Online Apprenticeship Learning

Lior Shani[1], **Tom Zahavy**[2], **Shie Mannor**[1,3]
[1] Technion – Israel Institute of Technology, Israel
[2] Deepmind, UK
[3] Nvidia Research, Israel
shanlior@gmail.com, tomzahavy@gmail.com, shie@ee.technion.ac.il

Abstract

In Apprenticeship Learning (AL), we are given a Markov Decision Process (MDP) without access to the cost function. Instead, we observe trajectories sampled by an expert that acts according to some policy. The goal is to find a policy that matches the expert's performance on some predefined set of cost functions. We introduce an online variant of AL (Online Apprenticeship Learning; OAL), where the agent is expected to perform comparably to the expert while interacting with the environment. We show that the OAL problem can be effectively solved by combining two mirror descent based no-regret algorithms: one for policy optimization and another for learning the worst case cost. By employing optimistic exploration, we derive a convergent algorithm with $O(\sqrt{K})$ regret, where K is the number of interactions with the MDP, and an additional linear error term that depends on the amount of expert trajectories available. Importantly, our algorithm avoids the need to solve an MDP at each iteration, making it more practical compared to prior AL methods. Finally, we implement a deep variant of our algorithm which shares some similarities to GAIL, but where the discriminator is replaced with the costs learned by OAL. Our simulations suggest that OAL performs well in high dimensional control problems.

1 Introduction

In Reinforcement Learning (Sutton and Barto 2018, RL) an agent interacts with an environment by following a policy. The environment is modeled as a Markov Decision Process (Puterman 1994, MDP), where in each state, the agent takes an action based on the policy, and as a result, pays a cost and transitions to a new state. The goal of RL is to learn an optimal policy that minimizes the long term cumulative cost. This makes RL useful when we can specify the MDP model appropriately. However, in many real-world problems, it is often hard to define a cost which induces the desired behaviour. E.g., an autonomous driver might suffer costs when driving slowly or in a hazardous way. Yet, prescribing these costs can be eluding.

A feasible solution to this problem is *Imitation Learning* (IL). This setup introduces the notion of an expert, typically a human, that provides us with a set of demonstrations. The agent's goal is to learn the optimal policy by imitating the expert's decisions. Methods such as *Behavioural Cloning* (BC) try to directly mimic the demonstrator by applying a supervised learning (SL) algorithm to learn a mapping from the states to actions. This literature is too vast to cover here and we refer the reader to (Schaal 1997; Argall et al. 2009).

Apprenticeship Learning (Abbeel and Ng 2004, AL) aims to address the same motivation using a different goal. Rather than learning a cost, its goal is to find a policy whose performance is close to that of the expert for *any possible cost* in a known set. This keeps the state-action occupancy of the agent and expert in proximity, requiring the AL agent to find a path back to the expert trajectories in states that are unobserved by the expert. This differs from BC, in which the agent's policy is undetermined in these unobserved states. Prior works on AL (Abbeel and Ng 2004; Syed and Schapire 2008; Zahavy et al. 2020) mostly considered a batch RL setting with the purpose of finding an ϵ-optimal solution, where the transition model is typically known or can be extracted from the data. However, in many real world applications, the model is unknown, and the learner is inflicted costs when performing poorly on the task, even if these costs are not properly specified to serve as an objective. This leads us to consider an online version of AL in which an agent should perform as close as possible to the expert on any possible cost, *while it is learning*. As a result, an *online* autonomous driver would try to imitate the expert *when learning in the real-world*, avoiding unnecessary costs.

AL is typically formulated as a min-max game between a policy and a cost "players". This problem was shown to be convex in the cost and in the feature expectations of the policy, but not in the policy itself. Abbeel and Ng (2004) proposed the projection algorithm, in which the policy player plays the best response and the cost player plays a follow-the-leader (FTL) step by utilizing the convexity in the feature expectations. Alternatively, in MWAL (Syed and Schapire 2008), the policy player applies a similar best response step and the cost player replace the FTL step with Mirror Descent. Unfortunately, this requires both algorithms to solve an MDP in each iteration (see Section 2.2).

Instead, in the convex games setting, min-max games can be approximately solved by simultaneously running two competing no-regret algorithms, preventing the inefficiency of finding the best response in each iteration (Abernethy and Wang 2017). This result builds on the notion of stability found in online convex optimization algorithms such as Mirror Descent (Beck and Teboulle 2003, MD). Interestingly, there has been a recent body of papers connecting policy optimization techniques and online convex optimization. Specifically, in (Geist, Scherrer, and Pietquin 2019; Shani, Efroni, and Mannor 2020), the authors prove global convergence for an MD-based policy optimization algorithm. Moreover, in (Cai et al. 2019; Efroni et al. 2020), the authors show that using Mirror Descent policy optimization together with optimistic exploration leads to no-regret policy optimization algorithms.

In this work, we take a similar approach and propose an online AL algorithm (OAL, pronounced Owl) that minimizes the AL regret: the difference between the agent's cumulative performance and that of the expert, for any possible cost. Our algorithm performs a dual MD step in which, (1) the policy is improved using one step of *optimistic* policy optimization MD-based update, and (2) the cost is updated by a single MD iteration. We show this leads to a *sample efficient* algorithm, even when the model is unknown. Importantly, our algorithm avoids solving an MDP in each iteration, making it *more practical* than previous approaches. Finally, we conduct an empirical study to verify the need for exploration in OAL.

To illustrate the benefits and practicality of our approach, we implement a deep RL variant of OAL, based on the Mirror Descent Policy Optimization (Tomar et al. 2020, MDPO) algorithm. Our deep OAL variant holds connection to the Generative Adversarial IL algorithm (Ho and Ermon 2016, GAIL): both OAL and GAIL use a generative cost function and take a single policy improvement step in each iteration. Differently from GAIL which learns a probabilistic discrimination between the policy and expert, OAL aims to optimize the value difference between the policy and expert, based on the min-max formulation. This is closely related to GAIL variants based on the Wasserstein distance (Xiao et al. 2019; Chen et al. 2020). Our experiments on continuous control tasks suggest that OAL is comparable to GAIL.

