Interpretable Batch IRL to Extract Clinician Goals in ICU Hypotension Management

Srivatsan Srinivasan, M.E. [1], **Finale Doshi-Velez, PhD**[1]
[1] **Harvard University, Paulson School of Engineering and Applied Sciences, Cambridge, MA**

Abstract

Exposing and understanding the motivations *of clinicians is an important step for building robust assistive agents as well as improving care. In this work, we focus on understanding the motivations for clinicians managing hypotension in the ICU. We model the ICU interventions as a batch, sequential decision making problem and develop a novel interpretable batch variant of Adversarial Inverse Reinforcement Learning algorithm that not only learns rewards which induce treatment policies similar to clinical treatments, but also ensure that the learned functional form of rewards is consistent with the decision mechanisms of clinicians in the ICU. We apply our approach to understanding vasopressor and IV fluid administration in the ICU and posit that this interpretability enables inspection and validation of the rewards robustly.*

Introduction

Decisions are generally made in pursuit of a goal: an intensivist may administer a vasopressor to increase a patient's blood pressure into a safer range; they may suggest a sedative to reduce a patient's agitation. Methods to assist with decision-making, such as reinforcement learning (RL), take these goals as input and attempt to find decisions that will support them. Understanding what goals clinicians seek to *achieve*, rather than rules of how to react (e.g., if the blood pressure is too low, administer vasopressor), can enable the design of agents that will generalize more robustly to new situations. Exposing often implicit goals can also be of inherent interest for clinicians wishing to introspect on their decision-making.

However, identifying these goals can be challenging. For example, in this work, we focus on hypotension management in the ICU. It is an area where data-driven analysis could assist with decision making — while there exist several guidelines for treatment,[1–3], there is no widespread consensus on how to apply these guidelines. When asked about the goals of hypotension management, an intensivist might explain that they administer a vasopressor dosage to increase the patient's blood pressure. This specification misses the fact that their decision also considers keeping the dose low to avoid the risk of vasopressor-induced shock. In general, we cannot expect people to perfectly list a complete set of goals; we tend to make assumptions about what behaviors are reasonable or what desiderata are obvious (e.g. temporary raising the blood pressure is useless if the patient does not survive their ICU stay). In such settings, incomplete or incorrect goal specifications can lead to RL agents learning unsafe and potentially, even adverse behaviors.[4,5]

Inverse Reinforcement Learning[6] (IRL) is a field within machine learning that attempts to identify the implicit goals—more formally in the form of rewards—given demonstrations of expert behavior (e.g. treatment histories). These methods are distinct from imitation learners[7] that simply try to mimic the expert without attempting the more strenuous process of learning the task by understanding the *why* underpinning the decisions. The rewards learned by an IRL algorithm, if formulated appropriately, can be checked by an expert to identify goals (through rewards) that they have forgotten to specify, as well as help experts quantify the relative importance of different goals. Beyond its substantial value as a tool toward building clinician-interpretable assistive agents, the rewards learned by IRL from demonstrations can act as data-driven validations of the extent of concord between clinician behaviour and their intended goals.

Unfortunately, standard IRL algorithms have two major shortcomings when it comes to the purposes above. First, the most popular IRL algorithms[6,8,9] are not designed to work with observational data alone; they require the ability to test arbitrary treatment policies. While there are off-policy (batch) IRL variants[10–12], all these methods suffer from either high bias or high variance estimates and hence have not scaled well to real-life tasks. Thus, identifying rewards from expert trajectories alone–with no other ability to experiment (which are commonly called batch settings), remains a challenging research frontier. Second, our setting requires that the recovered rewards be interpretable: we expect clinical experts to vet them so as to use these rewards in guiding assistive agents towards better treatment, and interpretability is also essential for general introspection of behaviors learned from data. However, none of the recent

batch methods[10–12] attempt to align the recovered reward structure with how experts may be framing their goals while administering patient treatments.

Contributions: Our work makes two core technical contributions toward addressing the challenges above.

- We develop a novel interpretable batch settings IRL algorithm based on Adversarial IRL (iAIRL) that is more robust than the current state-of-the-art in batch settings[12] both in terms of interpretability and performance.
- We also provide a theoretically-sound way of enforcing that the learned reward structure matches how experts frame their goals (in this case, a structure that splits continuous lab and vitals measurements into ranges, and places a value (reward) for each combination of these feature ranges).

Together, these contributions enable us to identify a reward structure from purely observational data that can be inspected by a clinical expert. In our application to hypotension management, our clinical expert was able to confirm what parts of the learned reward made sense–including exposing features that he may not have remembered to include–and what parts were perhaps artifacts.

Background and Related Work

Reinforcement Learning and Inverse Reinforcement Learning Formally, a *Markov Decision Process* (MDP) consists of a set of states S, actions A, a transition function $T(s'|s,a)$ that defines how states evolve over time, a reward function $R(s,a)$ that defines the immediate reward for each action, and a discount factor γ that manages the trade-off between immediate and future rewards. Solving an MDP corresponds to finding a policy $\pi^*(s,a)$ that optimizes the long-term expected reward $E[\sum_t \gamma^t r_t]$. In the (model-free) *reinforcement learning* setting, the transition function T is not given, and we must learn a policy π via interacting with the environment to collect trajectories. In the *Inverse Reinforcement Learning* (IRL) setting, an agent is given some trajectories from a policy which we are told is (near) optimal, and in turn, asked to determine what $R(s,a)$ must have been. (See Appendix A3 for a more detailed RL background.)

