Of Moments and Matching: A Game-Theoretic Framework for Closing the Imitation Gap

Gokul Swamy [1] **Sanjiban Choudhury** [2] **J. Andrew Bagnell** [1 2] **Zhiwei Steven Wu** [3]

Abstract

We provide a unifying view of a large family of previous imitation learning algorithms through the lens of *moment matching*. At its core, our classification scheme is based on whether the learner attempts to match (1) *reward* or (2) *action-value* moments of the expert's behavior, with each option leading to differing algorithmic approaches. By considering adversarially chosen divergences between learner and expert behavior, we are able to derive bounds on policy performance that apply for all algorithms in each of these classes, the first to our knowledge. We also introduce the notion of *moment recoverability*, implicit in many previous analyses of imitation learning, which allows us to cleanly delineate how well each algorithmic family is able to mitigate compounding errors. We derive three novel algorithm templates (`AdVIL`, `AdRIL`, and `DAeQuIL`) with strong guarantees, simple implementation, and competitive empirical performance.

1. Introduction

When formulated as a statistical learning problem, imitation learning is fundamentally concerned with finding policies that minimize some notion of divergence between learner behavior and that of an expert demonstrator. Existing work has explored various types of divergences including KL (Pomerleau 1989, Bojarski et al. 2016), Jensen-Shannon (Rhinehart et al. 2018), Reverse KL (Kostrikov et al. 2019), f (Ke et al. 2019), and Wasserstein (Dadashi et al. 2020).

At heart, though, we care about the performance of the learned policy under an objective function that is not known to the learner. As argued by (Abbeel and Ng 2004) and (Ziebart et al. 2008), this goal is most cleanly formulated as a problem of *moment matching*, or, equivalently, optimizing Integral Probability Metrics (Sun et al. 2019) (IPMs). This is because a learner that in expectation matches the expert on all the basis functions of a class that includes the expert's objective function, or *matches moments*, must achieve the same performance and will thus be indistinguishable in terms of quality. Additionally, in sharp contrast to recently proposed approaches (Ke et al. 2019, Kostrikov et al. 2019, Jarrett et al. 2020, Rhinehart et al. 2018), moments, due to their simple forms as expectations of basis functions, can be effectively estimated via demonstrator samples and the uncertainty in these estimates can often be quantified to regularize the matching objective (Dudik et al. 2004). In short: these moment matching procedures are simple, effective, and provide the strongest policy performance guarantees we are aware of for imitation learning.

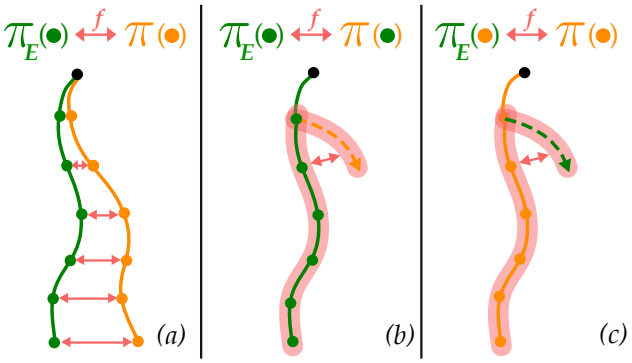

Figure 1. We consider three classes of imitation learning algorithms. **(a)** On-policy reward moment-matching algorithms require access to the environment to generate learner trajectories. **(b)** Off-policy Q-value moment-matching algorithms run completely offline but can produce policies with quadratically compounding errors. **(c)** On-policy Q-value moment-matching algorithms require access to the environment and a queryable expert but can produce strong policies in *recoverable* MDPs.

As illustrated in Fig. 1, there are three classes of moments a learner can focus on matching: (a) *on-policy reward moments*, (b) *off-policy Q-value moments*, and (c) *on-policy Q-value moments*, each of which have different requirements on the environment and on the expert. We abbreviate them as **reward**, **off-Q**, and **on-Q** moments, respectively.

[1]Robotics Institute, Carnegie Mellon University [2]Aurora Innovation [3]Institute for Software Research, Carnegie Mellon University. Correspondence to: Gokul Swamy <gswamy@cmu.edu>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

Our key insight is that reward moments have more *discriminative power* because they can pick up on differences in induced state visitation distributions rather than just action conditionals. Thus, *reward moment matching is a harder problem with stronger guarantees than off-Q and on-Q moment matching*.

Our work makes the following three contributions:

1. We present a unifying framework for moment matching in imitation learning. Our framework captures a wide range of prior approaches and allows us to construct, to our knowledge, the first formal lower bounds demonstrating that the choice between matching "reward", "off-Q", or "on-Q" moments is fundamental to the problem of imitation learning rather than an artifact of a particular algorithm or analysis.

2. We clarify the dependence of imitation learning bounds on problem structure. We introduce a joint property of an expert policy and moment class, *moment recoverability*, that helps us characterize the problems for which compounding errors are likely to occur, regardless of the kind of feedback the learner is exposed to.

3. We provide three novel algorithms with strong performance guarantees. We derive idealized algorithms that match each class of moments. We also provide practical instantiations of these ideas, `AdVIL`, `AdRIL`, and `DAeQuIL`. These algorithms have significantly different practical performance as well as theoretical guarantees in terms of compounding of errors over time steps of the problem.

2. Related Work

Imitation Learning. Imitation learning has been shown to be an effective method of solving a variety of problems, from getting cars to drive themselves (Pomerleau 1989), to achieving superhuman performance in games like Go (Silver et al. 2016, Sun et al. 2018), to sample-efficient learning of control policies for high DoF robots (Levine and Koltun 2013), to allowing human operators to effectively supervise and teach robot fleets (Swamy et al. 2020). The issue of compounding errors in imitation learning was first formalized by (Ross et al. 2011), with the authors proving that an interactive expert that can suggest actions in states generated via learner policy rollouts will be able to teach the learner to recover from mistakes.

Adversarial Imitation Learning. Starting with the seminal work of (Ho and Ermon 2016), numerous proposed approaches have framed imitation learning as a game between a learner's policy and another network that attempts to discriminate between learner rollouts and expert demonstrations (Fu et al. 2018, Song et al. 2018). We build upon this work by elucidating the properties that result from the kind of *feedback* the learner is exposed to – whether they are able to see the consequences of their own actions via rollouts or if they are only able to propose actions in states from expert trajectories. Our proposed approaches also have stronger guarantees and less brittle performance than the popular GAIL (Ho and Ermon 2016).

Mathematical Tools. Our algorithmic approach combines two tools that have enjoyed success in imitation learning: *functional gradients* (Ratliff et al. 2009) and the *Integral Probability Metric* (Sun et al. 2019). We define two algorithms, `AdRIL` and `AdVIL` that are based on optimizing the value-directed IPM, with `AdRIL` having the discriminative player perform updates via functional gradient descent. The IPM is linear in the discriminative function, unlike other proposed metrics like the Donsker-Varadhan bound on KL divergence. Specifically, the Donsker-Varadhan bound includes an expectation of the exponentiated discriminative function, which makes estimation difficult with a few samples (McAllester and Stratos 2020). Our analysis makes repeated use of the *Performance Difference Lemma* (Kakade and Langford 2002, Bagnell et al. 2003) or PDL, which allows us to bound the suboptimality of the learner's policy.

Our proposed algorithms bear some resemblance to previously proposed methods, with `AdRIL` resembling SQIL (Reddy et al. 2019) and `AdVIL` resembling ValueDICE (Kostrikov et al. 2019). We note that `AdVIL`, while cleanly derived from the PDL, can also be derived from an IPM by using a telescoping substitution similar to the ValueDICE derivation. Notably, because `AdVIL` is linear in the discriminator, it does not suffer from ValueDICE's difficulty in estimating the expectation of an exponential. This difficulty might help explain why ValueDICE can underperform the behavioral cloning baseline on several benchmark tasks (Jarrett et al. 2020). Similarly, `AdRIL` can avoid the sharp degradation in policy performance that SQIL demonstrates (Barde et al. 2020). This is because SQIL hard-codes the discriminator while `AdRIL` adaptively updates the discriminator to account for changes in the policy's trajectory distribution. `DAeQuIL` can be seen as the natural extension of DAgger (Ross et al. 2011) to the adversarial loss setting.

3. Moment Matching Imitation Learning

We begin by formalizing our setup and objective.

3.1. Problem Definition

Let $\Delta(\mathcal{X})$ denote the space of all probability distribution over a set $\mathcal{X}$. Consider an MDP parameterized by $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, r, T, P_0 \rangle$[1], where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the transition operator, $r : \mathcal{S} \times \mathcal{A} \to [-1, 1]$ is a reward function, T is the hori-

[1] We ignore the discount factor for simplicity.

zon, and P_0 is the initial state distribution. Let π denote the learner's policy and let π_E denote the demonstrator's policy. A trajectory $\tau \sim \pi = \{s_t, a_t\}_{t=1...T}$ refers to a sequence of state-action pairs generated by first sampling a state s_1 from P_0 and then repeatedly sampling actions a_t and next states s_{t+1} from π and $\mathcal{T}$ for $T-1$ time-steps. We also define our value and Q-value functions as $V_t^\pi(s) = \mathbb{E}_{\tau\sim\pi|s_t=s}[\sum_{t'=t}^{T} r(s_{t'}, a_{t'})]$, $Q_t^\pi(s,a) = \mathbb{E}_{\tau\sim\pi|s_t=s,a_t=a}[\sum_{t'=t}^{T} r(s_{t'}, a_{t'})]$. We also define the advantage function as $A_t^\pi(s,a) = Q_t^\pi(s,a) - V_t^\pi(s)$. Lastly, let performance be $J(\pi) = \mathbb{E}_{\tau\sim\pi}[\sum_{t=1}^{T} r(s_t, a_t)]$. Then,

Definition 1. *We define the **imitation gap** as:*

$$J(\pi_E) - J(\pi) \tag{1}$$

The goal of the learner is to minimize (1) and *close the imitation gap*. The unique challenge in imitation learning is that the reward function r is unknown and the learner must rely on demonstrations from the expert π_E to minimize the gap. A natural way to solve this problem is to empirically match *all* moments of $J(\pi_E)$. If all the moments are perfectly matched, regardless of the unknown reward function, the imitation gap must go to zero. We now delve into the various types of moments we can match.

3.2. Moment Taxonomy

Broadly speaking, a learner can focus on matching per-timestep *reward* or over-the-horizon *Q-value* moments of expert behavior. We use $\mathcal{F}_r : \mathcal{S} \times \mathcal{A} \to [-1, 1]$ to denote a class of reward functions, $\mathcal{F}_Q : \mathcal{S} \times \mathcal{A} \to [-T, T]$ to denote the set of Q functions induced by sampling actions from some $\pi \in \Pi$, and $\mathcal{F}_{Q_E} : \mathcal{S} \times \mathcal{A} \to [-\overline{Q}, \overline{Q}]$ to denote the set of Q functions induced by sampling actions from π_E. We assume all three function classes are closed under negation. Lastly, we refer to $H \in [0, 2\overline{Q}]$ as the *recoverability constant* of the problem and define it as follows:

Definition 2. *A pair $(\pi_E, \mathcal{F}_{Q_E})$ of an expert policy and set of expert Q-functions is said to be* **H-*recoverable*** *if $\forall s \in \mathcal{S}$, $\forall a \in \mathcal{A}$, $\forall f \in \mathcal{F}_{Q_E}$, $|f(s,a) - \mathbb{E}_{a'\sim\pi_E(s)}[f(s,a')]| < H$.*

H is an upper bound on all possible advantages that could be obtained by the expert under a Q-function in $\mathcal{F}_{Q_E}$. Intuitively, H tells us how many time-steps it takes the expert to recover from an arbitrary mistake. We defer a more in-depth discussion of the implications of this concept to Section 4.5.

