Defining Admissible Rewards for High-Confidence Policy Evaluation in Batch Reinforcement Learning

Niranjani Prasad
Princeton University
Princeton NJ, USA
np6@princeton.edu

Barbara E. Engelhardt
Princeton University
Princeton NJ, USA
bee@cs.princeton.edu

Finale Doshi-Velez
Harvard University
Cambridge MA, USA
finale@seas.harvard.edu

ABSTRACT

A key impediment to reinforcement learning (RL) in real applications with limited, batch data is in defining a reward function that reflects what we implicitly know about reasonable behaviour for a task and allows for robust off-policy evaluation. In this work, we develop a method to identify an *admissible* set of reward functions for policies that (a) do not deviate too far in performance from prior behaviour, and (b) can be evaluated with high confidence, given only a collection of past trajectories. Together, these ensure that we avoid proposing unreasonable policies in high-risk settings. We demonstrate our approach to reward design on synthetic domains as well as in a critical care context, to guide the design of a reward function that consolidates clinical objectives to learn a policy for weaning patients from mechanical ventilation.

CCS CONCEPTS

• **Computing methodologies** → **Batch learning**; **Sequential decision making**; • **Applied computing** → *Health care information systems.*

KEYWORDS

reward design, off-policy evaluation, reinforcement learning

ACM Reference Format:
Niranjani Prasad, Barbara E. Engelhardt, and Finale Doshi-Velez. 2020. Defining Admissible Rewards for High-Confidence Policy Evaluation in Batch Reinforcement Learning. In *ACM Conference on Health, Inference, and Learning (ACM CHIL '20), April 2–4, 2020, Toronto, ON, Canada.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3368555.3384450

1 INTRODUCTION

One fundamental challenge of reinforcement learning (RL) in practice is specifying the agent's reward. Reward functions implicitly define policy, and misspecified rewards can introduce severe, unexpected effects, from reward gaming to irreversible changes in parts of the environment we do not want to influence [17]. However, it can be difficult for domain experts to distill multiple (and often implicit) requisites for desired behaviour into a single scalar feedback signal. This is exemplified by efforts towards the application of reinforcement learning to decision-making in healthcare; in RL, an agent aims to choose the best action within a stochastic process given inherent time delay in feedback from a decision, making it an attractive framework for learning clinical treatment policies [30]. However, this feedback can be received over various timescales and represent clinical implications—such as treatment efficacy, side effects or patient discomfort—with widely different, and uncertain, priorities. Existing approaches to representing this scalar feedback in healthcare tasks range from taking reward to be a sparse, high-level signal such as mortality [15] or rewards based on a single physiological variable or severity score of interest [20, 25] to relatively ad-hoc weighting of clinically derived objectives [21].

Much work in reward design [26, 27] or inference using inverse reinforcement learning [1, 4, 10] focuses on online, interactive settings in which the agent has access either to human feedback [5, 18] or to a simulator with which to evaluate policies and compare against human performance. Here, we focus on reward design for *batch* RL: we assume access only to a set of past trajectories collected from sub-optimal experts, with which to train our policies. This is common in many real-world scenarios where the risks of deploying an agent are high but logging current practice is relatively easy, as in healthcare, as well as education or finance [2, 6].

Batch RL is distinguished by two key preconditions when performing reward design. First, as we assume that data are expensive to acquire, we must ensure that policies found using the reward function can be *evaluated* given existing data. Regardless of the true objectives of the designer, there exist fundamental limitations on reward functions that can be optimized and that also provide guarantees on performance. There have been a number of methods presented in the literature for safe, high-confidence policy improvement from batch data given some reward function, treating behaviour seen in the data as a baseline [9, 16, 23, 28]. In this work, we turn this question around to ask: *What is the class of reward functions for which high-confidence policy improvement is possible?*

Second, we typically assume that batch data are not random but produced by domain experts pursuing biased but reasonable policies. Thus if an expert-specified reward function results in behaviour that diverges substantially from past trajectories, we must ask whether that divergence was intentional or, as is more likely, simply because the designer omitted an important constraint, causing the agent to learn unintentional behaviour. This assumption can be formalized by treating the batch data as ε-optimal with respect to the true reward function, and searching for rewards that are consistent with this assumption [11]. Here, we extend these ideas to incorporate the uncertainty present when evaluating a policy in the batch setting, where trajectories from the estimated policy cannot be collected.

We note that these two constraints are not equivalent; the extent of overlap in reward functions satisfying these criteria depends,

ACM CHIL '20, April 2–4, 2020, Toronto, ON, Canada

ACM ISBN 978-1-4503-7046-2/20/04.
https://doi.org/10.1145/3368555.3384450

for example, on the homogeneity of behaviour in the batch data: if consistency is measured with respect to average behaviour in the data, and agents deviate substantially from this average—as may be across clinical care providers—then the space of policies that can be evaluated given the batch data may be larger than the policy space consistent with the average expert.

In this paper, we combine these two conditions to construct tests for *admissible* functions in reward design using available data. This yields a novel approach to the challenge of high-confidence policy evaluation given high-variance importance sampling-based value estimates over extended decision horizons—typical of batch RL problems—and encourages safe, incremental policy improvement. We illustrate our approach on several benchmark control tasks with continuous state spaces, and in reward design for the health care task of weaning a patient from a mechanical ventilator.

2 PRELIMINARIES AND NOTATION

A Markov decision process (MDP) is a tuple of the form $M = \{\mathcal{S}, \mathcal{A}, P_0, P, R, \gamma\}$, where $\mathcal{S}$ is the set of all possible states, and $\mathcal{A}$ are the available actions. $P_0(s)$ is the distribution over the initial state $s \in \mathcal{S}$; $P(s'|s,a)$ gives the probability of transition to s' given current state s and action $a \in \mathcal{A}$. The function $R(s,a,s')$ defines the reward for performing action a in state s, and observing new state s'. Lastly, the discount factor $\gamma \leq 1$ determines the relative importance of immediate and longer-term rewards.

Our objective is to learn a policy function $\pi^* : \mathcal{S} \rightarrow \mathcal{A}$ mapping states to actions that maximize the expected cumulative discounted reward—that is, $\pi^* = \text{argmax}_\pi \, \mathbb{E}_{s\sim P_0}[V^\pi(s)|M]$—where the value function $V^\pi(s)$ is defined as:

$$V^\pi = \mathbb{E}_{P_0,P,\pi}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1})\right]. \tag{1}$$

In batch RL, we have a collection of trajectories of the form $h = \{s_0, a_0, r_0, \ldots, s_T, a_T, r_T\}$. We do not have access to the transition function P or the initial state distribution P_0. Without loss of generality, we express the reward as a linear combination of some arbitrary function of the observed state transition: $R = w^T\phi(s,a,s')$, where $\phi \in \mathbb{R}^k$ is a vector function of state-action features relevant to learning an optimal policy, and $||w||_1 = 1$, to induce invariance to scaling factors in reward specification [4]. The value V^π of a policy π with reward weight w can then be written as:

$$V^\pi = \mathbb{E}_{P_0,P,\pi}\left[\sum_{t=0}^{\infty} \gamma^t w^T\phi(\cdot)\right] = w^T\mu^\pi, \text{ where}$$
$$\mu^\pi = \mathbb{E}_{P_0,P,\pi}\left[\sum_{t=0}^{\infty} \gamma^t \phi(\cdot)\right]. \tag{2}$$

where the vector μ^π denotes the *feature expectations* [1] of policy π, that is, the total expected discounted time an agent spends in each feature state. Thus, μ^π provides a representation of state dynamics of a policy that is decoupled from the reward function.

