On Efficient Online Imitation Learning via Classification

Anonymous Author(s)
Affiliation
Address
email

Abstract

Imitation learning (IL) is a general learning paradigm for sequential decision-making problems. Interactive imitation learning, where learners can interactively query for expert annotations, has been shown to achieve provably superior sample efficiency guarantees compared with its offline counterpart or reinforcement learning.

In this work, we study classification-based online imitation learning (abbrev. COIL) and the fundamental feasibility to design oracle-efficient regret-minimization algorithms in this setting. We make the following contributions: (1) we show that in the COIL problem, any proper online learning algorithm cannot guarantee a sublinear regret in general; (2) we propose LOGGER, an improper online learning algorithmic framework, that reduces COIL to *online linear optimization*, by utilizing a new definition of mixed policy class; (3) we design two oracle-efficient algorithms within the LOGGER framework that enjoy different sample and interaction round complexity tradeoffs, and show their improvements over behavior cloning; (4) we show that under standard complexity-theoretic assumptions, efficient dynamic regret minimization is infeasible in the LOGGER framework.

1 Introduction

Imitation learning (IL), also known as learning from expert demonstrations [37, 35], is a general paradigm for training intelligent behavior for sequential decision making tasks. IL has been successfully deployed in many applications, such as autonomous driving [37, 36], robot arm control [53], game playing [46], and sequence prediction [19, 6]. It is now well-known that with the help of a demonstrating expert, an IL agent can bypass the exploration challenges of reinforcement learning, to achieve a much lower sample requirement than reinforcement learning agents [48].

Two major IL paradigms have been studied in the literature: offline and interactive. In offline IL [1, 49, 57, 25], the learner receives a set of expert demonstrations ahead of time; in contrast, in interactive IL [19, 43, 42, 26], the learner has the ability to interactively query the expert for demonstrations on states at its disposal, allowing expert feedback to be provided in a targeted manner. In both settings, the goal of the learner is to output a policy $\hat{\pi}$ that competes with the expert's policy π^E, with as few resources (e.g. expert annotations) consumed as possible. Between these two, interactive IL is known to be able to achieve superior policy performance than its offline counterpart, in that learning agents can use interaction to address the compounding error challenge [41, 39].

An influential algorithmic framework for interactive IL, DAGGER (Data Aggregation), has been proposed in [43]. Its key insight is to reduce online IL to regret minimization in online learning [45]. Specifically, it constructs a N-round online learning game, where at every round $n \in [N]$, the learner outputs some policy π_n from a policy class $\mathcal{B}$, and incurs a loss $F_n(\pi)$; the loss is carefully constructed so that the learner's instantaneous loss $F_n(\pi_n)$ characterizes current policy π_n's competitiveness compared to the expert π^E. A representative example of $F_n(\pi)$ is the expected disagreement

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

between policy π and π^E on the state occupancy distribution induced by π_n, used in the original DAGGER paper [43], which can be expressed as the expected zero-one loss of π on a distribution of classification examples – we call such setting *Classification-based Online Imitation Learning* (abbrev. COIL). The DAGGER reduction framework has spurred an active line of research on IL [e.g. 42, 48, 11, 14, 13, 30]: it enables a conversion from stochastic online optimization algorithms with static or dynamic regret guarantees to IL algorithms with different policy suboptimality guarantees, allowing the research community to directly "port" new results from online learning to the field of IL.

Perhaps surprisingly, from a fundamental perspective, rigorous design of efficient regret minimization algorithms for COIL has been largely overlooked by the prior literature. Specifically, many works assume a fixed parameterization of policies in $\mathcal{B}$, and assume that $F_n(\pi)$'s are convex in π's underlying parameters [e.g. 43, 48, 11]. This viewpoint has two issues: (1) it makes the usage of policy classes with complex parameterization (e.g. rule-based policies such as decision trees) difficult, one key reason being that convexity is hard to establish for such classes; (2) in DAGGER's reduction, the learner uses finite-sample approximations of $F_n(\pi)$, which are often discontinuous in π's underlying parameters (e.g. given a policy $\pi_\theta(s) = \mathrm{sign}(\langle \theta, s \rangle)$ as a linear classifier, its zero-one loss on a state, $I(\pi_\theta(s) \neq \pi^E(s))$ is discontinuous in θ) , making stochastic gradient-based methods inapplicable. Convex surrogate loss functions has been proposed as a workaround [43], but it is well-known that even in the simpler supervised learning setting, minimizing convex surrogate losses can result in very different models compared to minimizing the original zero-one classification loss [5].

An overview of our results. In this paper, we bridge the above-mentioned gaps by studying the fundamental feasibility of designing efficient regret minimization algorithms for COIL. Our first result is that, analogous to Cover's impossibility result in online classification [16], in the COIL setting, any proper online learning algorithm (that outputs a sequence of policies $\{\pi_n\}_{n=1}^N$ from the original class $\mathcal{B}$) cannot guarantee a sublinear regret in general (§ 3.1).

The above negative result motivates the design of improper learning algorithm for regret minimization. We propose to choose policies from a mixed policy class $\Pi_{\mathcal{B}}$, and provide an algorithmic framework, LOGGER, that reduces COIL to *online linear optimization*. In a nutshell, LOGGER uses a natural parameterization on $\Pi_{\mathcal{B}}$ that allows to express $F_n(\pi)$ as a linear function of the underyling parameters of $\pi \in \Pi_{\mathcal{B}}$. We show that any online linear optimization algorithm with regret guarantees can be plugged into LOGGER to obtain an algorithm for COIL with policy suboptimality guarantees (§ 3.2).

Next, enabled by the LOGGER framework, we design computationally efficient algorithms for static regret minimization. Assuming access to an offline cost-sensitive classification oracle $\mathcal{O}$, and a set of unlabeled *separator examples* for $\mathcal{B}$ [51, 20], we design a sample and computationally efficient algorithm LOGGER-M. Using $O(1/\epsilon^2)$ interaction rounds and $O(1/\epsilon^2)$ expert annotations, LOGGER-M enjoys a per-round static regret of ϵ (§ 4.1). Underlying LOGGER-M is a delicate utilization of the connection between Follow-the-Perturbed-Leader and Follow-the-Regularized-Leader, two well-known online learning algorithm families, first observed by [2]. Moreover, by exploiting the predictability of the COIL problem [40, 14, 13], we design an efficient algorithm LOGGER-ME, that enjoys a per-round static regret of ϵ, with $O(1/\epsilon)$ interaction rounds and $O(1/\epsilon^2)$ expert annotations (§ 4.2). Its reduced number of interaction rounds enables a more practical deployment of IL agents, as expert annotations often come with delays.

Finally, we study efficient dynamic regret minimization in the LOGGER framework (§ 5). We show that it is unlikely to be feasible: under a standard complexity-theoretic assumption, no oracle-efficient algorithms can output policies in $\Pi_{\mathcal{B}}$ with sublinear dynamic regret. Our work puts classification-based online imitation learning, an important IL setup, into a firmer ground. Due to space constraints, we discuss key related works throughout the paper, but defer additional related work discussions to Appendix A.

