MobILE: Model-Based Imitation Learning From Observation Alone

Rahul Kidambi*
Amazon Search & AI
Berkeley CA 94704.
rk773@cornell.edu

Jonathan D. Chang
CS Department, Cornell University
Ithaca NY 14853.
jdc396@cornell.edu

Wen Sun
CS Department, Cornell University
Ithaca NY 14853.
ws455@cornell.edu

Abstract

This paper studies Imitation Learning from Observations alone (ILFO) where the learner is presented with expert demonstrations that consist only of states visited by an expert (without access to actions taken by the expert). We present a provably efficient model-based framework MobILE to solve the ILFO problem. MobILE involves carefully trading off strategic exploration against imitation - this is achieved by integrating the idea of optimism in the face of uncertainty into the distribution matching imitation learning (IL) framework. We provide a unified analysis for MobILE, and demonstrate that MobILE enjoys strong performance guarantees for classes of MDP dynamics that satisfy certain well studied notions of structural complexity. We also show that the ILFO problem is *strictly harder* than the standard IL problem by presenting an exponential sample complexity separation between IL and ILFO. We complement these theoretical results with experimental simulations on benchmark OpenAI Gym tasks that indicate the efficacy of MobILE. Code for implementing the MobILE framework is available at https://github.com/rahulkidambi/MobILE-NeurIPS2021.

1 Introduction

This paper considers *Imitation Learning from Observation Alone (ILFO)*. In ILFO, the learner is presented with sequences of states encountered by the expert, *without* access to the actions taken by the expert, meaning approaches based on a reduction to supervised learning (e.g., Behavior cloning (BC) [49], DAgger [50]) are not applicable. ILFO is more general and has potential for applications where the learner and expert have different action spaces, applications like sim-to-real [56, 14] etc.

Recently, [59] reduced the ILFO problem to a sequence of one-step distribution matching problems that results in obtaining a non-stationary policy. This approach, however, is sample inefficient for longer horizon tasks since the algorithm does not effectively reuse previously collected samples when solving the current sub-problem. Another line of work considers model-based methods to infer the expert's actions with either an inverse dynamics [63] or a forward dynamics [16] model; these recovered actions are then fed into an IL approach like BC to output the final policy. These works rely on stronger assumptions that are only satisfied for Markov Decision Processes (MDPs) with injective transition dynamics [68]; we return to this in the related works section.

*Work initiated when RK was a post-doc at Cornell University; work done outside Amazon.

35th Conference on Neural Information Processing Systems (NeurIPS 2021), virtual.

We introduce MobILE—Model-based Imitation Learning and Exploring, a model-based framework, to solve the ILFO problem. In contrast to existing model-based efforts, MobILE learns the forward transition dynamics model—a quantity that is well defined for any MDP. Importantly, MobILE *combines strategic exploration with imitation* by interleaving a model learning step with a bonus-based, optimistic distribution matching step – a perspective, to the best of our knowledge, that has not been considered in Imitation Learning. MobILE has the ability to automatically trade-off exploration and imitation. It simultaneously explores to collect data to refine the model and imitates the expert wherever the learned model is accurate and certain. At a high level, our theoretical results and experimental studies demonstrate that *systematic exploration is beneficial for solving ILFO reliably and efficiently,* and *optimism* is a both theoretically sound and practically effective approach for strategic exploration in ILFO (see Figure 1 for comparisons with other ILFO algorithms). This paper extends the realm of partial information problems (e.g. Reinforcement Learning and Bandits) where optimism has been shown to be crucial in obtaining strong performance, both in theory (e.g., E^3 [30], UCB [3]) and practice (e.g., RND [10]). This paper proves that incorporating optimism into the min-max IL framework [69, 22, 59] is *beneficial* for both the theoretical foundations and empirical performance of ILFO.

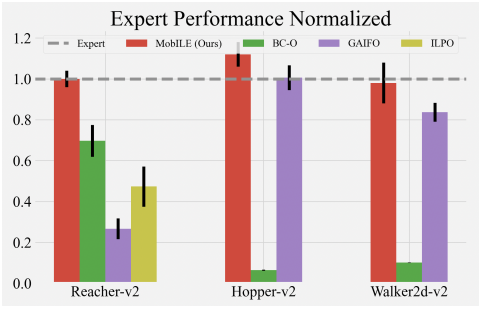

Figure 1: Expert performance normalized scores of ILFO algorithms averaged across 5 seeds in environments with discrete action spaces (Reacher-v2) and continuous action spaces (Hopper-v2 and Walker2d-v2).

Our Contributions: We present MobILE (Algorithm 1), a provably efficient, model-based framework for ILFO that offers competitive results in benchmark gym tasks. MobILE can be instantiated with various implementation choices owing to its modular design. This paper's contributions are:

1. The MobILE framework combines ideas of model-based learning, optimism for exploration, and adversarial imitation learning. MobILE achieves global optimality with near-optimal regret bounds for classes of MDP dynamics that satisfy certain well studied notions of complexity. The key idea of MobILE is to use optimism to *trade-off imitation and exploration*.
2. We show an exponential sample complexity gap between ILFO and classic IL where one has access to expert's actions. This indicates that ILFO is *fundamentally harder* than IL. Our lower bound on ILFO also indicates that to achieve near optimal regret, one needs to perform systematic exploration rather than random or no exploration, both of which will incur sub-optimal regret.
3. We instantiate MobILE with a model ensemble of neural networks and a disagreement-based bonus. We present experimental results on benchmark OpenAI Gym tasks, indicating MobILE compares favorably to or outperforms existing approaches. Ablation studies indicate that optimism indeed helps in significantly improving the performance in practice.

1.1 Related Works

Imitation Learning (IL) is considered through the lens of two types of approaches: (a) behavior cloning (BC) [45] which casts IL as a reduction to supervised or full-information online learning [49, 50], or, (b) (adversarial) inverse RL [40, 1, 69, 17, 22, 29, 18], which involves minimizing various distribution divergences to solve the IL problem, either with the transition dynamics known (e.g., [69]), or unknown (e.g., [22]). MobILE does not assume knowledge of the transition dynamics, is model-based, and operates without access to the expert's actions.

Imitation Learning from Observation Alone (ILFO) [59] presents a model-free approach FAIL that outputs a non-stationary policy by reducing the ILFO problem into a sequence of min-max problems, one per time-step. While being theoretically sound, this approach cannot share data across different time steps and thus is not data efficient for long horizon problems. Also FAIL in theory only works for discrete actions. In contrast, our paper learns a stationary policy using model-based approaches by reusing data across all time steps and extends to continuous action space. Another line of work [63, 16, 66] relies on learning an estimate of expert action, often through the use of an inverse dynamics models, $P^e(a|s;s')$. Unfortunately, an inverse dynamics model is not well defined

in many benign problem instances. For instance, [68, remark 1, section 9.3] presents an example showing that inverse dynamics isn't well defined except in the case when the MDP dynamics is injective (i.e., no two actions could lead to the same next state from the current state. Note that even deterministic transition dynamics doesn't imply injectivity of the MDP dynamics). Furthermore, ILPO [16] applies to MDPs with deterministic transition dynamics and discrete actions. `MobILE`, on the other hand, learns the forward dynamics model which is always unique and well-defined for both deterministic and stochastic transitions and works with discrete and continuous actions. Another line of work in ILFO revolves around using hand-crafted cost functions that may rely on task-specific knowledge [44, 4, 53]. The performance of policy outputted by these efforts relies on the quality of the engineered cost functions. In contrast, `MobILE` does not require cost function engineering.

Model-Based RL has seen several advances [61, 36, 13] including ones based on deep learning (e.g., [34, 19, 38, 24, 37, 65]). Given `MobILE`'s modularity, these advances in model-based RL can be translated to improved algorithms for the ILFO problem. `MobILE` bears parallels to provably efficient model-based RL approaches including E^3 [31, 27], R-MAX [7], UCRL [23], UCBVI [5], Linear MDP [67], LC^3 [25], Witness rank [58] which utilize optimism based approaches to trade-off exploration and exploitation. Our work utilizes optimism to trade-off *exploration and imitation*.

2 Setting

We consider episodic finite-horizon MDP $M = \{S, A, P^?, H, c, s_0\}$, where S, A are the state and action space, $P^? : S \times A \to \Delta(S)$ is the MDP's transition kernel, H is the horizon, s_0 is a fixed initial state (note that our work generalizes when we have a distribution over initial states), and c is the *state-dependent* cost function $c : S \to [0, 1]$. Our result can be extended to the setting where $c : S \times S \to [0, 1]$, i.e., the ground truth cost $c(s, s')$ depends on state and next state pairs. For analysis simplicity, we focus on $c : S \to [0, 1]$.[2]

We denote $d_P \in \Delta(S \times A)$ as the average state-action distribution of policy π under the transition kernel P, i.e., $d_P(s, a) := \frac{1}{H}\sum_{t=1}^{H} Pr(s_t = s, a_t = a | s_0, \pi, P)$, where $Pr(s_t = s, a_t = a | s_0, \pi, P)$ is the probability of reaching (s, a) at time step t starting from s_0 by following π under transition kernel P. We abuse notation and write $s \sim d_P$ to denote a state s is sampled from the state-wise distribution which marginalizes action over $d_P(s, a)$, i.e., $d_P(s) := \frac{1}{H}\sum_{t=1}^{H} Pr(s_t = s | s_0, \pi, P)$. For a given cost function $f : S \to [0, 1]$, $V_{P,f}^{\pi}$ denotes the expected total cost of π under transition P and cost function f. Similar to IL setting, in ILFO, the *ground truth cost c is unknown*. Instead, we can query the expert, denoted as $\pi_e : S \to \Delta(A)$. Note that the expert π_e could be stochastic and does not have to be the optimal policy. The expert, when queried, provides state-only demonstrations $\tau = \{s_0, s_1, \ldots, s_H\}$, where $s_{t+1} \sim P^?(\cdot | s_t, a_t)$ and $a_t \sim \pi_e(\cdot | s_t)$.