To quantify confidence in the estimated value V^π of policy π, we adapt the empirical Bernstein concentration inequality [19] to get a probabilistic lower bound V_{lb} on the estimated value [29]: consider a set of trajectories h_n, $n \in 1...N$ and let $\hat{V}_n$ be the value estimate for trajectory n. Then, with probability at least $1-\delta$:

$$V_{lb} = \frac{1}{N}\sum_{n=1}^{N}\hat{V}_n - \frac{1}{N}\sqrt{\frac{\ln(\frac{2}{\delta})}{N-1}\sum_{n,n'=1}^{N}(\hat{V}_n - \hat{V}_{n'})^2} - \frac{7b\ln(\frac{2}{\delta})}{3(N-1)}, \tag{3}$$

where b is the maximum achievable value of $V(\pi)$.

3 ADMISSIBLE REWARD SETS

We now turn to our task of identifying admissible reward sets – that is, defining the space of reward functions that yield policies that are *consistent* in performance with available observational data, as well as possible to *evaluate* off-policy for high-confidence performance lower bounds. In Sections 3.1 and 3.2, we define two sets of weights $\mathcal{P}_C$ and $\mathcal{P}_\mathcal{E}$ to be the consistent and evaluable sets, respectively, show that they are closed and convex, and define their intersection $\mathcal{P}_C \cap \mathcal{P}_\mathcal{E}$ as the set of *admissible* reward weights. In Sections 3.3 and 3.4, we describe how to test whether a given reward lies in the intersection of these polytopes, and, if not, how to find the closest points within this space of admissible reward functions given some initial reward proposed by the designer of the RL agent.

3.1 Consistent Reward Polytope

Given near-optimal expert demonstrations, the polytope of *consistent* rewards [11] may be defined as the set of all weight vectors w defining reward function $R = w^T\phi(s)$, that are consistent with the agent's existing knowledge. In the setting of learning from demonstrations, this knowledge is the assumption that demonstrations achieve ε-optimal performance with respect to the "true" reward. We denote the behaviour policy of experts as π_b with policy feature expectations μ_b , where $V(\pi_b) = w^T\mu_b$. The consistent weight vectors for this expert demonstration setting are then all w such that $w^T\mu \leq w^T\mu_b + \varepsilon$, $\mu \in \mathcal{P}_\mathcal{F}$, where $\mathcal{P}_\mathcal{F}$ is the space of all possible policy feature representations. It has been shown that this set is convex, given access to an exact MDP solver [11].

Translating this to the batch reinforcement learning setting, with a fixed set of *sub-optimal* trajectories, requires adaptations to both the constraints and their computation. First, we choose to constrain the relative rather than absolute difference in performance of the observed trajectories and that of the learnt optimal policy, in order to better handle high variance in the magnitudes of estimated values. Second, we make our constraint symmetric such that the value of the learnt policy can deviate equally above or below the value of the observed behaviour. This reflects the use of this constraint as a way to place metaphorical guardrails on the deviation of the behaviour of the learnt policy from the policy in the batch trajectories—rather than to impose optimality assumptions that only bound performance from above. That is, we want a reward that results in performance similar to the observed batch trajectories, where performance some factor Δ_c greater than or less than this established baseline should be equally admissible. Our new polytope $\mathcal{P}_C$ for the space of weights satisfying this is then:

$$\mathcal{P}_C = \left\{w \;:\; \frac{1}{\Delta_c} \leq \frac{w^T\mu_b}{w^T\mu} \leq \Delta_c \right\}, \tag{4}$$

where μ are the feature expectations of the optimal policy when solving an MDP with reward weights w, and value estimates are constrained to be positive, $w^T\mu > 0 \; \forall \mu \in \mathcal{P}_\mathcal{F}$. The parameter

$\Delta_c \geq 1$ that determines the threshold on the consistency polytope is tuned according to our confidence in the batch data; trajectories from severely biased experts may warrant larger Δ_c.

The batch setting also requires changes to the computation of these constraints, as we do not have access to a simulator to calculate exact feature expectations μ; we must instead estimate them from available data. We do so by adapting off-policy evaluation methods to estimate the representation of a policy in feature space. Specifically, we use per-decision importance sampling (PDIS [22]) to get a consistent, unbiased estimator of μ:

$$\hat{\mu} = \frac{1}{N}\sum_{n=1}^{N}\sum_{t=0}^{T}\gamma^t \rho_t^{(n)}\phi\left(s_t^{(n)}\right) \tag{5}$$

where importance weights $\rho_t^{(n)} = \Pi_{i=0}^{t}(\pi(a_i^n|s_i^n)/\pi_b(a_i^n|s_i^n))$. Together with the feature expectations of the observed experts (obtained by simple averaging across trajectories), we can evaluate the constraint in Eq 4.

Proposition 1. *The set of weights $\mathcal{P}_C$ defines a closed convex set, given access to exact MDP solver.*

Proof. The redefined constraints in Eq 4 can be rewritten as: $w^T(\mu - \Delta_c\mu_b) \leq 0$; $w^T(\mu_b - \Delta_c\mu) \leq 0$, where $\mu = \max_{\mu' \in \mathcal{P}_F} w^T\mu'$ is the feature expectations of the optimal policy obtained from the exact MDP solver. As these constraints are still linear in w—that is, of the form $w^T A \leq b$—the convexity argument in [11] holds. □

In Section 3.3, we discuss how this assumption of convexity changes given the presence of approximation error in the MDP solver and in estimated feature expectations.

Illustration. We first construct a simple, synthetic task to visualize a polytope of consistent rewards. Consider an agent in a two-dimensional continuous environment, with state defined by position $s_t = [x_t, y_t]$ for bounded x_t and y_t. At each time t, available actions are steps in one of four directions, with random step size $\delta_t \sim \mathcal{N}(0.4, 0.1)$. The reward is $r_t = [0.5, 0.5]^T s_t$: the agent's goal is to reach the top-right corner of the 2D map. We use fitted-Q iteration with tree-based approximation [7] to learn a deterministic policy π_b that optimizes the reward, then we sample 1000 trajectories from a biased policy (move left with probability ϵ, and π_b otherwise) to obtain batch data.