2 Preliminaries

Basic definitions. Define $[n] := \{1, \ldots, n\}$. Define indicator function $I(\cdot)$ such that $I(E) = 1$ if condition E is true, and $= 0$ otherwise. We use $\Delta(W)$ to denote the set of probability distributions over a finite set W, and use $\mathrm{Onehot}(w, W) \in \Delta(W)$ to denote the delta mass on $w \in W$. For a finite W, we sometimes treat $u \in \mathbb{R}^W$ (e.g. $u \in \Delta(W)$) as a $|W|$-dimensional vector; for $w \in W$, denote by $u[w]$ the w-th coordinate of u.

Episodic MDPs. We study imitation learning in episodic Markov decision processes (MDPs). An episodic MDP $\mathcal{M}$ is a tuple $(\mathcal{S}, \mathcal{A}, H, c, \rho, P)$, where $\mathcal{S}$ is a finite state space (that can be exponentially large), $\mathcal{A}$ is a finite action set, H is the episode length, $c : \mathcal{S} \times \mathcal{A} \to [0, 1]$ is the cost function, and $\rho \in \Delta(\mathcal{S})$ is the initial state distribution. Also, $P = \{P_t\}_{t=1}^{H-1}$ denotes $\mathcal{M}$'s transition dynamics, with $P_t : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ being the transition probability at step t. Throughout, we use S and A to denote $|\mathcal{S}|$ and $|\mathcal{A}|$, respectively. Without loss of generality, we assume that $\mathcal{M}$ is layered, where $\mathcal{S}$ can be partitioned into H disjoint sets $\{\mathcal{S}_t\}_{t=1}^{H}$; the initial distribution ρ is supported on $\mathcal{S}_1$, and transition distribution $P_t(\cdot \mid s, a)$ is supported on $\mathcal{S}_{t+1}$ for all t, s, a. For state $s \in \mathcal{S}$, define $\mathrm{Step}(s)$ as the step t such that $s \in \mathcal{S}_t$.

A learning agent interacts with $\mathcal{M}$ for one episode using the following protocol: for every step $t \in [H]$: it observes a state $s_t \in \mathcal{S}_t$, takes an action $a_t \in \mathcal{A}$, and transitions to next state $s_{t+1} \sim P_t(\cdot \mid s_t, a_t)$ except for the last step when it stops. Given a stationary policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$, we use $\pi(\cdot|s)$ to denote the action distribution of π on s. Denote by $\mathbb{E}_\pi$ and $\mathbb{P}_\pi$ the expectation and probability over executing policy π in $\mathcal{M}$. Given policy π, its state occupancy distribution at step t is defined as $d_\pi^t(\cdot) := \mathbb{P}_\pi(s_t = \cdot)$; its average state occupancy distribution is denoted as $d_\pi := \frac{1}{H}\sum_{t=1}^{H} d_\pi^t$. Let $J(\pi) := \mathbb{E}_\pi\left[\sum_{t=1}^{H} c(s_t, a_t)\right] = H \cdot \mathbb{E}_{s\sim d_\pi}\mathbb{E}_{a\sim\pi(\cdot|s)}\left[c(s, a)\right]$ denote the expected cumulative cost of π over an episode. For policy π, we denote its value function $V_\pi(s) := \mathbb{E}\left[\sum_{t=\mathrm{Step}(s)}^{H} c(s_t, a_t) \mid s, \pi\right]$ and action-value function $Q_\pi(s, a) := c(s, a) + \mathbb{E}\left[\sum_{t=\mathrm{Step}(s)+1}^{H} c(s_t, a_t) \mid s, a, \pi\right]$; in words, they are the expected cost of rolling out π starting from s and (s, a) respectively. For policy π, denote its advantage function $A_\pi(s, a) := Q_\pi(s, a) - V_\pi(s)$ that measures the expected performance difference by one step deviation of taking action a at state s. Also, we define the recoverability constant as the ability of π to recover from deviation when rolled out in $\mathcal{M}$:

Definition 1 (μ-recoverability)**.** *$(\mathcal{M}, \pi)$ is μ-recoverable if $\forall s \in \mathcal{S}, a \in \mathcal{A}$, $|A_\pi(s, a)| \leq \mu$.* [1]

Interactive IL. We study interactive imitation learning [19, 41], where the learner has access to a deterministic demonstrating expert π^E and would like to learn a policy with low expected cost. Throughout, we assume that $(\mathcal{M}, \pi^E)$ is μ-recoverable, for some $\mu \leq H$ that can potentially be $\ll H$. At each interaction round, the learner interacts with $\mathcal{M}$ for a few episodes, obtaining trajectories of the form $\tau = (s_1, a_1, s_2, a_2, \ldots, s_H, a_H)$ and queries the expert for feedback on some of the states. Specifically, given a state s, the feedback given by the expert is of the form $(\zeta_E(s, a))_{a\in\mathcal{A}} \in \mathbb{R}^{\mathcal{A}}$. Two notable examples are: (1) direct expert annotation [43], i.e. $\zeta_E(s, a) = \mu \cdot I(a \neq \pi^E(s))$; (2) estimates of value functions based on experts' rollout [42], i.e. $\zeta_E(s, a) = A^E(s, a) := A_{\pi^E}(s, a)$. Throughout, we assume $\zeta_E(s, a)$ satisfies $\forall s \in \mathcal{S}, a \in \mathcal{A}$, $A^E(s, a) \leq \zeta_E(s, a) \leq \mu \cdot I(a \neq \pi^E(s))$; this is satisfied by the two examples above. Define a policy π's *imitation loss* as: $L(\pi) := \mathbb{E}_{s\sim d_\pi}\mathbb{E}_{a\sim\pi(\cdot|s)}\left[\zeta_E(s, a)\right]$. By the performance difference lemma [28] (see Appendix H), $J(\pi) - J(\pi^E) \leq H \cdot L(\pi)$, implying that if π has a small imitation loss, it will have expected cost competitive with π^E. Because of this connection, in interactive IL, the learner would like to obtain policy $\hat{\pi}$ with low $L(\hat{\pi})$. Meanwhile, throughout the paper, we consider the learner to optimize two measures of data efficiency:

- *Sample complexity*: the total number of expert annotations ζ_E requested. A smaller sample complexity reduces the total cost of expert annotations (which often takes human effort).
- *Interaction round complexity*: total number of adaptive interaction rounds. A small number of interaction rounds enables more parallelized annotations within an interaction round, and mitigates the delayed annotation issue [55, 52].

The DAGGER reduction framework for interactive IL. The DAGGER framework reduces minimizing $L(\pi)$ to no-regret online learning [43, 42]. It constructs a N-round online learning game, where at every round $n \in [N]$, the learner outputs some policy π_n, and a loss function $F_n(\pi) = \mathbb{E}_{s\sim d_{\pi_n}}\mathbb{E}_{a\sim\pi(\cdot|s)}\left[\zeta_E(s, a)\right]$ is then defined. The key insight is that, by the definition of $\{F_n\}_{n=1}^{N}$, minimizing the online learning cumulative loss $\sum_{n=1}^{N} F_n(\pi_n)$ is equivalent to minimizing the cumulative imitation losses of π_n's, i.e. $\sum_{n=1}^{N} L(\pi_n)$. Research efforts in online learning [45] have focused on the design of algorithms that can output $\{\pi_n\}_{n=1}^{N}$ with static regret SReg_N or

[1] The μ-recoverability definition here is slightly different from the original ones in [43], in that it also requires that $A_\pi(s, a) \geq -\mu$; we can drop this assumption with a slightly worse sample complexity analysis.

