Truly Batch Apprenticeship Learning with Deep Successor Features [*]

Donghun Lee, **Srivatsan Srinivasan** and **Finale Doshi-Velez**
SEAS, Harvard University
{srivatsansrinivasan, donghunlee}@g.harvard.edu, finale@seas.harvard.edu,

Abstract

We introduce a novel Inverse Reinforcement Learning (IRL) method for *batch* settings where *only* expert demonstrations are given and no interaction with the environment is allowed. Such settings are common in health care, finance and education where environmental dynamics are unknown and no reliable simulator exists. Unlike existing IRL methods, our method does not require on-policy roll-outs or assume access to non-expert data. We introduce a robust epde off-policy estimator of feature expectations of any policy and also propose an IRL warm-start strategy that jointly learns a near-expert initial policy and an expressive feature representation directly from data, both of which together render batch IRL feasible. We demonstrate our model's superior performance in batch settings with both classical control tasks and a real-world clinical task of sepsis management in the ICU.

1 Introduction

Reward design is a key challenge in Reinforcement Learning (RL). Manually identifying an appropriate reward function is often difficult, and poorly specified rewards can pose safety risks [Leike *et al.*, 2017]. Apprenticeship learning is the process of learning to act from expert demonstrations. One type of apprenticeship learning, Imitation Learning (IL) (e.g. [Ross *et al.*, 2011; Ho and Ermon, 2016]), directly learns a policy from demonstrations. In contrast, Inverse Reinforcement Learning (IRL) aims to recover the expert policy by learning a reward function, which, when solved, induces policies that are similar to the expert. While there exist theoretical connections between IL and IRL [Piot *et al.*, 2017], one may be preferred depending on the application. In particular, directly learning a policy (imitation) can be brittle in cases of long-horizon planning, and environments with strong covariate or dynamics shifts [Piot *et al.*, 2013; Fu *et al.*, 2017]. Besides addressing these issues, the learned reward function in IRL can also be used to identify expert motivations underlying the actions: rewards describe what the expert *wishes* to achieve, rather than simply what they are *reacting to*, enabling agents to generalize better with the knowledge of these "intentions" in related environments.

In this work, we focus on IRL in *batch* settings: we must infer a reward function that induces the expert policy, given *only* a fixed set of expert demonstrations. No simulator exists and environment dynamics (transition model) is unknown. Performing analyses on batch data often ends up being the only reasonable alternative in domains such as healthcare, finance, education, or industrial engineering where precollected logs of expert behavior are relatively plentiful but new data acquisition or a policy roll-out is costly and risky.

Many existing IRL algorithms (e.g. [Abbeel and Ng, 2004; Ratliff *et al.*, 2006]) use the *feature expectations* of a policy as the proxy quantity that measures the similarity between expert policy and an arbitrary policy that IRL proposes. If a simulator is available, feature expectations can be computed by taking sample means across on-policy rollouts [Abbeel and Ng, 2004]. However, new rollouts are not possible in batch settings. To estimate feature expectations in batch settings, a few IRL algorithms exist that either use a linear estimator [Klein *et al.*, 2012] or bypass the estimation by assuming the existence of additional data from an arbitrary policy [Boularias *et al.*, 2011]. Often, linear estimators do not possess the representational power necessary to model real-world tasks, while the ability to access additional data is overly restrictive. Along with an off-policy estimator for feature expectations, successful batch IRL also requires methods to engineer expressive feature spaces and manage computational complexity. Our work addresses all these batch IRL challenges.

Contributions Specifically, our work makes two key contributions that together enable a batch version of max-margin IRL that empirically scales well across classical control and real-world tasks, *using only expert demonstrations* and no additional inputs whatsoever. Firstly, we propose the *Deep Successor Feature Network* (**DSFN**), an off policy feature expectations estimator that replaces the on-policy roll-outs in standard IRL algorithms. Secondly, to mitigate a potential high bias caused by data support mismatch in DSFN, we propose *Transition Regularized Imitation Learning* (**TRIL**), which warm-starts the IRL loop with a near-expert initial policy and simultaneously provides an expressive feature space for better IRL performance directly learned from the batch data without any manual feature engineering.

[*]Supplementary material (**Appendix**) to the main text can be found here : https://tinyurl.com/ijcai-dsfn-supplement

2 Related Work

This paper focuses on *batch* IRL, where the IRL evaluation metric (for evaluating the IRL policy's closeness with respect to expert) is feature expectations [Abbeel and Ng, 2004]. Our goal is to learn a reward function for imitating and interpreting the expert policy under unknown dynamics with *expert demonstrations alone*. This batch definition differs from other settings such as RE-IRL [Boularias *et al.*, 2011] that requires access to non-expert data from an arbitrary policy for importance sampling or DM-IRL [Burchfiel *et al.*, 2016] that needs additional demonstration queries made to the expert. Finally, while we adopt the max-margin IRL framework [Abbeel and Ng, 2004] in this work, our key contributions—TRIL, DSFN— are generic and can be applied across a broad class of IRL methods (e.g. probabilistic IRL methods such as [Ziebart *et al.*, 2008]).