We then train policies π_w optimizing for reward functions $r_t = w^T\phi(s)$ on a set of candidate weights $w \in \mathbb{R}^2$ on the unit $\ell 1$-norm ball. For each policy, a PDIS estimate of the feature expectations $\hat{\mu}_w$ is obtained using the collected batch data. The consistency constraint (Eq 4) is then evaluated for each candidate weight vector, with different thresholds Δ_c (Fig 1). Prior to evaluating constraints, we ensure our estimates $w^T\mu$ for discounted cumulative reward are positive, by augmenting w and $\phi(s)$ with a constant positive bias term: $w' = [w, 1]$, $\phi'(s) = [\phi(s), B]$ where $B = 14.0$ for this task. For large Δ_c ($\Delta_c \geq 17$), the set of consistent w includes approximately half of all test weights: given these thresholds, all w for which at least one dimension of the state vector was assigned a significant positive weight (greater than 0.5) in the reward function were determined to yield policies sufficiently close to the batch data, while vectors with large negative weights on either coordinate are rejected. When Δ_c is reduced to 3.0, only the reward originally optimized for the batch data, ($w = [0.5, 0.5]$) is admitted by $\mathcal{P}_C$.

3.2 Evaluable Reward Polytope

Our second set of constraints on reward design stem from the need to be able to confidently evaluate a policy in settings when further data collection is expensive or infeasible. We interpret this as a condition on confidence in the estimated policy performance: given an estimate for the expected value $\mathbb{E}[\hat{V}^\pi] = w^T\hat{\mu}$ of a policy π and corresponding probabilistic lower bound V_{lb}^π, we constrain the ratio of these values to lie within some threshold $\Delta_e \geq 0$. A reward function with weights w lies within the polytope of evaluable rewards if $V_{lb}^\pi \geq (1-\Delta_e)w^T\hat{\mu}$, where $\hat{\mu} \in \mathcal{P}_{\mathcal{F}}$ is our PDIS estimate of feature expectations. To formulate this as a linear constraint in the space of reward weights w, the value lower bound V_{lb}^π must be rewritten in terms of w. This is done by constructing a combination of upper and lower confidence bounds on the policy feature expectations, denoted μ^{lb}. Starting from the empirical Bernstein concentration inequality (Eq 3):

$$\begin{aligned} V_{lb} &= \frac{1}{N}\sum_{n=1}^{N}\hat{V}_n - \frac{1}{N}\sqrt{\underbrace{\frac{\ln(\frac{2}{\delta})}{N-1}}_{c_1}\sum_{n,n'=1}^{N}(\hat{V}_i - \hat{V}_j)^2} - \underbrace{\frac{7b\ln(\frac{2}{\delta})}{3(N-1)}}_{c_2} \\ &= \frac{1}{N}\sum_{n=1}^{N}\hat{\mu}^{(n)}w - sgn(w)\cdot\frac{1}{N}\sqrt{c_1\sum_{n,n'=1}^{N}\left(\hat{\mu}^{(n)}w - \hat{\mu}^{(n')}w\right)^2} - c_2 \quad (6) \\ &= \frac{1}{N}w\cdot\sum_{n=1}^{N}\hat{\mu}^{(n)} - sgn(w)\cdot\frac{1}{N}w\sqrt{c_1\cdot\sum_{n,n'=1}^{N}\left(\hat{\mu}^{(n)} - \hat{\mu}^{(n')}\right)^2} - c_2 \\ &= w^T\hat{\mu}^{lb} - c_2 \quad (7) \end{aligned}$$

where the k^{th} element of $\hat{\mu}^{lb}$—that is, the value of the k^{th} feature that yields the lower bound in the value of the policy—is dependent on the sign of the corresponding element of the weights, $w[k]$:

$$\hat{\mu}^{lb}[k] = \begin{cases} \frac{1}{N}\left[\sum_{n=1}^{N}\hat{\mu}^{(n)} - \sqrt{c_1\sum_{n,n'=1}^{N}\left(\hat{\mu}^{(n)} - \hat{\mu}^{(n')}\right)^2}\right]_k & w[k] \geq 0 \\ \frac{1}{N}\left[\sum_{n=1}^{N}\hat{\mu}^{(n)} + \sqrt{c_1\sum_{n,n'=1}^{N}\left(\hat{\mu}^{(n)} - \hat{\mu}^{(n')}\right)^2}\right]_k & w[k] < 0 \end{cases} \tag{8}$$

The definition in Eq 8 allows us to incorporate uncertainty in $\hat{\mu}$ when evaluating our confidence in a given policy: a lower bound for our value estimate requires the *lower bound* of $\hat{\mu}$ if the weight is positive, and the *upper bound* if the weight is negative. Thus, the *evaluable reward polytope* can be written as:

$$\mathcal{P}_{\mathcal{E}} = \left\{w \,:\, w^T\mu^{lb} \geq (1-\Delta_e)w^T\mu\right\} \tag{9}$$

where $\mu = \max_{\mu' \in \mathcal{P}_F} w^T\mu'$ is the expectation of state features for the optimal policy obtained from solving the MDP with reward weights w, and μ_{lb} is the corresponding lower bound. The constant c_2 in the performance lower bound (Eq 7) is absorbed by threshold parameter Δ_e on the tightness of the lower bound.

Proposition 2. *The set of weights $\mathcal{P}_{\mathcal{E}}$ defines a closed convex set, given access to an exact MDP solver.*

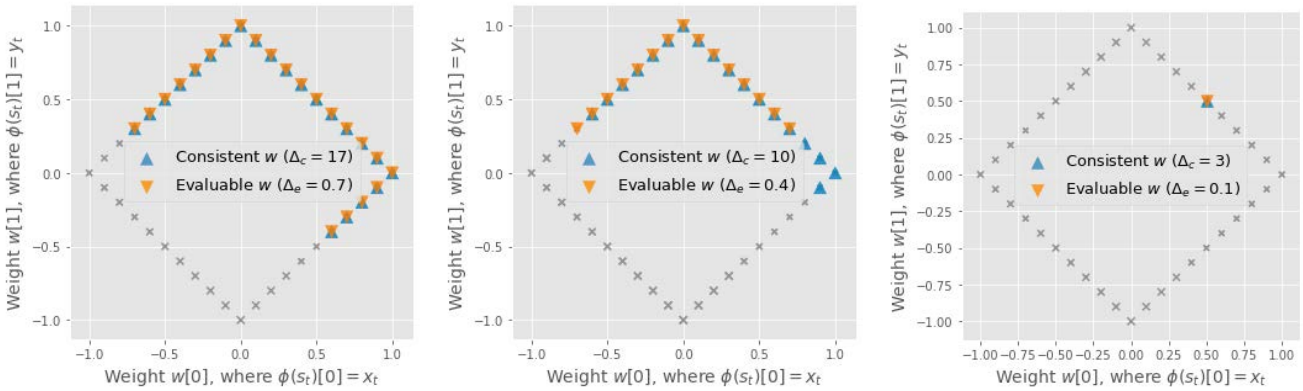

Figure 1: Consistent and evaluable polytopes, with different thresholds $\Delta_c > 1.0$ and $\Delta_e < 1.0$ respectively, for a continuous 2D map environment with true reward $r_t = [0.5, 0.5]^T \phi(s_t)$, where $\phi(s_t) = [x_t, y_t]$. In both cases, increasing Δ corresponds to relaxing constraints and expanding the satisfying set of weights w. Intersecting points comprise the final admissible set $\mathcal{P}_{adm}$.

