorly. Out of all models GAM-CAIRL achieves the highest reward and also recovers ground truth more faithfully with the lowest distance on the shape graph to the ground truth (Fig. 2). It also outperforms GAM-AIRL, which does not include what-if reasoning but still achieves reasonable performance. In Linear MDP, we see Linear-CAIRL performs the best. Linear-AIRL has similar reward but its distance on the graph is much larger (0.051 v.s. 0.016). And MMA and CIRL perform slightly worse than Linear-AIRL and Linear-CAIRL. GAM and FCNN also perform well without significant differences from Linear-CAIRL.

We repeat the experiment with \( \gamma = 0.5 \) and find the reward difference between models becomes smaller. In GAM MDP, GAM-CAIRL still achieves the smallest distance to the ground truth. In Linear MDP, however, GAM has a smaller distance than Linear in both CAIRL and AIRL settings; we find Linear has an opposite slope in feature Glucose that leads to a larger distance.

We visualize our shape graphs in GAM MDP when
Table 3: The test accuracy (with stdev) of actions matched to the expert in the MIMIC3. BC is behavior cloning.

| Action | Acc(%) | BC | Linear-AIRL | GAM-AIRL | FCNN-AIRL | Linear-CAIRL | GAM-CAIRL | FCNN-CAIRL |
|--------|--------|----|-------------|----------|---------|-------------|----------|---------|
| Heart Rate | 72.0 (1.0) | 74.1 (0.4) | 74.2 (0.9) | 73.8 (0.5) | **74.9** (0.4) | 74.7 (0.3) | 74.4 (0.4) |
| Systolic BP | 72.0 (1.0) | 74.1 (0.4) | 74.2 (0.9) | 73.8 (0.5) | **74.9** (0.4) | 74.7 (0.3) | 74.4 (0.4) |
| % of Oxyg | 72.0 (1.0) | 74.1 (0.4) | 74.2 (0.9) | 73.8 (0.5) | **74.9** (0.4) | 74.7 (0.3) | 74.4 (0.4) |
| Glucose | 72.0 (1.0) | 74.1 (0.4) | 74.2 (0.9) | 73.8 (0.5) | **74.9** (0.4) | 74.7 (0.3) | 74.4 (0.4) |

We find Linear (green) model can not handle non-linear reward and thus act as a straight line. And our GAM-CAIRL (red) is closest to GT (blue).

\( \gamma = 0.9 \) in Fig. 2. First, linear model as expected can not capture the non-linear relationship and thus is flat. In Heart Rate, Systolic BP and % of Oxygen, all models except linear capture the correct trend. For Glucose, only GAM-CAIRL capture the correct shape that finds value 2 produces the highest reward. We also show the shape graphs in Linear MDP when \( \gamma = 0.9 \) in Fig. 3. Overall all models capture the correct trend in all 4 features.

5.2 MIMIC3 Hypotension Datasets

To demonstrate the utility of our method, we experiment on a real-world medical decision making task of managing hypotensive patients in the ICU. Hypotension is correlated with high mortality [15]. Although there exists various clinical guidelines [5, 16], there is no standardized treatment strategy since there are many underlying causes of hypotension [29].

Preprocessing We use MIMIC-III [14], filtering to adult patients with at least 2 treatments within the first 72 hours into ICU resulting in 9,404 ICU stays. We discretize trajectories into 2-hour windows, so trajectories end either at ICU discharge or at 72 hours into the ICU admission with at most 36 timesteps and 35 actions taken. We follow the preprocessing of Futoma et al. [8] to select two common treatments: fluid bolus therapy and vasopressors. We discretize both treatments into 4 levels (none, low, medium and high). We extract 5 covariates and 29 time-varying features and impute the missing value with the forward imputation. For each model we perform 5-fold cross validation with each fold having 60-20-20 for train-val-test splits. We set \( \gamma = 1 \). More details are in Supp. A.