On-Policy IRL and Adversarial IRL The process of learning rewards from demonstrations places our work in the general category of Inverse Reinforcement Learning, which is a very broad area with typical applications in robotics and automated driving - e.g.[6,8,13]. Our work builds on Adversarial IRL[9,14], in which a discriminator tries to differentiate between the samples (s,a,s') of the expert policy and the samples generated by the optimal policy induced by our learned rewards (IRL policy). In turn, the model uses this distinction of samples to streamline the rewards towards producing more realistic samples - an iterative process whose equilibrium state is defined by the samples of the IRL policy and the expert policy being indistinguishable. Besides, Adversarial IRL[9] also applies shaping rewards to disentangle environmental dynamics from the state-only goals (rewards), an assumption that characterizes clinical settings such as the ICU. In our work, we extend this on-policy AIRL algorithm to batch settings, which mounts additional challenges of estimating transitions off-policy, restricting policies to stay close to the data during learning to prevent significant estimation bias, and more robust adversarial training procedures due to limited batch data coverage.

Fully-Batch IRL Compared to on-policy IRL, relatively very few works have considered situations in which the agent cannot interact with the environment to collect more data. Some approaches[10,12] cast the problem of estimating feature expectations—a key statistic for max-margin methods—as an off-policy evaluation problem which can potentially lead to high variance estimates. There exist other works[11] that optimize reward by setting action value function as a score metric of a multi-class classification problem, an approach that still suffers from a linear off-policy evaluation problem and the tuning of problem-specific heuristics. Besides, all these works require the reward to be linear in the chosen feature space, limiting the IRL model's expressive power. In contrast, our work allows arbitrary forms for the reward, enabling us to learn rewards that match clinician decision frameworks and avoids making off-policy feature expectation estimation all-together.

Reinforcement Learning in Intensive Care Settings There exists a growing body of work that applies RL to optimize treatments in ICU settings.[15–17] All these works take the reward as input and try to find optimal treatment policies for the ICU. Finding optimal policies from batch data could be potentially unsafe as the model could prescribe actions that

are medically ill-advised. In contrast, our goal is to learn treatment policies that mimic the expert and in the process, propose plausible reward structure (which could also be used for policy optimization in future) based on observing demonstrations of clinicians treating patients in the ICU.

Methods

We first develop a batch version of Adversarial Inverse Reinforcement Learning[9] (AIRL, see Appendix A3 for detailed background details on the AIRL algorithm). The original algorithm *requires* an ability to test arbitrary treatment policies; our *batch* version *only* requires the original observational data as input. We choose AIRL as our base not only due to its overall performance[9] compared to other IRL methods, but also because its specific reward formulation allows us to easily decompose the learned reward $R_{\theta,\phi}(s,a,s')$ into a state-only component $g_\theta(s)$ and a shaping reward. The state-only component $g_\theta(s)$ can be specified as the modeler desires—including constructing specific formulations for the interpretability of the learned rewards (specifically in our work, purely based on patient features in the ICU). Below, we first describe how we adapted the AIRL algorithm to purely observational settings following which we describe our interpretable reward formulation.

Batch-data Adversarial IRL

We present our algorithm for fully-batch AIRL in Algorithm 1. The AIRL algorithm outputs a reward $R_{\theta,\phi}(s,a,s')$ that can be decomposed into a state-dependent reward $g_\theta(s)$ and an additional shaping term.[18] The key difference between the on-policy version of Fu et al.[9] happens in step 2: in the on-policy case, it is possible to collect trajectories for some policy π by simply performing a roll-out in the environment by following the policy. Below, we describe how we *learn* a sufficiently-accurate transition model that we can use for simulating rollouts. We also apply the WGAN loss with weight clipping in step 3, which gives us additional robustness while training discriminators and apply a warm-start to ensure that our IRL agent starts and remains in a state-action basin close to our batch data's support - a major concern for fully-batch IRL.[12]

Algorithm 1 Adversarial IRL with batch data

Input: $\mathcal{D}$ (Expert Data), Transition Model $T(s'|s,a)$, T^{τ}_{max} (Max length of trajectories), γ, π, δ,N_{IRL}
Parameter: θ, ϕ. Initialize discriminator $D_{\theta,\phi}$
Output: Learned Rewards and IRL policy : $g, R_{\theta,\phi}, \pi^{IRL}$