Reward. Matching reward moments entails minimizing the following expansion of the imitation gap:

$$\begin{aligned}
&J(\pi_E) - J(\pi) \\
&= \mathop{\mathbb{E}}_{\tau\sim\pi_E}\sum_{t=1}^{T} r(s_t,a_t) - \mathop{\mathbb{E}}_{\tau\sim\pi}\sum_{t=1}^{T} r(s_t,a_t) \\
&= \mathop{\mathbb{E}}_{\tau\sim\pi}\sum_{t=1}^{T} -r(s_t,a_t) - \mathop{\mathbb{E}}_{\tau\sim\pi_E}\sum_{t=1}^{T} -r(s_t,a_t) \\
&\le \sup_{f\in\mathcal{F}_r} \mathop{\mathbb{E}}_{\tau\sim\pi}\sum_{t=1}^{T} f(s_t,a_t) - \mathop{\mathbb{E}}_{\tau\sim\pi_E}\sum_{t=1}^{T} f(s_t,a_t)
\end{aligned}$$

MOMENT	CLASS	ENV.	π_E QUERIES
REWARD	$\mathcal{F}_r : [-1, 1]$	✓	✗
OFF-POLICY Q	$\mathcal{F}_Q : [-T, T]$	✗	✗
ON-POLICY Q	$\mathcal{F}_{Q_E} : [-\overline{Q}, \overline{Q}]$	✓	✓

Table 1. An overview of the requirements for the three classes of moment matching.

In the last step, we use the fact that $-r(s,a) \in \mathcal{F}_r$. Crucially, reward moment-matching demands on-policy rollouts $\tau \sim \pi$ for the learner to calculate per-timestep divergences.

Instead of matching moments of the reward function, we can consider matching moments of the action-value function. We can apply the Performance Difference Lemma (PDL) to expand the imitation gap (1) into either on-policy or off-policy expressions.

Off-Policy Q. Starting from the PDL:

$$\begin{aligned}
&J(\pi_E) - J(\pi) \\
&= \mathop{\mathbb{E}}_{\tau\sim\pi_E}[\sum_{t=1}^{T} Q_t^\pi(s_t,a_t) - \mathop{\mathbb{E}}_{a\sim\pi(s_t)}[Q_t^\pi(s_t,a)]] \\
&\le \sup_{f\in\mathcal{F}_Q} \mathop{\mathbb{E}}_{\tau\sim\pi_E}[\sum_{t=1}^{T} \mathop{\mathbb{E}}_{a\sim\pi(s_t)}[f(s_t,a)] - f(s_t,a_t)]
\end{aligned}$$

In the last step, we use the fact that $Q_t^\pi(s,a) \in \mathcal{F}_Q$ for all $\pi \in \Pi$ and $r \in \mathcal{F}_r$. The above expression is off-policy – it only requires a collected dataset of expert trajectories to be evaluated and minimized. In general though, $\mathcal{F}_Q$ can be a far more complex class than $\mathcal{F}_r$ because it has to capture both the dynamics of the MDP and the choices of *any* policy.

On-Policy Q. Expanding in the reverse direction:

$$\begin{aligned}
&J(\pi_E) - J(\pi) \\
&= -\mathop{\mathbb{E}}_{\tau\sim\pi}[\sum_{t=1}^{T} Q_t^{\pi_E}(s_t,a_t) - \mathop{\mathbb{E}}_{a\sim\pi_E(s_t)}[Q_t^{\pi_E}(s_t,a)]] \\
&\le \sup_{f\in\mathcal{F}_{Q_E}} \mathop{\mathbb{E}}_{\tau\sim\pi}[\sum_{t=1}^{T} f(s_t,a_t) - \mathop{\mathbb{E}}_{a\sim\pi_E(s_t)}[f(s_t,a)]]
\end{aligned}$$

In the last step, we use the fact that $Q_t^{\pi_E}(s,a) \in \mathcal{F}_{Q_E}$ for all $r \in \mathcal{F}_r$. In the realizable setting, $\pi_E \in \Pi$, $\mathcal{F}_{Q_E} \subseteq \mathcal{F}_Q$. While $\mathcal{F}_{Q_E}$ is a smaller class, to actually evaluate this expression, we require an *interactive* expert that can tell us what action they would take in any state visited by the learner as well as on-policy samples from the learner's current policy $\tau \sim \pi$.

With this taxonomy in mind, we now turn our attention to deriving policy performance bounds. [2]

3.3. Moment Matching Games

A unifying perspective on the three moment matching variants can be achieved by viewing the learner as solv-

[2] See Sup. E for mixed moments and an alternative Q-moment scheme that can extend to the IL from observation alone setting.

ing a game. More specifically, we consider variants of a two-player minimax game between a learner and a discriminator. The learner selects a policy $\pi \in \Pi$, where $\Pi \triangleq \{\pi : \mathcal{S} \to \Delta(\mathcal{A})\}$. We assume Π is convex, compact and that $\pi_E \in \Pi$.[3] The discriminator (adversarially) selects a function $f \in \mathcal{F}$, where $\mathcal{F} \triangleq \{f : \mathcal{S} \times \mathcal{A} \to \mathbb{R}\}$. We assume that $\mathcal{F}$ is convex, compact, closed under negation, and finite dimensional.[4] Depending on the class of moments being matched, we assume that $\mathcal{F}$ is spanned by convex combinations of the elements of $\mathcal{F}_r/2$, $\mathcal{F}_Q/2T$, or $\mathcal{F}_{Q_E}/2T$. Lastly, we set the learner as the *minimization player* and the discriminator as the *maximization player*.

Definition 3. *The **on-policy reward**, **off-policy** Q, and **on-policy** Q payoff functions are, respectively:*

$$U_1(\pi, f) = \frac{1}{T}(\mathop{\mathbb{E}}_{\tau\sim\pi}[\sum_{t=1}^{T} f(s_t, a_t)] - \mathop{\mathbb{E}}_{\tau\sim\pi_E}[\sum_{t=1}^{T} f(s_t, a_t)])$$

$$U_2(\pi, f) = \frac{1}{T}(\mathop{\mathbb{E}}_{\substack{\tau\sim\pi_E \\ a\sim\pi(s_t)}}[\sum_{t=1}^{T} f(s_t, a)] - \mathop{\mathbb{E}}_{\tau\sim\pi_E}[\sum_{t=1}^{T} f(s_t, a_t)])$$

$$U_3(\pi, f) = \frac{1}{T}(\mathop{\mathbb{E}}_{\tau\sim\pi}[\sum_{t=1}^{T} f(s_t, a_t)] - \mathop{\mathbb{E}}_{\substack{\tau\sim\pi \\ a\sim\pi_E(s_t)}}[\sum_{t=1}^{T} f(s_t, a)])$$

When optimizing over the policy class Π which contains π_E, we have a minimax value of 0: for $j \in \{1, 2, 3\}$,

$$\min_{\pi\in\Pi} \max_{f\in\mathcal{F}} U_j(\pi, f) = 0$$

Furthermore, for certain representations of the policy,[5] strong duality holds: for $j \in \{1, 2, 3\}$,

$$\min_{\pi\in\Pi} \max_{f\in\mathcal{F}} U_j(\pi, f) = \max_{f\in\mathcal{F}} \min_{\pi\in\Pi} U_j(\pi, f)$$

We now study the properties that result from achieving an approximate equilibrium for each imitation game.

4. From Approximate Equilibria to Bounded Regret

A learner computing an equilibrium policy for any of the moment matching games will be imperfect due to many sources including restricted policy class, optimization error, or imperfect estimation of expert moments. More formally, in a game with payoff U_j, a pair $(\hat{\pi} \in \Pi, \hat{f} \in \mathcal{F})$ is a *δ-approximate equilibrium solution* if the following holds:

[3]The full policy class satisfies all these assumptions.

[4]Our results extend to infinite-dimensional Reproducing Kernel Hilbert Spaces.

[5]One option is a mixture distribution over class Π' where optimization is now performed over the mixture weights. This distribution can then be collapsed to a single policy (Syed et al. 2008). A second option is to optimize over the causal polytope $P(\mathbf{A}^T||\mathbf{S}^T)$, as defined in (Ziebart et al. 2010).

MOMENT MATCHED	UPPER BOUND	LOWER BOUND
REWARD	$O(\epsilon T)$	$\Omega(\epsilon T)$
OFF-POLICY Q	$O(\epsilon T^2)$	$\Omega(\epsilon T^2)$
ON-POLICY Q	$O(\epsilon HT)$	$\Omega(\epsilon T)$

Table 2. An overview of the difference in bounds between the three types of moment matching. All bounds are on imitation gap (1).

$$\sup_{f\in\mathcal{F}} U_j(f, \hat{\pi}) - \frac{\delta}{2} \leq U_j(\hat{f}, \hat{\pi}) \leq \inf_{\pi\in\Pi} U_j(\hat{f}, \pi) + \frac{\delta}{2}$$

We assume access to an *algorithmic primitive* capable of finding such strategies:

Definition 4. *An **imitation game** δ-**oracle** $\Psi\{\delta\}(\cdot)$ takes payoff function $U : \Pi \times \mathcal{F} \to [-k, k]$ and returns a $(k\delta)$-approximate equilibrium strategy for the policy player.*

We now bound the imitation gap of solutions returned by such an oracle.

4.1. Example MDPs

For use in our analysis, we first introduce two MDPs, LOOP and CLIFF. As seen in Fig. 2, LOOP is an MDP where a learner can enter a state where it has seen no expert demonstrations (s_2) and make errors for the rest of the horizon. CLIFF is an MDP where a single mistake can result in the learner being stuck in an absorbing state.

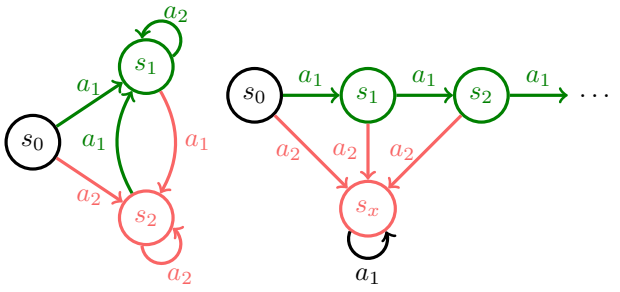

Figure 2. Left: Borrowed from (Ross et al. 2011), the goal of LOOP is to spend time in s_1. Right: a folklore MDP CLIFF, where the goal is to not "fall off the cliff" and end up in s_x evermore.

4.2. Reward Moment Performance Bounds

Let us first consider the reward moment-matching game.

Lemma 1. ***Reward Upper Bound:*** *If $\mathcal{F}_r$ spans $\mathcal{F}$, then for all MDPs, π_E, and $\pi \leftarrow \Psi\{\epsilon\}(U_1)$, $J(\pi_E) - J(\pi) \leq O(\epsilon T)$.*

Proof. We start by expanding the imitation gap:

$$
\begin{aligned}
&J(\pi_E) - J(\pi) \\
&\leq \sup_{f \in \mathcal{F}_r} \mathop{\mathbb{E}}_{\tau \sim \pi} \sum_{t=1}^{T} f(s_t, a_t) - \mathop{\mathbb{E}}_{\tau \sim \pi_E} \sum_{t=1}^{T} f(s_t, a_t) \\
&\leq \sup_{f \in \mathcal{F}} \mathop{\mathbb{E}}_{\tau \sim \pi} \sum_{t=1}^{T} 2f(s_t, a_t) - \mathop{\mathbb{E}}_{\tau \sim \pi_E} \sum_{t=1}^{T} 2f(s_t, a_t) \\
&= 2T \sup_{f \in \mathcal{F}} U_1(\pi, f) \leq 2T\epsilon
\end{aligned}
$$

The first line follows from the closure of $\mathcal{F}_r$ under negation. The last line follows from the definition of an ϵ-approximate equilibrium. □

In words, this bound means that in the worst case, we have an imitation gap that is $O(\epsilon T)$ rather than an imitation gap that compounds quadratically in time.

Lemma 2. ***Reward Lower Bound:*** *There exists an MDP, π_E, and $\pi \leftarrow \Psi\{\epsilon\}(U_1)$ such that $J(\pi_E) - J(\pi) \geq \Omega(\epsilon T)$.*

Proof. Consider CLIFF with a reward function composed of two indicators: $r(s, a) = -\mathbb{1}_{s_x} - \mathbb{1}_{a_2}$ and a perfect expert that never takes a_2. If with probability ϵ the learner's policy takes action a_2 only in s_0, the optimal discriminator would not only be able to penalize the learner for taking a_2 but also for the next $T - 1$ timesteps for being in s_x. Together, this would lead to an average cost of ϵ per timestep. Under r, this would make the learner ϵT worse than the expert, giving us $J(\pi_E) - J(\pi) = \epsilon T \geq \Omega(\epsilon T)$. □

Notably, both of these bounds are purely a function of the *game*, not the policy search algorithm and therefore apply for *all* algorithms that can be written in the form of a reward moment-matching imitation game. Our bounds do not depend on the size of the state space and therefore apply to continuous spaces, unlike those presented in (Rajaraman et al. 2020). Several recently proposed algorithms (Ho and Ermon 2016, Brantley et al. 2020, Spencer et al. 2021, Yang et al. 2020) including GAIL and SQIL can be understood as also solving this or a related game.