2 Preliminaries

In this work, we will deal with finite-horizon MDPs, defined by a tuple $\mathcal{M} := (\mathcal{S}, \mathcal{A}, p, c, H)$, where $\mathcal{S}, \mathcal{A}$ are the state and action spaces, respectively, and H is the length of an episode. $p_h(s' \mid s, a)$ is transition kernel describing the probability of transitioning to any state s', given the current state s and action a, for any $h \in [H]$. Similarly, $c_h(s, a)$ is the cost of applying action a at state s, during the h-th time-step. In adversarial MDPs, we allow the costs to change arbitrarily between episodes. A policy $\pi_h : \mathcal{S} \to \mathcal{A}$ is a mapping from state to action. The value function $V_h^{\pi,p,c}(s) = \mathbb{E}\Big[\sum_{t=h}^{H} c_h(s, a) \mid s_h = s, \pi\Big]$ is the cumulative expected costs of the agent, following π from state s at time-step h, over the MDP defined by the transition kernel p and costs c. Similarly, we define the Q-function, $Q_h^{\pi,p,c}(s, a) = \mathbb{E}\Big[\sum_{t=h}^{H} c_h(s, a) \mid s_h = s, a_h = a, \pi\Big]$. The occupancy measure $d_h^{\mu,\pi,p}(s, a) = \Pr(s_h = s, a_h = a \mid \mu, \pi, p)$ is the probability to reach state s and action a, at the h-th timestep, following π and starting from the initial distribution μ. Here throughout, we omit μ and assume without loss of generality that there exists a single starting state. Also, we omit p when clear from context. Notably, it holds for any π, that $\mathbb{E}^{s\sim\mu}[V_1^\pi(s)] = \langle c, d^\pi \rangle$, where $\langle c, d^\pi \rangle := \sum_{h=1}^{H} \langle c_h, d_h^\pi \rangle = \sum_{h,s,a} c_h(s, a) d_h^\pi(s, a)$. A mixed policy ψ over the set of all deterministic policies Π^{det} is executed by randomly selecting the policy $\pi_i \in \Pi^{det}$ at the beginning of an episode with probability $\psi(i)$, and exclusively following π_i thereafter. Finally, the filtration $\mathcal{F}_k$ includes all events in the k-th episode. We omit logarithmic factors when using the $O(\cdot)$ notation.

2.1 Mirror Descent in RL

The role of conservative updates in the convergence of policy optimization algorithms has been extensively studied in RL, going back to the analysis of the Conservative Policy Iteration (CPI) algorithm (Kakade and Langford 2002). Though sometimes motivated differently, the notion of conservative or stable updates is deeply related to ideas and analyses found in the convex optimization literature. Specifically, CPI can be considered an RL variant of the Frank-Wolfe (FW) algorithm (Scherrer and Geist 2014). Alternatively, the MD algorithm was also studied and applied to MDPs, allowing to provide theoretical guarantees for RL algorithms (Geist, Scherrer, and Pietquin 2019; Shani, Efroni, and Mannor 2020).

MD (Beck and Teboulle 2003) is a framework for solving convex optimization problems. At each iteration, the MD procedure minimizes the sum of a linear approximation of the current objective and a Bregman divergence term, aimed to keep consecutive iterates in proximity. For a set f_k of convex losses, and a constraint set $\mathcal{C}$, the k-th MD iterate is $x_{k+1} \in \arg\min_{x\in\mathcal{C}} \langle \nabla f_k(x)|_{x=x_k}, x - x_k \rangle + t_k B_\omega(x, x_k)$, where B_ω is a Bregman divergence and t_k is a step size. Finally, MD is known to be a no-regret online optimization algorithm (Hazan 2019). More formally, $\text{Reg}(K) := \max_x \sum_{k=1}^{K} f_k(x_k) - f_k(x) \le O(\sqrt{K})$.

The stability of the MD updates is crucial to obtain $O(\sqrt{K})$ regret in online optimization, where at each iteration the learner encounters an arbitrary loss function (Hazan 2019). This property was also exploited in RL to prove convergence in Adversarial MDPs, where the costs can change arbitrarily between episodes. Indeed, Neu, György, and Szepesvári (2010) provided such guarantees when the transition model is known. Recently, in (Cai et al. 2019; Efroni et al. 2020), the authors provided convergent MD policy optimization algorithms for adversarial MDPs when the model is unknown. These algorithms perform an *optimistic policy evaluation step to induce exploration*, followed by an MD policy update.

The benefits of MD in RL go beyond establishing convergence guarantees. Shani, Efroni, and Mannor (2020) shows that TRPO (Schulman et al. 2015), a widely used practical deep RL algorithm is actually an adaptation of the MD algorithm to MDPs. As a result, Tomar et al. (2020) derived Mirror Descent Policy Optimization (MDPO), a closer-to-theory deep RL algorithm based on the re-interpretation of TRPO, with on-policy and off-policy variants.

2.2 Apprenticeship Learning

In AL, we assume the existence of an *expert policy*, denoted by π^E. We assume access to N experts' trajectories sampled from π^E over the MDP, from which we construct an estimate of the occupancy measure d^E, denoted by $\hat{d}^E$. While the cost is unknown in AL, we assume it belongs to some set of costs $\mathcal{C}$. In the theoretical analysis, we focus on the following set:

$\mathcal{C}_b$ – **Bounded costs:** In this tabular case (Eq. (2.1)), the costs are of the form $c_h(s,a) \in [0,1], \forall h, s, a$.

In our experiments, we will refer to the following sets:

$\mathcal{C}_l$ – **Linear costs:** The states are assumed to be associated with features $\phi(s) \in [-1,1]^d$, and $\mathcal{C}_l$ is the costs that are linear in the features: i.e., $c(s) = w \cdot \phi(s)$. For any $w \in \mathcal{W}$, w is usually assumed to be the ℓ_2 unit ball (Abbeel and Ng 2004) or the simplex (Syed and Schapire 2008). The feature expectations of a policy π are $\Phi^\pi := \mathbb{E}_{d^\pi}\phi(s)$.

$\mathcal{C}_n$ – **Non-linear costs:** In this case the costs are some general non-linear function (typically a DNN) of the state features: $c(s) = f(\phi(s))$, where f is bounded. We will also consider the case that f is lipschitz continuous. In this case, AL is related to minimizing the Wasserstein distance between the agent and the expert (Zhang et al. 2020a,b).