1: **for** i={1,2,....N_{IRL}} **do**
2: Using $T(s'|s,a)$, collect trajectories for policy π: $\tau_i^{\pi} = \{s_0, a_0, ...s_{T^{\tau}_{max}}, a_{T^{\tau}_{max}}\}$ where $a_i \sim \pi(a|s_i)$
3: Using Equation 1 for discriminator loss, train the discriminator $D_{\theta,\phi}$.
4: Update shaped MDP rewards $R_{\theta,\phi}(s,a,s') \leftarrow \log D_{\theta,\phi}(s,a,s') - \log\big(1 - D_{\theta,\phi}(s,a,s')\big)$
5: Update π with respect to $R_{\theta,\phi}(s,a,s')$ using any policy optimization method (e.g. DDQN[19], TRPO[20]) : $\pi \leftarrow \pi^*(R_{\theta,\phi})$
6: **end for**
7: **return** $g_\theta, R_{\theta,\phi}$,

Learning Transition Dynamics As mentioned earlier, most standard IRL algorithms assume access to a simulator whereas in our case we have only samples (batch) of trajectories and must learn a dynamics model directly from data in order to simulate any candidate policy within the IRL iterations. Unfortunately, learning an accurate dynamics model in large state-action spaces and highly stochastic environments is challenging in general[19,21] and the fact that our observed trajectories span only a narrow part of the state-action space exacerbates our learning challenges. In this work, we forgo learning a parametric model but rather apply a transition model $T(s'|a,s)$ that selects the next state s' from the k-nearest state neighbors $s \in \mathcal{S}_s^k$ that were administered the action a in our batch data. Choosing k is domain-specific and depends mainly on the stochasticity of the expert policy within the state space. In cases where we did not find a non-trivial number of neighbors, we resort to either increasing k or finding transitions pertaining to the closest neighbors of a among the possible alternatives. Unlike parametric models, using a non-parametric model keeps our transitions s' close to the real data and prevents the model from extrapolating badly in less represented parts of the state-action space.

Warm-Start for Convergence Any transition model - parametric or non-parametric, cannot make accurate predictions far from the support of the batch data. Also, note that in the IRL learning process, one starts with some policy π to generate samples from (using our learned transition model) in Step 2 and then learns the rewards in Steps 3 and 4 to optimize this policy iteratively. If we start with some policy π that is already close to the expert policy, not only will our approximate rollouts in Step 2 be more accurate (as there is a higher likelihood of seeing those state-action decisions in the data), but the IRL procedure will also require fewer iterations to converge. Thus, before starting the AIRL loop, we first learn a (near-expert) policy using supervised learning. This data-informed choice of starting point ensures that our IRL is both feasible and accurate in complicated batch settings with limited data coverage.

Wasserstein GAN training objective for Robustness At times, when modeling complicated distributions, the traditional GAN objective[22] suffers from mode collapse and unstable gradients[23]. In our experiments, we found some evidence supporting this fact and hence, we use the discriminator training objective of a Wasserstein-GAN[24] using the weight clipping procedure to enforce K-Lipschitz continuity. This brings minor changes to any general discriminator D (parametrized by w) loss function and weight updates compared to a traditional GAN[22] as seen in Equation 1 (Refer to Arjovsky et al.[24] for more details).

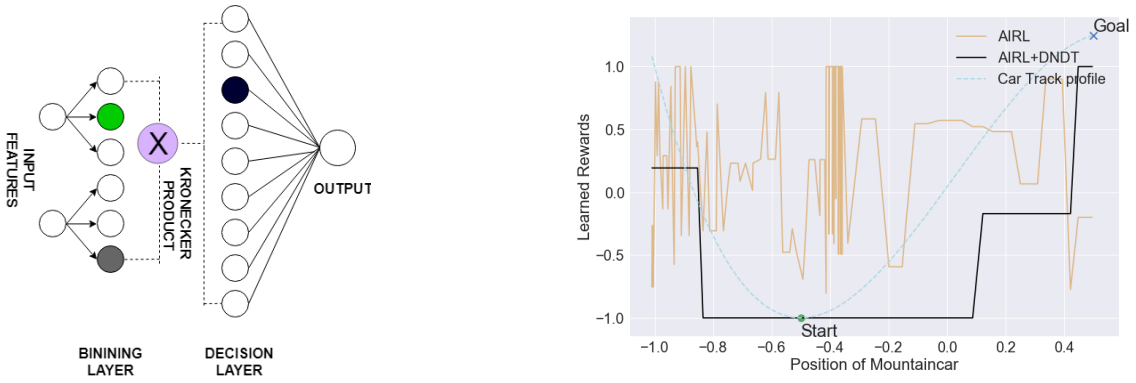
$$\begin{aligned} \nabla_w \mathcal{L}(D_w) &= \frac{1}{M} \nabla_w \left[\sum_{i=1}^{M} D_w \Big((s,a,s')^{(i)} \sim \mathcal{D} \Big) - \sum_{i=1}^{M} D_w \Big((s,a,s')^{(i)} \sim \tau^{\pi} \Big) \right] \\ w^{new} &= w^{old} + \text{RMSPROP}(w, \nabla_w \mathcal{L}(D_w)) \\ w &= \text{CLIP}(w, -c, c) \end{aligned} \tag{1}$$

We also train the discriminator for multiple epochs and take it closer to optimality in the earlier training phases as it helps stabilize the overall learning. Also, note that good training of Wasserstein distance based discriminators require the usage of non-momentum based optimizers such as RMSProp. Overall with the use of a transition model, warm-start and a Wasserstein Distance based discriminator, we are able to extend AIRL to batch settings reasonably even in environments with complicated dynamics such as the ICU.