Protocol 1 Classification-based Online Imitation learning (COIL)

for $n = 1, \ldots, N$ **do**
  Learner outputs policy π_n
  Loss function $F_n(\pi) := \mathbb{E}_{s\sim d_{\pi_n}} \mathbb{E}_{a\sim\pi(\cdot|s)} \left[\zeta_E(s,a)\right]$
  Learner draws samples from $\mathcal{D}^E_{\pi_n}$ to obtain information about loss F_n, via interacting with $\mathcal{M}$ and querying the expert for annotation ζ_E
end for
Goal of learner: minimize $\sum_{n=1}^N F_n(\pi_n) = \sum_{n=1}^N L(\pi_n)$

dynamic regret DReg_N against some benchmark policy class $\mathcal{B}$, formally:

$$\mathrm{SReg}_N := \sum_{n=1}^N F_n(\pi_n) - \min_{\pi\in\mathcal{B}} \sum_{n=1}^N F_n(\pi), \quad \mathrm{DReg}_N := \sum_{n=1}^N \left(F_n(\pi_n) - \min_{\pi\in\mathcal{B}} F_n(\pi) \right).$$

Assuming that the learner chooses policies $\{\pi_n\}_{n=1}^N$ from another policy class $\mathcal{B}_0$ (which may or may not be $\mathcal{B}$), the following proposition shows static and dynamic regret guarantees can be converted to policy suboptimality guarantees:

Proposition 2 (e.g. [11])**.** *For any $N \in \mathbb{N}^+$, define $\mathrm{Bias}(\mathcal{B}, \mathcal{B}_0, N) := \max_{\{v_n\}_{n=1}^N \in \mathcal{B}_0^N} \min_{\pi\in\mathcal{B}} \mathbb{E}_{s\sim\bar{d}_N} \mathbb{E}_{a\sim\pi(\cdot|s)} \left[I(a \neq \pi^E(s))\right]$, where $\bar{d}_N := \frac{1}{N}\sum_{n=1}^N d_{v_n}$. Then, choosing $\hat{\pi}$ uniformly at random from $\{\pi_n\}_{n=1}^N$ has guarantee:*

$$\mathbb{E}\left[J(\hat{\pi}) - J(\pi_E)\right] \leq H \cdot \min\left\{ \mu \cdot \mathrm{Bias}(\mathcal{B}, \mathcal{B}_0, N) + \frac{\mathbb{E}[\mathrm{SReg}_N]}{N},\ \mu \cdot \mathrm{Bias}(\mathcal{B}, \mathcal{B}_0, 1) + \frac{\mathbb{E}[\mathrm{DReg}_N]}{N} \right\}.$$

In the above proposition, $\mathrm{Bias}(\mathcal{B}, \mathcal{B}_0, N)$ takes the worst-case mixture of state occupancy distributions $\{d_{v_n}\}_{n=1}^N$ induced by N policies from $\mathcal{B}_0$, and measures the expected disagreement between π^E and its best approximating policy in $\mathcal{B}$. Informally, it measures the "approximation error" of benchmark class $\mathcal{B}$: it is always nonnegative, and when $\pi^E \in \mathcal{B}$ (which we call the *realizable case)*, $\mathrm{Bias}(\mathcal{B}, \mathcal{B}_0, N) = 0$. Proposition 2 gives two ways to obtain a competitive imitation policy: (1) choosing $\mathcal{B}$ and $\mathcal{B}_0$ with a small $\mathrm{Bias}(\mathcal{B}, \mathcal{B}_0, N)$ and achieving a low static regret; (2) choosing $\mathcal{B}$ and $\mathcal{B}_0$ with a small $\mathrm{Bias}(\mathcal{B}, \mathcal{B}_0, 1)$ and achieve a low dynamic regret. Although achieving low dynamic regret can be significantly more challenging than achieving low static regret, minimizing dynamic regret has the advantage that its approximation error term $\mathrm{Bias}(\mathcal{B}, \mathcal{B}_0, 1)$ is smaller than the static regret formulation's counterpart $\mathrm{Bias}(\mathcal{B}, \mathcal{B}_0, N)$.

Classification-based Online Imitation Learning (COIL). As we consider a finite action space $\mathcal{A}$, a policy π can be equivalently viewed as a (possibly randomized) multiclass classifier. A cost-sensitive example is defined to be a pair $(x, \vec{c})$, where $x \in \mathcal{S}$ is its feature part, and $\vec{c} \in \mathbb{R}^{\mathcal{A}}$ is its cost part. Under the DAGGER reduction framework, the loss at iteration n, $F_n(\pi) = \mathbb{E}_{s\sim d_{\pi_n}} \mathbb{E}_{a\sim\pi(\cdot|s)} \left[\zeta_E(s,a)\right]$, can be viewed as the expected cost of policy π on a distribution of cost-sensitive examples $\mathcal{D}^E_{\pi_n}$, where a sample $(s, \vec{c})$ is drawn from $\mathcal{D}^E_\pi$ by first rolling out π and drawing $s \sim d_\pi$, and query the expert on s to obtain $(\zeta_E(s,a))_{a\in\mathcal{A}}$ as its associated $\vec{c}$. The learner can obtain a finite-sample approximation to $F_n(\pi)$ by interacting with $\mathcal{M}$ and the expert to draw samples from $\mathcal{D}^E_{\pi_n}$. We will focus on designing efficient regret-minimizing algorithms in the COIL setting; see Protocol 1 for a summary.

We assume that benchmark policy class $\mathcal{B}$ is a collection of B deterministic policies $h : \mathcal{S} \to \mathcal{A}$. To design algorithms with computational efficiency guarantees, following a sequence of empirically and theoretically successful works on oracle-efficient learning [51, 21, 3, 17], throughout this paper, we assume access to the following computational oracle, with each call to it taking unit time:

Definition 3 (Cost-sensitive classification oracle)**.** *A cost-sensitive classification oracle $\mathcal{O}$ for policy class $\mathcal{B}$ is such that: given any input set of cost-sensitive examples $D = ((x_1, \vec{c}_1), \ldots, (x_n, \vec{c}_n)) \in (\mathcal{S} \times \mathbb{R}^{\mathcal{A}})^n$, it outputs the policy in $\mathcal{B}$ that has the smallest empirical cost, formally, $\mathcal{O}(D) := \mathrm{argmin}_{h\in\mathcal{B}} \mathbb{E}_{(x,c)\sim D} \left[\vec{c}(h(x))\right]$.*

Algorithm 2 LOGGER: reducing COIL to online linear optimization

Input: Online linear optimization algorithm OLOA with decision set $\Delta(\mathcal{B})$, sample size per iteration K.
for $n = 1, 2, \ldots, N$ **do**
  Choose $u_n \leftarrow \mathrm{OLOA}(\{g_i\}_{i=1}^{n-1})$, which induces policy $\pi_n := \pi_{u_n}$
  Draw K examples $D_n = \{(s, \vec{c})\}$ i.i.d. from $\mathcal{D}^E_{\pi_n}$, via interaction with $\mathcal{M}$ and expert ζ_E
  D_n induces $g_n = \left(\mathbb{E}_{(s,\vec{c})\sim D_n}\left[\vec{c}(h(s))\right]\right)_{h\in\mathcal{B}}$, an unbiased estimate of $\theta(u_n)$
  Suffer linear loss $u \mapsto \langle g_n, u\rangle$
end for

3 LOGGER: reducing COIL to online linear optimization

In this section, we introduce our main algorithmic framework, LOGGER (an abbreviation for Linear lOss aGGrEgation) for designing regret-minimizing algorithms in the COIL setting. Section 3.1 motivates our approach by showing that natural proper learning-based approaches fail to achieve sublinear regret; Section 3.2 introduces the main idea of performing improper learning using a carefully-defined mixture policy class, via a reduction to online linear optimization.