The goal of AL is to find a policy π with good performance, relative to the expert, for any possible cost within a set $\mathcal{C}$,

$$\text{AL:} \ \arg\min_\pi \max_{c\in\mathcal{C}} \ \langle c, d^\pi\rangle - \langle c, d^E\rangle. \tag{2.1}$$

Previous works mostly focus on the space of mixed policies, and linear costs (see Section D for a discussion on differences between the tabular and linear setting). In this case, eq. (2.1) is equivalent to $\arg\min_{\psi\in\Psi}\max_{w\in\mathcal{W}} \ \langle w, \Phi^\psi\rangle - \langle w, \Phi^E\rangle$. Abbeel and Ng (2004) analyzed this objective when $\mathcal{W}$ is the euclidean unit ball. In this setup, it is possible to compute the **best response** for the cost (the maximizer over $\mathcal{W}$), exactly, for any ψ, and get that $w = \frac{\Phi^\psi - \Phi^E}{\|\Phi^\psi - \Phi^E\|}$. Plugging this back in the objective, we get that solving eq. (2.1) is equivalent to Feature Expectation Matching (FEM), i.e., minimizing $\|\Phi^\psi - \Phi^E\|^2$. To solve the FEM objective, the authors propose the projection algorithm. This algorithm starts with an arbitrary policy π_0 and computes its feature expectations Φ^{π_0}. At step t they fix a cost $w_t = \bar{\Phi}_{t-1} - \Phi^E$ and find the policy π_t that minimizes it, where $\bar{\Phi}_t$ is a convex combination of the feature expectations of previous (deterministic) policies $\bar{\Phi}_t = \sum_{j=1}^t \alpha_j \Phi^{\pi_j}$. They show that in order to get that $\left\|\bar{\Phi}_T - \Phi_E\right\| \le \epsilon$, it suffices to run the algorithm for $O(\frac{d}{\epsilon^2}\log(\frac{d}{\epsilon}))$ iterations (where d is features dimension).

Another type of algorithms, based on online convex optimization, was proposed by Syed and Schapire (2008). Similarly to the projection algorithm, the cost player plays a no-regret algorithm and the policy player plays the best response, i.e., it plays the policy π_t that minimizes the cost at time t. The algorithm runs for T steps and returns a mixed policy ψ that assigns probability $1/T$ to each policy π_t. In (Syed and Schapire 2008), the authors prove their scheme is faster than the projection algorithm (Abbeel and Ng 2004), requiring only $O(\log(d)/\epsilon^2)$ iterations. This improvement follows from the analysis of MD and specifically the Multiplicative Weights algorithm (Freund and Schapire 1997; Littlestone and Warmuth 1994), giving the algorithm its name, Multiplicative Weights AL (MWAL).

Both types of AL algorithms we have described are based on the concept of solving the min-max game when one of the players plays the best response: the policy player in MWAL, and the cost player in the projection algorithm. The main limitation in implementing these algorithms in practice is that they both require to solve an MDP in each iteration.

3 Online Apprenticeship Learning

In this work, we study an online version of AL where an agent interacts with an environment with the goal of imitating an expert. Our focus is on algorithms that are sample efficient in the number of interactions with the environment. This is different from prior batch RL work (Abbeel and Ng 2004; Syed and Schapire 2008; Zahavy et al. 2020) which mostly focused on PAC bounds on the amount of optimization iterations needed to find an ϵ-optimal solution and typically assumed that the environment is known (or that the expert data is sufficient to approximate the model). Our formulation, on the other hand, puts emphasis on the performance of the agent *while it is learning*, which we believe is important in many real world applications.

Formally, we measure the performance of an online AL algorithm via the *regret* of the learning algorithm w.r.t the expert. In standard RL, when the costs are known, the regret of a learner is defined as the difference between the expected accumulated values of the learned policies and the value of the optimal policy (Jaksch, Ortner, and Auer 2010). However, in the absence of costs, the optimal policy is not defined. Therefore, it is most natural to compare the performance of the learner to the expert. With Eq. (2.1) in mind, this leads us to introduce the regret in Definition 1, which measures the *worst-case* difference between the accumulated values of the learner and the expert, *over all possible costs* in $\mathcal{C}$:

Definition 1 (Apprenticeship Learning Regret)**.** *The regret of an AL algorithm is:*

$$Reg_{AL}(K) := \max_{c\in\mathcal{C}} \sum_{k=1}^K \Big[V_1^{\pi_k,c} - V_1^{\pi^E,c}\Big], \tag{3.1}$$

Definition 1 suggests a notion of regret from the perspective of comparison to the expert as a reference policy. Instead, as an optimization problem, the regret of (2.1) is measured w.r.t. to its optimal solution, $\mathrm{Reg}(K) := \max_{c\in\mathcal{C}}\sum_{k=1}^K\langle c, d^{\pi^k} - d^E\rangle - \min_\pi \max_{c\in\mathcal{C}}\sum_{k=1}^K\langle c, d^\pi - d^E\rangle$. Importantly, in the following lemma, we show the two regret definitions coincide:

Lemma 1. *The online regret of the AL optimization problem* (2.1) *and the AL regret are equivalent.*

3.1 Online Apprenticeship Learning Scheme

In Algorithm 1, we present a scheme for solving the AL problem using online optimization tools. Specifically, we introduce a min-player to solve the minimization problem in eq. (2.1). This min-player is an RL agent that aims to find the optimal policy in an adversarial MDP in which a max-player chooses the cost in each round. In Section 3.2, we show that simultaneously optimizing both the policy and cost using no-regret algorithms leads to a *sample efficient* no-regret AL algorithm (see Definition 1). Our approach *averts the need*

Algorithm 1: OAL Scheme

```
1: for $k = 1, ..., K$ do
2:     Rollout a trajectory by acting $\pi_k$
3:     # Evaluation Step
4:     Evaluate $Q^{\pi_k}$ using the current cost $c^k$
5:     Evaluate $\nabla_c L(\pi^k, c; \pi^E)|_{c=c^k}$
6:     # Policy Update
7:     Update $\pi^{k+1}$ by an MD policy update with $Q^{\pi^k}$
8:     # Costs Update
9:     Update $c^{k+1}$ by an MD step on $\nabla_c L(\pi^k, c)|_{c=c^k}$
```

to solve an MDP in each iteration, as was typically done in previous work (see the discussion in Section 2.2), and therefore vastly reduces the computational complexity of the algorithm and makes it more practical.