Interpretable Reward Networks

The AIRL framework allows us to decompose the learned reward $R_{\theta,\phi}(s,a,s')$ into two terms: a *true* reward $g_\theta(s)$ and a shaping reward. Shaping rewards [18] refers to the process of modifying rewards from one form to another while keeping the optimal policy the same. Shaping was originally introduced to speed up the optimization process in traditional RL by incorporating some domain expertise into the reward formulation. Following[9], we use the idea of reward shaping in this work to transform a non-interpretable $R_{\theta,\phi}(s,a,s')$ into an interpretable $g_\theta(s)$. We do so by forcing an interpretable form on $g_\theta(s)$ and letting the shaping term account for the effects of the environment dynamics on the rewards.

Interpretable $g_\theta(s)$ via Neural Networks that Mimic Tree-Like Mechanisms Interviewing intensivists, we found that they tend to think about a patient's sickness or wellness in discrete terms—for example, a blood pressure value may be acceptable or concerning—and these discrete settings define their goals. However, such discrete structures are not easy to incorporate into gradient-based learning architectures for end-to-end learning. In this work, we build on a novel architecture, the Deep Neural Decision Tree (DNDT) introduced by Yang et al.,[25] which learns a discrete split structure on each feature and represents different possible combination of these splits in order to learn response prediction scores (rewards in our case) jointly via a single backpropagation step. This model also allows an arbitrary number of splits without any restriction on the structure of the tree. We briefly explain the working mechanism of DNDT. Assuming we can bin each of our features into a pre-specified number of discrete bins, we intend to set up our interpretable reward $g_{\theta(s)}$ using a neural network architecture that learns $g_{\theta(s)}$ based on combinations of these feature bins (e.g. high BP, low urine $\rightarrow$ low rewards). It is important to note that the binning boundaries are learned by the model and we need to specify only the number of bins along each feature. Figure 1a demonstrates the architecture of DNDT for our interpretable reward $g_{\theta(s)}$. The network essentially has two hidden layers - a binning layer that learns soft one-hot encodings (via activations) for each feature's value with respect to its corresponding binning boundaries and a decision layer that encodes every possible binary combination of these activations i.e. each possible combination



(a) DNDT Architecture **(b)** Mountain Car Example

Figure 1: (a) DNDT architecture for state-only reward approximator $g_\theta(s)$ assuming two features and three bins(two boundaries). The binning layer creates a soft one-hot encoding (colored nodes in the binning layer) of the bin. The decision layer is a result of Kronecker product of all nodes' activated values in the binning layer, producing one unique activation for each possible combination of feature bins. The entire model is differentiable via backpropagation. (b) Rewards learned by the IRL models for MountainCar-v0 across the position feature, where the goal is to reach the goal on right. AIRL learns a non-smooth and less interpretable rewards (yellow) while iAIRL(AIRL+DNDT) learns more interpretable rewards (black) that motivate the agent to swing towards the extremes.

of the feature bins activates exactly one node in the decision layer. The weights mapping the decision layer and the output node learn the value (in regression settings; or classification score in classification settings) of the response variable (in our case, $g_{\theta(s)}$) for different possible decision boundaries.

Towards creating sparse reward descriptions for better interpretability If we define the number of input features as D_{input} and the number of binning boundaries on each feature as N_f ($N_f + 1$ bins), the number of nodes in the decision layer turns out to be $(N_f + 1)^{D_{input}}$ and thus DNDTs do not scale well to a large number of features[25]. It is important to note that the number of non-trivial weights in the decision layer is the number of different decision boundary combinations we need to interpret. Also, it is obvious that we can understand clinicians' motivations more clearly if there are fewer decision boundaries with strong signals to interpret rather than an exploding number of decisions. Hence, imposing some kind of sparsity is essential on these weights and we tackle this problem by applying a L-1 weight regularization for the weights of the last layer[26]. If the usual loss of any network whose N weights are $w_{\{1,\dots N\}}$ is $\mathcal{L}_w = \phi(w)$, a L1-regularized variant of the same model has the loss function $\mathcal{L}_w^{L1} = \phi(w) + \lambda \sum_{i=1}^{N} |w_i|$, where λ is the hyperparameter which signifies the regularization strength. This enforced sparsity means that we need to form meaningful interpretations only about those rewards that have significant positive or negative values across the spectrum of decision - i.e. *most valuable or extreme decisions*.