Proof. The set $\mathcal{P}_{\mathcal{E}}$ contains all weights w that satisfy constraints linear in w: $w^T((1-\Delta_e)\mu - \mu_{lb}) \leq 0$. As in the case of $\mathcal{P}_C$, it follows from [11] that the set described by these constraints is convex. □

Illustration. In order to visualize an example polytope for evaluable rewards (Eq 9), we return to the two-dimensional map described in Section 3.1. As before, we begin with a batch of trajectories collected by a biased ϵ-greedy expert policy trained on the true reward. We use these trajectories to obtain PDIS estimates $\hat{\mu}$ for policies trained with a range of reward weights w on the $\ell 1$-norm ball. We then evaluate $\hat{\mu}^{lb}$, and in turn the hyperplanes defining the intersecting half-spaces of the evaluable reward polytope, for each w. Plotting the set of evaluable reward vectors for different thresholds Δ_e, we see substantial overlap with the consistent reward polytope in this environment, though neither polytope is a subset of the other (Fig 1(b)). We also find that in this setting, the value of the evaluability constraint is asymmetric about the true reward—more so than the the consistency metric—such that policies trained on penalizing x_t ($w[0] < 0$), hence favoring movement left, can be evaluated to obtain a tighter lower bound than weights that learn policies with movement down, which is rarely seen in the biased demonstration data (Fig 1(b)). Finally, tightening the threshold further to $\Delta_e = 0.1$ (Fig 1(c)) the set of accepted weights is again just the true reward, as for the consistency polytope.

3.3 Querying Admissible Reward Polytope

Given our criteria for consistency and evaluability of reward functions, we need a way to access the sets satisfying these constraints. These sets cannot be explicitly described as there are infinite policies with corresponding representations μ, and so infinite possible constraints; instead, we construct a *separation oracle* to access points in this set in polynomial time (Algorithm 1). A separation oracle tests whether a given point w' lies in polytope of interest $\mathcal{P}$, and if not, outputs a separating hyperplane defining some half-space $w^T A \leq b$, such that $\mathcal{P}$ lies inside this half-space and w' lies outside of it. The separation oracle for the polytope of *admissible* rewards evaluates both consistency and evaluability to determine whether w' lies in the intersection of the two polytopes, which we define as our admissible polytope $\mathcal{P}_{adm}$. If a constraint is not met, it outputs a new hyperplane accordingly.

Algorithm 1: Separation oracle SO_{adm} for admissible w

Input: Proposed weights $w \in \mathbb{R}^k$, behaviour policy μ_b, threshold parameters Δ_c, Δ_e
1. Solve MDP with weights w for optimal policy features $\mu = \text{argmax}_{\mu'} w^T \mu'$
2. Evaluate lower bound μ^{lb} for estimated policy features
if $w^T(\mu - \Delta_c \mu_b) > 0$ **then**
  $w \notin \mathcal{P}_C \Rightarrow$ Reject w
  Output: Halfspace $\{w^T(\mu - \Delta_c \mu_b) \leq 0\}$
else if $w^T(\mu_b - \Delta_c \mu) > 0$ **then**
  $w \notin \mathcal{P}_C \Rightarrow$ Reject w
  Output: Halfspace $\{w^T(\mu_b - \Delta_c \mu) \leq 0\}$
else if $w^T((1-\Delta_e)\mu - \mu_{lb}) > 0$ **then**
  $w \notin \mathcal{P}_{\mathcal{E}} \Rightarrow$ Reject w
  Output: Halfspace $\{w^T((1-\Delta_e)\mu - \mu_{lb}) \leq 0\}$
else
  $w \in \mathcal{P}_C \cap \mathcal{P}_{\mathcal{E}} = \mathcal{P}_{adm} \Rightarrow$ Accept w

It should be noted that the RL problems of interest to us are typically large MDPs with continuous state spaces, as in the clinical setting of managing mechanical ventilation in the ICU, and moreover, because we are optimizing policies given only batch data, we know we can only expect to find *approximately* optimal policies. The use of PDIS estimates $\hat{\mu}$ of the true feature expectations in the batch setting introduces an additional source of approximation error. It has been shown that Algorithm 1 with an approximate MDP solver produces a *weird separation oracle* [11], one that does not necessarily define a convex set. However, it does still accept all points in the queried polytope, and can thus still be used to test whether a proposed weight vector w lies within this set.

Returning to our 2D map (Fig 1), the *admissible reward polytope* $\mathcal{P}_{adm}$ is the set of weights accepted by both the consistent and evaluable polytopes. The choice of thresholds Δ_c and Δ_e respectively is important in obtaining a meaningfully restricted, non-empty set to choose rewards from. These thresholds will depend on the extent of exploratory or sub-optimal behaviour in the batch data, and the level of uncertainty acceptable when deploying a new policy. We find that in this toy 2D map setting, there is considerable overlap between the two polytopes defining the admissible set, though this is not always the case; from our earlier intuition, as the behaviour policy from which trajectories were generated is the same for all trajectories, there is limited "exploration", or deviation from average behaviour across trajectories, and the therefore the evaluability constraints admit reward weights that largely overlap with those consistent with average behaviour.

3.4 Finding the Nearest Admissible Reward

With a separation oracle SO_{adm} for querying whether a given w lies in the admissible reward polytope, we optimize linear functions over this set using, e.g., the ellipsoid method for exact solutions or—as considered here—the iterative *follow-perturbed-leader* (FPL) algorithm for computationally efficient approximate solutions [13]. To achieve our goal of aiding reward specification for a designer with existing but imperfectly known goals, we pose our optimization problem as follows (Alg 2): given initial reward weights w_0 proposed by the agent designer, we first test whether w_0, with some small perturbation, lies in the admissible polytope $\mathcal{P}_{adm}$, which we define by training a policy π_0 approximately optimizing this reward. If it does not lie in $\mathcal{P}_{adm}$, we return new weights $w \in \mathcal{P}_{adm}$ that minimize distance $\|w - w_{init}\|_2$ from the proposed weights. This solution is then perturbed and tested in turn. We note that constraints posed based on the behaviour μ_b observed in the available batch trajectories are encapsulated by this minimization over weights in set $\mathcal{P}_{adm}$, that is, solving a constrained linear optimization defined by the linear constraints on w from Eqs 4 and 9. The constraints at each iteration do not fully specify $\mathcal{P}_{adm}$, but instead give us a half-space to optimize over, at each step.