4.3. Off-Q Moment Performance Bounds

We contrast the preceding guarantees with those based on matching off-Q moments.

Lemma 3. ***Off-Q Upper Bound:*** *If $\mathcal{F}_Q$ spans $\mathcal{F}$, then for all MDPs, π_E, and $\pi \leftarrow \Psi\{\epsilon\}(U_2)$, $J(\pi_E) - J(\pi) \leq O(\epsilon T^2)$.*

Proof. Starting from the PDL:

$$
\begin{aligned}
&J(\pi_E) - J(\pi) \\
&\leq \sup_{f \in \mathcal{F}_\mathcal{Q}} \mathop{\mathbb{E}}_{\tau \sim \pi_E} [\sum_{t=1}^{T} \mathop{\mathbb{E}}_{a \sim \pi(s_t)} [f(s_t, a)] - f(s_t, a_t)] \\
&\leq \sup_{f \in \mathcal{F}} \mathop{\mathbb{E}}_{\tau \sim \pi_E} [\sum_{t=1}^{T} \mathop{\mathbb{E}}_{a \sim \pi(s_t)} [2Tf(s_t, a)] - 2Tf(s_t, a_t)] \\
&= 2T^2 \sup_{f \in \mathcal{F}} U_2(\pi, f) \leq 2\epsilon T^2
\end{aligned}
$$

The T in the second to last line comes from the fact that $\mathcal{F}_Q/2T \subseteq \mathcal{F}$. Thus, a policy π returned by $\Psi\{\epsilon\}(U_2)$ must satisfy $J(\pi_E) - J(\pi) \leq O(\epsilon T^2)$ – that is, it can do up to $O(\epsilon T^2)$ worse than the expert. □

Lemma 4. ***Off-Q Lower Bound:*** *There exists an MDP, π_E, and $\pi \leftarrow \Psi\{\epsilon\}(U_2)$ such that $J(\pi_E) - J(\pi) \geq \Omega(\epsilon T^2)$.*

Proof. Once again, consider CLIFF. If the learner policy instead takes a_2 with probability ϵT in s_0, the optimal discriminator would be able to penalize the learner up to ϵT for that timestep and ϵ on average. However, on rollouts, the learner would have an ϵT chance of paying a cost of 1 for the rest of the horizon, leading to a lower bound of $J(\pi_E) - J(\pi) = \epsilon T^2 \geq \Omega(\epsilon T^2)$. □

These bounds apply for *all* algorithms that can be written in the form of an off-Q imitation game, including behavioral cloning (Pomerleau 1989) and ValueDICE (Kostrikov et al. 2019).

4.4. On-Q Moment Performance Bounds

We now derive performance bounds for on-Q algorithms with interactive experts.

Lemma 5. ***On-Q Upper Bound:*** *If $\mathcal{F}_{Q_E}$ spans $\mathcal{F}$, then for all MDPs with H-recoverable $(\mathcal{F}_{Q_E}, \pi_E)$, and $\pi \leftarrow \Psi\{\epsilon\}(U_3)$, $J(\pi_E) - J(\pi) \leq O(\epsilon HT)$.*

Proof. Starting from the PDL:

$$
\begin{aligned}
&J(\pi_E) - J(\pi) \\
&\leq \sup_{f \in \mathcal{F}_{Q_E}} \mathop{\mathbb{E}}_{\tau \sim \pi} [\sum_{t=1}^{T} f(s_t, a_t) - \mathop{\mathbb{E}}_{a \sim \pi_E(s_t)} [f(s_t, a_t)]] \\
&\leq \sup_{f \in \mathcal{F}} \mathop{\mathbb{E}}_{\tau \sim \pi_E} [\sum_{t=1}^{T} 2Tf(s_t, a_t) - \mathop{\mathbb{E}}_{a \sim \pi_E(s_t)} [2Tf(s_t, a)]] \\
&= 2T^2 \sup_{f \in \mathcal{F}} U_3(\pi, f) \leq (\epsilon \frac{H}{2T}) 2T^2 = \epsilon HT
\end{aligned}
$$

As before, the T in the second to last line comes from the fact that $\mathcal{F}_{Q_E}/2T \subseteq \mathcal{F}$. The $H/2T$ factor comes from the scale of the payoff. Thus, a policy π returned by $\Psi\{\epsilon\}(U_3)$ must satisfy $J(\pi_E) - J(\pi) \leq O(\epsilon HT)$ – that is, it can do up to $O(\epsilon HT)$ worse than the expert. □

Lemma 6. ***On-Q Lower Bound:*** *There exists an MDP, π_E, and $\pi \leftarrow \Psi\{\epsilon\}(U_3)$ such that $J(\pi_E) - J(\pi) \geq \Omega(\epsilon T)$.*

Proof. The proof of the *reward lower bound* holds verbatim because every policy, including the previously considered π_E will be stuck in s_x after it falls in. □

As before, these bounds apply for all algorithms that can be written in the form of an on-Q imitation game, including DAgger (Ross et al. 2011) and AggreVaTe (Ross and Bagnell 2014). For example, in the bounds for AggreVaTe, Q_{max} is equivalent to the recoverability constant H.

4.5. Recoverability in Imitation Learning

The bounds we presented above beg the question of when on-Q moment matching has error properties similar to those

of reward moment matching versus those of off-Q moment matching. Recoverability allows us to cleanly answer this question and others. We begin by providing more intuition for said concept.

Concretely, in Fig. 2, LOOP is 1-recoverable for the expert policy that always moves towards s_1. CLIFF is not H-recoverable for any $H < T$ if the expert never ends up in s_x. A sufficient condition for H-recoverability is that the state occupancy distribution that results from taking an arbitrary action and then $H-1$ actions according to π_E is the same as that of taking H actions according to π_E. We emphasize that recoverability is a property of the *set of moments* matched and the expert, not just of the expert, as has been previously considered (Pan et al. 2019).

Bound Clarification. Our previously derived upper bound for on-Q moment matching ($J(\pi_E) - J(\pi) \leq O(\epsilon HT)$) tells us that for $O(1)$-recoverable MDPs, on-Q moment matching behaves like reward moment matching while for $O(T)$-recoverable MDPs, it instead behaves like off-Q moment matching and has an $O(\epsilon T^2)$ upper bound. Thus, $O(1)$-recoverability is in a certain sense necessary for achieving $O(\epsilon T)$ error with on-Q moment matching.

Another perspective on recoverability is that it helps us delineate problems where compounding errors are hard to avoid for both on-Q and reward moment matching. Let $l(s) = \sum_{a' \in \mathcal{A}} |\, \mathbb{E}_{a \sim \pi_E(s)}[\mathbb{1}_{a'}(a)] - \mathbb{E}_{a \sim \pi(s)}[\mathbb{1}_{a'}(a)]|$ be the classification error of a state s. We prove the following lemma in Supplementary Material A.1:

Lemma 7. *Let $\kappa > 0$. There exists a $(\pi_E, \{r, -r\})$ pair that for any $H < T$ is not H-recoverable such that $l(s) = \kappa$ on any state leads to $J(\pi_E) - J(\pi) \geq \Omega(\kappa T^2)$.*

Because some states might not appear on expert rollouts, evaluating $l(s)$ can require an interactive expert. However, even with this strong form of feedback, a classification error of κ on a single state can lead to $\Omega(\kappa T^2)$ imitation gap for $O(T)$-recoverable MDPs. This lemma also implies that in such an MDP, achieving $O(\kappa T)$ imitation gap via on-policy moment matching would require the learner to have a classification error $\propto \kappa/T$, or to make vanishingly rare errors as we increase the horizon. We note that this does not conflict with our previously stated bounds but reveals that achieving a moment-matching error of (time-independent) ϵ might require achieving a classification error that scales inversely with time for $O(T)$-recoverable MDPs. Practically, this can be rather challenging. Thus, neither on-Q nor reward moment matching is a silver bullet for getting $O(\epsilon T)$ error for $O(T)$-recoverable problems.

5. Finding Equilibria: Idealized Algorithms

We now provide reduction-based methods for computing (approximate) equilibria for these moment matching games which can be seen as blueprints for constructing our previously described oracle (Def. 4). We study, in particular, finite state problems and a complete policy class. We analyze an approach to equilibria finding where an outer player follows a *no-regret* strategy and the inner player follow a (modified) *best response* strategy, by which we can efficiently find policies with strong performance guarantees.

Algorithm 1 AdVIL

Input: Expert demonstrations $\mathcal{D}_E$, Policy class Π, Discriminator class $\mathcal{F}$, Performance threshold δ, Learning rates $\eta_f > \eta_\pi$
Output: Trained policy π
Set $\pi \in \Pi$, $f \in \mathcal{F}$, $L(\pi, f) = 2\delta$
while $L(\pi, f) > \delta$ **do**
  $L(\pi, f) = \mathbb{E}_{(s,a)\sim\mathcal{D}_E}[\mathbb{E}_{x\sim\pi(s)}[f(s,x)] - f(s,a)]$
  $f \leftarrow f + \eta_f \nabla_f L(\pi, f)$
  $\pi \leftarrow \pi - \eta_\pi \nabla_\pi L(\pi, f)$
end while

5.1. Preliminaries

An *efficient no-regret algorithm* over a class $\mathcal{X}$ produces iterates $x^1, \ldots, x^H \in \mathcal{X}$ that satisfy the following property for any sequence of loss functions $l^1, \ldots, l^H$:

$$\text{Regret}(H) = \sum_t^H l^t(x^t) - \min_{x \in \mathcal{X}} \sum_t^H l^t(x) \leq \beta_{\mathcal{X}}(H)$$

where $\beta_{\mathcal{X}}(H)/H \leq \epsilon$ holds for H that are $O(\text{poly}(\frac{1}{\epsilon}))$.

5.2. Theoretical Guarantees

We are interested in obtaining a policy efficiently that is a near-equilibrium solution to the game. We consider two general strategies to do so:

Primal. We execute a no-regret algorithm on the policy representation, while a maximization oracle over the space $\mathcal{F}$ computes the best response to those policies.

Dual. We execute a no-regret algorithm on the space $\mathcal{F}$, while a minimization oracle over policies computes *entropy regularized* best response policies.

The asymmetry in the above is driven by the need to recover the equilibrium strategy for the policy player and the fact that a dual approach on the original, unregularized objective $U_j(\cdot, f)$ will typical not converge to a single policy but rather shift rapidly as f changes.[6]

By choosing the the policy representation to be a causally conditioned probability distribution over actions,

[6]The average over iterations of the policies generated in an unregularized dual will also be near-equilibrium but can be inconvenient. Entropy regularization provides a convenient way to extract a single policy and meshes well with empirical practice.

$P(\mathbf{A}^T||\mathbf{S}^T) = \prod_{t=1}^{T} P(A_t|\mathbf{S}_{1:t}, \mathbf{A}_{1:t-1})$, we find each of the imitation games is *bilinear* in both policy and discriminator f and strongly dual .[7] Thus, we can efficiently compute a near-equilibrium assuming access to the optimization oracles in either *primal* or *dual* above:

Theorem 1. *Given access to the no-regret and maximization oracles in either **primal** or **dual** above, for all three imitation games we are able to compute a δ-approximate equilibrium strategy for the policy player in $poly(\frac{1}{\delta}, T, \ln|\mathcal{S}|, \ln|\mathcal{A}|)$ iterations of the outer player optimization.*

This result relies on a general lemma that establishes we can recover the equilibrium inner player by appropriately entropy regularizing that inner player's decisions. We prove both in Sup. A.

By substituting $\delta = \epsilon$ in Theorem 1 and combining with our previously derived bounds, we can cleanly show that these theorems can also be viewed as certificates of policy performance. Thus, our framework enables us to efficiently *close the imitation gap*.