The goal is to leverage expert's state-wise demonstrations to learn a policy π that performs as well as π_e in terms of optimizing the ground truth cost c, with polynomial sample complexity on problem parameters such as horizon, number of expert samples and online samples and underlying MDP's complexity measures (see section 4 for precise examples). We track the progress of any (randomized) algorithm by measuring the (expected) regret incurred by a policy π defined as $E[V^{\pi}]$ V^{π} as a function of number of online interactions utilized by the algorithm to compute π.

2.1 Function Approximation Setup

Since the ground truth cost c is unknown, we utilize the notion of a function class (i.e., discriminators) $F \subseteq S \to [0, 1]$ to define the costs that can then be utilized by a planning algorithm (e.g. NPG [26]) for purposes of distribution matching with expert states. If the ground truth c depends (s, s'), we use discriminators $F \subseteq S \times S \to [0, 1]$. Furthermore, we use a model class $P \subseteq S \times A \to \Delta(S)$ to capture the ground truth transition $P^?$. For the theoretical results in the paper, we assume realizability:

Assumption 1. *Assume F and P captures ground truth cost and transition, i.e., $c \in F$, $P^? \in P$.*

We will use Integral probability metric (IPM) with F as our divergence measure. Note that if $c \in F$ and $c : S \to [0, 1]$, then IPM defined as $\max_{f \in \mathcal{F}} \mathbb{E}_{s \sim d} f(s) - \mathbb{E}_{s \sim d_e} f(s)$ directly upper

[2] Without any additional assumptions, in ILFO, learning to optimize action-dependent cost $c(s, a)$ (or $c(s, a, s')$) is **not possible**. For example, if there are two sequences of actions that generate the same sequence of states, without seeing expert's preference over actions, we do not know which actions to commit to.

Algorithm 1 MobILE: The framework of **Mo**del-**b**ased **I**mitation **L**earning and **E**xploring for ILFO

1: **Require**: IPM class F, dynamics model class P, policy class , bonus function class B, expert dataset D_e $fs_i^e g_{i=1}^N$.
2: Initialize policy $_0$ 2 , replay buffer $D_{-1} = ;$.
3: **for** $t = 0;$ $;T$ 1 **do**
4: Execute $_t$ in true environment $P^?$ to get samples $_t = fs_k; a_k g_{k=0}^{H-1}$ $[$ s_H. Append to replay buffer $D_t = D_{t-1}$ $[$ $_t$.
5: Update model and bonus: $\widehat{P}_{t+1} : S$ A $!$ S and $b_{t+1} : S$ A $!$ R^+ using buffer D_t.
6: Optimistic model-based min-max IL: obtain $_{t+1}$ by solving equation (1) with $\widehat{P}_{t+1}; b_{t+1}; D_e$.
7: **end for**
8: **Return** $_T$.

bounds sub-optimality gap V V ^{e}, where V is the expected total cost of under cost function c. This justifies why minimizing IPM between two state distributions suffices [22, 59]. Similarly, if c depends on $s; s'$, we can simply minimize IPM between two state-next state distributions, i.e., $\max_f E_{s;s^0 \sim d}$ $f(s; s')$ $E_{s;s^0 \sim d}$ $^e f(s; s')$ where discriminators now take $(s; s')$ as input.[3]

To permit generalization, we require P to have bounded complexity. For analytical simplicity, we assume F is discrete (but exponentially large), and we require the sample complexity of any PAC algorithm to scale polynomially with respect to its complexity $\ln(jFj)$. The $\ln jFj$ complexity can be replaced to bounded conventional complexity measures such as Rademacher complexity and covering number for continuous F (e.g., F being a Reproducing Kernel Hilbert Space).

3 Algorithm

We introduce MobILE (Algorithm 1) for the ILFO problem. MobILE utilizes (a) a function class F for Integral Probability Metric (IPM) based distribution matching, (b) a transition dynamics model class P for model learning, (c) a bonus parameterization B for exploration, (d) a policy class for policy optimization. At every iteration, MobILE (in Algorithm 1) performs the following steps:

1. **Dynamics Model Learning:** execute policy in the environment online to obtain state-action-next state $(s; a; s')$ triples which are appended to the buffer D. Fit a transition model $\widehat{P}$ on D.
2. **Bonus Design:** design bonus to incentivize exploration where the learnt dynamics model is uncertain, i.e. the bonus $b(s; a)$ is large at state s where $\widehat{P}(\ js; a)$ is uncertain in terms of estimating $P^?(\ js; a)$, while $b(s; a)$ is small where $\widehat{P}(\ js; a)$ is certain.
3. **Imitation-Exploration tradeoff:** Given discriminators F, model $\widehat{P}$, bonus b and expert dataset D_e, perform distribution matching by solving the model-based IPM objective with bonus:

$$_{t+1} \quad \arg\min_{\in} \max_{f \in \mathcal{F}} L(\ ; f; \widehat{P}; b; D_e) := E_{(s;a)\sim d_{\widehat{P}}}[f(s) \quad b(s; a)] \quad E_{s\sim \mathcal{D}_e}[f(s)]; \quad (1)$$

where $E_{s\sim \mathcal{D}_e} f(s) := \sum_{s\in \mathcal{D}_e} f(s)=jD_ej$.

Intuitively, the bonus cancels out discriminator's power in parts of the state space where the dynamics model $\widehat{P}$ is not accurate, thus offering freedom for MobILE to explore. We first explain MobILE's components and then discuss MobILE's key property—which is to trade-off *exploration and imitation*.

3.1 Components of MobILE

This section details MobILE's components.

Dynamics model learning: For the model fitting step in line 5, we assume that we get a calibrated model in the sense that: $k\widehat{P}_t(\ js; a)$ $P^?(\ js; a)k_1$ $_t(s; a); 8s; a$ for some uncertainty measure $_t(s; a)$, similar to model-based RL works, e.g. [12]. We discuss ways to estimate $_t(s; a)$ in the bonus estimation below. There are many examples (discussed in Section 4) that permit efficient

[3] we slightly abuse notation here and denote d as the average state-next state distribution of , i.e., d $(s; s') := d$ $(s) \int_a$ $(ajs)daP^?(s'js; a)$.

estimation of these quantities including tabular MDPs, Kernelized nonlinear regulator, nonparametric model such as Gaussian Processes. Consider a general function class $G \quad S \quad A \; ! \; S$, one can learn $\widehat{g}_t$ via solving a regression problem, i.e.,

$$\widehat{g}_t = \operatorname*{argmin}_{g \in \mathcal{G}} \sum_{s;a;s' \in \mathcal{D}_t} k g(s;a) \quad s' k_2^2; \tag{2}$$

and setting $\widehat{P}_t(\, js;a) = N\left(\widehat{g}_t(s;a);\ ^2 I\right)$, where, is the standard deviation of error induced by $\widehat{g}_t$. In practice, such parameterizations have been employed in several settings in RL with G being a multi-layer perceptron (MLP) based function class (e.g.,[48]). In Section 4, we also connect this with prior works in provable model-based RL literature.

Bonus: We utilize bonuses as a means to incentivize the policy to efficiently explore unknown parts of the state space for improved model learning (and hence better distribution matching). With the uncertainty measure $_t(s;a)$ obtained from calibrated model fitting, we can simply set the bonus $b_t(s;a) = O(H \ _t(s;a))$. How do we obtain $_t(s;a)$ in practice? For a general class G, given the least square solution $\widehat{g}_t$, we can define a version space G_t as: $G_t = \left\{ g \; 2 \; G : \sum_{i=0}^{t-1} \sum_{h=0}^{H-1} k g(s_h^t; a_h^t) \quad \widehat{g}_t(s_h^t; a_h^t) k_2^2 \quad z_t \right\}$, with z_t being a hyper parameter. The version space G_t is an *ensemble of functions* $g \; 2 \; G$ which has training error on D_t almost as small as the training error of the least square solution $\widehat{g}_t$. In other words, version space G_t contains functions that agree on the training set D_t. The uncertainty measure at $(s;a)$ is then the *maximum disagreement* among models in G_t, with $_t(s;a) \; / \; \sup_{f_1; f_2 \in \mathcal{G}_t} k f_1(s;a) \quad f_2(s;a) k_2$. Since $g \; 2 \; G_t$ agree on D_t, a large $_t(s;a)$ indicates $(s;a)$ is novel. See example 3 for more theoretical details.

Empirically, disagreement among an ensemble [41, 6, 11, 43, 37] is used for designing bonuses that incentivize exploration. We utilize an neural network ensemble, where each model is trained on D_t (via SGD on squared loss Eq. 2) with different initialization. This approximates the version space G_t, and the bonus is set as a function of maximum disagreement among the ensemble's predictions.