The constrained linear program solved at each iteration scales in constant time with the dimensionality of w; although we only present results with functions $\phi(s)$ of dimensionality at most 3, for the sake of visualization, the iterative algorithm presented can be scaled to higher dimensional $\phi(s)$, as the complexity of the linear program solved at each iteration is dependent only on the number of constraints. Our final reward weights and a randomized policy are the average across the approximate solutions in each iteration. This policy optimizes a reward that is the closest admissible reward to the original goals of the designer of the RL agent.

Algorithm 2: *Follow-perturbed-leader* for admissible w.

INPUT: Initial weights $w_0 \in \mathbb{R}^k$, iterations T, $\delta = \frac{1}{k\sqrt{T}}$, $t = 0$

while $t \leq T$ **do**

1. Let $r_t = \sum_{i=1}^{t-1}(w_i + p_t) \cdot \phi(\cdot)$, where $p_t \sim \mathcal{U}[0, \frac{1}{\delta}]^k$; solve for $\pi_t = \text{argmax}_\pi V^\pi | r_t$ 2. Let $\mu_t = \mu(\pi_t) + q_t$, where $q_t \sim \mathcal{U}[0, \frac{1}{\delta}]^k$; evaluate constraints defining $\mathcal{P}_{adm}$ 3. Solve for $w_t := \text{argmin}_{w \in \mathcal{P}_{adm}} \|w - w_{init}\|_2$ 4. Let $t := t + 1$

end

OUTPUT: $\pi_{final} = \frac{1}{T}\sum_{i=1}^{T} \pi_t$; $\bar{w} = \frac{1}{T}\sum_{i=1}^{T} w_t$

4 EXPERIMENT DESIGN

4.1 Benchmark Tasks

We illustrate our approach to determining admissible reward functions on three benchmark domains with well-defined objectives: classical control tasks Mountain Car and Acrobot, and a simulation-based treatment task for HIV patients. The control tasks, implemented using OpenAI Gym [3], both have a continuous state space and discrete action space, and the objective is to reach a terminal goal state. To explore how the constrained polytopes inform reward design for these tasks, an expert behaviour policy is first trained with data collected from an exploratory policy receiving a reward of -1 at each time step, and 0 once the goal state is reached. A batch of 1000 trajectories is collected by following this expert policy with Boltzmann exploration, mimicking a sub-optimal expert. Given these trajectories, our task is to choose a reward function that allows us to efficiently learn an optimal policy that is i) consistent with the expert behaviour in the trajectories, and ii) evaluable with acceptably tight lower bounds on performance. We limit the reward function $r_t = w^T\phi(s_t)$ in each task to a weighted sum of three features, $\phi(s) \in \mathbb{R}^3$, chosen to include sufficient information to learn a meaningful policy while allowing for visualization. For Mountain Car, we use quantile-transformed position, velocity, and an indicator ± 1 of whether the goal state has been reached. For Acrobot, $\phi(s)$ comprises the quantile-transformed cosine of the angle of the first link, angular velocity of the link, and an indicator ± 1 of whether the goal link height is satisfied. We sweep over weight vectors on the 3D $\ell 1$-norm ball, training policies with the corresponding rewards, and filtering for admissible w.

The characterization of a good policy is more complex in our third benchmark task, namely treatment recommendation for HIV patients, modeled by a linear dynamical system [8]. We have a continuous state space and four discrete actions to choose from: no treatment, one of two possible drugs, or both in conjunction. The true reward in this domain is given by: $R = -0.1V + 10^3E - 2 \cdot 10^4(0.7d_1)^2 - 2 \cdot 10^3(0.3d_2)^2$, where V is the viral count, E is the count of white blood cells (WBC) targeting the virus, and d_1 and d_2 are indicators for drugs 1 and 2 respectively. We can rewrite this function as $r = w^T\phi(s)$, where $\phi(s) = [V, c_0E, c_1d_1 + c_2d_2] \in \mathbb{R}^3$, with constants c_0, c_1 and c_2 set such that weights $w = [-0.1, 0.5, 0.4]$ reproduce the original function. Again, the low dimensionality of $\phi(s)$ is simply for interpretability. An expert policy is trained using this true reward, and a set of sub-optimal trajectories are collected by following this policy with Boltzmann exploration. Policies are trained over weights w, $||w||_1 = 1$ and tested for admissibility.

4.2 Mechanical Ventilation in ICU

We use our methods to aid reward design for the management of invasive mechanical ventilation in critically ill patients [21]. Mechanical ventilation refers to the use of external breathing support to replace spontaneous breathing in patients with compromised

Table 1: MDP state features taken as input for learning an optimal policy for management of mechanical ventilation in ICU.

	State Features
Demographics	Age, Gender, Ethnicity, Admission Weight, First ICU Unit
Ventilator Settings	Ventilator mode, Inspired O_2 fraction (FiO_2), PEEP set, O_2 Flow
Measured Vitals	Heart Rate, Respiratory Rate, O_2 saturation pulseoxymetry, Arterial pH, Richmond-RAS Scale, Non Invasive Blood Pressure (systolic, diastolic, mean), Mean Airway Pressure, Tidal Volume, Peak Insp. Pressure, Plateau Pressure, Arterial CO_2 Pressure, Arterial O_2 pressure
Input Sedation	Propofol, Fentanyl, Midazolam, Dexmedetomidine, Morphine Sulfate, Hydromorphone, Lorazepam
Other	Consecutive duration into ventilation (D), Number of reintubations (N)

lung function. It is one of the most common, as well as most costly, interventions in the ICU [24]. Timely *weaning*, or removal of breathing support, is crucial to minimizing risks of ventilator-associated infection or over-sedation, while avoiding failed breathing tests or reintubation due to premature weaning. Expert opinion varies on how best to trade off these risks, and clinicians tend to err towards conservative estimates of patient wean readiness, resulting in extended ICU stays and inflated costs.