3.1 Can we achieve sublinear regret using proper learning?

A natural idea for minimizing regret is *proper learning*: at round n, the learner chooses some policy π_n (potentially at random) from $\mathcal{B}_0 = \mathcal{B}$, our benchmark policy class, using some online learning algorithm, based on all information collected in the first $n-1$ rounds; the learner collects information on F_n via rollouts of π_n and expert annotations, and continue to the next iteration.

While this approach has demonstrated sharp online regret guarantees in classical online cost-sensitive classification settings [32, 23], perhaps subtly, we show in the following theorem that, this approach is insufficient to guarantee sublinear regret in the COIL setting.

Theorem 4. *Suppose the expert's feedback is of the form $\zeta_E(s,a) = A^E(s,a)$. Then, for any $H \geq 2$, there exists an MDP $\mathcal{M}$ of time horizon H, a deterministic expert policy π^E, a benchmark policy class $\mathcal{B}$, such that for any learner that sequentially and possibly at random generates a sequence of policies $(\pi_n)_{n=1}^N \in \mathcal{B}^N$, its static regret satisfies $\mathrm{SReg}_N = \Omega(N)$.*

The proof of the theorem can be found at Appendix D.1. Its key insight is that, distinct from the classical online cost-sensitive classification setting, in COIL, the loss at round n, F_n, *depends on the policy chosen at that round* π_n, making standard regret minimization results in online classification [32, 23] inapplicable. In more detail, we construct MDPs that act "adversarially" to policies in $\mathcal{B}$, such that any $\pi_n \in \mathcal{B}$ has $F_n(\pi_n) = \frac{N(H-1)}{H}$, whereas $\min_{\pi\in\mathcal{B}} \sum_n^N F_n(\pi) \leq \frac{N(H-1)}{2H}$. Our theorem is akin to Cover's impossibility result in online classification [16], which shows that an adversary that adapts to the randomness of the learner at each round forces the learner to suffer linear regret.

3.2 A new hypothesis class and the LOGGER algorithmic framework

To sidestep the impossibility result in Theorem 4, we apply the convexification by randomization technique in online convex optimization [45] by doing improper learning on a mixed policy class.

Definition 5 (Mixed policy class)**.** *Given policy class $\mathcal{B}$, define its induced mixed policy class*

$$\Pi_{\mathcal{B}} := \left\{ \pi_u(\cdot|s) := \sum_{h\in\mathcal{B}} u[h] \cdot h(\cdot|s) : u \in \Delta(\mathcal{B}) \right\}.$$

We slightly abuse notation and use $h(\cdot \mid s) \in \Delta(\mathcal{A})$ to denote the delta mass on $h(s) \in \mathcal{A}$.[2]

At a first sight, choosing a policy from $\Pi_{\mathcal{B}}$ seems equivalent to choosing some policy at random from $\mathcal{B}$, which also falls into the failure mode of proper learning (Theorem 4). We remark that this is not true: rolling out a policy from $\Pi_{\mathcal{B}}$ is equivalent to drawing a freshly new policy at random from $\mathcal{B}$ *at every step of the episode* instead. We show below that, this avoids the above failure mode.

[2] [48, Theorem 5.3], [4, Section 15.5] propose to perform no-regret learning using another definition of mixed policy class; we identify an issue with this approach, and defer detailed discussions to Appendix D.2.

The key observation is with the learner outputting policies from the mixed policy class $\Pi_{\mathcal{B}}$, online regret minimization in IL becomes an *online linear optimization* problem. Recall that in online IL, the loss at round n is $F_n(\pi) = \mathbb{E}_{s\sim d_{\pi_v}} \mathbb{E}_{a\sim\pi(\cdot|s)} [\zeta_E(s,a)]$. By taking $\pi = \pi_u$ for $u \in \Delta(\mathcal{B})$, $F_n(\pi_u)$ can be viewed as a linear function of u:

$$F_n(\pi_u) = \sum_{h\in\mathcal{B}} u[h] \cdot \mathbb{E}_{s\sim d_{\pi_v}} \mathbb{E}_{a\sim h(\cdot|s)} [\zeta_E(s,a)] = \langle \theta(u_n), u \rangle ,$$

where $\theta(v) := \left(\mathbb{E}_{s\sim d_{\pi_v}} \mathbb{E}_{a\sim h(\cdot|s)} [\zeta_E(s,a)]\right)_{h\in\mathcal{B}}$. Denote $\pi_n = \pi_{u_n}$ for $u_n \in \Delta(\mathcal{B})$, we have

$$\sum_{n=1}^{N} \langle \theta(u_n), u_n \rangle = \sum_{n=1}^{N} F_n(\pi_n), \qquad \min_{u\in\Delta(\mathcal{B})} \sum_{n=1}^{N} \langle \theta(u_n), u \rangle = \min_{u\in\Delta(\mathcal{B})} F_n(\pi_u) = \min_{\pi\in\mathcal{B}} F_n(\pi),$$

and therefore, minimzing the static regret SReg_N is equivalent to minimizing the regret in the online linear optimization problem with losses $\{u \mapsto \langle \theta(u_n), u \rangle\}_{n=1}^N$. This motivates LOGGER (Algorithm 2), our main algorithmic framework. LOGGER takes into input an online linear optimization algorithm OLOA, and at every iteration n, LOGGER chooses $u_n \in \Delta(\mathcal{B})$, which corresponds to a policy $\pi_n \in \Pi_{\mathcal{B}}$; it then rolls out π_n in $\mathcal{M}$ for K times, obtaining K samples i.i.d. from d_{π_n}, and use them to construct unbiased estimate of $\theta(u_n)$, namely g_n, and use OLOA to perform online linear optimization with respect to linear losses $\{u \mapsto \langle g_n, u \rangle\}_{n=1}^N$.

Comparison to prior works. [12] considers a general online convex optimization formulation for online IL, dubbed "continuous online learning"; our loss function $\langle \theta(u), \cdot \rangle$ can be viewed as a concrete instantiation of the loss function $f_u(\cdot)$ therein. However, their results do not cover the setting when $f_u(\cdot)$ is linear, which occurs naturally in the COIL setting. We have the following proposition that relates the static regret with respect to $\{\langle g_n, \cdot \rangle\}_{n=1}^N$, to the static regret of the online IL problem.