In Table 3 we show the accuracy of the actions matched to the expert under different methods. We also compare with behavior cloning (BC) which does supervised learning directly from the logged expert data. We find Linear-CAIRL and GAM-CAIRL perform the best and outperform both BC and their AIRL counterparts. In Fig. 4 we evaluate the shape graphs derived from both our Linear-CAIRL and GAM-CAIRL. First, we compare with the clinician-designed reward responsible for hypotensive patients [8] of two features, MAP and Lactate, shown in Fig. 5. We find GAM-CAIRL recovers the right regions: in Fig. 4(a) the reward increases as MAP increases above 65 and decreases after 100, which matches the normal range of MAP between 65 and 100 in the clinical guideline. Similarly, in Fig. 4(b) the reward of lactate substantially drops as it grows be-
We illustrate other features (c)-(h). In Systolic BP (c), beyond the value 2 and keeps slowly decreasing matching the trend in Fig. 5. Despite the fact that Linear-CAIRL has similar accuracy to GAM-CAIRL (Table 3), it only learns a modest increase in MAP, and an opposite trend in Lactate which is counter to clinical intuitions.

We illustrate other features (c)-(h). In Systolic BP (c), since the goals of both fluids and vasopressors management are to increase blood pressure, it makes sense that the lower blood pressure has lower reward. Unfortunately, GAM assigns high reward to high Systolic BP even when it is around 200 which we think is an artifact due to the inductive bias of tree-based Node-GAM that remains flat. Linear model instead indicates a negative slope that suggests the lower the blood pressure the better which is clearly in violation of the goal of treating hypotension. Glasgow Coma Scale (GCS, (d)) describes the level of consciousness of patients with value 15 meaning high consciousness and 3 meaning deep coma. Our GAM model captures this notion by learning a steady increase of reward as GCS increases, while linear model learns the opposite trend which does not make sense. PO2 (e) measures the oxygen concentration in blood. Studies show that PO2 > 80 is likely another artifact. Again, linear model shows a sharp decrease when PO2 is right above 100 which confirms this. It is important to note that low PO2 (<80) should not be rewarded either, suggesting that the high reward learned by GAM at PO2 < 80 is likely another artifact. Again, linear model learns the completely opposite trend in PO2. For heart rate (f), both GAM and linear model correctly agree that slower heart rate is generally better although the
heart rate of 20 is likely too low and should not be rewarded. For potassium (g), high potassium is correlated with kidney disease and sometimes can cause a heart attack or death. Our GAM roughly matches the clinical guideline that assigns higher reward for normal range (2.5-5.1) and assigns much lower reward for high potassium. Instead linear model has the opposite trend again. In (h), hematocrit level (HCT) is the percentage of red cells in the blood, and normally when hematocrit is too low it indicates an insufficient supply of healthy red blood cells (anemia). GAM successfully captures this by assigning high reward in HCT but linear model again contradicts the clinical knowledge.

In the third row (i)-(l), urine volume (i) is correlated with blood pressure and usually high output is a good sign of health while low volume (<50) is an indicator of acute or chronic kidney disease. GAM successfully captures this trend by learning much lower reward for low urine especially below 50, while Linear model learns a higher reward for lower urine output. Body temperature (j) should be maintained between 36-38 degrees. Values that are higher or lower are concerning; GAM captures it perfectly while Linear model learns the upward trend that higher the temperature the better, failing to capture the needed non-linearity. WBC (k) has a normal range between 4.5−11, and high WBC often indicates an infection. GAM displays a steady decrease once the threshold of 10 is exceeded. FiO2 (l) is usually maintained below 50 even when ventilation is used to avoid oxygen toxicity, and we clearly see a sharp decrease of the reward after 1.1, while Linear model is unable to learn this threshold effect. BUN (o) measures the amount of urea nitrogen in the blood, and a high BUN level implies worse conditions like heart failure or shock. Both GAM and Linear model capture the correct downturn trend but GAM learns a sharp drop around 8 similar to the clinical guideline. Finally, Bilirubin (p) is a yellow pigment that occurs normally when part of one’s red blood cells break down. High bilirubin levels are a sign that the liver isn’t clearing the bilirubin from one’s blood as it should. GAM again captures the important clinical threshold of 1.2 while the Linear model has the opposite trend learning that higher BUN is better.