Batch Feature-Expectations IRL. A major challenge in batch IRL is estimating feature expectations. [Klein *et al.*, 2011] made an important observation that this problem is similar to off-policy policy evaluation and proposed an estimator based on LSTD-Q [Lagoudakis and Parr, 2003]; this estimator suffers from the high sensitivity and weak representational power of linear systems. To address the computational complexity aspect of IRL, [Klein *et al.*, 2012] proposed SCIRL that takes a supervised learning (classification) approach where feature expectations are estimated by a simple sample mean. SCIRL assumes a deterministic expert and relies on the heuristic assumption that feature expectations of non-expert actions can be approximated by a multiplicative constant factor of the feature expectations for expert actions (requires tuning for every domain). In contrast, our method is scalable across domains with little tuning required and only requires the expert demonstrations without any additional assumptions. Moreover, SCIRL uses a linear classifier while our method admits any parametric model (e.g. neural nets). Our flexible neural net parametrization for estimating feature expectations in DSFN (similar in spirit to [Kulkarni *et al.*, 2016]'s deep successor features for value functions in RL) along with a TRIL warm-start enables our model to scale the best among alternatives, to real-world tasks.

Warm-starting batch IRL. Computational complexity and sensitivity to features are unavoidable *practical* challenges in IRL, more so in batch settings in which off-policy evaluations are required to compute feature expectations. Standard IRL algorithms [Abbeel and Ng, 2004; Ratliff *et al.*, 2006] assume manually engineered features and initialize with a simple (random, sample mean) policy. However, in batch settings, initializing with such simple policies is not helpful as OPE on such policies can produce biased feature expectations estimates when expert actions may differ significantly from the policy itself. Our novel TRIL network warm-starts batch IRL to obtain a near-expert initial policy via supervised classification (similar ideas include [Klein *et al.*, 2013; Piot *et al.*, 2014]) regularized by the joint learning of dynamics, and simultaneously performs deep feature representation learning with the network's shared layers (e.g. [Song *et al.*, 2016]).

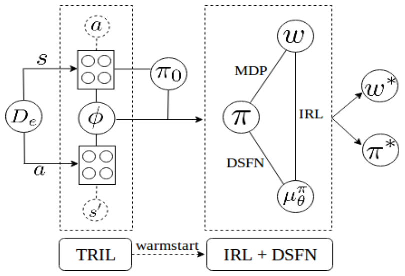

Figure 1: TRIL+DSFN: DSFN provides the crucial off-policy estimate of the feature expectations of IRL policies in batch settings using the three inputs D_e, π_0, ϕ provided by TRIL. TRIL warm-starts the IRL loop by learning an initial policy π_0(multi-class classification regularized by next state prediction s') and generates a feature map ϕ from the shared hidden layers in the process.

3 Background

Markov Decision Process (MDP). An MDP is a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{T}_0, R, \gamma)$ where $s \in \mathcal{S}$ states (continuous, in this work), $a \in \mathcal{A}$ actions (discrete, in this work), $\mathcal{T}(s'|s,a), \mathcal{T}_0$ the transition probabilities and the initial state distribution respectively, $R(s,a)$ 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. The state-value function is defined as $V^\pi(s) = \mathbb{E}_\pi[\sum_{t=0}^\infty \gamma^t R(s_t,a_t)|s_0 = s]$. The action-value function is defined as $Q^\pi(s,a) = R(s,a) + \mathbb{E}_{s_1,a_1\sim\pi,\ldots}[\sum_{t=1}^\infty \gamma^t R(s_t,a_t)]$. The optimal policy under an MDP is given by $\pi_e = \pi^* = \arg\max_\pi V^\pi(s)$ $(\forall s \in S)$.

Batch Max-Margin IRL. We assume the existence of an expert policy π_e that is optimal under some unknown, linear reward function of the form $R(s,a) = \mathbf{w} \cdot \phi$ for some reward weights $\mathbf{w} \in \mathbb{R}^d$ and some pre-defined feature map $\phi(s,a) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$. Both quantities are bounded i.e. $||w||_2 \leq 1, ||\phi(\cdot)||_2 \leq 1$. Let $\mathcal{D}_e = \{(s_0, a_0 \sim \pi_e, \ldots, s_T)\}$ be a set of N expert trajectories sampled according to π_e. Unlike traditional IRL methods, we assume we *cannot* sample trajectories from any other policy and that $\mathcal{T}$ is unknown (common in real-world batch settings). The state-action *feature expectations* of a policy $\mu^\pi(s,a) \in \mathbb{R}^d$ and the overall feature expectations $\mu^\pi \in \mathbb{R}^d$ of a policy are defined as

$$\begin{aligned} \mu^\pi(s,a) &= \phi(s,a) + \mathbb{E}_{s_1,a_1\sim\pi,\ldots}\Big[\sum_{t=1}^{\infty} \gamma^t \phi(s_t,a_t)\Big] \\ \mu^\pi &= \mathbb{E}_{s_0\sim\mathcal{T}_0}[\mu^\pi(s_0,\pi(s_0))] \end{aligned} \tag{1}$$

Conceptually, the feature expectation μ^π represents the (expected, discounted) amount of the feature accumulated while acting under a policy π. All IRL algorithms require a way to measure the similarity between the expert policy and candidate policies, and max-margin IRL uses $\|\mu^{\pi_e} - \mu^\pi\|_2$ to measure the same. Specifically, [Abbeel and Ng, 2004] showed that convergence in feature expectations implies convergence in the expected value function between two policies.