Optimistic model-based min-max IL: For model-based imitation (line 6), `MobILE` takes the current model $\widehat{P}_t$ and the discriminators F as inputs and performs policy search to minimize the divergence defined by $\widehat{P}_n$ and F: $d_t(\ ;\ ^e) := \max_{f \in \mathcal{F}} \left[\mathbb{E}_{s;a \sim d_{\widehat{P}_t}} (f(s) \quad b_t(s;a)) \quad \mathbb{E}_{s \sim d \ ^e} f(s) \right]$. Note that, for a fixed , the $\arg\max_{f \in \mathcal{F}}$ is identical with or without the bonus term, since $\mathbb{E}_{s;a \sim d_{\widehat{P}_t}} b_t(s;a)$ is independent of f. In our implementation, we use the Maximum Mean Discrepancy (MMD) with a Radial Basis Function (RBF) kernel to model discriminators F.[4] We compute $\operatorname{argmin} \ d_t(\ ;\ ^e)$ by iteratively (1) computing the argmax discriminator f given the current , and (2) using policy gradient methods (e.g., TRPO) to update inside $\widehat{P}_t$ with $f \quad b_t$ as the cost. Specifically, to find $_t$ (line 6), we iterate between the following two steps:

1. Cost update: $\hat{f} = \operatorname*{argmax}_{f \in \mathcal{F}} \mathbb{E}_{s \sim d^{\hat{}}_{\widehat{P}_t}} f(s) \quad \mathbb{E}_{s \sim \mathcal{D}^e} f(s)$; 2. PG Step: $\hat{} = \hat{} \qquad r \ \hat{V}_{\widehat{P}_t; \hat{f} - b_t}$;

where the PG step uses the learnt dynamics model $\widehat{P}_t$ and the optimistic IPM cost $\hat{f}(s) \quad b_t(s;a)$. Note that for MMD, the cost update step has a closed-form solution.

3.2 Exploration And Imitation Tradeoff

We note that `MobILE` is performing an automatic *trade-off between exploration and imitation*. More specifically, the bonus is designed such that it has high values in the state space that have not been visited, and low values in the state space that have been frequently visited by the sequence of learned policies so far. Thus, by incorporating the bonus into the discriminator $f \; 2 \; F$ (e.g., $\tilde{f}(s;a) = f(s) \quad b_t(s;a)$), we diminish the power of discriminator f at novel state-action space regions, which relaxes the state-matching constraint (as the bonus cancels the penalty from the discriminators) at those novel regions so that exploration is encouraged. For well explored states, we force the learner's states to match the expert's using the full power of the discriminators. Our work uses optimism (via coupling bonus and discriminators) to carefully balance imitation and exploration.

[4] For MMD with kernel k, $F = f w^\top \ (s;a) j k w k_2 \quad 1 g$ where $: h \ (s;a);\ (s';a') i = k((s;a);(s';a'))$.

4 Analysis

This section presents a general theorem for MobILE that uses the notion of *information gain* [57], and then specializes this result to common classes of stochastic MDPs such as discrete (tabular) MDPs, Kernelized nonlinear regulator [28], and general function class with bounded Eluder dimension [51].

Recall, Algorithm 1 generates one state-action trajectory $\tau^t := \{s_h^t, a_h^t\}_{h=0}^{H}$ at iteration t and estimates model $\widehat{P}_t$ based on $\mathcal{D}_t = \tau^0, \ldots, \tau^{t-1}$. We present our theorem under the assumption that model fitting gives us a model $\widehat{P}$ and a confidence interval of the model's prediction.

Assumption 2 (Calibrated Model)**.** *For all iteration t with $t \geq 0$, with probability $1-\delta$, we have a model $\widehat{P}_t$ and its associated uncertainty measure $\sigma_t : S \times A \mapsto \mathbb{R}^+$, such that for all $s, a \in S \times A$*[5]

$$\left\| \widehat{P}_t(\cdot|s,a) - P^\star(\cdot|s,a) \right\|_1 \leq \min\{\sigma_t(s,a), 2\}.$$

Assumption 2 has featured in prior works (e.g., [12]) to prove regret bounds in model-based RL. Below we demonstrate examples that satisfy the above assumption.

Example 1 (Discrete MDPs)**.** *Given $\mathcal{D}_t$, denote $N(s,a)$ as the number of times (s,a) appears in $\mathcal{D}_t$, and $N(s,a,s')$ number of times (s,a,s') appears in $\mathcal{D}_t$. We can set $\widehat{P}_t(s'|s,a) = N(s,a,s')/N(s,a), \forall s,a,s'$. We can set $\sigma_t(s,a) = \widetilde{O}\left(\sqrt{S/N(s,a)}\right)$.*

Example 2 (KNRs [28])**.** *For KNR, we have $P^\star(\cdot|s,a) = \mathcal{N}\left(W^\star \phi(s,a), \sigma^2 I\right)$ where feature mapping $\phi(s,a) \in \mathbb{R}^d$ and $\|\phi(s,a)\|_2 \leq 1$ for all s,a.*[6] *We can learn $\widehat{P}_t$ via Kernel Ridge regression, i.e., $\widehat{g}_t(s,a) = \widehat{W}_t \phi(s,a)$ where*

$$\widehat{W}_t = \operatorname*{argmin}_W \sum_{s,a,s' \in \mathcal{D}_t} \|W \phi(s,a) - s'\|_2^2 + \|W\|_F^2$$

where $\|\cdot\|_F$ is the Frobenius norm. The uncertainty measure $\sigma_t(s,a) = \gamma_t \|\phi(s,a)\|_{\Sigma_t^{-1}}$, $\gamma_t = \{2\|W^\star\|_2^2 + 8\sigma^2[d_s \ln(5) + 2\ln(t^2/\delta) + \ln(4) + \ln(\det(\Sigma_t)/\det(\lambda I))]\}^{1/2}$, and, $\Sigma_t = \sum_{k=0}^{t-1}\sum_{h=1}^{H-1} \phi(s_h^k, a_h^k)\phi(s_h^k, a_h^k)^\top + \lambda I$ with $\lambda > 0$. See Proposition 12 for more details.

Similar to RKHS, Gaussian processes (GPs) offers a calibrated model [57]. Note that GPs offer similar regret bounds as RKHS; so we do not discuss GPs and instead refer readers to [12].

Example 3 (General class $\mathcal{G}$)**.** *In this case, assume we have $P^\star(\cdot|s,a) = \mathcal{N}(g^\star(s,a), \sigma^2 I)$ with $g^\star \in \mathcal{G}$. Assume $\mathcal{G}$ is discrete (but could be exponentially large with complexity measure, $\ln(|\mathcal{G}|)$), and $\sup_{g\in\mathcal{G};s,a} \|g(s,a)\|_2 \leq \mathcal{G} \in \mathbb{R}^+$. Suppose model learning step is done by least square: $\widehat{g}_t = \operatorname{argmin}_{g\in\mathcal{G}} \sum_{k=0}^{t-1}\sum_{h=0}^{H-1} \|g(s_h^k, a_h^k) - s_{h+1}^k\|_2^2$. Compute a version space $\mathcal{G}_t = \left\{g \in \mathcal{G} : \sum_{k=0}^{t-1}\sum_{h=0}^{H-1} \|g(s_h^k, a_h^k) - \widehat{g}_t(s_h^k, a_h^k)\|_2^2 \leq z_t\right\}$, where $z_t = 2\sigma^2\mathcal{G}^2 \ln(2t^2|\mathcal{G}|/\delta)$ and use this for uncertainty computation. In particular, set uncertainty $\sigma_t(s,a) = \frac{1}{2}\max_{g_1\in\mathcal{G};g_2\in\mathcal{G}} \|g_1(s,a) - g_2(s,a)\|_2$, i.e., the maximum disagreement between any two functions in the version space $\mathcal{G}_t$. Refer to Proposition 14 for more details.*

The maximum disagreement above motivates our practical implementation where we use an ensemble of neural networks to approximate the version space and use the maximum disagreement among the models' predictions as the bonus. We refer readers to Section 6 for more details.

4.1 Regret Bound

We bound regret with the quantity named *Information Gain* $\mathcal{I}$ (up to some constant scaling factor) [57]:

$$\mathcal{I}_T := \max_{\text{Alg}} \mathbb{E}_{\text{Alg}}\left[\sum_{t=0}^{T-1}\sum_{h=0}^{H-1} \min\left\{\sigma_t^2(s_h^t, a_h^t), 1\right\}\right]; \tag{3}$$

[5] the uncertainty measure $\sigma_t(s,a)$ will depend on the input failure probability δ, which we drop here for notational simplicity. When we introduce specific examples, we will be explicit about the dependence on the failure probability δ which usually is in the order of $\ln(1/\delta)$.

[6] The covariance matrix can be generalized to any PSD matrix with bounded condition number.

where Alg is any adaptive algorithm (thus including Algorithm 1) that maps from history before iteration t to some policy $_t \; 2 \quad$. After the main theorem, we give concrete examples for I_T where we show that I_T has extremely mild growth rate with respect to T (i.e., logarithimic). Denote $V \quad$ as the expected total cost of $\quad$ under the true cost function c and the real dynamics $P^?$.

Theorem 3 (Main result)**.** *Assume model learning is calibrated (i.e., Assumption 2 holds for all t) and Assumption 1 holds. In Algorithm 1, set bonus $b_t(s;a) := H \min\{ _t(s;a); 2\}$. There exists a set of parameters, such that after running Algorithm 1 for T iterations, we have:*

$$\mathrm{E}\left[\min_{t\in[0;\cdots;T-1]} V^{\ t} \quad V^{\ e}\right] \quad O\left(\frac{H^{2:5}\sqrt{I_T}}{\sqrt{T}} + H\sqrt{\frac{\ln(THjFj)}{N}}\right):$$

Appendix A contains proof of Theorem 3. This theorem indicates that as long as I_T grows sublinearly $o(T)$, we find a policy that is at least as good as the expert policy when T and N approach infinity. For any discrete MDP, KNR [28], Gaussian Processes models [57], and general G with bounded Eluder dimension ([52, 42]), we can show that the growth rate of I_T with respect to T is mild.