Proposition 6. *If* LOGGER *uses online linear optimization algorithm* OLOA *that outputs* $\{u_n\}_{n=1}^N \subset \Delta(\mathcal{B})^N$ *s.t. with probability* $1-\delta/3$, $\sum_{n=1}^N \langle g_n, u_n \rangle - \min_{u\in\Delta(\mathcal{B})} \sum_{n=1}^N \langle g_n, u \rangle \le \mathrm{Reg}(N)$. *Then, with probability* $1-\delta$, *its output policies* $\{\pi_n\}_{n=1}^N$ *satisfy* $\mathrm{SReg}_N \le \mathrm{Reg}(N) + \tilde{O}\left(\mu\sqrt{\frac{N\ln(B)}{K}}\right)$.

Proposition 6 shows that Algorithm 2 is a regret-preserving reduction from online IL to online linear optimization over $\Delta(\mathcal{B})$, a B-dimensional probability simplex. The latter is well-known as the "prediction with expert advice" problem [23] (abbrev. expert problem), where algorithms with different guarantees abound, such as Follow-the-Regularized-Leader(FTRL), Hedge [23] and its adaptive and optimistic variants [e.g. 47, Section 1], many of which have a optimal worst-case regret $R(N) = O\left(\sqrt{N\ln(B)}\right)$. Instantiating Algorithm 2 with OLOA set as these algorithms, we obtain a family of online IL algorithms with expected regret of order $O(\sqrt{N})$.

Although satisfying from a statistical efficiency perspective, such online IL algorithms suffers from computational inefficiency: they require explicit maintenance of u_n, a B-dimensional vector whose all entries needs to be updated separately. For instance, when Hedge is set as OLOA, $u_n[h] \propto \exp(-\eta \sum_{i=1}^{n-1} g_n[h]))$ for all $h \in \mathcal{B}$, which naively requires B time per round to maintain. In the next section, we utilize the cost-sensitive classification oracle $\mathcal{O}$ to design sublinear-regret algorithms that have computational efficiency guarantees.

4 Efficient algorithms with static regret guarantees

Using the LOGGER framework, in this section, we propose two oracle-efficient COIL algorithms that have sublinear static regret guarantees against policy class $\mathcal{B}$, in subsections 4.1 and 4.2 respectively.

4.1 LOGGER-M: an efficient algorithm with $O\left(\sqrt{N}\right)$ static regret

The LOGGER reduction framework calls for a computationally and statistically efficient OLOA, which, if devised, yields an computationally and statistically efficient online IL learner. However, when viewed as a general adversarial online learning problem, computational hardness results [24] suggest that, even with access to classification oracle $\mathcal{O}$, a prohibitive $\Omega(\sqrt{B})$ time complexity is

Algorithm 3 MFTPL: an oracle-efficient approximation of FTRL

Input: Datasets $\{D_i\}_{i=1}^{n-1}$ representing linear losses $\{g_i\}_{i=1}^{n-1}$ (s.t. $g_i[h] = \mathbb{E}_{(s,\vec{c})\sim D_i}[\vec{c}(h(s))]$ for all $h \in \mathcal{B}$), separator set $\mathcal{X}$, learning rate η, sparsification parameter T
for $j = 1, 2, \ldots, T$ **do**
  Draw $(\ell_{x,j}(a))_{a\in\mathcal{A}} \sim \mathcal{N}(0, I_A)$ i.i.d. for each $x \in \mathcal{X}$.
  Define $Z_j = \left\{(x, \frac{K}{\eta}\ell_{x,j}) : x \in \mathcal{X}\right\}$
  Compute $u_j \leftarrow \mathrm{Onehot}(\mathcal{O}((\cup_{i=1}^{n-1} D_i) \cup Z_j), \mathcal{B})$.
end for
return $u \leftarrow \frac{1}{T}\sum_{j=1}^{T} u_j$

necessary for sublinear regret. Therefore, in subsequent sections, we adopt an assumption on $\mathcal{B}$ in [51], which, to the best of our knowledge, is the state-of-the-art weakest assumption that allows the design of oracle-efficient online cost-sensitive classification algorithms in the adversarial setting:

Definition 7 (Separator)**.** *There exists a set $\mathcal{X} \subset \mathcal{S}$ (called the separator set) such that, for every pair of distinct policies $h, h' \in \mathcal{B}$, there exist some $x \in \mathcal{X}$, such that $h(x) \neq h'(x)$. Denote by $X := |\mathcal{X}|$.*

Technical challenges. Even under the small separator assumption (that X is small), the design of low-regret oracle-efficient algorithms for imitation learning still remains nontrivial. A naive application of existing oracle-efficient online cost-sensitive classification algorithms, such as Contextual Follow the Perturbed Leader (CFTPL) [51, 20], still falls into the failure mode of proper learning (Theorem 4 in Section 3.1). To recap, at round n, these algorithms first constructing a random set of "hallucinated" cost-sensitive examples Z based on the separator set $\mathcal{X}$; they subsequently calls the cost-sensitive classification oracle $\mathcal{O}$ on the union of Z and the accumulated dataset $\cup_{i=1}^{n-1} D_i$ to obtain policy $\pi_n \in \mathcal{B}$. This $\Omega(N)$ regret lower bound is in sharp contrast to the classical online cost-sensitive classification setting, where CFTPL enjoys a $O(\sqrt{N})$ regret [51, 20].

Our approach. The above difficulty motivates the need of a new algorithmic approach to efficient online IL. In view of Section 3.2's observation that FTRL approaches enjoy a sublinear regret, we ask the question: is it possible to perform FTRL in an oracle-efficient manner? A positive answer will simultaneously address the computational and statistical challenges of COIL.

We answer this question in the affirmative, by utilizing a connection between FTRL and FTPL first observed in [2]: an in-expectation version of FTPL can be viewed as an FTRL algorithm. Using this observation, we design Algorithm 3, namely Mixed CFTPL (abbrev. MFTPL), which mimics an in-expectation version of CFTPL in an oracle-efficient manner. At a high level, it runs CFTPL for T times, and takes a uniform mixture of the T policies as its output policy, and guarantees that:

Lemma 8. *There exists some strongly convex function $R : \Delta(\mathcal{B}) \to \mathbb{R}$, such that the following holds. Suppose* MFTPL *receives datasets $\{D_i\}_{i=1}^{n-1}$, separator set $\mathcal{X}$, learning rate η, sparsification parameter T. Then, with probability at least $1-\delta$,* MFTPL *makes T calls to the cost-sensitive oracle $\mathcal{O}$, and outputs $u_n \in \Delta(\mathcal{B})$ such that $\forall s \in \mathcal{S}$, $\|\pi_{u_n}(\cdot|s) - \pi_{u_n^*}(\cdot|s)\|_1 \leq \sqrt{\frac{2A\left(\ln(S)+\ln(\frac{2}{\delta})\right)}{T}}$, with $u_n^* := \mathrm{argmin}_{u\in\Delta(\mathcal{B})}\left(\eta\sum_{i=1}^{n-1}\langle g_i, u\rangle + R(u)\right)$.*

Therefore, by setting $T = \Omega(A\ln(S))$, the policy π_n induced by the MFTPL's output u_n closely mimics $\pi_{u_n^*}$, a policy induced by the FTRL output u_n^*. The $\ln(\cdot)$ dependence on S makes the lemma useful in large state space settings. A naive attempt to show Lemma 8 is to establish the ℓ_1 closeness of u_n and u_n^* . This unavoidably carries a concentration factor of $O(\sqrt{B/T})$, which is impractical, as this requires T to be $\Omega(B)$ to be useful. We get around by directly showing the closeness of the action distributions π_{u_n} and $\pi_{u_n^*}$. We defer the full version of the Lemma to Appendix E.1.