Demonstration on Synthetic Examples

Before experimenting with our model on the ICU data, we tested our approach on some basic IRL benchmarks—a gridworld and mountaincar—where we had knowledge of the ground truth reward structure (unavailable in the clinical setting; we emphasize the ground truth was only used for validation at the end and not during training). Due to space constraints, the details of the experiments are included in Appendix A2 (where we compare our methods with other batch IRL baselines). However, we illustrate the value of our approach in figure 1b, where the agent's goal is to reach the far right by swinging higher and higher. *Our method iAIRL (batch AIRL + DNDT based state-only rewards) finds a simple, discrete reward structure (very close to the one that generated the ground-truth demonstrations) that recovers the optimal policy, unlike standard AIRL which learns a highly non-smooth, non-interpretable reward structure.*

Hypotension Management in the ICU: Cohort, Modeling, and Experimental-Setup

In this section, we provide the details of how we extracted our cohort and applied the general ideas developed in this work to the task of understanding the clinician motivations behind standard interventions for hypotension management

in the ICU.

Cohort, Data Collection, and Features Our cohort was drawn from the public MIMIC-III version 1.4 database,[27] which contains trajectories from patients treated at Beth Israel Deaconess Medical Center between 2001 and 2012, and our pre-processing closely follows the work of Ghassemi et al.[28] Our cohort contained adult patients over the age of 15 and we excluded patients with less than 6 hours or more than 360 hours of data. Applying these filters gave us a total of 16,502 patients. For each patient, we extracted four arrays of features

- **Static Features ($z^k \in R^{11}$)** : age, weight on admission, SOFA, OASIS and SAPS scores at ICU admission, indicator variables for gender, ethnicity, emergency, admission urgency and hours from admit to the ICU. These observations remain constant for a patient throughout his/her stay in the ICU.
- **Clinical Observations at time-step i ($\mathbf{x^k_{[1,2...T]}}; x_i^k \in R^{18}$)**: bicarbonate, bun, creatinine, fio2, glucose, hct, heart rate, lactate, magnesium, meanbp, platelets, potassium, sodium, spo2, spontaneousrr, temp, urine, wbc. The choice of these features are drawn from Ghassemi at al.[28], which used the same set of features to predict sepsis onset in patients.
- **Indicator Flags for Observations at time-step i ($\mathbf{o^k_{[1,2,.....T]}}; o_i^k \in \{0,1\}^{18}$)**: An indicator variable that denotes whether the observation actually changed from bin $i-1$ to i, i.e. whether a new measurement for the concerned feature was taken in the last 30 minutes (our time step size) in the ICU.
- **Interventions at time-step i ($\mathbf{y^k_{[1....T]}}; y_i^k \in R^2$)**: normalized vasopressor dosages and fluid boluses. These interventions will be referred to as vasopressors/vaso and fluids in the rest of the text. It is well-known that both these interventions are relevant towards managing blood pressure in the ICU[15].

Each of the time-series variables was aggregated into 30 minute time intervals with the mean or sum being recorded (as appropriate) when several data points were present in one window. All the features were then standardized using z-scores. Finally, we split the dataset into training (80%) and holdout (20%) sets by patient. *Further details on the construction of an MDP from the dataset and the specific model architecture developed for warm-start with supervised learning are discussed in Appendix A1.*

Baselines and Training Incorporating a DNDT based architecture into AIRL rewards creates a potential trade-off between reward function expressiveness and interpretability and it is important to understand the extent of this trade-off in the ICU data. Thus, we consider two alternatives (both developed in this paper) in our experiments (Model, training hyperparameters are given in Appendix A5.)

- **Off-policy AIRL (AIRL):** We consider our off-policy batch extension of AIRL with a feed-forward architecture for state-only reward function approximation i.e. $g_\theta(s) = \text{FEEDFORWARD}(s;\theta)$. Remember that $g_\theta(s) \approx R^*(s)$ (function approximator of ground-truth rewards) .
- **Interpretable AIRL (iAIRL):** In addition to our off-policy extension of AIRL, we use a DNDT to approximate the rewards, that is, $g_\theta(s) = \text{DNDT}(\hat{s};\theta)$. Since the number of decision mechanisms (nodes in last layer) grows exponentially with the number of layers, we only choose a subset of features (in consultation with clinicians):*Mean BP, Heart Rate, Urine Output, Creatinine and Lactate*.

Results on Understanding Hypotension Management in the ICU

iAIRL sacrifices little in action matching performance compared to AIRL. To evaluate the performance of our model, the first metric we consider is action matching, which indicates how closely our model's suggested actions match that of the expert. Remember that we have 25 action bins in total (5 for each intervention) and the overall action matching is calculated as the fraction of the number of observations in the hold-out set in which the model's predicted action (an action that maximizes the learned Q-values) matches the action taken by the expert. The overall action matching for our warm-start imitation policy is around 73.69%. Using this policy to kick start the IRL, our batch AIRL model learns a reward function with an action matching of 71.08% while our interpretable iAIRL learns a reward function with an overall action matching of 64.43%.