Corollary 4 (Discrete MDP)**.** *For discrete MDPs, $I_T = \widetilde{O}(HS^2A)$ where $S = jSj; A = jAj$. Thus:*

$$\mathrm{E}\left[\min_{t\in[0;\cdots;T-1]} V^{\ t} \quad V^{\ e}\right] = \widetilde{O}\left(\frac{H^3S\sqrt{A}}{\sqrt{T}} + H\sqrt{\frac{\ln(jFj)}{N}}\right):$$

Note that Corollary 4 (proof in Appendix A.1) hold for *any* MDPs (not just injective MDPs) and any stochastic expert policy. The dependence on $A; T$ is tight (see lower bound in 4.2). Now we specialize Theorem 3 to continuous MDPs below.

Corollary 5 (KNRs (Example 2))**.** *For simplicity, consider the finite dimension setting $: S \quad A \; 7!$ $\mathbb{R}^d$. We can show that $I_T = \widetilde{O}\left(Hd + Hdd_s + Hd^2\right)$ (see Proposition 13 for details), where d is the dimension of the feature $(s;a)$ and d_s is the dimension of the state space. Thus, we have* [7]

$$\mathrm{E}\left[\min_{t\in[0;\cdots;T-1]} V^{\ t} \quad V^{\ e}\right] = \widetilde{O}\left(\frac{H^3\sqrt{dd_s + d^2}}{\sqrt{T}} + H\sqrt{\frac{\ln(jFj)}{N}}\right):$$

Corollary 6 (General G with bounded Eluder dimension (Example 3))**.** *For general G, assume that G has Eluder-dimension $d_E(\ )$ (Definition 3 in [42]). Denote $d_E = d_E(1=TH)$. The information gain is upper bounded as $I_T = O\left(Hd_E + d_E \ln(T^3HjGj)\ln(TH)\right)$ (see Proposition 16). Thus,*

$$\mathrm{E}\left[\min_{t\in[0;\cdots;T-1]} V^{\ t} \quad V^{\ e}\right] = \widetilde{O}\left(\frac{H^3\sqrt{d_E \ln(THjGj)}}{\sqrt{T}} + H\sqrt{\frac{\ln(jFj)}{N}}\right):$$

Thus as long as G has bounded complexity in terms of the Eluder dimension [52, 42], `MobILE` with the maximum disagreement-based optimism leads to near-optimal guarantees.

4.2 Exploration in ILFO and the Exponential Gap between IL and ILFO

To show the benefit of strategic exploration over random exploration in ILFO, we present a *novel* reduction of the ILFO problem to a bandit optimization problem, for which strategic exploration is known to be *necessary* [9] for optimal bounds while random exploration is suboptimal; this reduction indicates that benefit of strategic exploration for solving ILFO efficiently. This reduction also demonstrate that there exists an exponential gap in terms of sample complexity between ILFO and classic IL that has access to expert actions. We leave the details of the reduction framework in Appendix A.4. The reduction allows us to derive the following lower bound for any ILFO algorithm.

Theorem 7. *There exists an MDP with number of actions $A \quad 2$, such that even with infinitely many expert data, any ILFO algorithm must occur expected commutative regret $(\sqrt{AT})$.*

Specifically we rely on the following reduction where solving ILFO, with even infinite expert data, is at least as hard as solving an MAB problem with the known optimal arm's mean reward which itself

[7] We use $\widetilde{O}$ to suppress log term except the $\ln(jGj)$ and $\ln(jFj)$ which present the complexity of F and G.

occurs the same worst case $\sqrt{AT}$ cumulative regret bound as the one in the classic MAB setting. For MAB, it is known that random exploration such as -greedy will occur suboptimal regret $O(T^{2=3})$. Thus to achieve optimal $\sqrt{T}$ rate, one needs to leverage strategic exploration (e.g., optimism).

Methods such as BC for IL have sample complexity that scales as poly $\ln(A)$, e.g., see [2, Theorem 14.3, Chapter 14] which shows that for tabular MDP, BC learns a policy whose performance is $O(H^2\sqrt{S\ln(A)=N})$ away from the expert's performance (here S is the number of states in the tabular MDP). Similarly, in interactive IL setting, DAgger [50] can also achieve poly $\ln(A)$ dependence in sample complexity. The *exponential gap* in the sample complexity dependence on A between IL and ILFO formalizes the additional difficulty encountered by learning algorithms in ILFO.

5 Practical Instantiation of `MobILE`

We present a brief practical instantiation `MobILE`'s components with details in Appendix Section C.
Dynamics model learning: We employ Gaussian Dynamics Models parameterized by an MLP [48, 32], i.e., $\widehat{P}(s;a) := N(h(s;a);\ ^2I)$, where, $h(s;a) = s +$ $_s$ MLP $(s_c;a_c)$, where, are MLP's trainable parameters, $s_c = (s \quad _s)=$ $_s$, $a_c = (a \quad _a)=$ $_a$ with $_s$; $_a$ (and $_s$; $_a$) being the mean of states, actions (and standard deviation of states and actions) in the replay buffer D. Next, for $(s;a;s') \ 2\ D$, $_s = s' \quad s$ and $_s$ is the standard deviation of the state differences $_s \ 2\ D$. We use SGD with momentum [60] for training the parameters of the MLP.
Discriminator parameterization: We utilize MMD as our choice of IPM and define the discriminator as $f(s) = w^\top$ (s), where, (s) are Random Fourier Features [46].
Bonus parameterization: We utilize the discrepancy between predictions of a pair of dynamics models $h_{\ 1}(s;a)$ and $h_{\ 2}(s;a)$ for designing the bonus. Empirically, we found that using more than two models in the ensemble offered little to no improvements. Denote the disagreement at any $(s;a)$ as $(s;a) = kh_{\ 1}(s;a) \quad h_{\ 2}(s;a)k_2$, and $_D = \max_{(s;a)\sim D}$ $(s;a)$ is the max discrepancy of a replay buffer D. We set bonus as $b(s;a) = \quad \min(\ (s;a)=\ _D$, where > 0 is a tunable parameter.
PG oracle: We use TRPO [54] to perform incremental policy optimization inside the learned model.

6 Experiments

This section seeks to answer the following questions: (1) How does `MobILE` compare against other benchmark algorithms? (2) How does optimism impact sample efficiency/final performance? (3) How does increasing the number of expert samples impact the quality of policy outputted by `MobILE`?

We consider tasks from Open AI Gym [8] simulated with Mujoco [62]: `Cartpole-v1`, `Reacher-v2`, `Swimmer-v2`, `Hopper-v2` and `Walker2d-v2`. We train an expert for each task using TRPO [54] until we obtain an expert policy of average value $460;\ \ 10;38;3000;2000$ respectively. We setup `Swimmer-v2`, `Hopper-v2`,`Walker2d-v2` similar to prior model-based RL works [33, 39, 38, 48, 32].

We compare `MobILE` against the following algorithms: Behavior Cloning (BC), GAIL [22], BC-O [63], ILPO [16] (for environments with discrete actions), GAIFO [64]. Furthermore, recall that BC and GAIL utilize both expert states and actions, information that is not available for ILFO. This makes both BC and GAIL idealistic targets for comparing ILFO methods like `MobILE` against. As reported by Torabi et al. [63], BC outperforms BC-O in all benchmark results. Moreover, our results indicate `MobILE` outperforms GAIL and GAIFO in terms of sample efficiency. With reasonable amount of parameter tuning, BC serves as a very strong baseline and nearly solves *deterministic* Mujoco environments. We use code released by the authors for BC-O and ILPO. For GAIL we use an open source implementation [21], and for GAIFO, we modify the GAIL implementation as described by the authors. We present our results through (a) learning curves obtained by averaging the progress of the algorithm across 5 seeds, and, (b) bar plot showing expert normalized scores averaged across 5 seeds using the best performing policy obtained with each seed. Normalized score refers to ratio of policy's score over the expert score (so that expert has normalized score of 1). For `Reacher-v2`, since the expert policy has a negative score, we add an constant before normalization. More details can be found in Appendix C.