6. Practical Moment Matching Algorithms

We now present three practical algorithms that match reward moments (`AdRIL`), off-Q moments (`AdVIL`), or on-Q moments (`DAeQuIL`). They are specific, implementable versions of our preceding abstract procedures. At their core, all three algorithms optimize an IPM. IPMs are a distance between two probability distributions (Müller et al. 1997):

$$\sup_{f \in \mathcal{F}} \mathop{\mathbb{E}}_{x \sim P_1}[f(x)] - \mathop{\mathbb{E}}_{x \sim P_2}[f(x)]$$

Plugging in the learner and expert trajectory distributions, we end up with our IPM-based objective:

$$\sup_{f \in \mathcal{F}} \sum_{t=1}^{T} \mathop{\mathbb{E}}_{\tau \sim \pi}[f(s_t, a_t)] - \mathop{\mathbb{E}}_{\tau \sim \pi_E}[f(s_t, a_t)] \tag{2}$$

As noted previously, this objective is equivalent to and inherits the strong guarantees of moment matching and allows our analysis to easily transfer.

6.1. `AdVIL`: Adversarial Value-moment Imitation Learning

We can transform our IPM-based objective (2) into an expression only over (s, a) pairs from expert data to fit into the off-Q moment matching framework by performing a series of substitutions. We refer interested readers to Supplementary Material B.1. We arrive at the following expression:

$$\sup_{v \in \mathcal{F}} \mathop{\mathbb{E}}_{\tau \sim \pi_E}[\sum_{t=1}^{T} \mathop{\mathbb{E}}_{a \sim \pi(s_t)}[v(s_t, a)] - v(s_t, a_t)] \tag{3}$$

Intuitively, by minimizing (3) over policies $\pi \in \Pi$, one is driving down the extra cost the learner has over the expert. We term this approach *Adversarial Value-moment Imitation Learning* (`AdVIL`) because if we view π as optimizing cumulative reward, $-v$ could be viewed as a value function. We set the learning rate for f to be greater than that for π, making `AdVIL` a primal algorithm.

Practically, `AdVIL` bears similarity to the Wasserstein GAN (WGAN) framework (Arjovsky et al. 2017), though we consider an IPM rather than the more restricted Wasserstein distances. However, the overlap is enough that techniques from the WGAN literature including gradient penalties on the discriminator (Gulrajani et al. 2017) and orthogonal regularization of the policy player (Brock et al. 2018) help.

Algorithm 2 `AdRIL`

Input: Expert demonstrations $\mathcal{D}_E$, Policy class Π, Dynamics $\mathcal{T}$, Kernel K, Performance threshold δ
Output: Trained policy π
Set $\pi \in \Pi$, $f = 0$, $\mathcal{D}_\pi = \{\}$, $\mathcal{D}' = \{\}$, $L(\pi, f) = 2\delta$
while $L(\pi, f) > \delta$ **do**
  $f \leftarrow \mathbb{E}_{\tau \sim D_\pi}[\sum_t K(sa, \cdot)] - \mathbb{E}_{\tau \sim D_E}[\sum_t K(sa, \cdot)]$
  $\pi, \mathcal{D}' \leftarrow \texttt{MaxEntRL}(\texttt{T} = \mathcal{T}, \texttt{r} = -f)$
  $\mathcal{D}_\pi \leftarrow \mathcal{D}_\pi \cup \mathcal{D}'$
  $L(\pi, f) = \mathbb{E}_{\tau \sim D'}[\sum_t f(s, a)] - \mathbb{E}_{\tau \sim D_E}[\sum_t f(s, a)]$
end while

6.2. `AdRIL`: Adversarial Reward-moment Imitation Learning

We now present our dual reward moment matching algorithm and refer interested readers to Supplementary Material B.2 for the full derivation. In brief, we solve for the discriminator in closed form via functional gradient descent in Reproducing Kernel Hilbert Space (RKHS) and have the policy player compute a best response via maximum entropy reinforcement learning. Let $\overline{\mathcal{D}_k}$ denote the aggregated dataset of policy rollouts. Assuming a constant number of training steps at each iteration and averaging functional gradients over k iterations of the algorithm, we get the cost function for the policy and round k:

$$\sum_{t=1}^{T} \frac{1}{|\overline{\mathcal{D}_k}|} \sum_{\tau}^{\overline{\mathcal{D}_k}} K([s_t, a_t], \cdot) - \frac{1}{|\mathcal{D}_E|} \sum_{\tau}^{\mathcal{D}_E} K([s_t, a_t], \cdot)$$

For an indicator kernel and under the assumption we never see the same state twice, this is equivalent to maximizing a reward function that is 1 at each expert datapoint,

[7] Following (Ziebart et al. 2010) we can represent the policy as an element of the causal conditioned polytope and regularize with the causal entropy $H(P(\mathbf{A^T}||\mathbf{S^T}))$ to denote the causal Shannon entropy of a policy. An equivalent result can be proved for optimizing over *occupancy measures* (Ho and Ermon 2016).

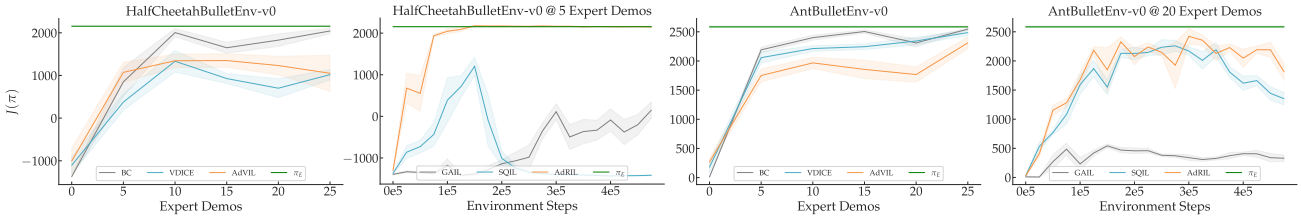

Figure 3. Our proposed methods (in orange) are able to match or out-perform the baselines across a variety of continuous control tasks. $J(\pi)$s are averaged across 10 evaluation episodes. Standard errors are across 5 runs of each algorithm.

$\propto \frac{-1}{k}$ along previous rollouts that do not perfectly match the expert, and 0 everywhere else. We term this approach *Adversarial Reward-moment Imitation Learning* (`AdRIL`).

We note that under these assumption, our objective resembles that of SQIL (Reddy et al. 2019). SQIL can be seen as a degenerate case of `AdRIL` that never updates the discriminator function. This oddity removes solution quality guarantees while introducing the need for early stopping (Arenz and Neumann 2020).

6.3. `DAeQuIL`: DAgger-esque Qu-moment Imitation Learning

We present the natural extension of DAgger (Ross et al. 2011) to the space of moments: `DAeQuIL` (DAgger-esque Qu-moment Imitation Learning) in Algorithm 3. Like DAgger, one can view `DAeQuIL` as a primal algorithm that uses Follow the Regularized Leader as the no-regret algorithm for the policy player. Per-round losses are adversarially chosen though. It is a subtle point but as written, `DAeQuIL` is technically not solving the on-Q game directly because it optimizes over a history of learner state distributions instead of the current learner's state distribution. However, it retains strong performance guarantees – see Sup. B.3 for more.

Algorithm 3 `DAeQuIL`

Input: Queryable expert π_E, Policy class Π, Discriminator class $\mathcal{F}$, Performance threshold δ, Behavioral cloning loss $\ell_{BC} : \Pi \to \mathbb{R}$, Strongly convex fn $R : \Pi \to \mathbb{R}$
Output: Trained policy π
Optimize: $\pi \leftarrow \arg\min_{\pi' \in \Pi} \ell_{BC}(\pi')$.
Set $L(\pi) = 2\delta$, $\mathcal{D} = []$, $F = []$, $t = 1$
while $L(\pi) > \delta$ **do**
  Rollout π to generate $\mathcal{D}_\pi \leftarrow [(s, a), \dots]$.
  Relabel $\mathcal{D}_\pi$ to $\mathcal{D}_E \leftarrow [(s, a') | a' \sim \pi_E(s), \forall s \in \mathcal{D}_\pi]$
  $L(f) = \mathbb{E}_{(s,a)\sim\mathcal{D}_\pi}[f(s, a)] - \mathbb{E}_{(s,a)\sim\mathcal{D}_E}[f(s, a)]$
  Append: $F \leftarrow F \cup [\arg\max_{f' \in \mathcal{F}} L(f')]$.
  Append: $\mathcal{D} \leftarrow \mathcal{D} \cup [(s, t) \,|\, \forall s \in \mathcal{D}_\pi]$.
  $L(\pi) = \mathbb{E}_{(s,t)\in\mathcal{D}}[F[t](s, \pi(s))] + \ell_{BC}(\pi) + R(\pi)$
  Optimize: $\pi \leftarrow \arg\min_{\pi' \in \Pi} L(\pi')$.
  $t \leftarrow t + 1$
end while

7. Experiments

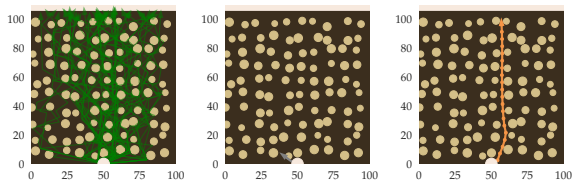

Figure 4. **Left:** The expert demonstrates many feasible trajectories, causing a learner that attempts to just match the mean action to crash directly into the tree the expert was avoiding. **Center:** On-policy corrections do not help DAgger as it still tries to match the mean action and crashes into the first tree it encounters. **Right:** `DAeQuIL`, when run with moments that allow the learner to focus on swerving out of the way of trees, regardless of direction, is able to produce policies that successfully navigate through the forest.

We test our algorithms against several baselines on several higher-dimensional continuous control tasks from the PyBullet suite (Coumans and Bai 2016–2019). We measure the performance of off-Q algorithms as a function of the amount of data provided with a fixed maximum computational budget and of reward moment-matching algorithms as a function of the amount of environment interactions. We see from Fig. 3 that `AdVIL` can match the performance of ValueDICE and Behavioral Cloning across most tasks. `AdRIL` performs better than GAIL across all environments and does not exhibit the catastrophic collapse in performance SQIL does on the tested environments. On both environments, behavioral cloning is able to recover the optimal policy with enough data, indicating there is little covariate shift (Spencer et al. 2021). However, on HalfCheetah, we see `AdRIL` recover a strong policy with far less data than it takes Behavioral Cloning to, showcasing the potential benefit of the learner observing the consequences of their own actions. We refer readers to Sup. C for a description of our hyparameters and setup. Notably, `AdVIL` is able to converge reliably to a policy of the same quality as that found by ValueDICE with an order of magnitude less compute. As seen in Fig. 4, `DAeQuIL` is able to significantly out-perform DAgger on a toy UAV navigation task – see Sup. D for full information.We release our code at **`https://github.com/gkswamy98/pillbox`**.

8. Discussion

8.1. A Unifying View of Moment Matching IL

We present a cohesive perspective of moment matching in imitation learning in Table 3. We note that reward moment-matching dual algorithms have been a repeated success story in imitation learning but that there has been comparatively less work done in off-Q and on-Q dual algorithms.

MOMENT	PRIMAL	DUAL
OFF-Q	VDICE, **ADVIL**	✗
REWARD	GAIL	MMP, LEARCH, MAXENT IOC, SQIL, **ADRIL**, A+N
ON-Q	DAGGER, GPS **IFAIL**, **DAEQUIL**	✗

Table 3. An taxonomy of moment matching algorithms. **Bold** text indicates algorithms that are IPM-based.

8.2. The Hidden Cost of Reward Moment Matching

At first glance, the reward moment matching bound might seem to good to be true – reward matching algorithms don't require a queryable expert like on-Q approaches yet their performance bound seems to be tighter. This better performance characteristic is a product of a potentially *exponentially* harder optimization problem for the learner. Consider the following tree-structured MDP with $|\mathcal{A}|$ actions at each step, each of which lead to a distinct state. Consider an

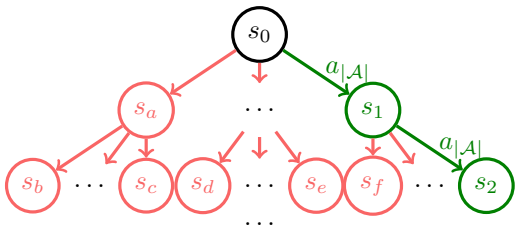

Figure 5. In TREE, the expert always takes the right-most action from the current node. The exponential number of T-length trajectories in this problem make it a challenge for reward moment matching approaches.

expert that takes $a_{|\mathcal{A}|}$ at each timestep. Solving the reward matching problem requires the learner to simultaneously optimize over all T timesteps of the problem while considering the effect of past actions on future states. If we set Π to be the class of deterministic policies, this is equivalent to optimizing over the set of all length T trajectories, of which there are $O(|\mathcal{A}|^T)$ in tree-structured problems. In contrast, for off-Q approaches, we attempt to match expert moments on a fixed expert state distribution. Similarly, we can optimize over a fixed history of past learner state distributions under a weak realizability assumption in the on-Q setting. In the preceding example, the Q-matching settings are like being handed a node at each level of the tree and being asked to choose between the $|\mathcal{A}|$ edges available, leading to a total of $O(|\mathcal{A}|T)$ options to choose between. As we saw in Sec. 5, the policy player sometimes needs to compute a best response over the entire set of choices it has available, which means these search space sizes directly affect the per-iteration complexity of the moment-matching algorithms. Concisely, the price we pay for solving an easier optimization problem is looser policy performance bounds.