Both models largely learn the same or similar treatments suggested by clinicians. Since the action distribution in our data is hugely imbalanced, studying just the overall action matching via accuracy does not provide the full picture. Figure 2 shows the confusion matrices for both our batch AIRL and our interpretable batch iAIRL. We see that both the models have strong diagonal terms indicating reasonable prediction accuracy for each class. In cases where the predictions don't match, the prediction vs. ground truth are close: the model predicts either the adjacent action bin (same fluid action bin, ± 1 vasopressor action bin) or an action bin 5 steps away (same vasopressor bin, ± 1 fluid action bin). The main mismatch occurs for higher dosages of vasopressors and fluids. We suspect it is because the features we have chosen for DNDT and the decision boundaries learned may not be expressive enough to handle extreme scenarios in which the clinicians could be taking their decisions based on myriad other factors. That said, the confusion matrix suggests that our matching is of sufficient quality for most common scenarios.

iAIRL provides a significantly more parsimonious reward description than AIRL. For initial quantitative validation of our claim that iAIRL learns more interpretable rewards than AIRL, we considered whether AIRL learned similarly parsimonious rewards as iAIRL. To do so, we fit a decision tree restricted to 243 (3^5, the same number of nodes in the last DNDT layer for a fair comparison) leaf nodes to the rewards learned by AIRL. The decision tree's regression RMSE of 0.8 on rewards normalized to fall between $[-1, 1]$ suggests poor learning of the reward function by the decision tree. This implies that the learned AIRL rewards are more complex and non-smooth compared to the iAIRL rewards as they couldn't be fit well into simple decision rules. This further reinforces the observation we learned from Figure 1b that iAIRL learns discrete, sparse and neatly interpretable rewards compared to AIRL which learns non-interpretable rewards.

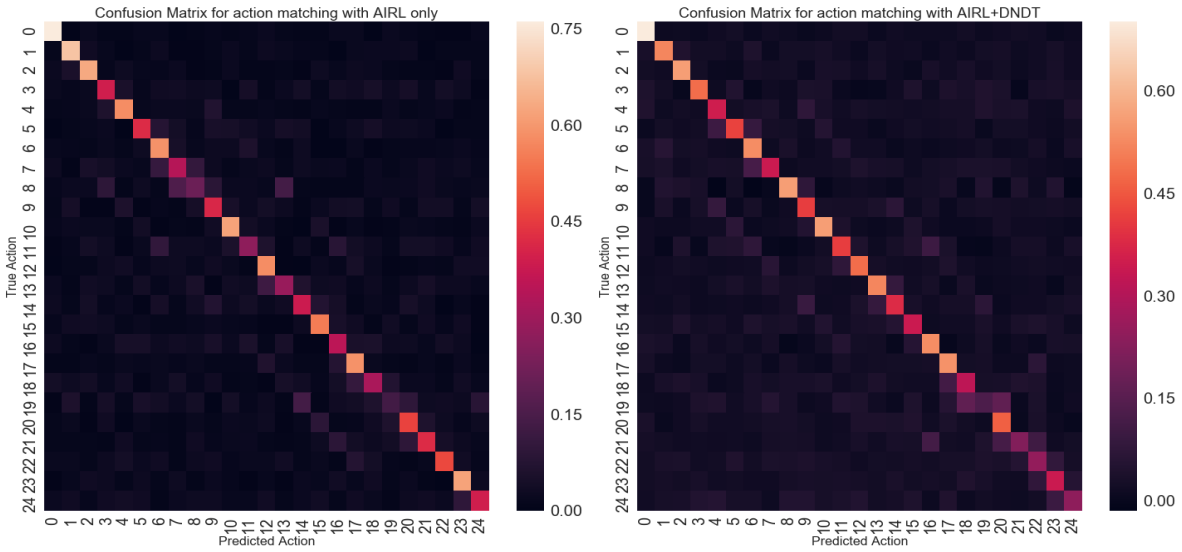


Figure 2: Confusion Matrices for action matching of AIRL (left) and iAIRL (right). The overall action matching of AIRL is about 71% and that of iAIRL is about 65% when both are warm-started with an imitation policy with action matching 73%. We see strong diagonal matrix meaning acceptable action matching and cases of wrong predictions have a higher probability of being either the adjacent action bin (same fluid bin, +/- 1 vaso bin) or action bins 5 steps away (same vaso bin, +/- 1 fluid bin). Remember that the overall action bin = fluid bin * 5 + vaso bin.

Clinical Perspective: Learned Rewards match clinician goals. Figure 3 shows the rewards along with the feature bins learned by our iAIRL model. From the two extremes of rewards, we can observe certain general trends: The model penalizes low BP, low urine, high heart rate and on the other hand, rewards stable to high BP, urine outputs and stable heart rates, lactate levels. Because of our weight regularization, sparsity was enforced and the model placed non-trivial rewards over only about 73 out of the 243 possible weights. There is no strong signal in creatinine possibly because our interventions don't directly control creatinine levels.