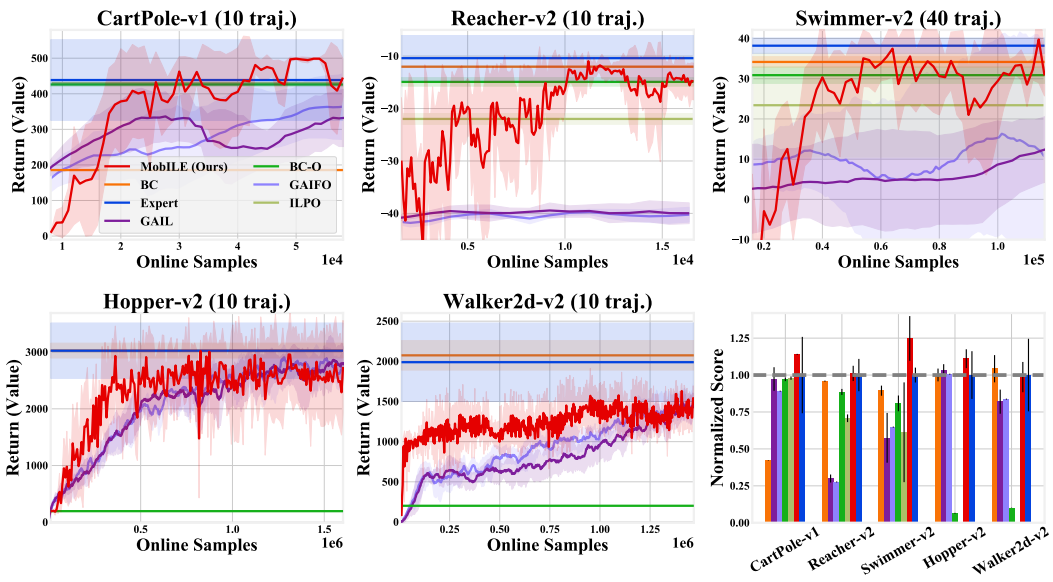

Figure 2: Comparing MobILE (red) against BC (orange), BC-O (green), GAIL (purple), GAIFO (periwinkle), ILPO (green olive). The learning curves are obtained by averaging all algorithms over 5 seeds. MobILE outperforms BC-O, GAIL and matches BC's behavior despite MobILE not having access to expert actions. The bar plot (bottom-right) presents the best performing policy outputted by each algorithm averaged across 5 seeds for each algorithm. MobILE clearly outperforms BC-O, GAIFO, ILPO while matching the behavior of IL algorithms like BC/GAIL which use expert actions.

6.1 Benchmarking MobILE on MuJoCo suite

Figure 2 compares MobILE with BC, BC-O, GAIL, GAIFO and ILPO. MobILE consistently matches or exceeds BC/GAIL's performance *despite BC/GAIL having access to actions taken by the expert* and MobILE functioning *without* expert action information. MobILE, also, consistently improves upon the behavior of ILFO methods such as BC-O, ILPO, and GAIFO. We see that BC does remarkably well in these benchmarks owing to determinism in the transition dynamics; in the appendix, we consider a variant of the cartpole environment with stochastic dynamics. Our results suggest that BC struggles with stochasticity in the dynamics and fails to solve this task, while MobILE continues to reliably solve this task. Also, note that we utilize 10 expert trajectories for all environments except Swimmer-v2; this is because all algorithms (including MobILE) present results with high variance. We include a learning curve for Swimmer-v2 with 10 expert trajectories in the appendix. The bar plot in Figure 2 shows that within the sample budget shown in the learning curves, MobILE (being a model-based algorithm), presents superior performance in terms of matching expert, thus indicating it is more sample efficient than GAIFO, GAIL (both being model-free methods), ILPO and BC-O.

6.2 Importance of the optimistic MDP construction

Figure 3 presents results obtained by running MobILE with and without optimism. In the absence of optimism, the algorithm either tends to be sample inefficient in achieving expert performance or

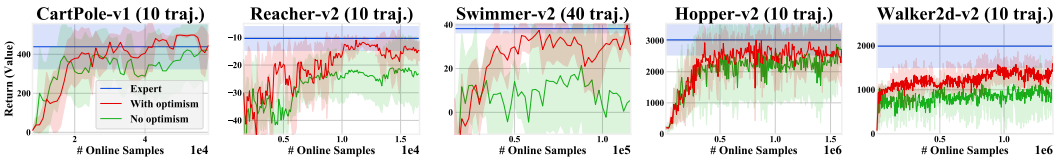

Figure 3: Learning curves obtained by running MobILE with (red) and without (green) optimism. Without optimism, the algorithm learns slowly or does not match the expert, whereas, with optimism, MobILE shows improved behavior by automatically trading off exploration and imitation.

completely fails to solve the problem. Note that without optimism, the algorithm isn't explicitly incentivized to explore – only implicitly exploring due to noise induced by sampling actions. This, however, is not sufficient to solve the problem efficiently. In contrast, MobILE with optimism presents improved behavior and in most cases, solves the environments with fewer online interactions.

6.3 Varying Number of Expert Samples

Table 1 shows the impact of increasing the number of samples drawn from the expert policy for solving the ILFO problem. The main takeaway is that increasing the number of expert samples aids MobILE in reliably solving the problem (i.e. with lesser variance).

Table 1: Expert normalized score and standard deviation of policy outputted by MobILE when varying number of expert trajectories as E_1 and E_2 (specific values represented in parentheses)

Environment	E_1		E_2		Expert	
Cartpole-v1	1:07	0:15 (5)	1:14	0 (10)	1	0:25
Reacher-v2	1:01	0:05 (10)	0:997	0:055 (20)	1	0:11
Swimmer-v2	1:54	1:1 (10)	1:25	0:15 (40)	1	0:05
Hopper-v2	1:11	0:064 (10)	1:16	0:03 (40)	1	0:16
Walker2d-v2	0:975	0:12 (10)	0:94	0:038 (50)	1	0:25

7 Conclusions

This paper introduces MobILE, a model-based ILFO approach that is applicable to MDPs with stochastic dynamics and continuous action spaces. MobILE trades-off exploration and imitation, and this perspective is shown to be important for solving the ILFO efficiently both in theory and in practice. Future works include exploring other means for learning dynamics models, performing strategic exploration and extending MobILE to problems with rich observation spaces (e.g. videos).

By not even needing the actions to imitate, ILFO algorithms allow for learning algorithms to capitalize on large amounts of video data available online. Moreover, in ILFO, the learner is successful if it learns to imitate the expert. Any expert policy designed by bad actors can naturally lead to obtaining new policies that continue to imitate and be a negative influence to the society. With this perspective in mind, any expert policy must be thoroughly vetted in order to ensure ILFO algorithms including MobILE are employed in ways that benefit the society.

Acknowledgements

Rahul Kidambi acknowledges funding from NSF TRIPODS Award CCF 1740822 at Cornell University. All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

References

[1] Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *ICML*. ACM, 2004.

[2] Alekh Agarwal, Nan Jiang, Sham M Kakade, and Wen Sun. Reinforcement learning: Theory and algorithms. *CS Dept., UW Seattle, Seattle, WA, USA, Tech. Rep*, 2019.

[3] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2):235–256, 2002.

[4] Yusuf Aytar, Tobias Pfaff, David Budden, Tom Le Paine, Ziyu Wang, and Nando de Freitas. Playing hard exploration games by watching youtube. In *NeurIPS*, pages 2935–2945, 2018.

[5] Mohammad Gheshlaghi Azar, Ian Osband, and Rémi Munos. Minimax regret bounds for reinforcement learning. In *International Conference on Machine Learning*, pages 263–272, 2017.

[6] Kamyar Azizzadenesheli, Emma Brunskill, and Animashree Anandkumar. Efficient exploration through bayesian deep q-networks. In *ITA*, pages 1–9. IEEE, 2018.

[7] Ronen I. Brafman and Moshe Tennenholtz. R-max - a general polynomial time algorithm for near-optimal reinforcement learning. *J. Mach. Learn. Res.*, 3:213–231, 2001.

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

[9] Sébastien Bubeck and Nicolò Cesa-Bianchi. Regret analysis of stochastic and non-stochastic multi-armed bandit problems. *Found. Trends Mach. Learn*, 5(1):1–122, 2012.

[10] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

[11] Yuri Burda, Harrison Edwards, Amos J. Storkey, and Oleg Klimov. Exploration by random network distillation. In *ICLR*. OpenReview.net, 2019.

[12] Sebastian Curi, Felix Berkenkamp, and Andreas Krause. Efficient model-based reinforcement learning through optimistic policy search and planning. *arXiv preprint arXiv:2006.08684*, 2020.

[13] Marc Deisenroth and Carl E. Rasmussen. PILCO: A model-based and data-efficient approach to policy search. In *International Conference on Machine Learning*, pages 465–472, 2011.

[14] Siddharth Desai, Ishan Durugkar, Haresh Karnan, Garrett Warnell, Josiah Hanna, Peter Stone, and AI Sony. An imitation from observation approach to transfer learning with dynamics mismatch. *Advances in Neural Information Processing Systems*, 33, 2020.

[15] Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, Yuhuai Wu, and Peter Zhokhov. Openai baselines. https://github.com/openai/baselines, 2017.

[16] Ashley D. Edwards, Himanshu Sahni, Yannick Schroecker, and Charles L. Isbell Jr. Imitating latent policies from observation. In *ICML*, 2019.

[17] Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *ICML*, 2016.

[18] Seyed Kamyar Seyed Ghasemipour, Richard Zemel, and Shixiang Gu. A divergence minimization perspective on imitation learning methods. In *Conference on Robot Learning*, pages 1259–1277. PMLR, 2020.

[19] Shixiang Gu, Timothy Lillicrap, Ilya Sutskever, and Sergey Levine. Continuous deep q-learning with model-based acceleration, 2016.

[20] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *CoRR*, abs/1801.01290, 2018.

[21] Ashley Hill, Antonin Raffin, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, Rene Traore, Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, and Yuhuai Wu. Stable baselines. https://github.com/hill-a/stable-baselines, 2018.

[22] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *CoRR*, abs/1606.03476, 2016.

[23] Thomas Jaksch, Ronald Ortner, and Peter Auer. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research*, 11(Apr):1563–1600, 2010.

[24] Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. *CoRR*, abs/1906.08253, 2019.

[25] Sham Kakade, Akshay Krishnamurthy, Kendall Lowrey, Motoya Ohnishi, and Wen Sun. Information theoretic regret bounds for online nonlinear control. *arXiv preprint arXiv:2006.12466*, 2020.

[26] Sham M. Kakade. A natural policy gradient. In *NIPS*, pages 1531–1538, 2001.