8.3. Takeaways

In this work, we tease apart the differences in requirements and performance guarantees that come from matching reward, on-Q, and off-Q adversarially chosen moments. Reward moment matching has strong guarantees but requires access to an accurate simulator or the real world. Off-Q moment matching can be done purely on collected data but incurs an $O(\epsilon T^2)$ imitation gap.

We formalize a notion of *recoverability* that is both necessary and sufficient to understand recovering from errors in imitation learning. If a problem (due to expert or the MDP itself) is $O(T)$-recoverable, there exist problems where no algorithm can escape an $O(T^2)$ compounding of errors; if it is $O(1)$-recoverable, we find on policy algorithms prevent compounding. Together, these constitute a cohesive picture of moment matching in imitation learning.

We derive idealized no-regret procedures and practical IPM-based algorithms that are conceptually elegant and correct for difficulties encountered by prior methods. While behavioral cloning equally weights action-conditional errors, `AdVIL` can prevent headaches with value moment-based weighting. `AdRIL` is simple to implement, does not require training a GAN, and enjoys strong performance both in theory and practice. `DAeQuIL`'s moment-based losses are able to treat aches from focusing on action-conditionals that can congest DAgger's path on some problems.

Acknowledgments

We thank Siddharth Reddy for helpful discussions. We also thank Allie Del Giorno, Anirudh Vemula, and the members of the Social AI Group for their comments on drafts of this work. ZSW was supported in part by the NSF FAI Award #1939606, a Google Faculty Research Award, a J.P. Morgan Faculty Award, a Facebook Research Award, and a Mozilla Research Grant. Lastly, GS would like to thank John Steinbeck for inspiring the title of this work:

> *"As happens sometimes, a moment settled and hovered and remained for much more than a moment."* – John Steinbeck, Of Mice and Men.

References

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

Oleg Arenz and Gerhard Neumann. Non-adversarial imitation learning and its connections to adversarial methods, 2020.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan, 2017.

James Bagnell, Sham M Kakade, Jeff Schneider, and Andrew Ng. Policy search by dynamic programming. *Advances in neural information processing systems*, 16:831–838, 2003.

Paul Barde, Julien Roy, Wonseok Jeon, Joelle Pineau, Christopher Pal, and Derek Nowrouzezahrai. Adversarial soft advantage fitting: Imitation learning without policy optimization, 2020.

Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to end learning for self-driving cars. *CoRR*, abs/1604.07316, 2016. URL `http://arxiv.org/abs/1604.07316`.

Kiante Brantley, Wen Sun, and Mikael Henaff. Disagreement-regularized imitation learning. In *Eighth International Conference on Learning Representations (ICLR)*, April 2020.

Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. `http://pybullet.org`, 2016–2019.

Robert Dadashi, Léonard Hussenot, Matthieu Geist, and Olivier Pietquin. Primal wasserstein imitation learning. *arXiv preprint arXiv:2006.04678*, 2020.

Miroslav Dudik, Steven J Phillips, and Robert E Schapire. Performance guarantees for regularized maximum entropy density estimation. In *International Conference on Computational Learning Theory*, pages 472–486. Springer, 2004.

Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55 (1):119–139, 1997.

Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning, 2018.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in neural information processing systems*, pages 5767–5777, 2017.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning, 2016.

Daniel Jarrett, Ioana Bica, and Mihaela van der Schaar. Strictly batch imitation learning by energy-based distribution matching, 2020.

Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *In Proc. 19th International Conference on Machine Learning*. Citeseer, 2002.

Liyiming Ke, Matt Barnes, Wen Sun, Gilwoo Lee, Sanjiban Choudhury, and Siddhartha S. Srinivasa. Imitation learning as f-divergence minimization. *CoRR*, abs/1905.12888, 2019. URL `http://arxiv.org/abs/1905.12888`.

Ilya Kostrikov, Ofir Nachum, and Jonathan Tompson. Imitation learning via off-policy distribution matching, 2019.

Sergey Levine and Vladlen Koltun. Guided policy search. In *International Conference on Machine Learning*, pages 1–9, 2013.

David McAllester and Karl Stratos. Formal limitations on the measurement of mutual information, 2020.

K-R Müller, Alexander J Smola, Gunnar Rätsch, Bernhard Schölkopf, Jens Kohlmorgen, and Vladimir Vapnik. Predicting time series with support vector machines. In *International Conference on Artificial Neural Networks*, pages 999–1004. Springer, 1997.

Yunpeng Pan, Ching-An Cheng, Kamil Saigol, Keuntaek Lee, Xinyan Yan, Evangelos Theodorou, and Byron Boots. Imitation learning for agile autonomous driving. In *The International Journal of Robotics Research (IJRR)*, 2019.

Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *Advances in neural information processing systems*, pages 305–313, 1989.

Antonin Raffin. Rl baselines3 zoo. https://github.com/DLR-RM/rl-baselines3-zoo, 2020.

Antonin Raffin, Ashley Hill, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, and Noah Dormann. Stable baselines3. https://github.com/DLR-RM/stable-baselines3, 2019.

Nived Rajaraman, Lin F. Yang, Jiantao Jiao, and Kannan Ramachandran. Toward the fundamental limits of imitation learning, 2020.

Nathan D Ratliff, David Silver, and J Andrew Bagnell. Learning to search: Functional gradient techniques for imitation learning. *Autonomous Robots*, 27(1):25–53, 2009.

Siddharth Reddy, Anca D. Dragan, and Sergey Levine. Sqil: Imitation learning via reinforcement learning with sparse rewards, 2019.

Nicholas Rhinehart, Kris M Kitani, and Paul Vernaza. R2p2: A reparameterized pushforward policy for diverse, precise generative path forecasting. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 772–788, 2018.

Stephane Ross and J. Andrew Bagnell. Reinforcement and imitation learning via interactive no-regret learning, 2014.

Stephane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning, 2011.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587): 484–489, 2016.

Jiaming Song, Hongyu Ren, Dorsa Sadigh, and Stefano Ermon. Multi-agent generative adversarial imitation learning. *CoRR*, abs/1807.09936, 2018. URL http://arxiv.org/abs/1807.09936.

Jonathan Spencer, Sanjiban Choudhury, Arun Venkatraman, Brian Ziebart, and J. Andrew Bagnell. Feedback in imitation learning: The three regimes of covariate shift, 2021.

Wen Sun, Geoffrey J Gordon, Byron Boots, and J Bagnell. Dual policy iteration. *Advances in Neural Information Processing Systems*, 31:7059–7069, 2018.

Wen Sun, Anirudh Vemula, Byron Boots, and J. Andrew Bagnell. Provably efficient imitation learning from observation alone, 2019.

Gokul Swamy, Siddharth Reddy, Sergey Levine, and Anca D Dragan. Scaled autonomy: Enabling human operators to control robot fleets. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5942–5948. IEEE, 2020.

Umar Syed, Michael Bowling, and Robert E Schapire. Apprenticeship learning using linear programming. In *Proceedings of the 25th international conference on Machine learning*, pages 1032–1039, 2008.

Gerald Tesauro. Temporal difference learning and td-gammon. *Communications of the ACM*, 38(3):58–68, 1995.

Steven Wang, Sam Toyer, Adam Gleave, and Scott Emmons. The imitation library for imitation learning and inverse reinforcement learning. https://github.com/HumanCompatibleAI/imitation, 2020.

Fan Yang, Alina Vereshchaka, Yufan Zhou, Changyou Chen, and Wen Dong. Variational adversarial kernel learned imitation learning. 34:6599–6606, Apr. 2020. doi: 10.1609/aaai.v34i04.6135. URL https://ojs.aaai.org/index.php/AAAI/article/view/6135.

Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, 2008.

Brian D Ziebart, J Andrew Bagnell, and Anind K Dey. Modeling interaction via the principle of maximum causal entropy. 2010.

Supplementary Material

A. Proofs

A.1. Proof of Lemma 7

Proof. Consider the following MDP:

Figure 6. In UNICYCLE, the goal is to stay in s_1 forever.

As illustrated above, $c(s,a) = \mathbb{1}_{s_2}$ is the cost function for this MDP. Let the expert perfectly optimize this function by always taking a_1 in s_1. Thus, we are in the $O(T)$-recoverable setting. Then, for any $\epsilon > 0$, if the learner takes a_2 in s_1 with probability ϵ, $J(\pi_E) - J(\pi) = \sum_{t=1}^{T} \epsilon(1-\epsilon)^{t-1}(T-t) = \Omega(\epsilon T^2)$. There is only one action in s_2 so it is not possible to have a nonzero classification error in this state. □

A.2. Proof of Entropy Regularization Lemma

Lemma 8. ***Entropy Regularization Lemma:*** *By optimizing $U_j(\pi, f) - \alpha H(\pi)$ to a δ-approximate equilibrium, one recovers at worst a $Q_M\sqrt{\frac{2\delta}{\alpha}} + \alpha T(\ln|\mathcal{A}| + \ln|\mathcal{S}|)$ equilibrium strategy for the policy player on the original game.*

Proof. We optimize in the $P(\mathbf{A}^T||\mathbf{S}^T)$ policy representation where strong duality holds and define the following:

$$\pi^R = \arg\min_{\pi\in\Pi}(\max_{f\in\mathcal{F}} U_j(\pi, f) - \alpha H(\pi))$$

First, we derive a bound on the distance between $\hat{\pi}$ and π^R. We define M as follows:

$$M(\pi) = \max_{f\in\mathcal{F}} U_j(\pi, f) - \alpha H(\pi) + \alpha T(\ln|\mathcal{A}| + \ln|\mathcal{S}|)$$

M is an α-strongly convex function with respect to $||\cdot||_1$ because U is a max of linear functions, $-H$ is 1-strongly convex, and the third term is a constant. This tells us that:

$$M(\pi^R) - M(\hat{\pi}) \leq \nabla M(\pi^R)^T(\pi^R - \hat{\pi}) - \frac{\alpha}{2}||\pi^R - \hat{\pi}||_1^2$$

We note that because π^R minimizes M, the first term on the RHS is negative, allowing us to simplify this expression to:

$$\frac{\alpha}{2}||\pi^R - \hat{\pi}||_1^2 \leq M(\hat{\pi}) - M(\pi^R)$$

We now upper bound the RHS of this expression via the following series of substitutions:

$$\begin{aligned}
M(\hat{\pi}) &= \max_{f\in\mathcal{F}} U_j(\hat{\pi}, f) - \alpha H(\hat{\pi}) + \alpha T(\ln|\mathcal{A}| + \ln|\mathcal{S}|) \\
&\leq U_j(\hat{\pi}, \hat{f}) - \alpha H(\hat{\pi}) + \alpha T(\ln|\mathcal{A}| + \ln|\mathcal{S}|) + \delta \\
&\leq U_j(\pi^R, \hat{f}) - \alpha H(\pi^R) + \alpha T(\ln|\mathcal{A}| + \ln|\mathcal{S}|) + \delta \\
&\leq \max_{f\in\mathcal{F}} U_j(\pi^R, f) - \alpha H(\pi^R) + \alpha T(\ln|\mathcal{A}| + \ln|\mathcal{S}|) + \delta \\
&= M(\pi^R) + \delta
\end{aligned}$$

Rearranging terms to get the desired bound on strategy distance:

$$\begin{aligned}
&M(\hat{\pi}) - M(\pi^R) \leq \delta \\
&\Rightarrow ||\pi^R - \hat{\pi}||_1^2 \leq \frac{2\delta}{\alpha} \\
&\Rightarrow ||\pi^R - \hat{\pi}||_1 \leq \sqrt{\frac{2\delta}{\alpha}}
\end{aligned}$$

Next, we prove that π^R is a $\alpha T(\ln|\mathcal{A}|+\ln|\mathcal{S}|)$-approximate equilibrium strategy for the original, unregularized game. We note that $H(\pi) \in [0, T(\ln|\mathcal{A}| + \ln|\mathcal{S}|)]$ and then proceed as follows:

$$\begin{aligned}
\max_{f\in\mathcal{F}} U_j(\pi^R, f) &= M(\pi^R) + \alpha H(\pi^R) - \alpha T(\ln|\mathcal{A}| + \ln|\mathcal{S}|) \\
&\leq M(\pi^R) \\
&\leq \alpha T(\ln|\mathcal{A}| + \ln|\mathcal{S}|)
\end{aligned}$$

The last line comes from the fact that playing the optimal strategy in the original game on the regularized game could at worst lead to a payoff of $\alpha T(\ln|\mathcal{A}| + \ln|\mathcal{S}|)$. Therefore, the value of the regularized game can at most be this quantity. Recalling that the value of the original game is 0 and rearranging terms, we get:

$$\max_{f\in\mathcal{F}} U_j(\pi^R, f) - \alpha T(\ln|\mathcal{A}| + \ln|\mathcal{S}|) \leq 0 = \max_{f\in\mathcal{F}} \min_{\pi\in\Pi} U_j(\pi, f)$$

Thus by definition, π^R must be half of an $\alpha T(\ln|\mathcal{A}| + \ln|\mathcal{S}|)$-approximate equilibrium strategy pair.