6 Limitations and Discussions

Our framework necessitates a good counterfactual transition model. If the transition model has not been properly tuned, the results might not be meaningful to humans, especially in the medical domain where measurements are usually long-tailed with lots of missing values. We find that performing quantile transformation to Gaussian is crucial to avoid outliers. Training with \( \ell_1 \) loss instead of Mean Squared Error (MSE) loss promotes the diversity of the future state predictions and prevents models from always predicting a value near zero (population mean).

Parameterizing reward functions using counterfactuals gives us an explanation of which outcomes experts likely prefer. As the reward could be sparse or dense, or depends on the current state or next state in different environments, we do not claim our use of counterfactuals recovers exactly the underlying expert reward. Neither do we assume experts actually compute these quantities nor that they adopt the same causal inference assumptions; rather we simply provide an interpretable way to explain how decision-makers are behaving in terms of future counterfactuals.

We observed that linear models often generated the opposite rewards from GAMs. We believe it happens because in the process of trying to fit data that is not linear it compensates by changing the signs of other correlated features thus creating counter-intuitive patterns.

The discretization in RL needs to be done carefully. For example, vasopressors and fluids should take effect within 2 hours, thus we discretized our data in 2 hour time windows. If the treatment takes longer to have an effect, a different discretization or considering multiple timesteps ahead might be needed.
References

[1] Deepmind ai reduces google data centre cooling bill by 40%. URL https://deepmind.com/blog/article/deepmind-ai-reduces-google-data-centre-cooling-bill-40.

[2] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In Proceedings of the twenty-first international conference on Machine learning, page 1, 2004.

[3] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. arXiv preprint arXiv:1606.06565, 2016.

[4] Ioana Bica, Daniel Jarrett, Alihan Hüyük, and Mihaela van der Schaar. Learning" what-if" explanations for sequential decision-making. arXiv preprint arXiv:2007.13531, 2020.

[5] Jessica Bunin. Diagnosis and management of hypotension and shock in the intensive care unit.

[6] Chun-Hao Chang, Rich Caruana, and Anna Goldenberg. Node-gam: Neural generalized additive model for interpretable deep learning. arXiv preprint arXiv:2106.01613, 2021.

[7] Chun-Hao Chang, Sarah Tan, Ben Lengerich, Anna Goldenberg, and Rich Caruana. How interpretable and trustworthy are gams? In Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, pages 95–105, 2021.

[8] Joseph Futoma, Michael C Hughes, and Finale Doshi-Velez. Popcorn: Partially observed prediction constrained reinforcement learning. arXiv preprint arXiv:2001.04032, 2020.

[9] Seyed Kamyar Seyed Ghasemipour, Richard Zemel, and Shixiang Gu. A divergence minimization perspective on imitation learning methods. In Conference on Robot Learning, pages 1259–1277. PMLR, 2020.

[10] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In International Conference on Machine Learning, pages 1352–1361. PMLR, 2017.

[11] Trevor J Hastie and Robert J Tibshirani. Generalized additive models. Routledge, 2017.

[12] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. Advances in neural information processing systems, 29:4565–4573, 2016.

[13] Simon Jenni and Paolo Favaro. On stabilizing generative adversarial training with noise. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 12145–12153, 2019.

[14] Alistair EW Johnson, Tom J Pollard, Lu Shen, Michael Li-Wei, Mengling Feng, Mohammad Ghasemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database. Scientific data, 3(1):1–9, 2016.

[15] Alan E Jones, Lyn S Aborn, and Jeffrey A Kline. Severity of emergency department hypotension predicts adverse hospital outcome. Shock, 22(5):410–414, 2004.

[16] Ashish K Khanna. Defending a mean arterial pressure in the intensive care unit: Are we there yet? Annals of intensive care, 8(1):1–2, 2018.

[17] Donghun Lee, Srivatsan Srinivasan, and Finale Doshi-Velez. Truly batch apprenticeship learning with deep successor features. arXiv preprint arXiv:1903.10077, 2019.

[18] Bryan Lim, Ahmed M Alaa, and Mihaela van der Schaar. Forecasting treatment responses over time using recurrent marginal structural networks. NeurIPS, 18:7483–7493, 2018.