Lemma 9. MFTPL*, with input step size $\eta = O\left(\frac{1}{\mu\sqrt{NA}}(\frac{\ln(B)}{X})^{\frac{1}{4}}\right)$ and sparsification parameter $T = \frac{N\ln(SN/\delta)}{\sqrt{X^3\ln(B)}}$, outputs a sequence of $\{u_n\}_{n=1}^{N}$, such that with probability $1-\delta$:*

$$\sum_{n=1}^{N}\langle g_n, u_n\rangle - \min_{u\in\Delta(\mathcal{B})}\sum_{n=1}^{N}\langle g_n, u\rangle \leq O\left(\mu\sqrt{NA}(X^3\log(B))^{\frac{1}{4}}\right)$$

Composing LOGGER with MFTPL, we obtain an efficient online IL algorithm LOGGER-M. Its regret guarantees immediately follow from combining Lemma 9 with Proposition 6:

Theorem 10. LOGGER-M*, with* MFTPL *setting its parameters as in Lemma 9, satisfies that: (1) with probability* $1-\delta$*, it outputs a sequence of policies* $\{\pi_n\}_{n=1}^N$ *whose static regret satisfies that* $\mathrm{SReg}_N \leq O\left(\mu\sqrt{NA}(X^3\log(B))^{\frac{1}{4}}\right)$*; (2) it queries* $O(N)$ *annotations from the expert; (3) it calls the cost-sensitive classification oracle* $\mathcal{O}$ *for* $O\left(\frac{N^2\ln(SN/\delta)}{\sqrt{X^3\ln(B)}}\right)$ *times.*

4.2 LOGGER-ME: an efficient algorithm with $O(1)$ static regret

Although LOGGER-M is oracle-efficient, it is unclear whether its $O(\sqrt{N})$ regret guarantee is optimal. As a lower regret can translate to lower sample and interaction round complexity guarantees, it is of importance to design algorithms with regret as low as possible.

A key observation from prior works [e.g. 11, 12, 30] is that, online IL is a predictable online learning problem [15, 40], and is thus not completely adversarial. This opens up possibilities to bypass the $O(\sqrt{N})$ worst-case regret barrier. Specifically, in the LOGGER framework, the coefficient of the linear loss at round n, $\theta(u_n)$, depends *continuously* on u_n; more concretely:

Lemma 11. *For* $u, v \in \Delta(\mathcal{B})$, $\|\theta(u)-\theta(v)\|_\infty \leq \mu H \max_{s\in\mathcal{S}} \|\pi_u(\cdot|s)-\pi_v(\cdot|s)\|_1 \leq \mu H\|u-v\|_1$.

This property and its variants, termed *distributional continuity*, has been utilized in many online IL algorithms to achieve sharper regret guarantees. These works additionally exploit the strong convexity on the loss functions $F_n(\pi)$ [11, 12, 30], or use some external predictive model that can predict $\nabla F_n(\pi)$ well [14, 13]. Unfortunately, in our COIL setting, neither is the loss function $F_n(\pi_u) = \langle\theta(u_n), u\rangle$ strongly convex in the policy parameter u, nor we have access to an external predictive model, rendering these approaches inapplicable.

We get around these challenges and design an oracle-efficient algorithm, namely MFTPL-EG (where EG stands for extra-gradient), with $O(1)$ regret for the online linear optimization problem, which, when composed with the LOGGER framework, yields the LOGGER-ME algorithm with $O(1)$ regret in the COIL setting. MFTPL-EG is largely inspired by the predictor-corrector framework for policy optimization [13] and extragradient methods in smooth optimization [33, 27]. Our key insight is that, although we do not have a prediction model for $\theta(u_n)$, we can use an extra round of interaction with $\mathcal{M}$ and expert annotations to obtain a good estimate of it. Using these insights, we derive an online linear optimization regret guarantee of MFTPL-EG, deferred to Appendix E.2. It immediately implies the following guarantee of LOGGER-ME:

Theorem 12. LOGGER-ME*, with* MFTPL-EG *setting its parameters appropriately, satisfies that: (1) with probability* $1-\delta$*, its output* $\{\pi_n\}_{n=1}^N$ *satisfies:* $\mathrm{SReg}_N \leq O(\mu HA\sqrt{X^3\log(B)})$*; (2) it queries* $O\left(\frac{N^2}{H^2A\sqrt{X^3\log(B)}}\ln(\frac{NB}{\delta})\right)$ *annotations from the expert; (3) it calls the cost-sensitive classification oracle* $\mathcal{O}$ *for* $O\left(\frac{N^3\ln(SN/\delta)}{H^2AX^3\ln(B)}\right)$ *times.*

Discussion and comparison. We now compare the guarantees of LOGGER-M, LOGGER-ME, and the behavior cloning baseline. All algorithms' output policy suboptimality guarantees have the following decomposition:

$$\mathbb{E}\left[J(\hat{\pi}) - J(\pi^E)\right] \leq \mathrm{ApproxErr} + \mathrm{EstimErr}, \tag{1}$$

where ApproxErr measures the approximation error of the policy class $\mathcal{B}$ to the expert policy π^E, and EstimErr is an estimation error term that vanishes with the number of examples and iterations increasing. By Proposition 2, for LOGGER-M and LOGGER-ME, their ApproxErr terms are both $\mu H \cdot \mathrm{Bias}(\mathcal{B}, \Pi_{\mathcal{B}}, N)$. For their $\mathrm{EstimErr} = \frac{H\cdot\mathrm{SReg}_N}{N}$, we use Theorems 10 and 12 to calculate the minimum number of interaction rounds $I(\epsilon)$, total number of expert annotations $A(\epsilon)$, and total number of oracle calls $C(\epsilon)$, so that EstimErr is at most ϵ, and present it in Table 1. It can be seen that, LOGGER-ME has the same sample complexity order as LOGGER-M, but has a much lower interaction round complexity ($\mu H^2/\epsilon$ vs. μ^2H^2/ϵ^2).

On the other hand, by standard ERM analysis [44] and conversion from supervised learning to imitation learning guarantees ([50, 28]), behavior cloning on $\mathcal{B}$ using K samples returns a policy

Algorithm	ApproxErr	$I(\epsilon)$	$A(\epsilon)$	$C(\epsilon)$
LOGGER-M	$\mu H \cdot \mathrm{Bias}(\mathcal{B}, \Pi_{\mathcal{B}}, N)$	$\tilde{O}\left(\frac{\mu^2 H^2}{\epsilon^2}\right)$	$\tilde{O}\left(\frac{\mu^2 H^2}{\epsilon^2}\right)$	$\tilde{O}\left(\frac{\mu^4 H^4}{\epsilon^4}\right)$
LOGGER-ME	$\mu H \cdot \mathrm{Bias}(\mathcal{B}, \Pi_{\mathcal{B}}, N)$	$\tilde{O}\left(\frac{\mu H^2}{\epsilon}\right)$	$\tilde{O}\left(\frac{\mu^2 H^2}{\epsilon^2}\right)$	$\tilde{O}\left(\frac{\mu^3 H^4}{\epsilon^3}\right)$
Behavior cloning	$H^2 \cdot \mathrm{Bias}(\mathcal{B}, \{\pi^E\}, 1)$	1	$\tilde{O}(\frac{H^4}{\epsilon^2})$	1

Table 1: A comparison between our algorithms and the behavior cloning baseline, in terms of approximation error, and number of interaction rounds $I(\epsilon)$, expert annotations $A(\epsilon)$, oracle calls $C(\epsilon)$, needed for estimation error to be at most ϵ. Here $\tilde{O}(\cdot)$ hides dependence on $X, A, \log S, \log B, \log 1/\delta$. See Appendix E.3 for the full version of the table.