We look to design a reward function for a weaning policy that penalizes prolonged ventilation, while weighing the relative risks of premature weaning such that the optimal policy does not recommend strategies starkly different from clinician behaviour, and the policies can be evaluated for acceptably robust bounds on performance using existing trajectories. We train and test our policies on data filtered from the MIMIC III data [12] with 6,883 ICU admissions from successfully discharged patients following mechanical ventilation, preprocessed and resampled in hourly intervals. The MDP for this task is adapted from [21]: the patient state s_t at time t is a 32-dimensional vector that includes demographic data, ventilator settings, and relevant vitals (Table 1. We learn a policy with binary action space $a_t \in [0, 1]$, for keeping the patient off or on the ventilator, respectively. The reward function $r_t = w^T \phi(s_t, a_t)$ with $\phi(s, a) \in \mathbb{R}^3$ includes (i) a penalty for more than 48 hours on the ventilator, (ii) a penalty for reintubation due to unsuccessful weaning, and (iii) a penalty on physiological instability when the patient is *off* the ventilator based on abnormal vitals:

$$\phi = \begin{bmatrix} -\min(0, \tanh 0.1(D_t - 48)) \cdot \mathbb{1}[a_t = 1] \\ -\mathbb{1}[\exists t' > t \text{ such that } N_{t'} > N_t) \cdot \mathbb{1}[a_t = 0] \\ -\frac{1}{|V|} \sum_v^V (v < v_{\min} || v < v_{\max}) \cdot \mathbb{1}[a_t = 0] \end{bmatrix} \quad (10)$$

where D_t is duration into ventilation at time t in an admission, N_t is the number of reintubations, $v \in V$ are physiological parameters each with normal range $[v_{\min}, v_{\max}]$, and V = {Ventilator settings, Measured vitals}. The three terms in $\phi(\cdot)$ represent penalties on duration of ventilation, reintubation, and abnormal vitals, respectively. Our goal is to learn the relative weights of these feedback signals to produce a consistent, evaluable reward function and learn a policy optimizing this reward. As before, we train our optimal policies using Fitted Q-iteration (FQI) with function approximation using extremely randomized trees [7]. We partition our dataset into 3,000 training episodes and 3,883 test episodes, and run FQI over 100 iterations on the training set, with discount factor $\gamma = 0.9$. We then use the learnt Q-function to train our binary treatment policy.

5 RESULTS AND DISCUSSION

5.1 Benchmark Control Tasks

5.1.1 Admissible $\mathbf{w}$ are clustered near true rewards. We analyze reward functions from the sweep over weight vectors on the $\ell 1$-norm unit ball for each benchmark task (Section 4.1) by first visualizing how the space of weights accepted by the consistency and evaluability polytopes—and therefore the space $\mathcal{P}_{adm}$ at the intersection of these polytopes—changes with the values of thresholds Δ_c and Δ_e. Alongside this, we plot the set of admitted weights produced by arbitrarily chosen thresholds (Fig 2). In all three tasks, we find that the admitted weights form distinct clusters; these are typically at positive weights on goal states in the classic control tasks, and at positive weights on WBC count for the HIV simulator, in keeping with the rewards optimized by the batch data in each case. We could therefore use this naive sweep over weights to choose a vector within the admitted cluster that is closest to our initial proposed function, or to our overall objective. For instance, if in the HIV task we want a policy that prioritizes minimization of side effects from administered drugs, we can choose specifically from admissible rewards with negative weight on the treatment term.

5.1.2 Analysis of admissible $\mathbf{w}$ can lend insight into reward shaping for faster policy learning. We may wish to shortlist candidate weights by setting more stringent thresholds for admissibility. We mimic this design process as follows: prioritizing evaluability in each of our benchmark environments, we choose the smallest possible Δ_e and large Δ_c for an admissible set of exactly three weights (Table 2). This reflects a typical batch setting, in which we want high-confidence performance guarantees; we also want to allow our policy to deviate when necessary from the available sub-optimal expert trajectories. For Mountain Car, our results show that two of the three vectors assign large positive weights to reaching the goal state; all assign zero or positive weight to the position of the car. The third, $w = [0.4, -0.4, 0.2]$ is dominated by a significant positive weight on position and a significant *negative* weight on velocity; this may be interpreted as a kind of reward shaping: the agent is encouraged to first move in reverse to achieve a negative velocity, as is necessary to reach the goal state in the under-powered mountain car problem. The top three w for Acrobot also place either positive

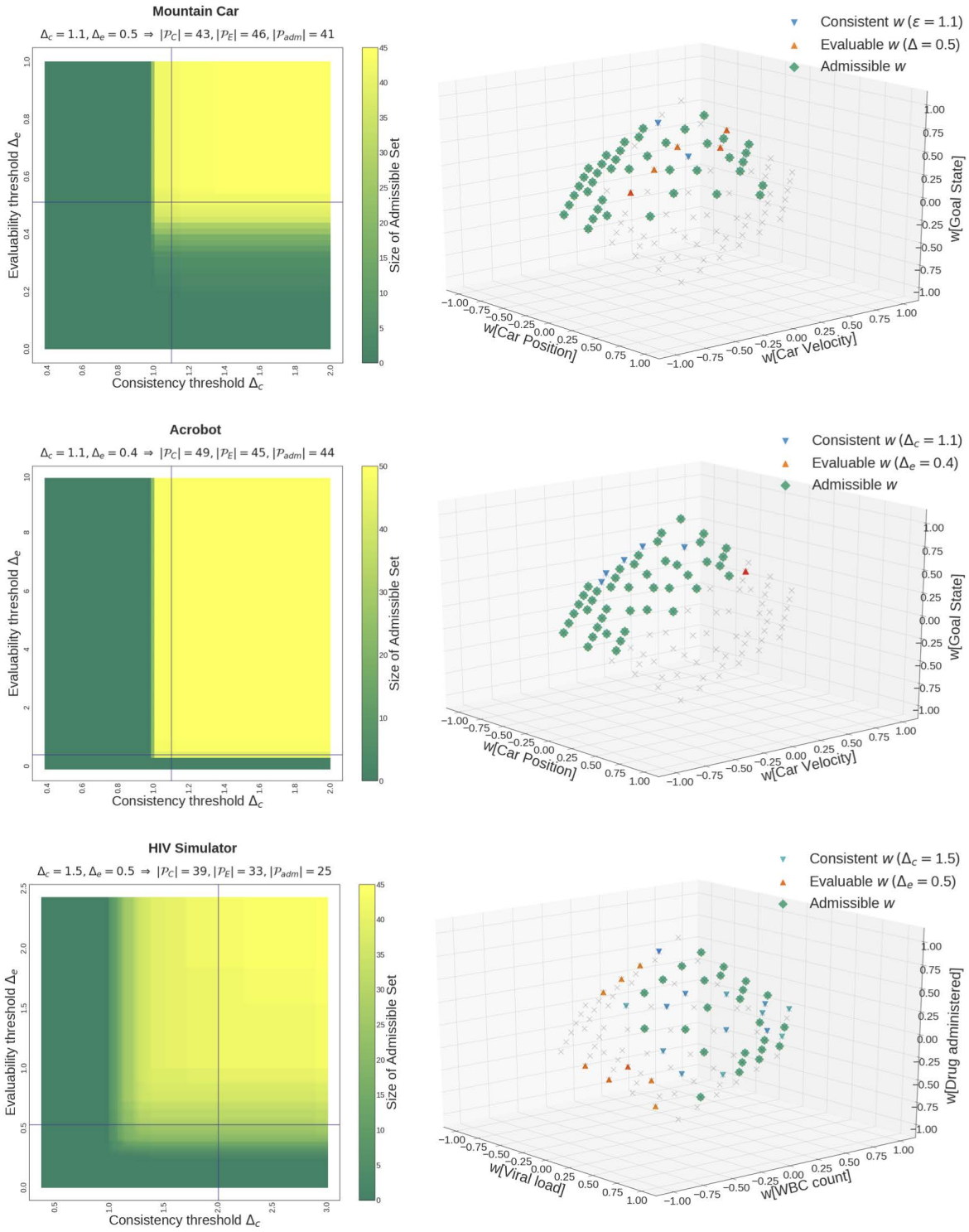

Figure 2: Admissible polytope size for varying thresholds on consistency (Δ_c) and evaluability (Δ_e), and distribution of admitted weights for fixed Δ_c, Δ_e, in: (a) Mountain Car (b) Acrobot (c) HIV Simulator. Note that admitted rewards for each task typically correspond to positive weights on the goal state.