This is attained in the following manner. Each OAL iteration consists of two phases: **(1) evaluation phase**, in which the gradients of the objective w.r.t. the policy and costs are estimated, and **(2) optimization phase**, where both the policy and cost are updated by two separated MD iterates (see Section 2.1).To specify the updates, we need calculate the gradient of the AL objective w.r.t. to the policy or cost, and choose an appropriate Bregman divergence.

Policy update. Because V^{π^E, c^k} does not depend on the current policy, the optimization objective is just V^{π, c^k}, which is the exactly the RL objective w.r.t. to the current costs. Thus, the gradient of the AL objective w.r.t. policy is the Q-function of the current policy and costs. The KL-divergence is a natural choice for the Bregman term, when optimizing over the set of stochastic policies (Shani, Efroni, and Mannor 2020). Using the stepsize t_k^π, the OAL policy update is

$$\pi_h^{k+1} \in \arg\min_{\pi} \langle Q_h^{\pi_k, c_k}, \pi_h \rangle + t_k^\pi d_{KL}(\pi_h || \pi_h^k). \quad (3.2)$$

Notably, this update only requires to evaluate the current Q-function, and does not to solve an MDP.

Cost update. Denoting the cost AL objective $L(\pi, c) := -(V_1^{\pi,c}(s_1) - V_1^{\pi^E,c}(s_1))$, the gradient w.r.t. the cost is $\nabla_c L(\pi, c)|_{c=c^k}$. The preferable choice of Bregman depends the cost set $\mathcal{C}$. We use the euclidean norm, but other choices are also possible. With stepsize t_k^c, the OAL cost update is

$$c^{k+1} \in \arg\min_{c} \langle \nabla_c L(\pi^k, c)|_{c^k}, c \rangle + \frac{t_k^c}{2} \left\| c - c^k \right\|^2. \quad (3.3)$$

In the next section, we use the scheme of Algorithm 1 to develop a no-regret AL algorithm.

3.2 Convergent Online Apprenticeship Learning

The updates of OAL in Eqs. (3.2) and (3.3) rely on the exact evaluation of Q^{π_k, c_k} and $\nabla L(\pi, c; \pi^E)|_{c^k}$. However, in most cases, the transition model is unknown, and therefore, assuming access to these quantities is unrealistic. Nevertheless, we now introduce Algorithm 2, an OAL variant which provably minimizes the AL regret (see Definition 1) in the tabular settings, without any restrictive assumptions. Intuitively, an AL agent should try and follow the expert's path. However, due to the environment's randomness and the possible scarcity of expert's demonstrations, it can stray afar from such path. Thus, it is crucial to explore the environment to learn a policy which keeps proximity to that of the expert (see the discussion in Section 4). To this end, Algorithm 2 uses *optimistic UCB-bonuses* to explore the MDP.

The policy player has access to the costs of all state-action pairs, in each iteration. Thus, from the policy player perspective, it interacts with an adversarial MDP with full information of the costs and unknown transitions. To solve this MDP, at each iteration, Algorithm 2 improves the policy by applying an MD policy update w.r.t. a UCB-based optimistic estimation of the current Q-function (Line 15), relying on the techniques of Cai et al. (2019). The UCB bonus, b_h^{k-1}, added to the costs, accounts for the uncertainty in the estimation of the transitions, driving the policy to explore (Line 8).

The cost player update in the tabular setting is given by $\nabla_c L(\pi^k, c)|_{c=c^k} = d^E - d^{\pi^k}$, which can be evaluated using the learned model. We use costs of the form $c \in \mathcal{C}_b$, which make the cost optimization in Eq. (3.3) separable at each time-step, state and action. In this case, the euclidean distance is a natural candidate for the choice of Bregman divergence, reducing the MD update to (1) performing a gradient step towards the difference between the expert's and the agent's probability of encountering the specific state-action pair, and (2) projecting the result back to $[0, 1]$ (Lines 17 and 18). We are now ready to state our **main theoretical result:**

Theorem 1. *The regret of the OAL algorithm (Algorithm 2) satisfies with probability of* $1 - \delta$,

$$Reg_{AL}(K) \leq O\Big(\sqrt{H^4 S^2 AK} + \sqrt{H^3 SAK^2/N}\Big).$$

The regret bound in Theorem 1 consists of two terms. The first term shows an $O(\sqrt{K})$ rate similar to the optimal regret of solving an MDP. Perhaps surprisingly, the fact that we solve the AL problem for *any possible costs* does not hurt the sample efficiency of our algorithm. The second term is a statistical error term due to the fact that we only have a limited amount of expert data. This error is independent of the AL algorithm used, and is a reminder of the fact we would like to mimic the expert itself and not the data. In this sense, it is closely related to the generalization bound in (Chen et al. 2020), discussed in Section 4. When expert data is scarce, it could be hard to mimic the true expert's policy, and the linear error dominates the bound. Still, when the amount of experts' trajectories is of the order of the number of environment interactions, $N \propto K$, the dominant term becomes $O(\sqrt{K})$.

Recall that previous AL results require to solve an MDP in each update, and therefore, their bounds on the amount of iterations refers to the *amount of times an MDP is solved*. In stark contrast, Algorithm 2 avoids solving an MDP, and the regret bound measures the *interactions with the MDP*.

In what follows we give some intuition regarding the proof of Theorem 1. The full proof is found in Appendix A. In the proof, we adapt the the analysis for solving repeated games using Online Optimization (Freund and Schapire 1999; Abernethy and Wang 2017) to the min-max AL problem. This allows us to prove the following key inequality (Lemma 2, Appendix A), which bounds the AL regret:

$$\mathrm{Reg}_{AL}(K) \le \underbrace{\mathrm{Reg}_\pi(K)}_{(i)} + \underbrace{\mathrm{Reg}_c(K)}_{(ii)} + \underbrace{2K\max_c |\langle c, d^E - \hat{d}^E\rangle|}_{(iii)}$$