[27] Sham M. Kakade, Michael J. Kearns, and John Langford. Exploration in metric state spaces. In *ICML*, 2003.

[28] Sham M. Kakade, Akshay Krishnamurthy, Kendall Lowrey, Motoya Ohnishi, and Wen Sun. Information theoretic regret bounds for online nonlinear control. In *NeurIPS*, 2020.

[29] Liyiming Ke, Matt Barnes, Wen Sun, Gilwoo Lee, Sanjiban Choudhury, and Siddhartha Srinivasa. Imitation learning as f-divergence minimization. *arXiv preprint arXiv:1905.12888*, 2019.

[30] Michael Kearns and Satinder Singh. Near-optimal reinforcement learning in polynomial time. *Machine learning*, 49(2-3):209–232, 2002.

[31] Michael Kearns and Satinder Singh. Near optimal reinforcement learning in polynomial time. *Machine Learning*, 49(2-3):209–232, 2002.

[32] Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel: Model-based offline reinforcement learning. *CoRR*, abs/2005.05951, 2020.

[33] Thanard Kurutach, Ignasi Clavera, Yan Duan, Aviv Tamar, and Pieter Abbeel. Model-ensemble trust-region policy optimization. In *ICLR*. OpenReview.net, 2018.

[34] Thomas Lampe and Martin A. Riedmiller. Approximate model-assisted neural fitted q-iteration. In *IJCNN*, pages 2698–2704. IEEE, 2014.

[35] Tor Lattimore and Csaba Szepesvári. *Bandit Algorithms*. Cambridge University Press, 2020.

[36] Weiwei Li and Emanuel Todorov. Iterative linear quadratic regulator design for nonlinear biological movement systems. In *ICINCO*, pages 222–229, 2004.

[37] Kendall Lowrey, Aravind Rajeswaran, Sham Kakade, Emanuel Todorov, and Igor Mordatch. Plan Online, Learn Offline: Efficient Learning and Exploration via Model-Based Control. In *International Conference on Learning Representations (ICLR)*, 2019.

[38] Yuping Luo, Huazhe Xu, Yuanzhi Li, Yuandong Tian, Trevor Darrell, and Tengyu Ma. Algorithmic framework for model-based deep reinforcement learning with theoretical guarantees. *arXiv preprint arXiv:1807.03858*, 2018.

[39] Anusha Nagabandi, Gregory Kahn, Ronald S. Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *IEEE International Conference on Robotics and Automation*, pages 7559–7566, 2018.

[40] Andrew Y. Ng and Stuart Russell. Algorithms for inverse reinforcement learning. In *Proc. ICML*, pages 663–670, 2000.

[41] Ian Osband, John Aslanides, and Albin Cassirer. Randomized prior functions for deep reinforcement learning. *CoRR*, abs/1806.03335, 2018.

[42] Ian Osband and Benjamin Van Roy. Model-based reinforcement learning and the Eluder dimension. In *Advances in Neural Information Processing Systems*, pages 1466–1474, 2014.

[43] Deepak Pathak, Dhiraj Gandhi, and Abhinav Gupta. Self-supervised exploration via disagreement. In *ICML*, pages 5062–5071, 2019.

[44] Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. Deepmimic: example-guided deep reinforcement learning of physics-based character skills. *ACM Trans. Graphics*, 2018.

[45] D. A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. Technical report, CMU, 1989.

[46] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems*, pages 1177–1184, 2008.

[47] Aravind Rajeswaran, Kendall Lowrey, Emanuel Todorov, and Sham Kakade. Towards Generalization and Simplicity in Continuous Control. In *NIPS*, 2017.

[48] Aravind Rajeswaran, Igor Mordatch, and Vikash Kumar. A game theoretic framework for model based reinforcement learning. *ArXiv*, abs/2004.07804, 2020.

[49] Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. In Yee Whye Teh and D. Mike Titterington, editors, *AISTATS*, JMLR Proceedings, pages 661–668, 2010.

[50] Stéphane Ross, Geoffrey J. Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*, pages 627–635, 2011.

[51] Daniel Russo and Benjamin Van Roy. Eluder dimension and the sample complexity of optimistic exploration. In *NIPS*, pages 2256–2264, 2013.

[52] Daniel Russo and Benjamin Van Roy. Learning to optimize via posterior sampling. *Mathematics of Operations Research*, 39(4):1221–1243, 2014.

[53] Karl Schmeckpeper, Oleh Rybkin, Kostas Daniilidis, Sergey Levine, and Chelsea Finn. Reinforcement learning with videos: Combining offline observations with interaction. *CoRR*, abs/2011.06507, 2020.

[54] John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization. *CoRR*, abs/1502.05477, 2015.

[55] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

[56] Yuda Song, Aditi Mavalankar, Wen Sun, and Sicun Gao. Provably efficient model-based policy adaptation. In *International Conference on Machine Learning*, pages 9088–9098. PMLR, 2020.

[57] Niranjan Srinivas, Andreas Krause, Sham M. Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009.

[58] Wen Sun, Nan Jiang, Akshay Krishnamurthy, Alekh Agarwal, and John Langford. Model-based rl in contextual decision processes: Pac bounds and exponential improvements over model-free approaches. In *Conference on Learning Theory*, pages 2898–2933. PMLR, 2019.

[59] Wen Sun, Anirudh Vemula, Byron Boots, and Drew Bagnell. Provably efficient imitation learning from observation alone. In *ICML*, volume 97. PMLR, 2019.

[60] Ilya Sutskever, James Martens, George E. Dahl, and Geoffrey E. Hinton. On the importance of initialization and momentum in deep learning. In *ICML*, volume 28, 2013.

[61] R. S. Sutton. First results with dyna, an integrated architecture for learning, planning, and reacting. In *Neural Networks for Control*, pages 179–189. The MIT Press: Cambridge, MA, USA, 1990.

[62] Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *IEEE International Conference on Intelligent Robots and Systems*, pages 5026–5033, 2012.

[63] Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. In *IJCAI*, pages 4950–4957, 2018.

[64] Faraz Torabi, Garrett Warnell, and Peter Stone. Generative adversarial imitation from observation. *arXiv preprint arXiv:1807.06158*, 2018.

[65] Tingwu Wang, Xuchan Bao, Ignasi Clavera, Jerrick Hoang, Yeming Wen, Eric Langlois, Shunshi Zhang, Guodong Zhang, Pieter Abbeel, and Jimmy Ba. Benchmarking model-based reinforcement learning. *arXiv preprint arXiv:1907.02057*, 2019.

[66] Chao Yang, Xiaojian Ma, Wenbing Huang, Fuchun Sun, Huaping Liu, Junzhou Huang, and Chuang Gan. Imitation learning from observations by minimizing inverse dynamics disagreement. In *NeurIPS*, 2019.

[67] Lin F Yang and Mengdi Wang. Reinforcement leaning in feature space: Matrix bandit, kernels, and regret bound. *arXiv preprint arXiv:1905.10389*, 2019.

[68] Zhuangdi Zhu, Kaixiang Lin, Bo Dai, and Jiayu Zhou. Off-policy imitation learning from observations. In *NeurIPS*, 2020.

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

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 1
   (b) Did you describe the limitations of your work? [Yes] See Section 4
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 7
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes] We have read the ethics review guidelines to ensure that this paper conforms to them.
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See section 4
   (b) Did you include complete proofs of all theoretical results? [Yes] See section 4 and supplemental results for all 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)? [Yes] See section 5. All hyperparameters, exact values, and environmental stats are detailed in the supplemental material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See section 6 and supplemental material
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See section 6. All of our results are presented with error bars across 5 random seeds.
   (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)? [No] We do not use any specialized hardware or large scale compute that would be prohibitive for an average end-user to run our experiments. All experiments were run on 1 core and 12 GB of RAM.
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? [Yes] See Section 6 for citations for OpenAI Gym and MuJoCo.
   (b) Did you mention the license of the assets? [No] OpenAI Gym is an opensource software and we paid for our license for MuJoCo.
   (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]

Contents

A Analysis of Algorithm 1

We start by presenting the proof for the unified main result in Theorem 3. We then discuss the bounds for special instances individually.

The following lemma shows that under Assumption 2, with $b_t(s;a) = H \min\{ _t(s;a); 2g$, we achieve *optimism* at all iterations.

Lemma 8 (Optimism)**.** *Assume Assumption 2 holds, and set* $b_t(s;a) = H \min\{ _t(s;a); 2g$. *For all state-wise cost function* $f : S \mapsto [0;1]$, *denote the bonus enhance cost as* $\widetilde{f}_t(s;a) := f(s) \quad b_t(s;a)$. *For all policy* , *we have the following optimism:*

$$V_{\widehat{P}_t;\widetilde{f}_t} \quad V_{P;f}; 8t:$$

Proof. In the proof, we drop subscript t for notation simplicity. We consider a fixed function f and policy . Also let us denote $\widehat{V}$ as the value function of under $(\widehat{P}; \widetilde{f})$, and V as the value function under $(P; f)$.