4 Method: Batch Feature-Expectations IRL

In this section, we present our model—a batch version of max-margin IRL [Abbeel and Ng, 2004] that consists of two novel neural-network models: one designed to address the off-policy evaluation challenge intrinsic to batch settings and the other to warm-start the former and reduce both the biased estimations of OPE and IRL's computational complexity.

The main roadblock in batch IRL is the inability to estimate feature expectations by taking a sample mean of on-policy rollouts: $\mu^\pi \not\approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{T}\gamma^t\phi(s_t,\pi_e(s_t)))$ due to the lack of a simulator. If a candidate policy π disagrees with the expert demos such that $a_t \sim \pi_e \neq \pi(s_t)$ for some $(s_t, a_t, s_{t+1}) \in \mathcal{D}_e$, we have no way of collecting next state samples to know how the trajectory evolves from s_t using the demonstrations alone. Computing the feature expectations now becomes an Off-Policy Evaluation (OPE) problem (e.g. [Thomas and Brunskill, 2016]) whose target is μ^π instead of V^π. In the following, we develop a model-free approach for the OPE, which avoids the bias of model-based methods and the high variance of importance sampling-based methods.

4.1 Estimating μ^π via DSFN

Inspired by the LSTD-based approach [Klein *et al.*, 2011], we observe an analogous formulation between feature expectations $\mu^\pi \in \mathbb{R}^d$ and action value function $Q^\pi \in \mathbb{R}$. Note that μ^π's i-th component $\mu_i^\pi(s,a) = \phi_i(s,a) + \mathbb{E}_{s_1,a_1\sim\pi,\ldots}[\sum_{t=1}^{\infty}\gamma^t\phi_i(s_t,a_t)] \in \mathbb{R}$, while $Q^\pi(s,a) = R(s,a) + \mathbb{E}_{s_1,a_1\sim\pi,\ldots}\left[\sum_{t=1}^{\infty}\gamma^t R(s_t,a_t)\right] \in \mathbb{R}$. Thus, learning μ^π can be setup as a system of d Temporal Difference (TD) learning problems [Sutton *et al.*, 1998] which we can solve by assuming that each feature dimension μ_i is *independently* associated with its own reward function ϕ_i. Leveraging this similarity, we derive our method, Deep Successor Feature Networks (DSFN), in a way analogous to deep Q-learning algorithms such as DQN [Mnih *et al.*, 2015]. Let $\mu^\pi(s,a;\theta)$ be the feature expectations estimator whose parameters θ (feed-forward neural network) are to be learned from expert demonstrations $\mathcal{D}_e$. The model is trained using gradient descent on MSE loss $\mathcal{L}(\theta,\pi)$ with respect to the TD targets that follow from Bellman equation [Sutton *et al.*, 1998]. Given π, we set the TD targets $\mu_+^\pi\ \ \forall\ (s_i,a_i,s_i') \in D_e,\ |D_e| = N$ as follows:

$$\mu_+^\pi(\mathbf{s},\mathbf{a}) = \begin{cases}\phi(s,a) & \text{if } s' = s_T\\ \phi(s,a) + \gamma\mathbb{E}_{s'}[\mu^\pi(s',\pi(s');\theta)] & \text{otherwise}\end{cases}$$
$$\mathcal{L}(\theta,\pi) \approx \frac{1}{2N}\sum_{i=1}^{N}||\mu^\pi(s_i,a_i;\theta) - \mu_+^\pi(\mathbf{s_i},\mathbf{a_i})||^2$$
$$\nabla_\theta\mathcal{L}(\theta,\pi) \approx \frac{1}{N}\sum_{i=1}^{N}\big[\ \big(\mu^\pi(s_i,a_i;\theta) - \mu_+^\pi(\mathbf{s_i},\mathbf{a_i})\big)\cdot \nabla\mu^\pi(s_i,a_i;\theta)\ \big] \tag{2}$$

Algorithm (1) presents a batch version of max-margin IRL using DSFN. DSFN's training procedure (Appendix Section 2.1) is similar to DQN with a subtle difference: DSFN evaluates a policy while DQN optimizes and hence it has a max operator in its target. While typical max-margin IRL returns a set of policies and one must choose after convergence (no guarantee that the last solution is the best), we observed empirically in batch problems that using *validation error of the IRL policy's actions* compared to expert actions was a useful convergence criterion, particularly given that μ^π computed via OPE could be biased. We take a validation set from D_e and terminate the algorithm when validation error $\mathcal{L}_{\text{val}}$ goes below some $\delta > 0$. DSFN's parametric details are given in Appendix Table 2.