Next, let Q_M denote the absolute difference between the minimum and maximum Q-value. For a fixed f, the maximum amount the policy player could gain from switching to policies within an L_1 ball of radius r centered at the original

policy is rQ_M by the bilinearity of the game and Hölder's inequality. Because the supremum over k-Lipschitz functions is known to be k-Lipschitz, this implies the same is true for the payoff against the best response f. To complete the proof, we can set $r = \sqrt{\frac{2\delta}{\alpha}}$ and combine this with the fact that π^R achieves in the worst case a payoff of $\alpha T(\ln|\mathcal{A}|+\ln|\mathcal{S}|)$ to prove that $\hat{\pi}$ can at most achieve a payoff of $Q_M\sqrt{\frac{2\delta}{\alpha}} + \alpha T(\ln|\mathcal{A}| + \ln|\mathcal{S}|)$ on the original game, which establishes $\hat{\pi}$ as a $(Q_M\sqrt{\frac{2\delta}{\alpha}} + \alpha T(\ln|\mathcal{A}| + \ln|\mathcal{S}|))$-approximate equilibrium solution.

□

A.3. Proof of Theorem 1

We proceed in cases.

Proof. We first consider the **primal case**. Our goal is to compute a policy $\hat{\pi}$ such that:

$$\max_{f\in\mathcal{F}} U_j(\hat{\pi}, f) \leq \delta$$

We prove that such a policy can be found efficiently by executing the following procedure for a polynomially large number of iterations:

1. For $t = 1 \dots N$, do:
   2. No-regret algorithm computes π^t.
   3. Set f^t to the best response to π^t.
4. Return $\hat{\pi} = \pi^{t^*}, t^* = \arg\min_t U_j(\pi^t, f^t)$.

Recall that via our no-regret assumption we know that

$$\frac{1}{N}\sum_t^N U_j(\pi^t, f^t) - \frac{1}{N}\min_{\pi\in\Pi}\sum_t^N U_j(\pi, f^t) \leq \frac{\beta_\Pi(N)}{N} \leq \delta$$

for some N that is poly$(\frac{1}{\delta})$. We can rearrange terms and use the fact that $\pi_E \in \Pi$ to upper bound the average payoff:

$$\frac{1}{N}\sum_t^N U_j(\pi^t, f^t) \leq \delta + \frac{1}{N}\min_{\pi\in\Pi}\sum_t^N U_j(\pi, f^t) \leq \delta$$

Using the property that there must be at least one element in an average that is at most the value of the average:

$$\min_t U_j(\pi^t, f^t) \leq \frac{1}{N}\sum_t^N U_j(\pi^t, f^t) \leq \delta$$

To complete the proof, we recall that f^t is chosen as the best response to π^t, giving us that:

$$\min_t \max_{f\in\mathcal{F}} U_j(\pi^t, f) \leq \delta$$

In words, this means that by setting $\hat{\pi}$ to the policy with the lowest loss out of the N computed, we are able to efficiently (within poly$(\frac{1}{\delta})$ iterations) find a δ-approximate equilibrium strategy for the policy player. Note that this result holds without assuming a finite $\mathcal{S}$ and $\mathcal{A}$ and does not require regularization of the policy. However, it requires us to have a no-regret algorithm over Π which can be a challenge for the reward moment-matching game.

We now consider the **dual case**. As before, we wish to find a policy $\hat{\pi}$ such that:

$$\max_{f\in\mathcal{F}} U_j(\hat{\pi}, f) \leq \delta$$

We run the following procedure on $U_j(\pi, f) - \alpha H(\pi)$:

1. For $t = 1 \dots N$, do:
   2. No-regret algorithm computes f^t.
   3. Set π^t to the best response to f^t.
4. Return $\hat{\pi} = \arg\min_{\pi\in\Pi} U_j(\pi, \bar{f}) - \alpha H(\pi)$.

By the classic result of (Freund and Schapire 1997), we know that the average of the N iterates produced by the above procedure (which we denote $\bar{f}$ and $\bar{\pi}$) is a δ'-approximate equilibrium strategy for some N that is poly$(\frac{1}{\delta'})$. Applying our Entropy Regularization Lemma, we can upper bound the payoff of $\bar{\pi}$ on the original game:

$$\sup_{f\in\mathcal{F}} U_j(\bar{\pi}, f) \leq Q_M\sqrt{\frac{2\delta'}{\alpha}} + \alpha T(\ln|\mathcal{A}| + \ln|\mathcal{S}|)$$

We now proceed similarly to our proof of the Entropy Regularization Lemma by first bounding the distance between $\bar{\pi}$ and $\hat{\pi}$ and the appealing to the Q_m-Lipschitzness of U_j. Let $l(\pi) = U_j(\pi, \bar{f}) - \alpha H(\pi)$. Then, while keeping the fact that l is α-strongly convex in mind:

$$\begin{aligned}
&l(\hat{\pi}) - l(\bar{\pi}) \leq \nabla l(\hat{\pi})^T(\hat{\pi} - \bar{\pi}) - \frac{\alpha}{2}||\bar{\pi} - \hat{\pi}||_1^2 \\
&\Rightarrow \frac{\alpha}{2}||\bar{\pi} - \hat{\pi}||_1^2 \leq l(\bar{\pi}) - l(\hat{\pi}) + \nabla l(\hat{\pi})^T(\hat{\pi} - \bar{\pi}) \\
&\Rightarrow \frac{\alpha}{2}||\bar{\pi} - \hat{\pi}||_1^2 \leq l(\bar{\pi}) - l(\hat{\pi}) \\
&\Rightarrow \frac{\alpha}{2}||\bar{\pi} - \hat{\pi}||_1^2 \leq \delta' \\
&\Rightarrow ||\bar{\pi} - \hat{\pi}||_1 \leq \sqrt{\frac{2\delta'}{\alpha}}
\end{aligned}$$

As before, the second to last step follows from the definition of a δ'-approximate equilibrium. Now, by the bilinearity of the game, Hölder's inequality, and the fact that supremum over k-Lipschitz functions is known to be k-Lipschitz, we can state that:

$$\sup_{f\in\mathcal{F}} U_j(\hat{\pi}, f) \leq 2Q_M\sqrt{\frac{2\delta'}{\alpha}} + \alpha T(\ln|\mathcal{A}| + \ln|\mathcal{S}|)$$

To ensure that the LHS of this expression is upper bounded by δ, it is sufficient to set $\alpha = \frac{\delta}{2T(\ln|\mathcal{A}|+\ln|\mathcal{S}|)}$ and $\delta' = \frac{\delta^2\alpha}{32Q_M^2}$. Plugging in these terms, we arrive at:

$$\sup_{f\in\mathcal{F}} U_j(\hat{\pi}, f) \leq \frac{\delta}{2} + \frac{\delta}{2} \leq \delta$$

We note that in practice, α is rather sensitive hyperparameter of maximum entropy reinforcement learning algorithms (Haarnoja et al. 2018) and hope that the above expression might provide some rough guidance for how to set α. To complete the proof, note that N is poly$(\frac{1}{\delta'})$ and $\frac{1}{\delta'} = \frac{64Q_M^2T(\ln|\mathcal{A}|+\ln|\mathcal{S}|)}{\delta^3}$. Thus, N is poly$(\frac{1}{\delta}, T, \ln|\mathcal{A}|, \ln|\mathcal{S}|))$.

□

B. Algorithm Derivations

B.1. AdVIL Derivation

We begin by performing the following substitution: $f = v - \mathcal{B}^\pi v$, where

$$\mathcal{B}^\pi v = \mathop{\mathbb{E}}_{\substack{s_{t+1}\sim\mathcal{T}(\cdot|s_t,a_t),\\ a_{t+1}\sim\pi(s_{t+1})}}[v]$$

is the expected Bellman operator under the learner's current policy. Our objective (2) then becomes:

$$\sup_{v\in\mathcal{F}} \sum_{t=1}^{T} \mathop{\mathbb{E}}_{\tau\sim\pi}[v(s_t,a_t) - \mathcal{B}^\pi v(s_t,a_t)] - \mathop{\mathbb{E}}_{\tau\sim\pi_E}[v(s_t,a_t) - \mathcal{B}^\pi v(s_t,a_t)]$$

This expression telescopes over time, simplifying to:

$$\sup_{v\in\mathcal{F}} \mathop{\mathbb{E}}_{\tau\sim\pi}[v(s_0,a_0)] - \sum_{t=1}^{T} \mathop{\mathbb{E}}_{\tau\sim\pi_E}[v(s_t,a_t) - \mathcal{B}^\pi v(s_t,a_t)]$$

We approximate $\mathcal{B}^\pi v$ via a single-sample estimate from the respective expert trajectory, yielding the following off-policy expression:

$$\sup_{v\in\mathcal{F}} \mathop{\mathbb{E}}_{\tau\sim\pi}[v(s_0,a_0)] - \sum_{t=1}^{T} \mathop{\mathbb{E}}_{\tau\sim\pi_E}[v(s_t,a_t) - \mathop{\mathbb{E}}_{a\sim\pi(s_{t+1})}[v(s_{t+1},a)]]$$

This resembles the form of the objective in ValueDICE (Kostrikov et al. 2019) but without requiring us to take the expectation of the exponentiated discriminator. We can further simplify this objective by noticing that trajectories generated by π_E and π have the same starting state distribution:

$$\sup_{v\in\mathcal{F}} \mathop{\mathbb{E}}_{\tau\sim\pi_E}[\sum_{t=1}^{T} \mathop{\mathbb{E}}_{a\sim\pi(s_t)}[v(s_t,a)] - v(s_t,a_t)] \tag{4}$$

We also note that this AdVIL objective can be derived straightforwardly via the Performance Difference Lemma.