[19] Harsha Nori, Samuel Jenkins, Paul Koch, and Rich Caruana. Interpretml: A unified framework for machine learning interpretability. arXiv preprint arXiv:1909.09223, 2019.

[20] Michael Oberst and David Sontag. Counterfactual off-policy evaluation with gumbel-max structural causal models. In International Conference on Machine Learning, pages 4881–4890. PMLR, 2019.

[21] Ahmed H Qureshi, Byron Boots, and Michael C Yip. Adversarial imitation via variational inverse reinforcement learning. arXiv preprint arXiv:1809.06404, 2018.

[22] James M Robins, Miguel Angel Hernan, and Babette Brumback. Marginal structural models and causal inference in epidemiology, 2000.

[23] Paul R Rosenbaum and Donald B Rubin. The central role of the propensity score in observational studies for causal effects. Biometrika, 70(1):41–55, 1983.

[24] Donald B Rubin. Bayesian inference for causal effects: The role of randomization. The Annals of statistics, pages 34–58, 1978.

[25] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. Advances in neural information processing systems, 29:2234–2242, 2016.
[26] Ahmad EL Sallab, Mohammed Abdou, Etienne Perot, and Senthil Yogamani. Deep reinforcement learning framework for autonomous driving. *Electronic Imaging*, 2017(19):70–76, 2017.

[27] Peter Schulam and Suchi Saria. Reliable decision support using counterfactual models. *Advances in Neural Information Processing Systems*, 30:1697–1708, 2017.

[28] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

[29] Srivatsan Srinivasan and Finale Doshi-Velez. Interpretable batch irl to extract clinician goals in icu hypotension management. *AMIA Summits on Translational Science Proceedings*, 2020:636, 2020.

[30] Sarah Tan, Rich Caruana, Giles Hooker, and Yin Lou. Distill-and-compare: Auditing black-box models using transparent model distillation. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, pages 303–310, 2018.

[31] Sarah Tan, Rich Caruana, Giles Hooker, Paul Koch, and Albert Gordo. Learning global additive explanations of black-box models. 2019.

[32] Yongxin Yang, Irene Garcia Morillo, and Timothy M Hospedales. Deep neural decision trees. *arXiv preprint arXiv:1806.06988*, 2018.

[33] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, Anind K Dey, et al. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.
A MIMIC3 Preprocessing

We follow Futoma et al. [8] to extract 5 covariates, 29 time-varying features and 10 features related to actions. We use the quantile transformation to Gaussian distribution and finds models provide more meaningful results than the log transformation used in Futoma et al. [8]. For action features, we calculate the past treatment values including the treatment in the last time point (e.g. last_fluid_2 means if the patient gets treated in the last time point with value 2 (medium)), and the treatment value in the last 8 hours, and the total treatment values so far. We also include the missingness indicator for each feature since medical data is not missing at random which results in total 73 features.

- covariates: age, is_F, surg_ICU, is_not_white, is_emergency, is_urgent
- features: dbp, fio2, hr, map, sbp, spontaneousrr, spo2, temp, urine, weight, bun, magnesium, platelets, sodium, alt, hct, po2, ast, potassium, wbc, bicarbonate, creatinine, lactate, pco2, bilirubin_total, glucose, inr, hgb, GCS
- action features: last_vaso_1, last_vaso_2, last_vaso_3, last_fluid_1, last_fluid_2, last_fluid_3, total_all_prev_vasos, total_all_prev_fluids, total_last_8hrs_vasos, total_last_8hrs_fluids

B Sepsis simulation reward design

In GAM MDP, we simulate the reward using the following state value pair:

- Heart rate: {0: -0.8, 1: 0, 2: -1}
- Systolic BP: {0: -1.2, 1: 0, 2: -0.6}
- % of Oxygen: {0: 0, 1: 0}
- Glucose: {0: -0.8, 1: -0.4, 2: 0, 3: -0.4, 4: -0.8}

In Linear MDP, we simulate the reward using the following state value pair:

- Heart rate: {0: -0.3, 1: -0.6, 2: -0.9}
- Systolic BP: {0: -0.4, 1: -0.8, 2: -1.2}
- % of Oxygen: {0: 0, 1: 0.6}
- Glucose: {0: 0, 1: 0.2, 2: 0.4, 3: 0.6, 4: 0.8}

C GRU training for behavior cloning and counterfactual transition model

Behavior Cloning (BC) Behavior cloning model takes in the history \( h_t \) to predict the current action \( a_t \) in the expert batch data. We use GRU to model the prediction. So we feed the history \( h_t \) into the GRU to produce output \( o_t \), and then feed the output into several layers of fully-connected layers (FC) with dropout and batchnorm to produce the final classification of action \( a_t \). We list the hyperparameters in Table 4.

Counterfactual Transition Model Learning Counterfactual models take in the current history \( h_t \) and action \( a_t \) to predict the next states \( s_{t+1} \). We use the similar architecture as the GRU for behavior cloning, except we also use action \( a_t \) as inputs and do the regression to predict \( s_t \). Specifically, for action \( a_t \), we go through several layers of Fully Connected Layers to produce action embeddings \( \phi_a \), and concatenate with the state \( s_t \) as the inputs to the GRU model. For output, we use the Huber loss (smoothed \( \ell_1 \) loss) that we find it produces more diverse states than Mean Squared Error (MSE) loss. Finally, we weight the samples by stabilized Inverse Propensity Weighting (IPTW) that gives different sample weights \( w_t \). We list the hyperparameters in Table 4.
### Table 4: Hyperparameters for GRU training for Behavior Cloning (BC) and Transition Model. We use random search to find the best hyperparameters.

|                      | GRU BC | GRU Transition Model |
|----------------------|--------|----------------------|
| **Epochs**           | 200    | 200                  |
| **Batch size**       | 64, 128, 256 | 128, 256           |
| **Learning Rate**    | 5e-4, 1e-3, 2e-3 | 5e-4, 1e-3         |
| **Weight Decay**     | 0, 1e-6, 1e-5, 1e-4 | 0, 1e-5          |
| **GRU Num Hidden**   | 64, 128, 256 | 64                 |
| **GRU Num layers**   | 1      | 1                    |
| **GRU Dropout**      | 0.3, 0.5 | 0.3, 0.5            |
| **FC Num Hidden**    | 128, 256, 384, 512 | 256, 384, 512   |
| **FC Num Layers**    | 2, 3, 4 | 2                   |
| **FC Dropout**       | 0.15, 0.3, 0.5 | 0.15                |
| **FC Activation**    | ELU    | ELU                  |
| **Act Num Hidden**   | -      | 64, 128              |
| **Act Num Layers**   | -      | 0, 1, 2              |
| **Act Num output**   | -      | 32, 64, 96           |
| **Act Dropout**      | -      | 0.3                  |

Specifically, given the behavior cloning policy $\pi_{bc}$, action embedding layers (Act) $A$, GRU model $G$ and output fully connected layer $F$, we have

\[
\phi_a = A(a_t) \\
\phi = \text{concat}(s_t, \phi_a) \\
g_t = M(\phi) \\
o_t = F(g_t) \\
w_t = \frac{P(a_t)}{\pi_{bc}(a_t)} \\
L = w_t \cdot \text{Huber}(o_t, s_{t+1}) \\
\theta \leftarrow \theta - \nabla_\theta L \quad \text{(Updated by Adam)}
\]

### D AIRL training

#### D.1 Discriminator Training

Here we train a discriminator that can distinguish expert batch data as class 1 and generated experience as class 0 in the binary classification setting. And its logit would represent the expert reward $r$. Given a batch of expert data $X_E$, we generate the data $X_G$ by executing the generator policy $\pi$ in the trained counterfactual transition model $T$. Note that to explain the expert in terms of future counterfactuals, we exclude the static covariates and action features and only use the time-varying features of next state when training the discriminator.