Algorithm 1 Batch Max-Margin IRL

```
Input: $\pi_0$ (Init. Policy), $\phi$ (Feature Map), $\mathcal{D}_e$ (Demo.), $n$ (max. iteration) $\epsilon$, $\delta$ (convergence thresholds)
Parameter: $\mathbf{w}$, $\theta$, Output: $R_w$
 1: Estimate $\mu^{\pi_e}$ by taking a sample mean over $\mathcal{D}_e$
 2: for $i = 0 : n$ do
 3:     Estimate $\mu^\pi_{(i)}$ with $\mu(\theta)$ DSFN and $\mathcal{D}_e$
 4:     Obtain a reward function $\mathbf{w}_{(i)}$ by solving QP:
            $w_{(i)} = \min_{\mathbf{w}\in\mathbb{R}^d, \|\mathbf{w}\|_2\le 1} \|\mathbf{w}\|_2^2$
            s.t. $\mathbf{w}\cdot\mu^\pi_j \le \mathbf{w}\cdot\mu^{\pi_e} + 1, \forall j \in [i-1]$
 5:     Obtain $\pi_{(i+1)}$ by solving MDP with $R_w = \langle \mathbf{w}_{(i)}, \phi\rangle$.
 6:     if $\left\|\mu^{\pi_e} - \mu^\pi_{(i)}\right\|_2 < \epsilon$ or $\mathcal{L}_{\text{val}}(\pi_{(i+1)}) < \delta$ then
 7:         Terminate the loop.
 8:     end if
 9: end for
10: return $R_w = \langle \mathbf{w}_{(i)}, \phi\rangle$
```

Necessity of warm-starting. As with any off-policy evaluation, our model-free DSFN approach to computing feature expectations can only be expected to be reliable if the candidate policy π has sufficient overlap or support under the transitions from the expert policy π_e. Specifically, note that Eqn. (2) depends on $(s,a,s') \in \mathcal{D}_e$. If an evaluation policy π generates trajectories far from demonstrated trajectories where data are almost absent (i.e. $\mathcal{D} \sim \pi, \mathcal{D}\cap\mathcal{D}_e \approx \varnothing$), then DSFN's off-policy estimates can suffer from a high bias, for there would be no samples that characterize $(s, a \sim \pi(s))$. One simple but important workaround is to ensure that π_0 is already a near-expert policy, ensuring successive policy updates do not differ much from the expert. Starting near the expert policy also reduces the number of iterations of the IRL (each requires an MDP solution) loop till convergence.

4.2 Warm-starting DSFN-IRL with TRIL

The goal of our warm-start is to obtain good π_0 and ϕ. It is known that a naive supervised learning approach tends to *over-fit* expert demonstrations due to correlated samples [Ross *et al.*, 2011]. Our solution is to regularize the learned hidden layers by jointly requiring them to predict environmental dynamics in order to prevent over-fitting. Specifically, we propose Transition-Regularized Imitation Learning (TRIL), a two-channel network (Figure 2) that *simultaneously* obtains π_0 via regularized multi-class classification and an empirically successful feature representation ϕ via deep representation learning in the process.

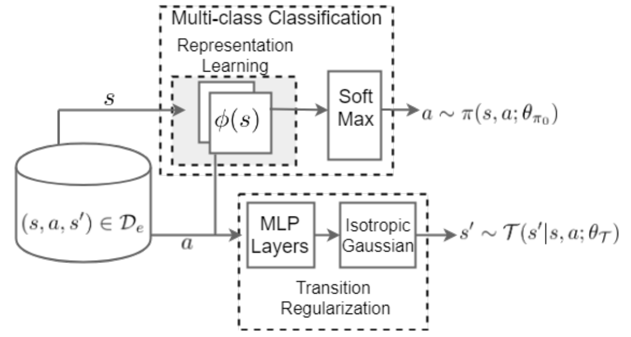
Multi-class Classification
Representation Learning
s
$\phi(s)$
Soft Max
$a \sim \pi(s, a; \theta_{\pi_0})$
$(s, a, s') \in \mathcal{D}_e$
a
MLP Layers
Isotropic Gaussian
$s' \sim \mathcal{T}(s'|s, a; \theta_{\mathcal{T}})$
Transition Regularization

Figure 2: TRIL : An initial policy π_0 is obtained by multi-class classification (action predictions - top channel) with transition regularization (s' predictions - bottom channel). A feature map $\phi(s)$ is learned by the shared hidden layers (shaded box). The Gaussian layer (learns μ, σ) handles stochastic environment dynamics better.

Regularized multi-class classification. Let θ_{π_0} be the parameters for expert action prediction (upper channel in Fig. 2) and $\theta_{\mathcal{T}}$ for next state prediction (lower channel in Fig. 2). The network is trained using the following loss: $\forall(s, a, s') \in D_e$

$$L(\theta_{\pi_0}, \theta_{\mathcal{T}}) = L_{\text{CE}}(a, \pi_0(s; \theta_{\pi_0})) + \lambda L_{\text{MSE}}(T(s, a; \theta_{\mathcal{T}}), s')$$

where L_{CE} denotes the cross entropy loss for expert action prediction given $(s, a) \in D_e$, L_{MSE} the mean squared error loss given $(s, a, s') \in D_e$, and λ the regularization strength.