$\hat{\pi}$, such that Equation (1) holds with $\mathrm{ApproxErr} = H^2 \cdot \mathrm{Bias}(\mathcal{B}, \{\pi^E\}, 1)$, and $\mathrm{EstimErr} = \tilde{O}(\sqrt{\ln(B/\delta)/K})$ (see Appendix E.3 for a detailed derivation), where $\mathrm{Bias}(\mathcal{B}, \{\pi^E\}, 1) = \min_{h \in \mathcal{B}} \mathbb{E}_{s \sim d_{\pi^E}} \left[I(h(s) \neq \pi^E(s)) \right]$, the smallest disagreement rate between policies in $\mathcal{B}$ and π^E on expert's state occupancy distribution. Compared with the two interactive IL algorithms above, behavior cloning requires only one interaction round and one call to the oracle $\mathcal{O}$, however its $\mathrm{ApproxErr}$ has a larger coefficient on the optimal classification loss (H^2 vs. μH), and needs a larger number of expert annotations (H^4/ϵ^2 vs. $H^2\mu^2/\epsilon^2$) to achieve approximation error smaller than ϵ.

5 Computational hardness of sublinear dynamic regret guarantees

Finally, we study dynamic regret minimization for COIL in the LOGGER framework. Although in the abstract continuous online learning setup [11, 30, 12], dynamic regret minimization has been shown to be computationally hard, given the peculiar linear loss structure of the LOGGER framework, the computational tractability of dynamic regret minimization within this framework still remains open.

We fill this gap by showing that, under a standard complexity-theoretic assumptions (that PPAD-complete problems do not admit randomized polynomial-time algorithms), there do not exist polynomial-time algorithms that achieve sublinear dynamic regret in COIL. Specifically, we have:

Theorem 13. *For all $\alpha > 0$, if there exist a COIL algorithm that interacts with $\mathcal{M}$, cost-sensitive classification oracle $\mathcal{O}$, expert feedback ζ_E, and returns a sequence of $\{\pi_n\}_{n=1}^N \subset \Pi_{\mathcal{B}}$ such that with probability at least $1/2$,*

$$\mathrm{DReg}_N(\{\pi_n\}_{n=1}^N) \leq O(\mathrm{poly}(S, A, B) \cdot N^{1-\alpha}),$$

with $\mathrm{poly}(N, S, A, B)$ time, then all problems in PPAD are solvable in randomized polynomial time.

The key insight behind our proof of Theorem 13 is that, achieving a sublinear dynamic regret in the COIL setup is as hard as finding an approximate Nash equilibrium in a two-player general-sum game, a well-known PPAD-complete problem [10]. To establish a reduction from a two-player general sum game to a COIL problem, we carefully construct a tree-structured MDP whose leave states can be divided to two major groups: one group has costs encoding the two players' payoffs, and the other group has a large constant cost, ensuring that any policy in $\Pi_{\mathcal{B}}$ with small dynamic regret encodes near-optimal strategies of both players. See Appendix F for details.

6 Conclusion

In this work, we investigate the fundamental statistical and computational limits of classification-based online imitation learning (COIL). On the positive side, we propose the LOGGER framework that enables the design of oracle and regret efficient COIL algorithms with different sample and interaction round complexity tradeoffs, outperforming the behavior cloning baseline. On the negative side, we establish impossibility results for sublinear static regret using proper learning in the COIL setting, a subtle but important observation overlooked by prior works. We also show the impossibility to design sublinear dynamic regret algorithms in the LOGGER framework.

Looking forward, we would like to investigate the optimality of our sample complexity and interaction round complexity guarantees; we are also eager to see the possibility of relaxing the small separator assumption for broader applications; we would also like to empirically evaluate our algorithms.

Broader Impact: To the best of our knowledge, there is no negative social impact by our work.

References

[1] 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.

[2] Jacob Abernethy, Chansoo Lee, Abhinav Sinha, and Ambuj Tewari. Online linear optimization via smoothing. In *Conference on Learning Theory*, pages 807–823. PMLR, 2014.

[3] Alekh Agarwal, Daniel Hsu, Satyen Kale, John Langford, Lihong Li, and Robert Schapire. Taming the monster: A fast and simple algorithm for contextual bandits. In *International Conference on Machine Learning*, pages 1638–1646. PMLR, 2014.

[4] Alekh Agarwal, Nan Jiang, Sham M. Kakade, and Wen Sun. Reinforcement learning: Theory and algorithms, 2020.

[5] Shai Ben-David, David Loker, Nathan Srebro, and Karthik Sridharan. Minimizing the misclassification error rate using a surrogate convex loss. In *ICML*, 2012.

[6] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. *Advances in neural information processing systems*, 28, 2015.

[7] Dimitri P Bertsekas. Stochastic optimization problems with nondifferentiable cost functionals. *Journal of Optimization Theory and Applications*, 12(2):218–231, 1973.

[8] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[9] Nicolo Cesa-Bianchi and Gábor Lugosi. *Prediction, learning, and games*. Cambridge university press, 2006.

[10] Xi Chen and Xiaotie Deng. Settling the complexity of two-player nash equilibrium. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 261–272. IEEE, 2006.

[11] Ching-An Cheng and Byron Boots. Convergence of value aggregation for imitation learning. In *International Conference on Artificial Intelligence and Statistics*, pages 1801–1809. PMLR, 2018.

[12] Ching-An Cheng, Jonathan Lee, Ken Goldberg, and Byron Boots. Online learning with continuous variations: Dynamic regret and reductions. In *International Conference on Artificial Intelligence and Statistics*, pages 2218–2228. PMLR, 2020.

[13] Ching-An Cheng, Xinyan Yan, Nathan Ratliff, and Byron Boots. Predictor-corrector policy optimization. In *International Conference on Machine Learning*, pages 1151–1161. PMLR, 2019.

[14] Ching-An Cheng, Xinyan Yan, Evangelos Theodorou, and Byron Boots. Accelerating imitation learning with predictive models. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 3187–3196. PMLR, 2019.

[15] Chao-Kai Chiang, Tianbao Yang, Chia-Jung Lee, Mehrdad Mahdavi, Chi-Jen Lu, Rong Jin, and Shenghuo Zhu. Online optimization with gradual variations. In *Conference on Learning Theory*, pages 6–1. JMLR Workshop and Conference Proceedings, 2012.

[16] Thomas M Cover. Behavior of sequential predictors of binary sequences. Technical report, STANFORD UNIV CALIF STANFORD ELECTRONICS LABS, 1966.