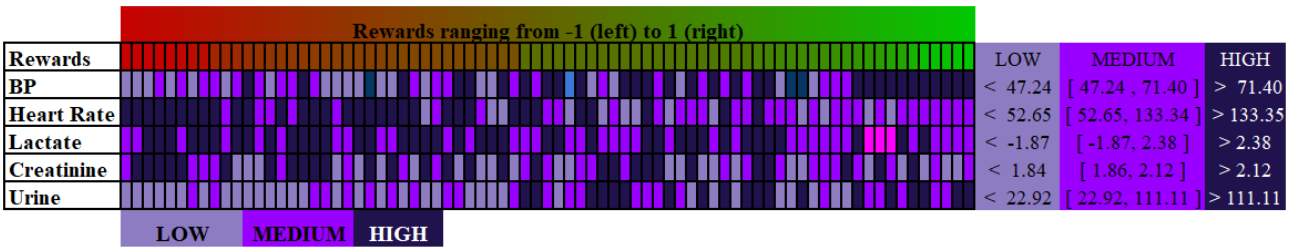


Figure 3: A heatmap showing normalized rewards learned by the iAIRL model with respect to the 5 chosen features and their corresponding value ranges. The high, medium and low ranges of features have been assigned distinct colors here and the value ranges (Low, Medium and High) can be seen from the last 3 columns. The first reward (first column) can be read as extremely low (bad) reward for low BP and Urine, high Heart Rate and Medium Lactate and Creatinine.

These learned reward patterns were shared with a practicing intensivist to study how the low, medium and high reward decision boundaries learned by the model for each bio-marker compares to ranges that induce clinicians to intensify patient treatment. (We only shared the rewards learned from our iAIRL model because there was no clear way to summarize the complex network learned by unconstrained AIRL.) The higher range for blood pressure made sense; the lower range (around 47) was lower than he expected but potentially made sense as a situation to avoid (that is, one might act on a blood pressure of 55 to avoid a blood pressure of 47). Similarly, the range for lower urine matched his cut-offs for actions. Where the ranges differed for the remaining variables, his hypothesis was that this reflected the fact that blood pressure and urine were the main targets for vasopressors and fluids, and hence it could be reasonable to see weaker signals on the other set of chosen features.

He noted a similar trend in the actual reward assignments: our algorithm recovered a function that gave higher rewards to higher blood pressures and higher urine outputs, which he confirmed is the goal of vasopressor and fluid administration. The effect of heart rate, lactate, and creatinine on the reward also had a trend that was consistent with his notions of better health, and it was interesting that these trends were discovered even though the interventions are not directly targeting those measures; it suggests that clinician behavior may be implicitly trying to protect those aspects even though they are not the goals they are shooting for. *Overall, being able to see the learned IRL rewards confirmed that intensivist behavior matched the goals that he believed he (and his colleagues) were aiming for and helped increase his confidence in choosing a reward for RL tasks*.

While both ends of the reward spectrum—the really high rewards and the really low rewards—matched his intuition, the learned structure had imperfections for smaller / intermediate reward values. For example, there were times when the model penalized creatinine levels over blood pressure levels, which ran counter to the intuition of our critical care expert—especially as he noted that there was little that these interventions (vasopressors and fluids) could do to adjust creatinine levels compared to their impact on a more obvious control feature such as blood pressure. Since the action-matching performance of our algorithm, while appreciably high for fully-batch IRL, was still far from perfect, these inconsistencies might simply be due to the fact that our reward function does not perfectly induce the expert behavior and only approximates it reasonably; improving the IRL based on these kinds of expert inputs and the model, data engineering suggestions discussed throughout this work are definitely interesting avenues for future work.

Discussion

In the previous section, we discussed many of the qualities of our batch AIRL and iAIRL algorithms in the context of recovering rewards associated with hypotension management in the ICU. We provide an in-depth technical discussion on certain key implementation aspects of the model in Appendix A4 and focus on the broader implications here. Being able to extract possible reward structures from observational data has important applications, both for the design of assistive agents—who need very precise descriptions of what they are to assist in optimizing—as well as general introspection. While our work makes significant progress toward this goal, we do have some fundamental limitations. One is that rewards are inherently non-identifiable (imagine adding 1 to every reward; the decisions will not change). Our approach uses that non-identifiability as a *feature* to find an interpretable reward among many potential rewards. However, there may be other interpretable rewards that recover expert policies. That said, we noticed that the feature

bins combinations constituting extreme rewards (close to +1 or -1) were consistently similar across different runs of our model while we observed variation in the relative ordering of the feature combinations signifying small/intermediate rewards, implying that our model learns the decisions at the extremes of the reward spectrum robustly. Moreover, the work here is intended to provide clinicians a starting point for defining and quantifying rewards, and that human inspection will identify if a particular structure is not sensible.

Another key consideration is that all of our work is predicated on some definition of patient state. In this work, we assume that the most recent raw observations are sufficient to capture the patient's state. Future works could benefit from more sophisticated latent space encodings. We also acknowledge the fact that several clinical decisions are made using factors that are not part of our data, and that the "clinician policy" that we see is, in fact, a mixture of decisions made by many people. Our approach can be applied to any set of input, so advances in capturing better notions of patient state, new variables, and specific decision-makers could be included; we again emphasize that the ability to validate with experts that our interpretability provides protects us from catastrophic outputs.