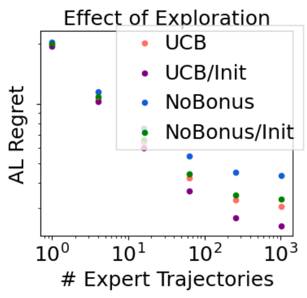
Algorithm 2: Online Apprenticeship Learning (OAL)

```
1:  for $k = 1, ..., K$ do
2:      Rollout a trajectory by acting $\pi_k$
3:      Estimate $\hat{d}_k$ using the empirical model $\bar{p}^{k-1}$
4:      # Policy Evaluation
5:      $\forall s \in \mathcal{S},\ V^k_{H+1}(s) \leftarrow 0$
6:      for $\forall h = H, .., 1, s \in \mathcal{S}, a \in \mathcal{A}$ do
7:          $Q^k_h(s,a) \leftarrow (c^k_h - b^{k-1}_h + \bar{p}^{k-1}_h V^k_{h+1})(s,a)$
8:          $Q^k_h(s,a) \leftarrow \max\{Q^k_h(s,a), 0\}$
9:          $V^k_h(s) \leftarrow \langle Q^k_h(s,\cdot), \pi^k_h(\cdot \mid s)\rangle$
10:     # Update Step
11:     for $\forall h, s, a \in [H] \times \mathcal{S} \times \mathcal{A}$ do
12:         # Policy Update
13:         $\pi^{k+1}_h(a \mid s) \propto \pi^k_h(a \mid s)\exp\big(-t^\pi_k Q^k_h(s,a)\big)$
14:         # Costs Update
15:         $c^{k+1}_h(s,a) \leftarrow c^k_h(s,a) + t^c_k\big(\hat{d}^k_h - \hat{d}^E_h\big)(s,a)$
16:         $c^{k+1}_h(s,a) \leftarrow \mathrm{Clip}\{c^{k+1}_h(s,a), 0, 1\}$
17:     Update counters and empirical model, $n_k, \bar{p}^k$
```

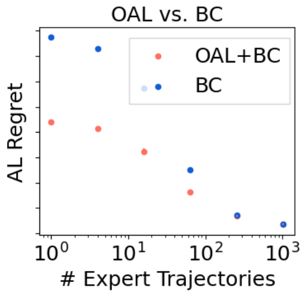
Importantly, this decomposes the regret of the policy and cost players, enabling the algorithm to perform the updates in (3.2) and (3.3) separately. We address each of the terms:
Term (i). The regret of the policy player in the adversarial MDP defined by the *known* costs $\{c_k\}_{k=1}^K$. By applying the MD-based policy update (Eq. (3.2)) with an optimistic Q-function in Algorithm 2, we follow the analysis in (Cai et al. 2019; Efroni et al. 2020) to bound this term by $O\big(\sqrt{H^4S^2AK}\big)$ (Lemma 4, Appendix A).
Term (ii). The regret of the cost player, $\max_c \sum_{k=1}^K \langle c, d^{\pi_k,p} - \hat{d}^E\rangle - \sum_{k=1}^K \langle c^k, d^{\pi_k,p} - \hat{d}^E\rangle$. Following MD updates in (3.3) w.r.t. the estimated occupancy measure of the current policy, this term is bounded by $O\big(\sqrt{H^4S^2AK}\big)$ (Lemma 5, Appendix A).
Term (iii). This describes the discrepancy between the expert's data and the true expert. It is bounded using Hoeffding's inequality, leading to the linear regret part of Theorem 1, $O\big(\sqrt{H^3SAK^2/N}\big)$.

4 Discussion and Related Work

The role of exploration in AL. A key component in our analysis for proving Theorem 1, is using a UCB cost bonus to induce exploration. *But should the agent explore at all, if its goal is to follow an expert?* Indeed, with infinite amount of expert trajectories, deriving a stochastic policy $\hat{\pi}^E$ using $\hat{d}^E$ (which is equivalent to BC without function approximation), allows for an exact retrieval of π^E, leading to zero regret *without any exploration*. Also, in a slightly different setting where the model is unknown and the true costs are observed, Abbeel and Ng (2005) provided a polynomial PAC sample complexity guarantee for imitating an expert. In contrast to our approach, they argue there is no need to encourage the algorithm to explore. Thus, intuitively, it might seem that the exploration procedure used in OAL wastefully forces the agent to explore unnecessary regions of the state-space.



(a)



(b)

Figure 1: (a) Exploration and minimizing the AL regret. (b) Comparison between BC and AL with BC initialization.

However, when the number of expert trajectories is finite, directly deriving $\hat{\pi}$ from $\hat{d}^\pi$ can often lead to states unseen in the data, resulting in undetermined policies in these states which can cause an unwanted behaviour. Instead, *the goal of AL is to learn policies which try to stick as close as possible to the experts trajectories*, even when unobserved states are encountered. Still, when the expert trajectories are abundant, the model can be accurately estimated in states that the expert can visit, mitigating the need for exploration. Indeed, the guarantee in (Abbeel and Ng 2005) (only) holds in this regime, when the number of trajectories is of the order of $O(1/\epsilon^3)$, where ϵ is the acceptable error.

In stark contrast, our bounds for AL do not rely on any assumption on the number of expert trajectories. As a result, our work suggests that *when expert trajectories are scarce, one do need exploration to learn the transition model efficiently*, due to the fact that the model cannot be accurately estimated even in states that the expert policy might reach. Note that even in the regime where expert data is abundant, the algorithm in (Abbeel and Ng 2005) requires $O(1/\epsilon^5)$ MDP interactions to converge, which is roughly comparable to $O(K^{4/5})$ regret. In this sense, our algorithm achieves $O(\sqrt{K})$ regret, and hence requires much less interactions, in the more intricate AL setting and for any amount of trajectories.

To further address this discussion, we empirically tested the necessity of exploration for different amounts of experts' trajectories, by running Algorithm 2 with and without UCB bonuses in a tabular MDP. A fixed amount of episodes was used in all runs. For any number of trajectories, the plot was averaged over 400 seeds, and the error bars represent 95% confidence intervals. The full experimental details are found in Section E. The results in Figure 1a show that the AL regret is consistently lower when using optimistic bonuses. This also holds when initializing the transition model in OAL by estimating it using the expert trajectories. This suggests that exploration is crucial to optimize the online performance of OAL. Moreover, Figure 1a shows that when more expert trajectories are available, the regret decreases down to a fixed value corresponding to the second term in Theorem 1.

Finally, Zhang et al. (2020b) elegantly proved PAC convergence for an algorithm that resembles the updates in Eqs. (3.2) and (3.3), using neural networks for approximation. In their work, they assume bounded Radon-Nikodym derivatives, which is similar to having finite concentrability coefficients (Kakade and Langford 2002). This bypasses the need to explore by assuming the agent policies can always reach any state that the expert reaches. By employing proper exploration, we refrained from such an assumption when proving the regret bound in Theorem 1 for the simpler tabular case.