Table 2: Analysing top three admitted weights w for each of the three benchmark control enviroments. Admissibility polytope thresholds are set by choosing a small Δ_c and required corresponding threshold Δ_e for an admissible set of size $|\mathcal{P}_{\mathbf{adm}}| = 3$.

Task	Top 3 Admissible weights	Δ_c ($\mathcal{P}_C$)	Δ_e ($\mathcal{P}_\mathcal{E}$)
Mountain Car	$[0.0, 0.2, 0.8]^T$, $[0.2, 0.2, 0.6]^T$, $[0.4, -0.4, 0.2]^T$	1.10	0.27
Acrobot	$[-0.2, 0.0, 0.8]^T$, $[-0.8, -0.2, 0.0]^T$, $[-0.2, -0.2, 0.6]^T$	1.10	0.29
HIV Simulator	$[0.0, 0.4, 0.6]^T$, $[-0.6, 0.2, 0.2]^T$, $[-0.2, 0.4, -0.4]^T$	1.20	0.28

weights on the goal state, or negative weights on the position of the first link. Again, the latter reward definition likely plays a shaping role in policy optimization by rewarding link displacement.

5.1.3 FPL can be used to correct biased reward specification in the direction of true reward. We use the HIV treatment task to explore how iterative solutions for admissible reward (Alg 2) can improve a partial or flawed reward specified by a designer. For instance, a simple first attempt by the designer at a reward function may place equal weights on each component of $\phi(s)$, with the polarity of weights—whether each component should elicit positive feedback or incur a penalty—decided by the designer's domain knowledge; here, the designer may suggest $w_0 = \frac{1}{3}[-1, 1, -1]^T$. We run Alg 2 for twenty iterations with this initial vector and thresholds $\Delta_c = 2.0, \Delta_e = 0.8$ and average over the weights from each iteration. This yields weights $\bar{w} = [-0.11, 0.57, -0.32]^T$, redistributed to be closer to the reward function being optimized in the batch data. This pattern is observed with more extreme initial rewards functions too; if e.g., the reward proposed depends solely on WBC count, $w_0 = [0, 1, 0]$, then we obtain weights $\bar{w} = [-0.14, 0.83, -0.04]$ after twenty iterations of this algorithm such that appropriate penalties are introduced on viral load and administered drugs.

5.2 Mechanical Ventilation in ICU

5.2.1 Admissible $\mathbf{w}$ may highlight bias in expert behaviour. We apply our methods to choose a reward function for a ventilator weaning policy in the ICU, given that we have access only to historical ICU trajectories with which to train and validate our policies. When visualizing the admissible set, with $\Delta_c = 1.8, \Delta_e = 0.4$, we find substantial intersection in the consistent and evaluable polytopes (Fig 3). Admitted weights are clustered at large negative weights on the duration penalty term favouring policies that are conservative in weaning patients (that is, those that keep patients *longer* on the ventilator), which is the direction of bias we expect in the past clinical behaviour. We can tether a naive reward that instead penalizes duration on the ventilator, $w = [1, 0, 0]$ to the space of rewards that are consistent with this conservative behaviour as follows: using FPL to search for a reward within the admissible set given this initial vector yields $\bar{w} = [0.72, 0.14, 0.14]$, introducing non-zero penalties on reintubation and physiological instability when off ventilation. This allows us to learn behaviour that is averse to premature extubation (consistent with historical clinical behaviour) without simply rewarding long durations on the ventilator.

5.2.2 FPL improves effective sample size for learnt policies. To verify whether weights from the admissible polytope enable higher

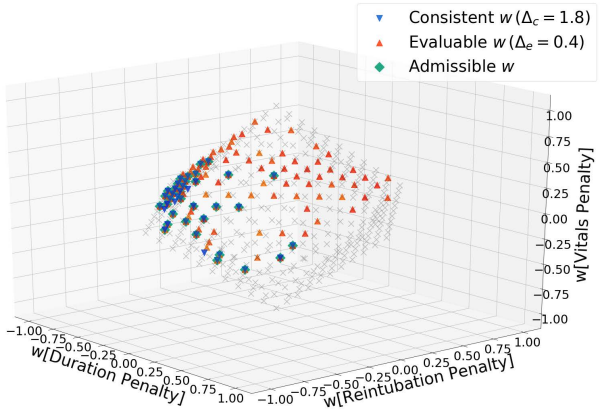

Figure 3: Mechanical Ventilation in the ICU: Admitted reward weights for fixed polytope thresholds Δ_c, Δ_e.

confidence policy evaluation, we explore a simple proxy for variance of an importance sampling-based estimate of performance: the effective sample size $N_{eff} = (\sum_n^N \rho_n)^2 / \sum_n^N \rho_n^2$ of the batch data [14], where ρ_n is importance weight of trajectory n for a given policy. In order to evaluate the Kish effective sample size N_{eff} for a given policy, we subsample admissions in our test data to obtain trajectories of approximately 20 timesteps in length, and calculate importance weights ρ_n for the policy considered using these subsampled trajectories. Testing a number of naive initializations of w, we find that effective sample size is consistently higher for weights following FPL (Table 3). This indicates that the final weights induce an optimal policy that is better represented in the batch data than the policy from the original weights.

Table 3: Mechanical ventilation in the ICU: Influence of FPL algorithm on Kish effective sample size of learnt policies.

Initial w	N_{EFF}	Final w	N_{EFF}
[1., 0., 0.]	8	[0.72, 0.14, 0.14]	14
[0., 1., 0.]	304	[-0.07, 0.77, 0.16]	352
[0., 0., 1.]	32	[0.15, -0.21, 0.66]	37
$\frac{1}{3}$[1., 1., 1.]	16	[0.24, 0.51, 0.25]	33

6 CONCLUSION

In this work, we present a method for reward design in reinforcement learning using batch data collected from sub-optimal experts. We do this by constraining rewards to those yielding policies within some distance of the policies of domain experts; the policies inferred from the admissible rewards also provide reasonable bounds on performance. Our experiments show how rewards can be chosen in practice from the space of functions satisfying these constraints, and illustrate this on the problem of weaning clinical patients from mechanical ventilation.