Conclusion

Understanding motivations behind ICU interventions purely from logs of patient data is an important and challenging task. In this work, we developed a robust, batch IRL algorithm (iAIRL) to learn these motivations in a clinically interpretable fashion. With these enhancements, we learned a parsimonious description of the motivations behind vasopressor and IV fluids prescription for hypotension management in the ICU from MIMIC-III data and also found that the learned motivations are largely consistent with clinical practice. Apart from refining the algorithms, future work can use this model of learning interpretable rewards for building reliable assistive agents, transferring the learned motivations to similar clinical treatment tasks and further data-driven quantification of the differences between patients.

Acknowledgments

SS and FDV acknowledge support from the Sloan Foundation for this work.

Appendix - Link to the appendix for more technical and experimental details: https://github.com/dtak/i-airl

References

1. Jessica Bunin. Diagnosis and management of hypotension and shock in the intensive care unit. https://www.cs.amedd.army.mil/FileDownloadpublic.aspx?docid=77d4b6b0-bfa2-4293-9509-375d2998f116.

2. Ashish K. Khanna. Defending a mean arterial pressure in the intensive care unit: Are we there yet? *Annals of Intensive Care*, 2018.

3. Gunnar Gampar, Christof Havel, Jasmin Arrich, Heidrun Losert, Nathan L Pace, Marcus Mllner, and Harald Herkner. Vasopressors for hypotensive shock. *Cochrane Database of Systematic Reviews*, (2), 2016.

4. Jan Leike, Miljan Martic, Victoria Krakovna, Pedro A Ortega, Tom Everitt, Andrew Lefrancq, Laurent Orseau, and Shane Legg. Ai safety gridworlds. 2017.

5. Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete Problems in AI Safety. *arXiv e-prints*, page arXiv:1606.06565, 2016.

6. 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. ACM, 2004.

7. Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, pages 4565–4573, 2016.

8. Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *AAAI*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.

9. Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. In *arXiv preprint arXiv:1710.11248*, 2017.

10. Edouard Klein, Matthieu Geist, and Olivier Pietquin. In *European Workshop on Reinforcement Learning*, pages 285–296. Springer, 2011.

11. Edouard Klein, Bilal Piot, Matthieu Geist, and Olivier Pietquin. A cascaded supervised learning approach to inverse reinforcement learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 1–16. Springer, 2013.

12. Donghun Lee, Srivatsan Srinivasan, and Finale Doshi-Velez. Truly Batch Apprenticeship Learning with Deep Successor Features. *arXiv e-prints*, page arXiv:1903.10077, Mar 2019.

13. Nathan D Ratliff, J Andrew Bagnell, and Martin A Zinkevich. Maximum margin planning. In *Proceedings of the 23rd international conference on Machine learning*, pages 729–736. ACM, 2006.

14. Chelsea Finn, Paul Christiano, Pieter Abbeel, and Sergey Levine. A Connection between Generative Adversarial Networks, Inverse Reinforcement Learning, and Energy-Based Models. *arXiv e-prints*, page arXiv:1611.03852, 2016.

15. Aniruddh Raghu, Matthieu Komorowski, Leo Celi Ahmed, Imran, Peter Szolovits, and Marzyeh Ghassemi. Deep reinforcement learning for sepsis treatment. 2017.

16. Xuefeng Peng, Yi Ding, David Wihl, Omer Gottesman, Matthieu Komorowski, Li-wei H. Lehman, Andrew Ross, Aldo Faisal, and Finale Doshi-Velez. Improving Sepsis Treatment Strategies by Combining Deep and Kernel-Based Reinforcement Learning. *arXiv e-prints*, page arXiv:1901.04670, 2019.

17. Niranjani Prasad, Li-Fang Cheng, Corey Chivers, Michael Draugelis, and Barbara E Engelhardt. A Reinforcement Learning Approach to Weaning of Mechanical Ventilation in Intensive Care Units. *arXiv e-prints*, page arXiv:1704.06300, 2017.

18. Andrew Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *International Conference on Machine Learning*, 1999.

19. Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. volume 518, page 529. Nature Publishing Group, 2015.

20. John Schulman, Sergei Levine, Phillip Moritz, and Michael and Jordan. Trust region policy optimization. 2015.

21. Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. 2015.

22. Ian GoodFellow, Mehdi Pouget-Abadie, Jean andMirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Neural Information Processing Systems*, 2014.

23. Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Chen Xi. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, 2016.

24. Martin Arjovsky, Soumith Chintala, , and Leon Bottou. Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 214–223, 2017.

25. Yonxin Yang, Irene Garcia Morillo, and Timothy M Hospedales. Deep neural decision trees. In *2018 ICML Workshop on Human Interpretability in Machine Learning*, 2018.

26. Andrew Ng. Feature selection, l1 vs. l2 regularization, and rotational invariance. In *21st International Conference on Machine Learning*, pages 49–58, 2004.

27. Alistair Johnson, Tom Pollard, Shen Lu, Li-Wei Lehmann, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, and Anthony Celi. Mimic-iii, a freely accessible critical care database. *Scientific Data*, 3, 2016.

28. Marzyeh Ghassemi, Michael Wu, Peter Szolovits, and Finale Doshi-Velez. Predicting intervention onset in the icu with switching state space models. pages 82–91, 2017.