On the generalization of OAL. Chen et al. (2020) analyzed the generalization of an AL-like algorithm in the average cost setting, which is defined as the gap between how the learned policy performs w.r.t. the true expert and its performance w.r.t the expert demonstrations. They show the generalization error depends on $O(\sqrt{(\log \mathcal{N})/N})$, where $\mathcal{N}$ is the covering number of the cost class. Specifically, when using $\mathcal{C}_b$ as in Algorithm 2, this becomes $O(\sqrt{HSA/N})$, which matches the dependence on SA/N in the linear term of Theorem 1. This result implies that even in the tabular case, different cost classes can lead to improved generalization. However, in cost classes other than $\mathcal{C}_b$, the projection required to solve Eq. (3.3) can be much harder. Still, it is valuable to understand how different cost classes affects the performance of OAL when the expert trajectories are limited.

Differences from BC. Instead of minimizing the value difference directly, BC algorithms directly minimize the zero-one or ℓ_1 loss between the agent and expert policies. A main caveat of this approach is that it fails to treat states unobserved by the expert. In this case, an ϵ error can lead to a value difference of $H^2\epsilon$ (Ross and Bagnell 2010). This can be improved by further assumptions: Ross and Bagnell (2010) assume that one can query the expert online; Brantley, Sun, and Henaff (2019) add an external mechanism to attract the agent towards the expert state-action distribution, and assume the agent is concentrated around the expert. Concretely, Rajaraman et al. (2020) shows that additional knowledge of the transition model is required to avoid this compounding error. Instead, learning a policy close to the state-action distribution of the expert is a core built-in mechanism in AL algorithms, exploiting the transition model to optimize the value-difference directly for any possible cost. This prevents the unwanted behaviour experienced in BC algorithms without relying on any additional assumptions. Another difference between the two approaches, is that analyses of BC algorithms focus on the performance difference between the final learned policy and the optimal policy (Ross and Bagnell 2010). Yet, this does not capture the online performance of the algorithm. Instead, our work uses the more common form of regret which measures the difference between the *online performance of the agent* and the best policy. Notably, many BC algorithms do interact with the environment online (e.g. (Ross and Bagnell 2010; Brantley, Sun, and Henaff 2019)), and it could be useful to analyze them with a similar criterion.

Finally, note that the two paradigms can be used in a complementary fashion. We ran an experiment to compare BC and Algorithm 2 with BC initialization (See Section E for details). The results in Figure 1b show that using OAL together with BC leads to an improved online regret. Yet, when expert trajectories are abundant, BC is sufficient to learn the expert policy, as discussed in the beginning of this section.

5 Deep Online Apprenticeship Learning

We now present a practical implementation of Algorithm 1 using Deep RL algorithms. We implement two separate modules, a policy and a generative costs module, both of which are updated based on the MD updates in Eqs. (3.2) and (3.3).

For policy optimization, we use MDPO (Tomar et al. 2020), an MD-based off-policy deep RL algorithm. To update the policy, MDPO approximately solves (3.2) by performing several SGD steps w.r.t. its objective, keeping the target policy fixed. This enforces the stability of the policy updates required by Algorithm 1.

For the generative costs, we consider two modules. (1) A **linear costs generator** based on $\mathcal{C}_l$ (see Section 2.2). Using this set of costs, Eq. (3.3) can be solved in close-form. (2) A **neural network costs generator** (see $\mathcal{C}_n$ in Section 2.2). Note that any cost in this set must be bounded, so it can serve as a cost of an MDP. In theory, this is easily achieved by the projection step in Eq. (3.3), which corresponds to clipping the cost to reside within the set. Yet, when using neural networks, this clipping procedure can hurt the gradient flow. Instead, we use different techniques to keep the cost bounded. First, we penalize the network's output to be close to zero, to effectively limit the size of the costs. Second, we apply the technique proposed in (Gulrajani et al. 2017) to enforce a Lipschitz constraint on the costs. Specifically, we use a convex sum of state-actions pairs encountered by the agent's policy and the expert as an input to the costs network, and penalize the costs updates so that the gradient of the costs w.r.t. to the input would be close to 1. Third, we perform several gradient steps on Eq. (3.3) to force the updated costs to be close to the old ones, instead of updating the costs using gradient steps w.r.t. to the AL objective (this technique is only applied in the on-policy case discussed in Section 5.1). This prevents the costs from diverging too quickly. Finally, the costs given to the policy player are clipped.

5.1 Experiments

(Ho and Ermon 2016) pioneered the idea of solving the AL problem without solving an MDP in each step. To this end, they propose GAIL, an AL algorithm inspired by generative adversarial networks (GAN; Goodfellow et al. 2014). GAIL uses a neural network, which learns to differentiate between the policy and the expert using the GAN loss, as a *surrogate* for the AL problem. In turn, the GAN loss is given as the cost of the MDP. In this section, we demonstrate that it is possible to *directly solve the AL paradigm* using either *a linear or a NN-based family of costs*, by following the OAL scheme.

Experimental Setup. We evaluated deep OAL (Section 5) on the MuJoCo (Todorov, Erez, and Tassa 2012) set of continuous control tasks. To show the online convergence properties of AL algorithms, we present the full learning curves. We used 10 expert trajectories in all our experiments, roughly the average amount in (Ho and Ermon 2016; Kostrikov et al.

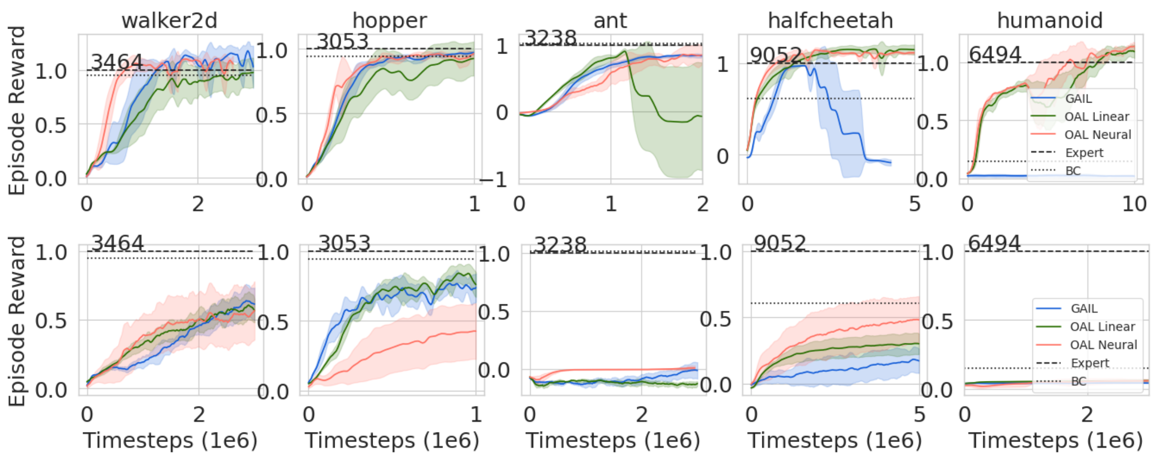


Figure 2: OAL vs. GAIL. Policy optimizer used: (Top) off-policy MDPO; (Bottom) on-policy TRPO.