Let us start from $h = H$, where we have $\widehat{V}_H(s) = V_H(s) = 0$. Assume inductive hypothesis holds at $h + 1$, i.e., for any $s; a$, we have $\widehat{Q}_{h+1}(s;a) \quad Q_{h+1}(s;a)$. Now let us move to h. We have:

$$\begin{aligned}
\widehat{Q}_h(s;a) \quad Q_h(s;a) &= \widetilde{f}(s;a) + \mathbb{E}_{s' \sim \widehat{P}(\cdot|s;a)} \widehat{V}_{h+1}(s') \quad f(s) \quad \mathbb{E}_{s' \sim P(\cdot|s;a)} V_{h+1}(s') \\
&\quad H \min\{ (s;a); 2g + \mathbb{E}_{s' \sim \widehat{P}(\cdot|s;a)} V_{h+1}(s') \quad \mathbb{E}_{s' \sim P(\cdot|s;a)} V_{h+1}(s') \\
&\quad H \min\{ (s;a); 2g + H \left\| \widehat{P}(js;a) \quad P(js;a) \right\|_1 \\
&\quad H \min\{ (s;a); 2g + H \min\{ (s;a); 2g = 0;
\end{aligned}$$

where the first inequality uses the inductive hypothesis at time step $h + 1$. Finally, note that $V_h(s) = \mathbb{E}_{a \sim (s)} Q_h(s;a)$, which leads to $\widehat{V}_h(s) \quad V_h(s)$. This concludes the induction step. □

The next lemma concerns the statistical error from finite sample estimation of $\mathbb{E}_{s \sim d^e} f(s)$.

Lemma 9. *Fix* $2 (0;1)$. *For all* t, *we have that with probability at least* 1 ,

$$\left| \mathbb{E}_{s \sim d^e} f(s) \quad \sum_{i=1}^{N} f(s_i^e)=N \right| \quad 2\sqrt{\frac{\ln(2t^2 jF j=)}{N}}; 8f \ 2 \ F:$$

Proof. For any t, we set the failure probability to be $6 =(t^2 \ ^2)$ at iteration t where we abuse notation and point out that $= 3{:}14159:::$. Thus the total failure probability for all $t \ 2 \ \mathbb{N}$ is at most . We then apply classic Hoeffding inequality to bound $\mathbb{E}_{s \sim d^e} f(s) \quad \sum_{i=1}^{N} f(s_i^e)=N$ with the fact that $f(s) \ 2 \ [0;1]$ for all s. We conclude the proof by taking a union bound over all $f \ 2 \ F$. □

Note that here we have assumed $s_i^e \quad d^e$ is i.i.d sampled from d^e. This can easily be achieved by randomly sampling a state from each expert trajectory. Note that we can easily deal with i.i.d trajectories, i.e., if our expert data contains N many i.i.d trajectories $\{ ^1; : : : ; \ ^N\}$, we can apply concentration on the trajectory level, and get:

$$\left| \mathbb{E}_{ \sim ^e} \left[\sum_{h=0}^{H-1} f(s_h) \right] \quad \frac{1}{N} \sum_{i=1}^{N} \sum_{h=0}^{H-1} f(s_h^i) \right| \quad O\left(H\sqrt{\frac{\ln(t^2 jF j=)}{N}} \right);$$

where denotes that a trajectory being sampled based on , s_h^i denotes the state at time step h on the i-th expert trajectory. Also note that we have $\mathbb{E}_{s\sim d^{\ }} f(s) = \frac{1}{H}\mathbb{E}_{\ \sim}\left[\sum_{h=0}^{H-1} f(s_h)\right]$ for any ; f. Together this immediately implies that:

$$\left|\mathbb{E}_{s\sim d^{\ e}} f(s) \quad \frac{1}{NH}\sum_{i=1}^{N}\sum_{h=0}^{H-1} f(s_h^i)\right| \quad O\left(\sqrt{\frac{\ln(t^2 jFj= \ )}{N}}\right);$$

which matches to the bound in Lemma 9.

Now we conclude the proof for Theorem 3.

Proof of Theorem 3. Assume that Assumption 2 and the event in Lemma 9 hold. Denote the joint of these two events as E. Note that the probability of $\bar{E}$ is at most 2 . For notation simplicity, denote $_{stats} = 2\sqrt{\frac{\ln(2T^2|\mathcal{F}|= \ )}{N}}$.

In each model-based planning phase, recall that we perform model-based optimization on the following objective:

$$_t = \operatorname*{argmin}_{\in} \max_{f\in F}\left[\mathbb{E}_{s;a\sim d^{\ }_{\widehat{P}_t}}[f(s) \quad b_t(s;a)] \quad \sum_{i=1}^{N} f(s_i^e)=N\right]:$$

Note that for any , using the inequality in Lemma 9, we have:

$$\begin{aligned}
&\max_{f\in\mathcal{F}_t}\left[\mathbb{E}_{s;a\sim d^{\ }_{\widehat{P}_t}}(f(s) \quad b_t(s;a)) \quad \sum_{i=1}^{N} f(s_i^e)=N\right] \\
&= \max_{f\in\mathcal{F}}\left[\mathbb{E}_{s;a\sim d^{\ }_{\widehat{P}_t}}(f(s) \quad b_t(s;a)) \quad \mathbb{E}_{s\sim d^{\ e}} f(s) + \mathbb{E}_{s\sim d^{\ e}} f(s) \quad \sum_{i=1}^{N} f(s_i^e)=N\right] \\
&\quad \max_{f\in\mathcal{F}}\left[\mathbb{E}_{s;a\sim d^{\ }_{\widehat{P}_t}}(f(s) \quad b_t(s;a)) \quad \mathbb{E}_{s\sim d^{\ e}} f(s)\right] + \max_{f\in F}\left[\mathbb{E}_{s\sim d^{\ e}} f(s) \quad \sum_{i=1}^{N} f(s_i^e)=N\right] \\
&\quad \max_{f\in\mathcal{F}}\left[\mathbb{E}_{s;a\sim d^{\ }_{\widehat{P}_t}}(f(s) \quad b_t(s;a)) \quad \mathbb{E}_{s;a\sim d^{\ e}_{\widehat{P}_t}}(f(s) \quad b_t(s;a))\right] + \ _{stats}
\end{aligned}$$

where in the last inequality we use optimism from Lemma 8, i.e., $\mathbb{E}_{s;a\sim d^{\ e}_{\widehat{P}_t}}(f(s) \quad b_t(s;a))$ $\mathbb{E}_{s\sim d^{\ e}} f(s)$.

Hence, for $_t$, since it is the minimizer and $^{\ e}$ 2 , we must have:

$$\begin{aligned}
&\max_{f\in\mathcal{F}}\left[\mathbb{E}_{s;a\sim d^{\ t}_{\widehat{P}_t}}(f(s) \quad b_t(s;a)) \quad \sum_{i=1}^{N} f(s_i^e)=N\right] \\
&\quad \max_{f\in\mathcal{F}}\left[\mathbb{E}_{s;a\sim d^{\ e}_{\widehat{P}_t}}(f(s) \quad b_t(s;a)) \quad \sum_{i=1}^{N} f(s_i^e)=N\right] \\
&\quad \max_{f\in\mathcal{F}}\left[\mathbb{E}_{s;a\sim d^{\ e}_{\widehat{P}_t}}(f(s) \quad b_t(s;a)) \quad \mathbb{E}_{s;a\sim d^{\ e}_{\widehat{P}_t}}(f(s) \quad b_t(s;a))\right] + \ _{stats} = \ _{stats};
\end{aligned}$$

Note that F contains c, we must have:

$$\mathbb{E}_{s;a\sim d^{\ t}_{\widehat{P}_t}}[c(s) \quad b_t(s;a)] \quad \sum_{i=1}^{N} c(s_i^e)=N + \ _{stats} \quad \mathbb{E}_{s\sim d^{\ e}} c(s) + 2\ _{stats};$$

which means that $V^{\ t}_{\widehat{P}_t;\widetilde{c}_t} \quad V^{\ e} + 2H\ _{stats}$.

Now we compute the regret in episode t. First recall that $b_t(s;a) = H\min f\ _t(s;a);2g$, which means that $kb_tk_\infty \quad 2H$ as $kck_\infty \quad 1$, which means that $kc \quad b_tk_\infty \quad 2H$. Thus, $\left\|V_{\widehat{P};c-b_t}\right\|_\infty \quad 2H^2$. Recall simulation lemma (Lemma 18), we have:

$$V^{\ t} \quad V^{\ e} \quad V^{\ t} \quad V^{\ t}_{\widehat{P}_t;\widetilde{c}_t} + 2H\ _{stats}$$

$$
\begin{aligned}
&= H\mathbb{E}_{s,a\sim d_t}\left[\|\widetilde{c}_t(s,a) - c(s)\|_\infty + 2H^2\left\|\widehat{P}_t(\cdot|s,a) - P^\star(\cdot|s,a)\right\|_1\right] + 2H\,\epsilon_{stat}\\
&= H\mathbb{E}_{s,a\sim d_t}\left[H\min\{\Gamma_t(s,a), 2\} + 2H^2\left\|\widehat{P}_t(\cdot|s,a) - P^\star(\cdot|s,a)\right\|_1\right] + 2H\,\epsilon_{stat}\\
&\quad H\mathbb{E}_{s,a\sim d_t}\left[H\min\{\Gamma_t(s,a), 2\} + 2H^2\min\{\Gamma_t(s,a), 2\}\right] + 2H\,\epsilon_{stat}\\
&\quad 3H^3\mathbb{E}_{s,a\sim d_t}\min\{\Gamma_t(s,a), 2\} + 2H\,\epsilon_{stat}\\
&\quad 6H^3\mathbb{E}_{s,a\sim d_t}\min\{\Gamma_t(s,a), 1\} + 2H\,\epsilon_{stat}
\end{aligned}
$$

Now sum over t, and denote $\mathbb{E}_t$ as the conditional expectation conditioned on the history from iteration 0 to $t-1$, we get:

$$
\begin{aligned}
\sum_{t=0}^{T-1}\left[V^{\widetilde{\pi}_t} - V^{\pi_e}\right] &\le 6H^2\sum_{t=0}^{T-1}\mathbb{E}_t\left[\sum_{h=0}^{H-1}\min\{\Gamma_t(s_h^t,a_h^t), 1\}\right] + 2HT\,\epsilon_{stat}\\
&\le 6H^2\sum_{t=0}^{T-1}\left[\rho\sqrt{H}\sqrt{\mathbb{E}_t\sum_{h=0}^{H-1}\min\{\Gamma_t^2(s_h^t,a_h^t), 1\}}\right] + 2HT\,\epsilon_{stat},
\end{aligned}
$$

where in the last inequality we use $\mathbb{E}[a^\top b] \le \sqrt{\mathbb{E}[\|a\|_2^2]\mathbb{E}[\|b\|_2^2]}$.