Deep representation learning. We observed *empirically* that the feature map ϕ learned from the shared layers (Figure 2 - notice that TRIL extracts only the hidden layers of θ_{π_0} (shared across both channels) for obtaining $\phi(s)$) of the TRIL network provides a significant performance boost to our batch IRL. Intuitively, the learned feature space extracts sufficient information from the current state to predict both expert actions and the next state dynamics in the environment, both of which we consider important to model the unknown rewards. Training a feature representation on these quantities avoids the need for expert-specified features, providing a scalable approach across tasks to learn good feature maps for IRL from the data itself. The feature learning aspect of TRIL model is inspired from the work of [Song *et al.*, 2016] which states that feature encoders learned from dynamics prediction based errors are very successful value approximators in RL problems by demonstrating the theoretical connections between the model error for such a feature set and the Bellman error of the corresponding Q-function.

For discrete action problems (our experiments), $\phi(s, a)$ can be easily setup as the learned representation $\phi(s)$ from TRIL concatenated with a one-hot encoding that indicates the discrete action a i.e. $\phi(s, a) = \text{concat}(\phi(s), \{\mathbb{I}(a = a_i)\}_{i=1}^{|\mathcal{A}|})$.

5 Experimental Setup

We evaluated our model (TRIL+DSFN) on two sets of tasks: three simulated classical control tasks and a real-world clinical management task. *All experiments were done in a batch setting* where we assumed that only the expert demonstrations were provided bereft of any ability to collect additional transitions or access the simulator while training. Models were tested for expert imitation and reward interpretability.

Evaluation and Model setup For the classical control tasks, we evaluated the learned policies in terms of cumulative rewards that the learned IRL policy acquires on the simulators *after* the independent batch IRL training. For sepsis management, we used action-matching validation accuracy to evaluate goodness of policies. Both TRIL and DSFN are multi-layer feed-forward neural networks with a Gaussian output layer. DDQN [van Hasselt *et al.*, 2015] is used as the (near-optimal) MDP solver in our experiments. Parametric details of all the models are in Appendix Table 2.

Baselines. We compared our model to *batch* IRL baselines that use feature expectations explicitly and do not assume the existence of non-expert data: LSTD-μ [Klein *et al.*, 2011] and SCIRL [Klein *et al.*, 2012] (We did not include methods such as CSI, RCAL [Klein *et al.*, 2013; Piot *et al.*, 2014] as they do not use feature expectations; we leave comparison to a broader class of non-feature expectation-based IRL algorithms as future work). LSTD-μ is based on LSTD-Q [Lagoudakis and Parr, 2003] where it approximates the fixed points (μ^π) of the Bellman Equation by solving a linear system. SCIRL trains a linear multi-class classifier where expert feature expectations are computed using either a heuristic with Monte Carlo estimates or LSTD-μ. We chose LSTD-μ as we found it hard to apply the heuristic approach across tasks: the discount factor has to be tuned separately for each task and more importantly, it makes a relatively strong assumption that the effect of taking a non-expert action on its feature expectations can be expressed via the expert feature expectations up to the discount factor, which is limiting in practical batch setting tasks. Even with a tuned discount factor, a large, stochastic problem usually does not conform to the assumptions of the heuristic and the LSTD approach is more stable and generalizable across tasks with reasonably large amounts of data (e.g. sepsis). In general, we drew all experimental details from the authors' original source code (https://github.com/edouardklein/RL-and-IRL). To make the experiments fair, all batch IRL models had similar initializations, as appropriate. Link to our code repository can be found in Section 4 of the Appendix.

5.1 Classical Control Tasks

Task descriptions. MountainCar-v0, CartPole-v0 and Acrobot-v1 (https://github.com/openai/gym) are common simulated RL benchmarks. The environments are relatively small: MountainCar-v0 ($\mathcal{S} \subset \mathbb{R}^2, |\mathcal{A}| = 3$), CartPole-v0 ($\mathcal{S} \subset \mathbb{R}^4, |\mathcal{A}| = 2$), Acrobot-v1 ($\mathcal{S} \subset \mathbb{R}^6, |\mathcal{A}| = 3$). We obtained near-optimal policies using DDQN on the simulators and generated demonstration data by varying the number of trajectories $|\mathcal{D}_e| = \{1, 10, 100, 1000\}$. We note that while DDQN may not have produced an "optimal" expert, our goal in IRL is to simply identify rewards under which the observed behavior is optimal; our objective is not to recover the "true" reward that the expert may have been imperfectly targeting. We followed a $70 - 30$ train-validation split in our batch data.

DSFN(+TRIL) outperformed baselines, had better data efficiency and performed comparably as the pure imitator (TRIL). Fig. (3) shows TRIL+DSFN (red) outperforms all batch IRL baselines, works admirably in low batch data