2018). We tested OAL with both linear and neural costs (see Section 5), and compared them with GAIL. The same policy and cost networks were used for OAL and GAIL.

Our theoretical analysis dictates to optimize the policy using stable updates. Thus, we used two policy optimization algorithms applying the MD update: (1) on-policy TRPO, which can be seen as a hard-constraint version of MDPO (Shani, Efroni, and Mannor 2020). (2) off-policy MDPO, which directly solves the policy updates in Eq. (3.2).

The experimental results in Figure 2 show that both the linear (green) and neural (orange) versions of OAL are successful at imitating the expert. We turn to analyze the results:

OAL vs. GAIL. The results in Figure 2 show both the linear (green) and neural (orange) versions of OAL outperform GAIL (blue), implying it is not necessary to introduce a discriminator in AL. This holds independently on the policy optimization algorithm. Note that the performance drop of GAIL in "Humanoid" can be explained by the fact that Ho and Ermon (2016) had to increase the amount of MDP interactions and expert trajectories in this environment.

Neural vs. Linear. Surprisingly, the *linear version* (green) of OAL performs almost as good as the neural one (orange). This comes with the additional benefits that linear rewards are more interpretable, they do not require to design and tune an architecture, and are faster to compute. Our results suggest that linear costs might be sufficient for solving the AL problem even in complex environment, countering the intuition and empirical results found in (Ho and Ermon 2016). There, the authors argue that the main pitfall of AL is its reliance on a predetermined structured cost, which does not necessarily contains the true MDP cost. However, even if the *true cost* cannot be perfectly represented by a linear function, it might still be sufficient to obtain an *optimal policy*.

MDPO vs. TRPO. Inspecting Figure 2, one can see that the off-policy MDPO version (top) significantly outperforms the on-policy TRPO (bottom) on all three algorithms. This can be attributed to two reasons: First, in our analysis, the MD policy update in Eq. (3.3) is required for efficiently solving the AL problem. MDPO is explicitly designed to optimize this policy update and therefore closer to theory. Instead, TRPO only implicitly solves this equation. Note that Ho and Ermon (2016) motivated using TRPO in GAIL as preventing noisy Q-function estimates. Our work suggests *the need for stable policy updates* as an alternative motivation. Second, as was reported in other works (Kostrikov et al. 2018; Blondé and Kalousis 2019), using GAIL together with an off-policy policy algorithm allows a significant boost in data efficiency. Our results strongly imply a similar conclusion.

On Lipschitz Costs. In Figure 3, we study the dependence on the Lipschitz regularization coefficient in the HalfCheetah-v3 domain. Our results implies that restricting the cost to be Lipschitz is important for OAL. Interestingly, in (Kostrikov et al. 2018; Blondé and Kalousis 2019) the authors apply the same regularization for GAIL, even though Lipschitzness is not necessarily required in GAIL. Indeed, Figure 3 suggests that enforcing this regularity condition increases the stability of both GAIL and OAL. However, when used in GAIL, this technique might hurt convergence speed. Interestingly, Xiao et al. (2019) showed that solving the AL problem with Lipschitz costs is similar to GAIL with a Wasserstein distance between the occupancy measures of the agent and expert. They employ several regularizations to enforce Lipschitzness and optimize the policy using TRPO. Deep OAL is different from their implementation in the following ways: they focus only on on-policy scenario using TRPO; they enforce Lipschitzness by L_2-regularization, while we regularize the cost network gradients as proposed in (Gulrajani et al. 2017).

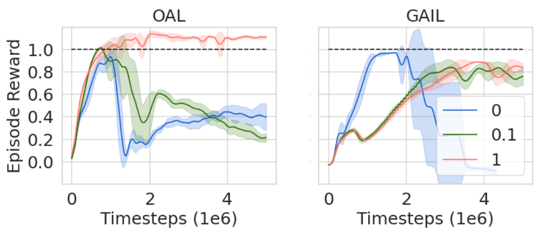


Figure 3: The effect of the Lipschitz regularization.

Acknowledgments

This research was partially supported by the ISF under contract 2199/20.

References

Abbeel, P.; and Ng, A. Y. 2004. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, 1. ACM.

Abbeel, P.; and Ng, A. Y. 2005. Exploration and apprenticeship learning in reinforcement learning. In *Proceedings of the 22nd international conference on Machine learning*, 1–8.

Abernethy, J. D.; and Wang, J.-K. 2017. On Frank-Wolfe and Equilibrium Computation. In *NIPS*, 6584–6593.

Argall, B. D.; Chernova, S.; Veloso, M.; and Browning, B. 2009. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5): 469–483.

Beck, A.; and Teboulle, M. 2003. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31: 167–175.

Blondé, L.; and Kalousis, A. 2019. Sample-efficient imitation learning via generative adversarial nets. In *The 22nd International Conference on Artificial Intelligence and Statistics*, 3138–3148. PMLR.

Brantley, K.; Sun, W.; and Henaff, M. 2019. Disagreement-regularized imitation learning. In *International Conference on Learning Representations*.

Cai, Q.; Yang, Z.; Jin, C.; and Wang, Z. 2019. Provably Efficient Exploration in Policy Optimization. *arXiv preprint arXiv:1912.05830*.

Chen, M.; Wang, Y.; Liu, T.; Yang, Z. Y.; Li, X.; Wang, Z.; and Zhao, T. 2020. On Computation and Generalization of Generative Adversarial Imitation Learning. In *International Conference on Learning Representations*.