**Discriminator Stabilizing Tricks** When optimizing discriminator, we use both one-sided label smoothing \[25\] which reduces the label confidence for the expert batch data. We also add a small input Gaussian noise to the inputs for both expert and generated data, and linearly decayed the noise throughout the training. It has been shown useful to stabilize GAN adversarial optimization \[13\]. Specifically, given label smoothing $\delta$, input noise $\delta_n$, the discriminator model $D$, expert batch data $D_E$, the transition model $T$, the generator policy $\pi_G$ and binary
Table 5: Hyperparameters for Node-GAM training for discriminator training. We use random search to find the best hyperparameters.

| Hyperparameter                | Simulation | MIMIC3 |
|-------------------------------|-----------|--------|
| Epochs                        | 100       | 100    |
| Input Noise                   | 0         | 0.1    |
| Noise decay                   | 0         | 80%    |
| LR                            | 2e-4, 4e-4| 5e-4, 8e-4, 1e-3 |
| Label Smoothing δ             | 0         | 0, 0.005, 0.01 |
| Num Layers                    | 1, 2      | 1, 2, 3 |
| Num Trees                     | 100, 200, 400 | 200, 300, 400 |
| Addi Tree Dim                 | 0, 1      | 0, 1   |
| Depth                         | 1, 2      | 2, 3, 4 |
| Output Dropout                | 0, 0.1    | 0.1, 0.2 |
| Last Dropout                  | 0, 0.3    | 0.3, 0.5 |
| Column Subsample              | 1, 0.5    | 0.5    |
| Temp Annealing                | 3000      | 3000   |

Table 6: Hyperparameters for Linear model and FCNN model for discriminator training.

| Hyperparameter | Linear Simulation | MIMIC3 | FCNN Simulation | MIMIC3 |
|----------------|-------------------|--------|----------------|--------|
| Epochs         | 100               | 100    | 100            | 100    |
| Input Noise    | 0                 | 0.1    | 0              | 0.1    |
| Noise decay    | 0                 | 0.8    | 0              | 0.8    |
| LR             | 2e-4, 4e-4        | 5e-4, 8e-4, 1e-3 | 2e-4, 4e-4 | 5e-4, 8e-4, 1e-3 |
| Label Smoothing δ | 0           | 0, 0.005, 0.01 | 0          | 0, 0.005, 0.01 |
| Num Layers     | -                 | -      | 2, 3, 4        | 2, 3, 4 |
| Num Hidden     | -                 | -      | 32, 64, 128, 256 | 32, 64, 128, 256 |
| Dropout        | -                 | -      | 0.1, 0.3, 0.5  | 0.1, 0.3, 0.5 |

cross entropy loss (BCE):

\[
X_E = (s, a, s') \sim D_E \\
X_G = (s, a, s') \text{ where } s \sim X_E, a \sim \pi_G(\cdot|s) \text{ and } s' \sim T(\cdot|s, a) \\
n \sim \mathcal{N}(0, 1) \\
X_E = X_E + \delta_n \cdot n, \ X_G = X_G + \delta_n \cdot n \\
L = \text{BCELoss}(D(X_E), 1 - \delta) + \text{BCELoss}(D(X_G), 0)
\]

We list the hyperparameters we use for Node-GAM in Table 5. And we list the linear and FCNN model’s hyperparameters in Table 6.

D.2 Generator Training

In Sepsis simulation, since we use the value iteration to solve the exact optimal policy for our generator, there is no hyperparameter to tune. To save the computation, we only update the generator after the discriminator updates for 20 steps.

In MIMIC3, we use the soft-Q learning as our generator as described in Sec. 4. We use a fully connected neural net with dropout, ELU activation function and batchnorm as our architecture. We show the hyperparameters in Table 7.
Table 7: Hyperparameters for generator training in Sepsis and MIMIC3 datasets.

|                      | Sepsis | MIMIC3 |
|----------------------|--------|--------|
| Update Freq          | 20     | 1      |
| Epochs               | -      | 100    |
| Entropy Coeff α      | -      | 0.25, 0.5 |
| Sample weights for gen data (δ) | -      | 0.5    |
| Sync Rate            | -      | 200    |
| LR                   | -      | 4e-4, 8e-4 |
| Num Layer            | -      | 3, 4   |
| Dropout              | -      | 0.3, 0.5 |
| BC Reg               | -      | 10     |
| BC Reg decay         | -      | 0.5    |