Translating policies learned from retrospective (batch) data into prospective results is a process that involves many parts and many steps. Effective reward design for RL is one of those parts; using admissible reward functions is one step toward not proposing problematic trajectories. Of course, it is not the only part: state spaces must still be defined, confounders identified, etc. That said, our approach to reward design provides a way to iteratively push the space of observed behaviour towards policies consistent and evaluable with respect to some ideal reward. There are a number of ways in which the methods here could be extended however, to better use the information available in existing data on what constitutes a safe policy, and in turn what reward function can ensure this. For instance, different care providers in clinical settings likely follow policies with different levels of precision, or perhaps even optimize for different reward functions; modeling this heterogeneity in behaviour and weighting experts appropriately can enable learnt behaviour closer to the best, rather than the average, expert. In addition, going beyond the use of summary statistics provided by policy feature expectations to explore more complex representations of behaviour that are still decoupled from rewards, and in turn better metrics for similarity in behaviour, can aid in more meaningful choices in reward function.

REFERENCES

[1] Pieter Abbeel and Andrew Y Ng. 2004. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first International Conference on Machine Learning*. ACM, 1.

[2] Onur Atan, William R Zame, and Mihaela van der Schaar. 2018. Learning Optimal Policies from Observational Data. *arXiv preprint arXiv:1802.08679* (2018).

[3] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. OpenAI Gym. *arXiv preprint arXiv:1606.01540* (2016).

[4] Daniel S Brown and Scott Niekum. 2018. Efficient probabilistic performance bounds for inverse reinforcement learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[5] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*. 4299–4307.

[6] Shayan Doroudi, Kenneth Holstein, Vincent Aleven, and Emma Brunskill. 2016. Sequence Matters, but How Exactly? A Method for Evaluating Activity Sequences from Data. *International Educational Data Mining Society* (2016).

[7] Damien Ernst, Pierre Geurts, and Louis Wehenkel. 2005. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research* 6, Apr (2005), 503–556.

[8] Damien Ernst, Guy-Bart Stan, Jorge Goncalves, and Louis Wehenkel. 2006. Clinical data based optimal STI strategies for HIV: a reinforcement learning approach. In *Proceedings of the 45th IEEE Conference on Decision and Control*. IEEE, 667–672.

[9] Mohammad Ghavamzadeh, Marek Petrik, and Yinlam Chow. 2016. Safe policy improvement by minimizing robust baseline regret. In *Advances in Neural Information Processing Systems*. 2298–2306.

[10] Dylan Hadfield-Menell, Smitha Milli, Pieter Abbeel, Stuart J Russell, and Anca Dragan. 2017. Inverse reward design. In *Advances in Neural Information Processing Systems*. 6765–6774.

[11] Jessie Huang, Fa Wu, Doina Precup, and Yang Cai. 2018. Learning safe policies with expert guidance. In *Advances in Neural Information Processing Systems*. 9105–9114.

[12] Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. 2016. MIMIC-III, a freely accessible critical care database. *Scientific data* 3 (2016), 160035.

[13] Adam Kalai and Santosh Vempala. 2016. Efficient algorithms for on-line optimization. *J. Comput. System Sci.* 71 (2016).

[14] L Kish. 1968. Survey Sampling. John Wiley & Sons, Inc., New York, London 1965, IX+ 643 S., 31 Abb., 56 Tab., Preis 83 s. *Biometrische Zeitschrift* 10, 1 (1968), 88–89.

[15] Matthieu Komorowski, Leo A Celi, Omar Badawi, Anthony C Gordon, and A Aldo Faisal. 2018. The Artificial Intelligence Clinician learns optimal treatment strategies for sepsis in intensive care. *Nature Medicine* 24, 11 (2018), 1716.

[16] Romain Laroche, Paul Trichelair, and Layla El Asri. 2017. Safe Policy Improvement with Baseline Bootstrapping. *arXiv preprint arXiv:1712.06924* (2017).

[17] Jan Leike, Miljan Martic, Victoria Krakovna, Pedro A Ortega, Tom Everitt, Andrew Lefrancq, Laurent Orseau, and Shane Legg. 2017. Ai safety gridworlds. *arXiv preprint arXiv:1711.09883* (2017).

[18] Robert Tyler Loftin, James MacGlashan, Bei Peng, Matthew E Taylor, Michael L Littman, Jeff Huang, and David L Roberts. 2014. A strategy-aware technique for learning behaviors from discrete human feedback. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*.

[19] Andreas Maurer and Massimiliano Pontil. 2009. Empirical Bernstein bounds and sample variance penalization. *arXiv preprint arXiv:0907.3740* (2009).

[20] Shamim Nemati, Mohammad M Ghassemi, and Gari D Clifford. 2016. Optimal medication dosing from suboptimal clinical examples: A deep reinforcement learning approach. In *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE, 2978–2981.

[21] Niranjani Prasad, Li-Fang Cheng, Corey Chivers, Michael Draugelis, and Barbara E Engelhardt. 2017. A reinforcement learning approach to weaning of mechanical ventilation in intensive care units, In Proceedings of the Conference on Uncertainty in Artificial Intelligence. *arXiv preprint arXiv:1704.06300*.

[22] Doina Precup, Richard S Sutton, and Satinder Singh. 2000. Eligibility Traces for Off-Policy Policy Evaluation. In *ICML'00 Proceedings of the Seventeenth International Conference on Machine Learning*.

[23] Elad Sarafian, Aviv Tamar, and Sarit Kraus. 2018. Safe Policy Learning from Observations. *arXiv preprint arXiv:1805.07805* (2018).

[24] Rhodri Saunders and Dimitris Geogopoulos. 2018. Evaluating the Cost-Effectiveness of Proportional-Assist Ventilation Plus vs. Pressure Support Ventilation in the Intensive Care Unit in Two Countries. *Frontiers in public health* 6 (2018).

[25] Susan M Shortreed, Eric Laber, Daniel J Lizotte, T Scott Stroup, Joelle Pineau, and Susan A Murphy. 2011. Informing sequential clinical decision-making through reinforcement learning: an empirical study. *Machine learning* 84, 1-2 (2011), 109–136.

[26] Jonathan Sorg, Satinder P Singh, and Richard L Lewis. 2010. Internal rewards mitigate agent boundedness. In *Proceedings of the 27th international conference on machine learning (ICML-10)*. 1007–1014.

[27] Jonathan Daniel Sorg. 2011. The optimal reward problem: Designing effective reward for bounded agents. *University of Michigan* (2011).

[28] Philip Thomas, Georgios Theocharous, and Mohammad Ghavamzadeh. 2015. High confidence policy improvement. In *International Conference on Machine Learning*. 2380–2388.

[29] Philip S Thomas, Georgios Theocharous, and Mohammad Ghavamzadeh. 2015. High-confidence off-policy evaluation. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

[30] Chao Yu, Jiming Liu, and Shamim Nemati. 2019. Reinforcement learning in healthcare: a survey. *arXiv preprint arXiv:1908.08796* (2019).