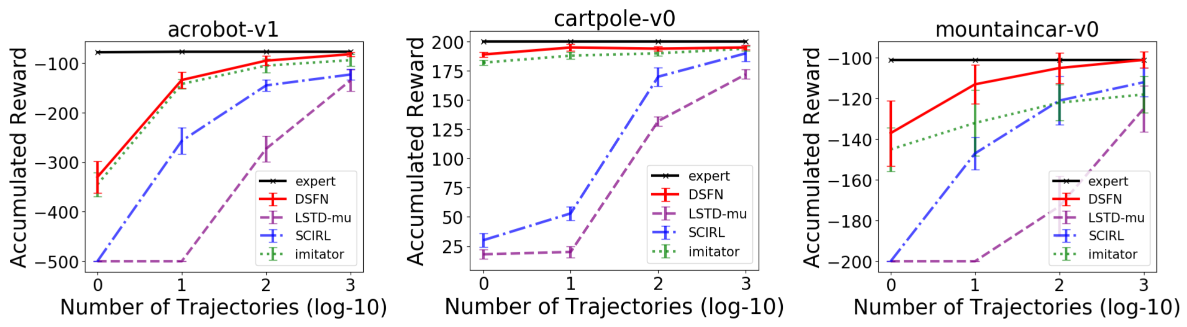

Figure 3: Cumulative Rewards in Classical Control Tasks: Batch IRL methods on three classical control benchmarks across varying number of demonstrated trajectories $|\mathcal{D}_e| = \{1, 10, 100, 1000\}$ (x axis on the log scale). The training was done in a batch setting while the final evaluations were done on the simulators over 5 independent trials with one standard error shown. Our method (DSFN initialized with TRIL) outperformed the batch IRL baselines in all data regimes.

regimes and scales well as the data sizes increase. We believe that LSTD-μ and SCIRL did not perform well in the low data regime because limited data led to an under-determined linear system and such LSTD models are known to be highly sensitive to data coverage (of the state-action space) of the expert demonstrations. Notably, the results also demonstrate that DSFN's action-matching performance is almost as good as pure imitator (TRIL) even when DSFN performs the *harder and useful* task of learning a reward function from demonstrations compared to pure policy imitation done by TRIL.

5.2 Clinical Task: Sepsis Management in ICU

Task descriptions. We evaluated TRIL+DSFN on a large-scale real-world Sepsis management task with the goal of imitating and interpreting clinicians' demonstrations. Sepsis is a leading cause of cost and mortality in Intensive Care Units (ICU) [Mervyn *et al.*, 2016]. Recently, [Raghu *et al.*, 2017] performed deep RL to *optimize* a policy for intravenous fluids and vasopressor interventions for sepsis patients on a *manually engineered* reward function — an RL problem. Our work performs a complementary task: we take the same dataset and learn a reward function via batch IRL that helps *imitate* clinicians by inducing similar treatment policies. Learning such a reward function is valuable because it may expose components of the reward that experts forgot to code (e.g. ways to reduce morbidity as well as mortality; if patients cannot be saved, there may still be appropriate palliative actions to take). Second, we may expose aspects that experts are optimizing for unknowingly and unnecessarily, thus promoting behavior change. Thus, IRL provides a starting point for designing an appropriate reward function for future RL works.

Data and MDP formulation. The input data was obtained from the Multiparameter Intelligent Monitoring in Intensive Care (MIMIC-III v1.4) database [Johnson *et al.*, 2016] and included a cohort of 17,898 patients fulfilling Sepsis-3 criteria. We then performed a 60-20-20 train-val-test split on our dataset. For the state space, we included 46 physiological features, including patient attributes, vitals and lab tests. For the action space, we included only the vasopressor intervention which was discretized to 5 bins (one for no action and four bins indicating 4 dosage quartiles) similar to [Raghu *et al.*, 2017]. We included other interventions such as IV fluids, mechanical ventilations within the state space. Overall, this led to an MDP with $\mathcal{S} \subset \mathbb{R}^{46}, |\mathcal{A}| = 5$ and unknown transition and reward functions. In-hospitality mortality indicator defined the terminal state S_T. (More details on data, setup can be found in Appendix Section 3).

Method	Top-1 Matching	Top-3 Matching
DSFN	$\mathbf{79 \pm 5\%}$	$\mathbf{90 \pm 3\%}$
LSTD-μ	$39 \pm 4\%$	$69 \pm 3\%$
SCIRL	$36 \pm 5\%$	$61 \pm 4\%$
TRIL	$\mathbf{80 \pm 2\%}$	$\mathbf{91 \pm 1\%}$
IL (unregularized)	$29 \pm 5\%$	$58 \pm 4\%$
Random	$20 \pm 1\%$	$49 \pm 6\%$

Table 1: Action Matching in Sepsis: Proportion of data (with std. error over 3 trials on test data) where the expert action is in the best or the best three predictions of each method. Our model outperformed all batch IRL baselines (all warm-started with with TRIL)

DSFN imitates clinicians better than the batch IRL baselines. Table (1) shows the action matching accuracy on the validation dataset where Top-$\{1, 3\}$ measures the proportion of transitions in the test data in which the expert action is either the best or among the three best predictions of each method. We observe that our models (TRIL, TRIL+DSFN) outperform all baselines in the action-matching accuracy by $\sim$ 40%. Similar to control experiments, DSFN performs comparably as the pure imitator TRIL while undertaking the harder task of learning a reward function. For a fair comparison, we also warm-started the two batch IRL baselines with TRIL. In fact, when these baselines were not initialized with TRIL, their performance was even worse, likely due to the limited representation power of linear models and their sensitivity to batch data coverage (limited in clinical data). Notably, the poor performance of the unregularized supervised classifier ($\sim 30\%$ vs. $\sim 80\%$ of TRIL) reinforces the value of transition-based regularization towards action imitation performance by preventing over-fitting to training batch data.