E  Reward scaling

Since the reward can be arbitrarily shifted and scaled without changing the resulting optimal policy, comparing the reward across models requires setting the scale of the reward for each model when showing the GAM plots and calculating distance. Therefore, in the simulation for each model, we shift the average reward to 0 and set the scaling $a$ that has the smallest $\ell_1$ distance to the ground truth reward under the state distribution of the expert batch data. Given the ground truth model $G$ and its GAM main effect $f_G(x_j)$ of each feature $j$, model $M$ and $f_M(x_j)$, $V_j$ as all the values of feature $j$, with each value $v \in V_j$, and the counts $c(v)$ in the expert batch data, we derive $a$ by convex optimization:

$$\min_a \sum_{j=1}^{D} \sum_{v \in V_j} |(f_G(v) - a f_M(v))| c(v)$$

In real-world data where there is no ground truth, we choose the scale $a$ to minimize the difference of max and min value in each feature of two models $G,M$ to make them display in the similar range:

$$\min_a \sum_{j=1}^{D} \left( \min_{v \in V_G} f_G(v) - a \min_{v \in V_M} f_M(v) \right) + \left( \max_{v \in V_G} f_G(v) - a \max_{v \in V_M} f_M(v) \right).$$

F  Clinical guidelines sources

Here we list the lower and upper bound, and the sources of the normal ranges we use in Fig. 4.

- MAP: (70, 100) [https://www.healthline.com/health/mean-arterial-pressure#:~:text=What%20is%20a%20normal%20MAP,100%20mmHg%20to%20be%20normal.]
- Lactate: (0, 2)
- Systolic BP: upper bound 180 [https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3704960/] and lower bound 90 [https://www.mayoclinic.org/diseases-conditions/low-blood-pressure/symptoms-causes/syc-20355465#:~:text=What’s%20considered%20low%20blood%20pressure,pressure%20to%20lower%20than%20normal.]
- Bicarbonate: (23, 30) [https://www.urmc.rochester.edu/encyclopedia/content.aspx?contenttypeid=167&contentid=bicarbonate#:~:text=Normal%20bicarbonate%20levels%20are%3A,30%20mEq%20FL%20in%20adults]
- pO2: (75, 100) [https://www.medicalnewstoday.com/articles/322343#:~:text=Most%20healthy%20adults%20have%20pO2%20values%20above%2090%,emphysema]
• Heart Rate: (40, 100) [https://health.clevelandclinic.org/is-a-slow-heart-rate-good-or-bad-for-you/]

• Potassium: (2.5, 5.1) [https://my.clevelandclinic.org/health/diseases/17740-low-potassium-levels-in-your-blood-hypokalemia]

• HCT: (35.5, 48.6) [https://www.mayoclinic.org/tests-procedures/hematocrit/about/pac-20384728]

• Urine: (400, inf) [https://www.healthline.com/health/urine-output-decreased#:~:text=Oliguria%20is%20considered%20to%20be%2C%20is%20considered%20to%20be%20anuria.]

• WBC: (4.5, 11) [https://my.clevelandclinic.org/health/diagnostics/17704-high-white-blood-cell-count]

• FiO2: (21, 50) [https://en.wikipedia.org/wiki/Fraction_of_inspired_oxygen#:~:text=Natural%20air%20includes%2021%25%20oxygen%2C%20to%20be%20routinely%20used.]

• ALT: (0, 55) [https://www.mayoclinic.org/tests-procedures/liver-function-tests/about/pac-20394595]

• INR: (0, 1.1) [https://my.clevelandclinic.org/health/diagnostics/17691-prothrombin-time-pt-test]

• BUN: (2.1, 8.5) [https://www.mayoclinic.org/tests-procedures/blood-urea-nitrogen/about/pac-20384821]

• Bilirubin total: (0, 1.2) [https://www.webmd.com/a-to-z-guides/bilirubin-test]

G Computing Resources Used

All experiments are run on 1 P100 GPU, 4 CPU and 16G RAM on a cluster.

H Complete shape graphs

We show the complete shape graphs of 29 features in MIMIC-III in Fig. 6.
Figure 6: The complete shape plots of MIMIC3.