[17] Christoph Dann, Nan Jiang, Akshay Krishnamurthy, Alekh Agarwal, John Langford, and Robert E Schapire. On oracle-efficient pac rl with rich observations. *Advances in neural information processing systems*, 31, 2018.

[18] Constantinos Daskalakis, Paul W Goldberg, and Christos H Papadimitriou. The complexity of computing a nash equilibrium. *SIAM Journal on Computing*, 39(1):195–259, 2009.

[19] Hal Daumé, John Langford, and Daniel Marcu. Search-based structured prediction. *Machine learning*, 75(3):297–325, 2009.

[20] Miroslav Dudík, Nika Haghtalab, Haipeng Luo, Robert E Schapire, Vasilis Syrgkanis, and Jennifer Wortman Vaughan. Oracle-efficient online learning and auction design. *Journal of the ACM (JACM)*, 67(5):1–57, 2020.

[21] Miroslav Dudik, Daniel Hsu, Satyen Kale, Nikos Karampatziakis, John Langford, Lev Reyzin, and Tong Zhang. Efficient optimal learning for contextual bandits. In *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, pages 169–178, 2011.

[22] Francisco Facchinei and Jong-Shi Pang. *Finite-dimensional variational inequalities and complementarity problems*. Springer Science & Business Media, 2007.

[23] 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.

[24] Elad Hazan and Tomer Koren. The computational power of optimization in online learning. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 128–141, 2016.

[25] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016.

[26] Kshitij Judah, Alan P Fern, Thomas G Dietterich, and Prasad Tadepalli. Active imitation learning: Formal and practical reductions to iid learning. *Journal of Machine Learning Research*, 15(120):4105–4143, 2014.

[27] Anatoli Juditsky, Arkadi Nemirovski, and Claire Tauvel. Solving variational inequalities with stochastic mirror-prox algorithm. *Stochastic Systems*, 1(1):17–58, 2011.

[28] Sham M. Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *ICML*, 2002.

[29] Sham M Kakade, Shai Shalev-Shwartz, and Ambuj Tewari. Regularization techniques for learning with matrices. *The Journal of Machine Learning Research*, 13(1):1865–1890, 2012.

[30] Jonathan N Lee, Michael Laskey, Ajay Kumar Tanwani, Anil Aswani, and Ken Goldberg. Dynamic regret convergence analysis and an adaptive regularization algorithm for on-policy robot imitation learning. *The International Journal of Robotics Research*, 40(10-11):1284–1305, 2021.

[31] Carlton E Lemke and Joseph T Howson, Jr. Equilibrium points of bimatrix games. *Journal of the Society for industrial and Applied Mathematics*, 12(2):413–423, 1964.

[32] Nick Littlestone and Manfred K Warmuth. The weighted majority algorithm. *Information and computation*, 108(2):212–261, 1994.

[33] Arkadi Nemirovski. Prox-method with rate of convergence o (1/t) for variational inequalities with lipschitz continuous monotone operators and smooth convex-concave saddle point problems. *SIAM Journal on Optimization*, 15(1):229–251, 2004.

[34] Francesco Orabona. A modern introduction to online learning. *arXiv preprint arXiv:1912.13213*, 2019.

[35] Takayuki Osa, Joni Pajarinen, Gerhard Neumann, J Andrew Bagnell, Pieter Abbeel, Jan Peters, et al. An algorithmic perspective on imitation learning. *Foundations and Trends® in Robotics*, 7(1-2):1–179, 2018.

[36] Yunpeng Pan, Ching-An Cheng, Kamil Saigol, Keuntaek Lee, Xinyan Yan, Evangelos A Theodorou, and Byron Boots. Imitation learning for agile autonomous driving. *The International Journal of Robotics Research*, 39(2-3):286–302, 2020.

[37] Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. *Advances in neural information processing systems*, 1, 1988.

[38] Jian Qian, Ronan Fruit, Matteo Pirotta, and Alessandro Lazaric. Concentration inequalities for multinoulli random variables. *arXiv preprint arXiv:2001.11595*, 2020.

[39] Nived Rajaraman, Yanjun Han, Lin Yang, Jingbo Liu, Jiantao Jiao, and Kannan Ramchandran. On the value of interaction and function approximation in imitation learning. *Advances in Neural Information Processing Systems*, 34, 2021.

[40] Alexander Rakhlin and Karthik Sridharan. Online learning with predictable sequences. In *Conference on Learning Theory*, pages 993–1019. PMLR, 2013.

[41] Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 661–668. JMLR Workshop and Conference Proceedings, 2010.

[42] Stephane Ross and J Andrew Bagnell. Reinforcement and imitation learning via interactive no-regret learning. *arXiv preprint arXiv:1406.5979*, 2014.

[43] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635, 2011.

[44] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[45] Shai Shalev-Shwartz et al. Online learning and online convex optimization. *Foundations and trends in Machine Learning*, 4(2):107–194, 2011.

[46] 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.

[47] Jacob Steinhardt and Percy Liang. Adaptivity and optimism: An improved exponentiated gradient algorithm. In *International Conference on Machine Learning*, pages 1593–1601. PMLR, 2014.

[48] Wen Sun, Arun Venkatraman, Geoffrey J Gordon, Byron Boots, and J Andrew Bagnell. Deeply aggrevated: Differentiable imitation learning for sequential prediction. In *International Conference on Machine Learning*, pages 3309–3318. PMLR, 2017.

[49] Umar Syed and Robert E Schapire. A game-theoretic approach to apprenticeship learning. *Advances in neural information processing systems*, 20, 2007.

[50] Umar Syed and Robert E Schapire. A reduction from apprenticeship learning to classification. *Advances in neural information processing systems*, 23, 2010.

[51] Vasilis Syrgkanis, Akshay Krishnamurthy, and Robert Schapire. Efficient algorithms for adversarial contextual learning. In *International Conference on Machine Learning*, pages 2159–2168. PMLR, 2016.

[52] Tianhao Wang, Si Chen, and Ruoxi Jia. One-round active learning. *arXiv preprint arXiv:2104.11843*, 2021.

[53] Ziyu Wang, Josh S Merel, Scott E Reed, Nando de Freitas, Gregory Wayne, and Nicolas Heess. Robust imitation of diverse behaviors. *Advances in Neural Information Processing Systems*, 30, 2017.

[54] Tsachy Weissman, Erik Ordentlich, Gadiel Seroussi, Sergio Verdu, and Marcelo J Weinberger. Inequalities for the l1 deviation of the empirical distribution. *Hewlett-Packard Labs, Tech. Rep*, 2003.

[55] Liu Yang and Jaime Carbonell. Buy-in-bulk active learning. *Advances in neural information processing systems*, 26, 2013.

[56] Constantin Zalinescu. *Convex analysis in general vector spaces*. World scientific, 2002.

[57] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, Anind K Dey, et al. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See Section 3.1, Section 3.2, Section 4, and Section 5.
   (b) Did you describe the limitations of your work? [Yes] See Section 2, Section 3.2, Definition 7 and Section 5.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Section 2, Section 3.2, Definition 7.
   (b) Did you include complete proofs of all theoretical results? [Yes] Please refer to Appendix for detailed proofs.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [N/A]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [N/A]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [N/A]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]