Dann, C.; Lattimore, T.; and Brunskill, E. 2017. Unifying PAC and regret: Uniform PAC bounds for episodic reinforcement learning. In *Advances in Neural Information Processing Systems*, 5713–5723.

Dhariwal, P.; Hesse, C.; Klimov, O.; Nichol, A.; Plappert, M.; Radford, A.; Schulman, J.; Sidor, S.; Wu, Y.; and Zhokhov, P. 2017. OpenAI Baselines. https://github.com/openai/baselines.

Efroni, Y.; Merlis, N.; Ghavamzadeh, M.; and Mannor, S. 2019. Tight regret bounds for model-based reinforcement learning with greedy policies. In *Advances in Neural Information Processing Systems*, 12203–12213.

Efroni, Y.; Shani, L.; Rosenberg, A.; and Mannor, S. 2020. Optimistic Policy Optimization with Bandit Feedback. *arXiv preprint arXiv:2002.08243*.

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

Freund, Y.; and Schapire, R. E. 1999. Adaptive game playing using multiplicative weights. *Games and Economic Behavior*, 29(1-2): 79–103.

Geist, M.; Scherrer, B.; and Pietquin, O. 2019. A Theory of Regularized Markov Decision Processes. In *International Conference on Machine Learning*, 2160–2169.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. *Advances in neural information processing systems*, 27: 2672–2680.

Gulrajani, I.; Ahmed, F.; Arjovsky, M.; Dumoulin, V.; and Courville, A. C. 2017. Improved training of wasserstein gans. In *Advances in neural information processing systems*, 5767–5777.

Hazan, E. 2019. Introduction to online convex optimization. *arXiv preprint arXiv:1909.05207*.

Hill, A.; Raffin, A.; Ernestus, M.; Gleave, A.; Kanervisto, A.; Traore, R.; Dhariwal, P.; Hesse, C.; Klimov, O.; Nichol, A.; Plappert, M.; Radford, A.; Schulman, J.; Sidor, S.; and Wu, Y. 2018. Stable Baselines. https://github.com/hill-a/stable-baselines.

Ho, J.; and Ermon, S. 2016. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, 4565–4573.

Jaksch, T.; Ortner, R.; and Auer, P. 2010. Near-optimal Regret Bounds for Reinforcement Learning. *Journal of Machine Learning Research*, 11(4).

Jin, C.; Yang, Z.; Wang, Z.; and Jordan, M. I. 2020. Provably efficient reinforcement learning with linear function approximation. In *Conference on Learning Theory*, 2137–2143. PMLR.

Kakade, S.; and Langford, J. 2002. Approximately optimal approximate reinforcement learning. In *ICML*, volume 2, 267–274.

Kostrikov, I.; Agrawal, K. K.; Dwibedi, D.; Levine, S.; and Tompson, J. 2018. Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning. *arXiv preprint arXiv:1809.02925*.

Littlestone, N.; and Warmuth, M. K. 1994. The weighted majority algorithm. *Information and computation*, 108(2): 212–261.

Neu, G.; György, A.; and Szepesvári, C. 2010. The Online Loop-free Stochastic Shortest-Path Problem. In *COLT*, volume 2010, 231–243. Citeseer.

Orabona, F. 2019. A Modern Introduction to Online Learning. *arXiv preprint arXiv:1912.13213*.

Osband, I.; Blundell, C.; Pritzel, A.; and Van Roy, B. 2016. Deep exploration via bootstrapped DQN. *arXiv preprint arXiv:1602.04621*.

Puterman, M. L. 1994. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

Rajaraman, N.; Yang, L.; Jiao, J.; and Ramchandran, K. 2020. Toward the Fundamental Limits of Imitation Learning. *Advances in Neural Information Processing Systems*, 33.

Ross, S.; and Bagnell, D. 2010. Efficient reductions for imitation learning. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 661–668.

Schaal, S. 1997. Learning from demonstration. In *Advances in neural information processing systems*, 1040–1046.

Scherrer, B.; and Geist, M. 2014. Local policy search in a convex space and conservative policy iteration as boosted policy search. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 35–50. Springer.

Schulman, J.; Levine, S.; Abbeel, P.; Jordan, M.; and Moritz, P. 2015. Trust region policy optimization. In *International conference on machine learning*, 1889–1897.

Shani, L.; Efroni, Y.; and Mannor, S. 2020. Adaptive trust region policy optimization: Global convergence and faster rates for regularized mdps. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 5668–5675.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press.

Syed, U.; and Schapire, R. E. 2008. A game-theoretic approach to apprenticeship learning. In *Advances in neural information processing systems*, 1449–1456.

Todorov, E.; Erez, T.; and Tassa, Y. 2012. MuJoCo: A physics engine for model-based control. In *IEEE International Conference on Intelligent Robots and Systems*, 5026–5033.

Tomar, M.; Shani, L.; Efroni, Y.; and Ghavamzadeh, M. 2020. Mirror Descent Policy Optimization. *arXiv preprint arXiv:2005.09814*.

Weissman, T.; Ordentlich, E.; Seroussi, G.; Verdu, S.; and Weinberger, M. J. 2003. Inequalities for the L1 deviation of the empirical distribution. *Hewlett-Packard Labs, Tech. Rep*.

Xiao, H.; Herman, M.; Wagner, J.; Ziesche, S.; Etesami, J.; and Linh, T. H. 2019. Wasserstein adversarial imitation learning. *arXiv preprint arXiv:1906.08113*.

Zahavy, T.; Cohen, A.; Kaplan, H.; and Mansour, Y. 2020. Apprenticeship learning via frank-wolfe. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 6720–6728.

Zanette, A.; and Brunskill, E. 2019. Tighter Problem-Dependent Regret Bounds in Reinforcement Learning without Domain Knowledge using Value Function Bounds. In *International Conference on Machine Learning*, 7304–7312.

Zhang, M.; Wang, Y.; Ma, X.; Xia, L.; Yang, J.; Li, Z.; and Li, X. 2020a. Wasserstein Distance guided Adversarial Imitation Learning with Reward Shape Exploration. *arXiv preprint arXiv:2006.03503*.

Zhang, Y.; Cai, Q.; Yang, Z.; and Wang, Z. 2020b. Generative adversarial imitation learning with neural networks: Global optimality and convergence rate. *arXiv preprint arXiv:2003.03709*.