B.2. AdRIL Derivation

Let $\mathcal{F}$ be a RKHS be equipped with kernel $K : (\mathcal{S}\times\mathcal{A}) \times (\mathcal{S}\times\mathcal{A}) \to \mathbb{R}$. On iteration k of the algorithm, consider a purely cosmetic variation of our IPM-based objective (2):

$$\sup_{c\in\mathcal{F}} \sum_{t=1}^{T} (\mathop{\mathbb{E}}_{\tau\sim\pi_k}[c(s_t,a_t)] - \mathop{\mathbb{E}}_{\tau\sim\pi_E}[c(s_t,a_t)]) = \sup_{c\in\mathcal{F}} L_k(c)$$

We evaluate the first expectation by collecting on-policy rollouts into a dataset $\mathcal{D}_k$ and the second by sampling from a fixed set of expert demonstrations $\mathcal{D}_E$. Assume that $|\mathcal{D}_k|$ is constant across iterations. Let $\mathcal{E}$ be the evaluation functional. Then, taking the functional gradient:

$$\begin{aligned}\nabla_c L_k(c) &= \sum_{t=1}^{T} \frac{1}{|\mathcal{D}_k|} \sum_{\tau}^{\mathcal{D}_k} \nabla_c \mathcal{E}[c;(s_t,a_t)] - \frac{1}{|\mathcal{D}_E|} \sum_{\tau}^{\mathcal{D}_E} \nabla_c \mathcal{E}[c;(s_t,a_t)] \\ &= \sum_{t=1}^{T} \frac{1}{|\mathcal{D}_k|} \sum_{\tau}^{\mathcal{D}_k} K([s_t,a_t],\cdot) - \frac{1}{|\mathcal{D}_E|} \sum_{\tau}^{\mathcal{D}_E} K([s_t,a_t],\cdot)\end{aligned}$$

where K could be an state-action indicator ($\mathbb{1}_{s,a}$) in discrete spaces and relaxed to a Gaussian in continuous spaces. Let $\overline{\mathcal{D}_k} = \bigcup_{i=0}^{k} \mathcal{D}_i$ be the aggregation of all previous $\mathcal{D}_i$. Averaging functional gradients over iterations of the algorithm (which, other than a scale factor that does not affect the optimal policy, is equivalent to having a constant learning rate of 1), we get the cost function our policy tries to minimize:

$$\begin{aligned}C(\pi_k) &= \sum_{i=0}^{k} \nabla_c L_i(c) \\ &= \sum_{t=1}^{T} \frac{1}{|\overline{\mathcal{D}_k}|} \sum_{\tau}^{\overline{\mathcal{D}_k}} K([s_t,a_t],\cdot) - \frac{1}{|\mathcal{D}_E|} \sum_{\tau}^{\mathcal{D}_E} K([s_t,a_t],\cdot)\end{aligned} \tag{5}$$

B.3. DAeQuIL Derivations

Let d_π denote the state-action visitation distribution of π. Then, DAeQuIL can be seen as Follow The Regularized Leader on the following sequence of losses:

1. $f_i = \arg\max_{f\in\mathcal{F}} \mathbb{E}_{s,a\sim d_{\pi_i}}[f(s,a) - f(s,\pi_E(s))]$
2. $l_i(\pi) = \mathbb{E}_{s\sim d_{\pi_i}}[f_i(s,\pi(s)) - f_i(s,\pi_E(s))]$

Solving the on-Q game proper would instead require $l_i'(\pi) = \mathbb{E}_{s\sim d_\pi}[f_i(s,\pi(s)) - f_i(s,\pi_E(s))]$ – for the state distribution to depend on the policy that is passed to the loss. While this would allow our previous no-regret analysis to apply as written, we would need to re-sample trajectories after every gradient step, a burden we'd like to avoid.

Let us consider the no-regret guarantee we get from the DAeQuIL losses:

$$\frac{1}{N}\sum_{t}^{N} l_t(\pi^t) - \frac{1}{N}\min_{\pi\in\Pi}\sum_{t}^{N} l_t(\pi) \leq \frac{\beta_\Pi(N)}{N} \leq \delta$$

Notice that $l_t(\pi^t) = \max_{f \in \mathcal{F}} U_3(\pi^t, f)$, the exact quantity we'd like to bound. The tricky part comes from the second term in the regret – under realizability, ($\pi_E \in \Pi$), this term is 0 and `DAeQuIL` directly finds a δ-approximate equilibrium for the on-Q game. Otherwise, we require the following weak notion of realizability to maintain the on-Q moment matching bounds: $\exists \pi' \in \Pi$ s.t.

$$\max_{d_\pi \in d_\Pi} \max_{f \in \mathcal{F}} \mathbb{E}_{s \sim d_\pi}[f(s, \pi'(a)) - f(s, \pi_E(a))] \leq O(\epsilon)$$

In words, this is saying that there exists a policy π' that can match expert moments up to ϵ on any state visitation distribution generated by a policy in Π. If we instead solved the on-Q game directly by using $l_i'(\pi)$, we would instead need the condition: $\exists \pi' \in \Pi$ s.t.

$$\max_{f \in \mathcal{F}} \mathbb{E}_{s \sim d_{\pi'}}[f(s, \pi'(a)) - f(s, \pi_E(a))] \leq O(\epsilon)$$

This weaker condition is concomitant with a much more computationally expensive optimization procedure.

C. Experimental Setup

C.1. Expert

We use the Stable Baselines 3 (Raffin et al. 2019) implementation of PPO (Schulman et al. 2017) and SAC (Haarnoja et al. 2018) to train experts for each environment, mostly using the already tuned hyperparameters from (Raffin 2020). Specifically, we use the modifications in Tables 4 and 5 to the Stable Baselines Defaults.

Parameter	Value
Buffer Size	300000
Batch Size	256
γ	0.98
τ	0.02
Training Freq.	64
Gradient Steps	64
Learning Rate	7.3e-4
Policy Architecture	256 x 2
State-Dependent Exploration	True
Training Timesteps	1e6

Table 4. Expert hyperparameters for HalfCheetah Bullet Task.

Parameter	Value
Buffer Size	300000
Batch Size	256
γ	0.98
τ	0.02
Training Freq.	64
Gradient Steps	64
Learning Rate	7.3e-4
Policy Architecture	256 x 2
State-Dependent Exploration	True
Training Timesteps	1e6

Table 5. Expert hyperparameters for Ant Bullet Task.

Env.	Expert	BC Performance
HalfCheetah	2154	2083
Ant	2585	2526

Table 6. With enough data (25 trajectories) and 100k steps of gradient descent, behavioral cloning is able to solve all tasks considered, replicating the results of (Spencer et al. 2021). However, other approaches are able to perform better when there is less data available.

C.2. Baselines

For all learning algorithms, we perform 5 runs and use a common architecture of 256 x 2 with ReLU activations. For each datapoint, we average the cumulative reward of 10 trajectories. For offline algorithms, we train on $\{5, 10, 15, 20, 25\}$ expert trajectories with a maximum of 500k iterations of the optimization procedure. For online algorithms, we train on a fixed number of trajectories (5 for HalfCheetah and 20 for Ant) for 500k environment steps. For GAIL (Ho and Ermon 2016) and behavioral cloning (Pomerleau 1989), we use the implementation produced by (Wang et al. 2020). We use the changes from the default values in Tables 6 and 7 for all tasks.

Parameter	Value
Entropy Weight	0
L2 Weight	0
Training Timesteps	5e5

Table 7. Learner hyperparameters for Behavioral Cloning.

For SQIL (Reddy et al. 2019), we build a custom implementation on top of Stable Baselines with feedback from the authors. As seen in Table 9, we use the similar parameters for SAC as we did for training the expert.

We modify the open-sourced code for ValueDICE (Kostrikov et al. 2019) to be actually off-policy with feedback from the authors. The publically available version of the ValueDICE code uses on-policy samples to compute a regularization term, even when it is turned off in the flags. We release our version.[8] We use the default hyperparameters for all experiments (and thus, train for 500k steps).

PARAMETER	VALUE
NUM STEPS	1024
EXPERT BATCH SIZE	32

Table 8. Learner hyperparameters for GAIL.

PARAMETER	VALUE
γ	0.98
τ	0.02
TRAINING FREQ.	64
GRADIENT STEPS	64
LEARNING RATE	LINEAR SCHEDULE OF 7.3E-4

Table 9. Learner hyperparameters for SQIL.

C.3. Our Algorithms

In this section, we use **bold text** to highlight sensitive hyperparameters. Similarly to SQIL, `AdRIL` is built on top of the Stable Baselines implementation of SAC. `AdVIL` is written in pure PyTorch. We use the same network architecture choices as for the baselines. For `AdRIL` we use the hyperparameters in Table 10 across all experiments.

PARAMETER	VALUE
γ	0.98
τ	0.02
TRAINING FREQ.	64
GRADIENT STEPS	64
LEARNING RATE	LINEAR SCHEDULE OF 7.3E-4
f **UPDATE FREQ.**	1250

Table 10. Learner hyperparameters for `AdRIL`.

We note that `AdRIL` requires careful tuning of f **Update Freq.** for strong performance. To find the value specified, we ran trials with $\{1250, 2500, 5000, 12500, 25000, 50000\}$ and selected the one that achieved the most stable updates. In practice, we would recommend evaluating a trained policy on a validation set to set this parameter. We also note because SAC is an off-policy algorithm, we are free to initialize the learner by adding all expert samples to the replay buffer at the start, as is done for SQIL.

We change one parameter between environments for `AdRIL` – for HalfCheetah, we perform standard sampling from the replay buffer while for Ant we sample an expert trajectory with $p = \frac{1}{2}$ and a learner trajectory otherwise, similar to SQIL. We find that for certain environments, this modification can somewhat increase the stability of updates while for other environments it can significantly hamper learner performance. We recommend trying both options if possible but defaulting to standard sampling.

[8] `https://github.com/gkswamy98/valuedice`

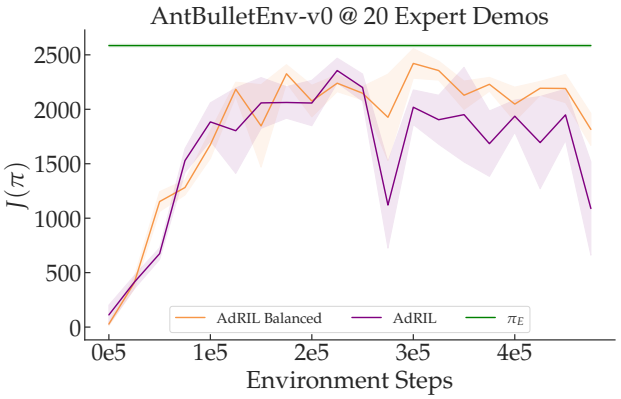

Figure 7. A comparison of balanced vs. unbalanced sampling for `AdRIL` on the Ant Environment. For certain tasks, balanced sampling can help with the stability of updates.

For `AdVIL`, we use the hyperparameters in Table 11 across all tasks. Emperically, small learning rates, large batch sizes, and regularization of both players are critical to stable convergence. We find that `AdVIL` converges significantly more quickly than ValueDICE, requiring only **50k steps for HalfCheetah and 100k Steps for Ant** instead of 500k steps for both tasks. However, we also find that running `AdVIL` for longer than these prescribed amounts can lead to a collapse of policy performance. Fortunately, this can easily be caught by watching for sudden and large fluctuations in policy loss after a long period of steady decreases. One can perform this early-stopping check without access to the environment.

PARAMETER	VALUE
η_π	8E-6
η_f	8E-4
BATCH SIZE	1024
f GRADIENT TARGET	0.4
f GRADIENT PENALTY WEIGHT	10
π ORTHOGONAL REGULARIZATION	1E-4
π MSE REGULARIZATION WEIGHT	0.2
NORMALIZE STATES WITH EXPERT DATA	TRUE
NORMALIZE ACTIONS TO [-1, 1]	TRUE
GRADIENT NORM CLIPPING	[-40, 40]

Table 11. Learner hyperparameters for `AdVIL`.

D. On-Q Experiments

We perform two experiments to tease out when one should apply `DAeQuIL` over DAgger. We first present results on a rocket-landing task from OpenAI Gym where behavioral cloning by itself is able to nearly solve the task, as has been

previously noted (Spencer et al. 2021). To make the task more challenging, we truncate the last two dimensions of the state for the policy class, which corresponds to masking the location of the legs of the lander. We use two-layer neural networks with 64 hidden units as all our function classes, perform the optimization steps via ADAM with learning rate $3e-4$, and sample 10 trajectories per update. Here, we see DAeQuIL do around as well as DAgger (Fig. 8), with both algorithms quickly learning a policy of quality equivalent to that of the expert. We list the full parameters of the algorithms in Tables 12 and 13. As in the previous section, **bold text** highlights sensitive hyperparameters.

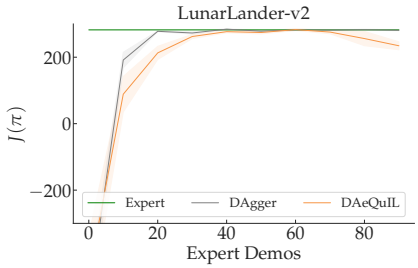

Figure 8. As behavioral cloning alone is able to nearly match the expert, DAgger and DAeQuIL perform around the same.

PARAMETER	VALUE
BATCH SIZE	32
GRADIENT STEPS π UPDATE	3E3
GRADIENT STEPS f UPDATE	**1E3**
f GRADIENT PENALTY TARGET	0
f GRADIENT PENALTY WEIGHT	5

Table 12. Learner hyperparameters for DAeQuIL on LunarLander-v2.