DSFN learns a clinically intuitive reward function. Fig. (4) analyzes the learned reward function with respect to three important vitals that are extreme in patients with Sep-

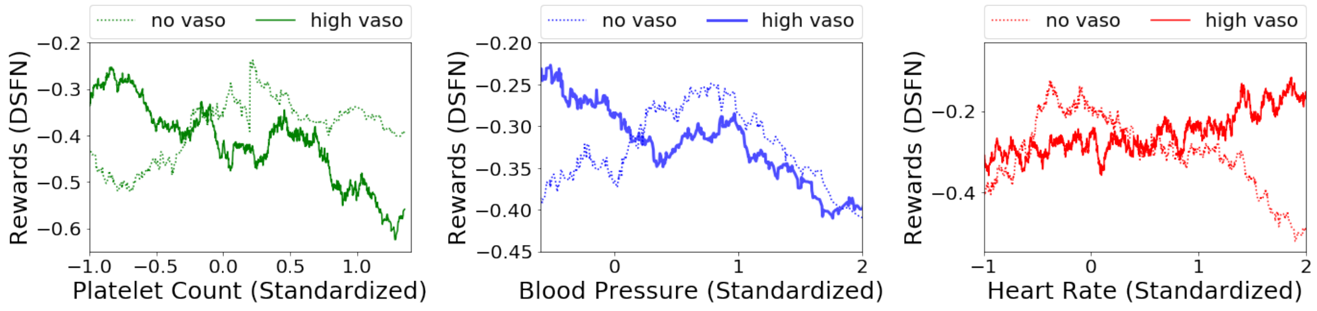

Figure 4: Clinically intuitive rewards in Sepsis: Patient vitals (x-axis, standardized) against the learned reward function (y-axis) for two actions—no vasopressor administered (dashed) and high amounts administered (solid). Septic shock typically causes low platelets, low BP, and high heart rates (HR) and requires vasopressor administration as treatment. We see that the learned reward function penalizes non-action and rewards high vasopressor during these severe scenarios, matching standard clinical intuition.

tic shock. Septic shock usually results in low platelet counts (PC), low blood pressures (BP), high heart rates (HR) and clinicians encourage strong dosages of vasopressors in these cases. We observed that the reward function complies with this intuition by giving low rewards for *not* administering vasopressor (dashed) and high rewards for administering vasopressor (solid) in the low BP, low platelets and high HR regimes. A practising intensivist confirmed this intuition.

6 Discussion

Performing IRL in a truly batch setting is challenging for three reasons: the estimation of feature expectations (specific to batch settings), IRL's computational complexity, and high sensitivity to the feature representation (generic across all IRL algorithms). In practice, it is essential to address *all* these challenges for batch IRL methods to perform consistently well across tasks. Our primary contribution, DSFN is key to address the OPE challenge for feature expectations and produces solid IRL performance compared to other baselines, even with the same initialization. TRIL, on the other hand, helps mitigate the other two challenges along with the potential data support mismatch issue of DSFN. With a TRIL warm-start, we typically converge in ≤ 5 iterations—significantly reducing computational burden—and the learned feature space from TRIL is expressive enough across tasks to find a reward function that induces policies almost as good as the expert.

In our efforts to solve these batch challenges, the role of few small, yet powerful, model engineering ideas cannot be overlooked. For instance, in cases where expert demonstrations are highly stochastic (such as the sepsis management), having an isotropic Gaussian output layer seems to provide a significant performance gain. The network learns the mean and variance of a Gaussian distribution that produces final samples similar to the work of [Duan *et al.*, 2016]. We also normalize states on a rolling basis to provide a consistent range of input values to the networks [Henderson *et al.*, 2017]. Similarly, transition-based regularization provides unprecedented performance boost on the imitation front by reducing over-fitting to batch data. In cases with highly correlated samples such as sepsis data, TRIL outperforms simple supervised imitation by about 50%. Besides, it is important to note that although TRIL learns a dynamics model, we use model training as a pure regularizer and our IRL algorithm remains model-free since the learned dynamics model (particularly for sepsis) is not powerful enough to simulate trajectories of IRL policies without accumulating huge errors.

As IRL is an ill-posed problem [Ng *et al.*, 2000], the reward functions that our method (true for all IRL methods in general) learns is *one among many* reward functions that explain the observed demonstrations and no guarantee exists that the learned reward function matches the true (unknown) expert's reward function. Rather, the only possible assertion one can make is that the model learns *a* reward function that induces a policy whose value is sufficiently close to that of the expert. In future, one can look to restrict the possible class of reward functions to enhance reward identifiability [Fu *et al.*, 2017].

Finally, we remark that TRIL and DSFN serve different purposes. If one only requires an expert-imitating policy under fixed dynamics, using TRIL alone may suffice. However, if one's focus is on understanding expert motivations via rewards or on building generalizable agents under shifting dynamics, learning rewards via DSFN (IRL) in batch settings is appropriate [Piot *et al.*, 2013; Fu *et al.*, 2017].

7 Conclusion

We proposed a novel IRL method that works well in *batch* settings without simulators and under unknown dynamics. Our method DSFN (+TRIL) outperforms all batch IRL baselines that use feature expectations on both simulated and real-world tasks, and the rewards learned in the ICU task match clinical intuition. Though we test our models on healthcare, our models make no domain specific assumptions and should work off-the-shelf for other domains. Also, the OPE and warm-start models we propose are generic and could be applied to develop batch versions of other popular IRL algorithms. Overall, our model is among the first ones to produce good imitation and learn rewards via IRL to infer the motivations underlying expert decisions in real-world batch settings.

Acknowledgements

We acknowledge and thank our funding source AFOSR FA 9550-17-1-0155.

References

[Abbeel and Ng, 2004] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In $21^{st} ICML$, page 1, 2004.

[Boularias *et al.*, 2011] Abdeslam Boularias, Jens Kober, and Jan Peters. Relative entropy inverse reinforcement learning. In 14^{th} *AISTATS*, pages 182–189, 2011.

[Burchfiel *et al.*, 2016] Benjamin Burchfiel, Carlo Tomasi, and Ronald Parr. Distance minimization for reward learning from scored trajectories. In 30^{th} *AAAI*, pages 3330–3336, 2016.

[Duan *et al.*, 2016] Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In 33^{rd} *ICML*, pages 1329–1338, 2016.

[Fu *et al.*, 2017] Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. *arXiv:1710.11248*, 2017.

[Henderson *et al.*, 2017] Peter Henderson, Risashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. *arXiv:1709.06560v3*, 2017.

[Ho and Ermon, 2016] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In 30^{th} *NeurIPS*, pages 4565–4573, 2016.

[Johnson *et al.*, 2016] Alistair Johnson, Tom Pollard, Lu Shen, Lehman Li-wei, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger Mark. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3:160035, 2016.

[Klein *et al.*, 2011] Edouard Klein, Matthieu Geist, and Olivier Pietquin. Batch, off-policy and model-free apprenticeship learning. In *European Workshop on Reinforcement Learning*, pages 285–296. Springer, 2011.

[Klein *et al.*, 2012] Edouard Klein, Matthieu Geist, Bilal Piot, and Olivier Pietquin. Inverse reinforcement learning through structured classification. In 26^{th} *NIPS*, pages 1007–1015, 2012.

[Klein *et al.*, 2013] 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.

[Kulkarni *et al.*, 2016] Tejas D Kulkarni, Ardavan Saeedi, Simanta Gautam, and Samuel J Gershman. Deep successor reinforcement learning. *arXiv:1606.02396*, 2016.

[Lagoudakis and Parr, 2003] Michail G Lagoudakis and Ronald Parr. Least-squares policy iteration. *JMLR*, 4:1107–1149, 2003.

[Leike *et al.*, 2017] Jan Leike, Miljan Martic, Victoria Krakovna, Pedro Ortega, Laurent Orseau, and Shane Legg. Ai safety gridworlds. *arXiv:1711.09883*, 2017.

[Mervyn *et al.*, 2016] Singer Mervyn, Deutschman Clifford S., Seymour Cristopher, and et al. The third international consensus definitions for sepsis and septic shock (sepsis-3). *JAMA*, 315(8):801–810, 2016.

[Mnih *et al.*, 2015] 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.

[Ng *et al.*, 2000] Andrew Y Ng, Stuart J Russell, et al. Algorithms for inverse reinforcement learning. In 17^{th} *ICML*, pages 663–670, 2000.

[Piot *et al.*, 2013] Bilal Piot, Matthieu Geist, and Olivier Pietquin. Learning from demonstrations: Is it worth estimating a reward function? In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 17–32. Springer, 2013.

[Piot *et al.*, 2014] Bilal Piot, Matthieu Geist, and Olivier Pietquin. Boosted and reward-regularized classification for apprenticeship learning. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, pages 1249–1256. International Foundation for Autonomous Agents and Multiagent Systems, 2014.

[Piot *et al.*, 2017] Bilal Piot, Matthieu Geist, and Olivier Pietquin. Bridging the gap between imitation learning and inverse reinforcement learning. *IEEE transactions on neural nets and learning systems*, 28(8):1814–1826, 2017.

[Raghu *et al.*, 2017] Aniruddh Raghu, Matthieu Komorowski, Leo Celi Ahmed, Imran, Peter Szolovits, and Marzyeh Ghassemi. Deep reinforcement learning for sepsis treatment. *arXiv:1711.09062*, 2017.

[Ratliff *et al.*, 2006] Nathan D Ratliff, J Andrew Bagnell, and Martin A Zinkevich. Maximum margin planning. In 23^{rd} *ICML*, pages 729–736, 2006.

[Ross *et al.*, 2011] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In 14^{th} *AISTATS*, pages 627–635, 2011.

[Song *et al.*, 2016] Zhao Song, Ronald Parr, Xuejun Liao, and Lawrence Carin. Linear feature encoding for reinforcement learning. In 30^{th} *NeurIPS*, pages 4224–4232, 2016.

[Sutton *et al.*, 1998] Richard S Sutton, Andrew G Barto, et al. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

[Thomas and Brunskill, 2016] Philip Thomas and Emma Brunskill. Data-efficient off-policy policy evaluation for rl. In 33^{rd} *ICML*, pages 2139–2148, 2016.

[van Hasselt *et al.*, 2015] Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. *arXiv:1509.06461*, 2015.

[Ziebart *et al.*, 2008] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In 23^{rd} *AAAI*, pages 1433–1438, 2008.