PARAMETER	VALUE
BATCH SIZE	32
GRADIENT STEPS π UPDATE	1E4

Table 13. Learner hyperparameters for DAgger on LunarLander-v2.

We next perform an experiment to show how careful curation of moments can allow DAeQuIL to significantly outperform DAgger at some tasks. Consider an operator trying to teach a drone to fly through a cluttered forest filled with trees. The operator has already trained a perception system that provides state information to the drone about whether a tree is infront of it. Because the operator is primarily concerned with safety, she only cares about making it through the forest, not the lateral location of the drone on the other side. She also tries to demonstrate a wide variety of evasive maneuvers as to hopefully teach the drone to generalize. We simulate such an operator and visualize the trajectories in Fig. 4, left.

Standard behavioral cloning with an ℓ_2 loss would fail at this task because it would attempt to reproduce the conditional mean action, leading the drone to fly straight into the tree. Unfortunately, DAgger inherits this flaw, and is therefore prone to producing a policy that crashes into the first tree it sees, as shown in Fig. 4, center.

For DAeQuIL, the operator leverages her knowledge of the problem and passes in two important moments: the perception system's imminent crash indicator and the absolute difference between the current and proposed headings. Whenever the former is on, the latter is a large value under the expert's distribution as they are trying to avoid the tree. So, the learner figures out that it should swerve out of the way of the tree. This leads to policies learned via DAeQuIL to be able to progress much further into the forest, as seen in Fig. 4, right.

Using the final position of executed trajectories as the cumulative reward, we see the following learning curves with DAeQuIL clearly out-performing DAgger (Fig. 9).

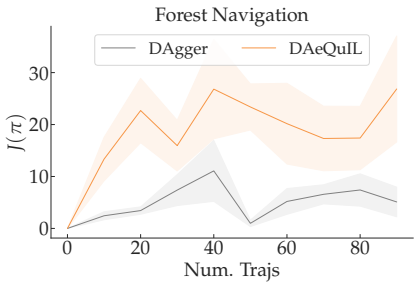

Figure 9. $J(\pi)$ is the longitudinal distance into the forest the learner is able to progress. All experiments are run on the forest layout shown in Fig. 4 and standard errors are computed across 10 trials.

We use the same function classes as the previous experiment but use a hidden size of 32 for the discriminator of DAeQuIL. We list the full set of parameters in Tables 14 and 15.

E. Additional Moment Types

E.1. A Fourth Moment Class: Mixed-Moment Value

We could instead plug in Q-moments to the reward moment payoff function U_1. Let $\mathcal{F}_V$ and $\mathcal{F}_{V_E}$ refer to the classes of policy and expert value functions. As before, we assume both of these classes are closed under negation and include the true value and expert value functions. For notational convenience, we assume both classes contain functions with

PARAMETER	VALUE
BATCH SIZE	32
GRADIENT STEPS π UPDATE	2E3
ℓ_{BC} **SCALE**	**5E-2**
GRADIENT STEPS f **UPDATE**	**1E3**
f GRADIENT PENALTY TARGET	0
f GRADIENT PENALTY WEIGHT	5

Table 14. Learner hyperparameters for `DAeQuIL` on Forest Navigation.

PARAMETER	VALUE
BATCH SIZE	32
GRADIENT STEPS π UPDATE	5E3

Table 15. Learner hyperparameters for DAgger on Forest Navigation.

type signatures $\mathcal{S} \times \mathcal{A} \to \mathbb{R}$, with the second argument being ignored. Starting from the PDL, we can expand as follows:

$$
\begin{aligned}
&J(\pi_E) - J(\pi) \\
&= \sum_{t=1}^{T} \mathop{\mathbb{E}}_{\tau\sim\pi_E}[Q_t^\pi(s_t,a_t) - \mathop{\mathbb{E}}_{a\sim\pi(s_t)}[Q_t^\pi(s_t,a)]] \\
&= \sum_{t=1}^{T} \mathop{\mathbb{E}}_{\tau\sim\pi_E}[Q_t^\pi(s_t,a_t) - \mathop{\mathbb{E}}_{a\sim\pi(s_t)}[Q_t^\pi(s_t,a)]] \\
&\qquad + \mathop{\mathbb{E}}_{\tau\sim\pi}[Q_t^\pi(s_t,a_t) - Q_t^\pi(s_t,a_t)] \\
&= \sum_{t=1}^{T} \mathop{\mathbb{E}}_{\tau\sim\pi_E}[Q_t^\pi(s_t,a_t)]] - \mathop{\mathbb{E}}_{\tau\sim\pi}[Q_t^\pi(s_t,a_t)]] \\
&\qquad + \mathop{\mathbb{E}}_{\substack{\tau\sim\pi\\ a\sim\pi(s_t)}}[Q_t^\pi(s_t,a)] - \mathop{\mathbb{E}}_{\substack{\tau\sim\pi_E\\ a\sim\pi(s_t)}}[Q_t^\pi(s_t,a)] \\
&\leq \sup_{f\in\mathcal{F}_Q\cup\mathcal{F}_V} 2\sum_{t=1}^{T} \mathop{\mathbb{E}}_{\tau\sim\pi}[f(s_t,a_t)] - \mathop{\mathbb{E}}_{\tau\sim\pi_E}[f(s_t,a_t)]
\end{aligned}
$$

The last step follows from the fact that $\sup_{a\in A} f(a) + \sup_{b\in B} f(b) \leq \sup_{c\in A\cup B} 2f(c)$. An analogous bound for $\mathcal{F}_{Q_E}$ and $\mathcal{F}_{V_E}$ can be proved by expanding the PDL in the reverse direction. We can use these expansions to provide bounds related to the reward-moment bound:

Lemma 9. ***Mixed Moment Value Upper Bound:*** *If $\mathcal{F}_Q/2T$ and $\mathcal{F}_V/2T$ spans $\mathcal{F}$ or $\mathcal{F}_{Q_E}/2T$ and $\mathcal{F}_{V_E}/2T$ do, then for all MDPs, π_E, and $\pi \leftarrow \Psi\{\epsilon\}(U_1)$, $J(\pi_E) - J(\pi) \leq O(\epsilon T^2)$.*

Proof. We start by expanding the imitation gap:

$$
\begin{aligned}
&J(\pi_E) - J(\pi) \\
&\leq \sup_{f\in\mathcal{F}_Q\cup\mathcal{F}_V} 2\sum_{t=1}^{T} \mathop{\mathbb{E}}_{\tau\sim\pi}[f(s_t,a_t)] - \mathop{\mathbb{E}}_{\tau\sim\pi_E}[f(s_t,a_t)] \\
&\leq \sup_{f\in\mathcal{F}} 2\mathop{\mathbb{E}}_{\tau\sim\pi}\sum_{t=1}^{T} 2Tf(s_t,a_t) - \mathop{\mathbb{E}}_{\tau\sim\pi_E}\sum_{t=1}^{T} 2Tf(s_t,a_t) \\
&= 4T^2 \sup_{f\in\mathcal{F}} U_1(\pi,f) \leq 4\epsilon T^2
\end{aligned}
$$

The T in the second to last line comes from the scaling down of either the $(\mathcal{F}_Q, \mathcal{F}_V)$ or the $(\mathcal{F}_{Q_E}, \mathcal{F}_{V_E})$ pairs by T to fit into the function class $\mathcal{F}$. □

Lemma 10. ***Mixed Moment Value Lower Bound:*** *There exists an MDP, π_E, and $\pi \leftarrow \Psi\{\epsilon\}(U_1)$ such that $J(\pi_E) - J(\pi) \geq \Omega(\epsilon T)$.*

Proof. The proof of the *reward lower bound* holds verbatim. □

These bounds show that solving this game, which might be more challenging than the reward-moment game, appears to offer no policy performance gains. However, in the imitation learning from observation alone setting, where one does not have access to action labels, reward-matching might be impossible, forcing one to use an approach similar to the above. This is because value functions are pure functions of state, not actions. (Sun et al. 2019) give an efficient algorithm for this setting.

E.2. Combining Reward and Value Moments

For both the *off-Q* and *on-Q* setups, one can leverage the standard expansion of a Q-function into a sum of rewards to derive a flexible family of algorithms that allow one to include knowledge of both reward and Q moments. Explicitly, for the off-Q case:

$$
\begin{aligned}
&J(\pi_E) - J(\pi) \\
&= \frac{1}{T}\Big(\mathop{\mathbb{E}}_{\substack{\tau\sim\pi_E\\ a\sim\pi(s_t)}}\Big[\sum_{t=1}^{T} Q^\pi(s_t,a) - Q^\pi(s_t,a_t)\Big]\Big) \\
&= \frac{1}{T}\Big(\mathop{\mathbb{E}}_{\substack{\tau\sim\pi_E\\ a\sim\pi(s_{t'})}}\Big[\sum_{t=1}^{T}\sum_{t'=1}^{T'} r(s_{t'},a) - r(s_{t'},a_{t'}) \\
&\quad + Q_{T'}^\pi(s_{T'},a) - Q_{T'}^\pi(s_{T'},a_{T'})\Big]\Big) \\
&\leq \max_{\substack{f\in\mathcal{F}_r\\ g\in\mathcal{F}_Q}} \frac{1}{T}\Big(\mathop{\mathbb{E}}_{\substack{\tau\sim\pi_E\\ a\sim\pi(s_t)}}\Big[\sum_{t=1}^{T}\sum_{t'=t}^{T'} f(s_{t'},a) - f(s_{t'},a_{t'}) \\
&\quad + g(s_{T'},a) - g(s_{T'},a_{T'})\Big]\Big) \qquad (6)
\end{aligned}
$$

Passing such a payoff to our oracle with $\mathcal{F}$ spanned by $\mathcal{F}_r/2 \times \mathcal{F}_Q/2T$ would recover the off-Q bounds.

This expansion begs the question of when it is useful. One answer is a standard bias/variance trade-off with different values of T', as has been explored in TD-Gammon (Tesauro 1995). We can provide an alternative answer by considering the limiting case – when the Q function is decomposed entirely into reward functions, the learner is required at timestep t to match the sum of future reward moments. An efficient algorithm for such a problem can be derived as a natural extension of Policy Search by Dynamic Programming (PSDP) (Bagnell et al. 2003), where, starting from $t = T - 1$, the learner matches expert moments one timestep in the future, before moving one step backwards in

time along the expert's trajectory. While this approach has the same performance characteristics as off-Q algorithms, matching the class of reward moments might be simpler for some types of problems, like those with sparse rewards. However, it has the added complexity of producing a non-stationary policy.

We can perform an analogous expansion for the on-Q case by utilizing the reverse direction of the PDL:

$$
\begin{aligned}
&J(\pi_E) - J(\pi) \\
&= \frac{1}{T}\Big(\mathop{\mathbb{E}}_{\substack{\tau\sim\pi \\ a\sim\pi_E(s_t)}} \Big[\sum_{t=1}^{T} Q^{\pi_E}(s_t, a_t) - Q^{\pi_E}(s_t, a)\Big]\Big) \\
&= \frac{1}{T}\Big(\mathop{\mathbb{E}}_{\substack{\tau\sim\pi \\ a\sim\pi_E(s_{t'})}} \Big[\sum_{t=1}^{T}\sum_{t'=1}^{T'} r(s_{t'}, a_{t'}) - r(s_{t'}, a) \\
&+ Q^{\pi_E}_{T'}(s_{T'}, a_{T'}) - Q^{\pi_E}_{T'}(s_{T'}, a)\Big]\Big) \\
&\leq \min_{\pi\in\Pi} \max_{\substack{f\in\mathcal{F}_r \\ g\in\mathcal{F}_{Q_E}}} \frac{1}{T}\Big(\mathop{\mathbb{E}}_{\substack{\tau\sim\pi \\ a\sim\pi_E(s_{t'})}} \Big[\sum_{t=1}^{T}\sum_{t'=t}^{T'} f(s_{t'}, a_{t'}) - f(s_{t'}, a) \\
&+ g(s_{T'}, a_{T'}) - g(s_{T'}, a)\Big]\Big) \qquad (7)
\end{aligned}
$$

Passing such a payoff to our oracle with $\mathcal{F}$ spanned by $\mathcal{F}_r/2 \times \mathcal{F}_{Q_E}/2T$ would recover the on-Q bounds. A backwards-in-time dynamic-programming procedure is not possible for this expansion because of the need to sample trajectories from the policy at previous timesteps.