Recall that $\widetilde{\pi}_t$ are random quantities, add expectation on both sides of the above inequality, and consider the case where E holds and $\bar{E}$ holds, we have:

$$
\begin{aligned}
\mathbb{E}\left[\sum_{t=0}^{T-1}\left(V^{\widetilde{\pi}_t} - V^{\pi_e}\right)\right] &\le 6H^{2.5}\mathbb{E}\left[\sum_{t=0}^{T-1}\sqrt{\mathbb{E}_t\sum_{h=0}^{H-1}\min\{\Gamma_t^2(s_h^t,a_h^t), 1\}}\right] + 2HT\,\epsilon_{stat} + \mathbb{P}(\bar{E})TH\\
&\le 6H^{2.5}\left[\rho\sqrt{T}\sqrt{\mathbb{E}\left[\sum_{t=0}^{T-1}\sum_{h=0}^{H-1}\min\{\Gamma_t^2(s_h^t,a_h^t), 1\}\right]}\right] + 2HT\,\epsilon_{stat} + 2\,\delta TH,
\end{aligned}
$$

where in the last inequality, we use $\mathbb{E}[a^\top b] \le \sqrt{\mathbb{E}[\|a\|_2^2]\mathbb{E}[\|b\|_2^2]}$. This implies that that:

$$
\mathbb{E}\left[\min_t V^{\widetilde{\pi}_t} - V^{\pi_e}\right] \le \frac{6H^{2.5}\rho}{\sqrt{T}}\sqrt{\max_{\text{Alg}}\mathbb{E}_{\text{Alg}}\left[\sum_{t=0}^{T-1}\sum_{h=0}^{H-1}\min\{\Gamma_t^2(s_h^t,a_h^t), 1\}\right]} + 2H\,\epsilon_{stats} + 2H\delta.
$$

Set $\delta = 1/(HT)$, we get:

$$
\mathbb{E}\left[V^{\star} - V^{\pi_e}\right] \le \frac{6H^{2.5}\rho}{\sqrt{T}}\sqrt{\max_{\text{Alg}}\mathbb{E}_{\text{Alg}}\left[\sum_{t=0}^{T-1}\sum_{h=0}^{H-1}\min\{\Gamma_t^2(s_h^t,a_h^t), 1\}\right]} + 2H\sqrt{\frac{\ln(T^3HjFj)}{N}} + \frac{2}{T}
$$

where Alg is any adaptive mapping that maps from history from $t=0$ to the end of the $t-1$ iteration to to some policy $\widetilde{\pi}_t$. This concludes the proof. □

Below we discuss special cases.

A.1 Discrete MDPs

Proposition 10 (Discrete MDP Bonus)**.** *With* $\in (0,1)$. *With probability at least* $1-$ *, for all* $t \in \mathbb{N}$*, we have:*

$$
\left\|\widehat{P}_t(\cdot|s,a) - P^\star(\cdot|s,a)\right\|_1 \le \min\left\{\sqrt{\frac{S\ln(t^2SA/)}{N_t(s,a)}}, 2\right\}.
$$

Proof. The proof simply uses the concentration result for $\widehat{P}_t$ under the ℓ_1 norm. For a fixed t and s,a pair, using Lemma 6.2 in [2], we have that with probability at least $1-$,

$$
\left\|\widehat{P}_t(\cdot|s,a) - P^\star(\cdot|s,a)\right\|_1 \le \sqrt{\frac{S\ln(1/)}{N_t(s,a)}}.
$$

Applying union bound over all iterations and all s, a pairs, we conclude the proof. □

What left is to bound the information gain I for the tabular case. For this, we can simply use the Proposition 13 that we develop in the next section for KNR. This is because in KNR, when we set the feature mapping $(s,a) \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$ to be a one-hot vector with zero everywhere except one in the entry corresponding to (s,a) pair, the information gain in KNR is reduced to the information gain in the tabular model.

Proposition 11 (Information Gain in discrete MDPs)**.** *We have:*

$$I_T = O\left(HS^2A \ln(TSA/\delta)\ln(1+TH)\right).$$

Proof. Using Lemma B.6 in [25], we have:

$$\sum_{t=0}^{T-1} \min\left\{\sum_{h=0}^{H-1} \frac{1}{N_t(s_h^t, a_h^t)}, 1\right\} \le 2SA\ln(1+TH).$$

Now using the definition of information gain, we have:

$$I_T = \sum_{t=0}^{T-1}\sum_{h=0}^{H-1} \min\left\{\beta_t^2(s_h^t,a_h^t), 1\right\} \le S\ln(T^2SA/\delta)H\sum_{t=0}^{T-1}\min\left\{\sum_{h=0}^{H-1}\frac{1}{N_t(s_h^t,a_h^t)}, 1\right\}$$
$$\le 2HS^2A\ln(T^2SA/\delta)\ln(1+TH)$$

This concludes the proof. □

A.2 KNRs

Recall the KNR setting from Example 2. The following proposition shows that the bonus designed in Example 2 is valid.

Proposition 12 (KNR Bonus)**.** *Fix* $\delta \in (0,1)$*. With probability at least* $1-\delta$*, for all* $t \in \mathbb{N}$*, we have:*

$$\left\|\widehat{P}_t(\cdot|s,a) - P^*(\cdot|s,a)\right\|_1 \le \min\left\{\frac{\beta_t}{\sqrt{n}}\|\phi(s,a)\|_{\Lambda_t^{-1}}, 2\right\}, \forall s,a,$$

where $\beta_t = \sqrt{2\log(1/\delta) + 8\log(t)(d_s\ln(5) + 2\ln(t^2/\delta) + \ln(4) + \ln(\det(\Lambda_t)/\det(\lambda I)))}$.

Proof. The proof directly follows the confidence ball construction and proof from [25]. Specifically, from Lemma B.5 in [25], we have that with probability at least $1-\delta$, for all t:

$$\left\|\left(\widehat{W}_t - W^*\right)(\Lambda_t)^{1/2}\right\|_2^2 \le \beta_t^2.$$

Thus, with Lemma 19, we have:

$$\left\|\widehat{P}_t(\cdot|s,a) - P^*(\cdot|s,a)\right\|_1 \le \frac{1}{\sqrt{n}}\left\|(\widehat{W}_t - W^*)\phi(s,a)\right\|_2 \le \left\|(\widehat{W}_t - W^*)(\Lambda_t)^{1/2}\right\|\|\phi(s,a)\|_{\Lambda_t^{-1}} \le \frac{\beta_t}{\sqrt{n}}\|\phi(s,a)\|_{\Lambda_t^{-1}}.$$

This concludes the proof. □

The following proposition bounds the information gain quantity.

Proposition 13 (Information Gain on KNRs)**.** *For simplicity, let us assume* $\phi : \mathcal{S}\times\mathcal{A} \mapsto \mathbb{R}^d$*, i.e.,* $\phi(s,a)$ *is a d-dim feature vector. In this case, we will have:*

$$I_T = O\left(H\left(d\ln(T^2/\delta) + dd_s + d^2\ln\left(1 + kW^*k_2^2TH/\delta^2\right)\right)\ln\left(1 + kW^*k_2^2TH/\delta^2\right)\right).$$

Proof. From the previous proposition, we know that $\beta_t^2(s,a) = (\beta_t^2/\delta^2)\|\phi(s,a)\|^2_{\Lambda_t^{-1}}$. Setting $\lambda = \delta^2/kW^*k_2^2$, we will have $\beta_t^2/\delta^2 \ge 1$, which means that $\min\{\beta_t^2(s,a), 1\} \le (\beta_t^2/\delta^2)\min\left\{\|\phi(s,a)\|^2_{\Lambda_t^{-1}}, 1\right\}$.

Note that $_t$ is non-decreasing with respect to t, so $_t$ $_T$ for T t, where

$$_T = \sqrt{2 \ ^2 + 8 \ ^2(d_s \ln(5) + 2\ln(T^2=) + \ln(4) + d\ln(1 + THkW^?k_2^2= \ ^2))}$$

Also we have $\sum_{t=0}^{T-1}\sum_{h=0}^{H-1}\min\left\{k \ (s_h^t; a_h^t)k^2 \ _{t \ 1}; 1\right\}$ $H\sum_{t=0}^{T-1}\min\left\{\sum_{h=0}^{H-1} k \ (s_h^t; a_h^t)k^2 \ _{t \ 1}; 1\right\}$, since $\min fa_1; b_1g + \min fa_2; b_2g$ $\min fa_1 + a_2; b_1 + b_2g$. Now call Lemma B.6 in [25], we have:

$$\sum_{t=0}^{T-1}\min\left\{\sum_{h=0}^{H-1} k \ (s_h^t; a_h^t)k^2 \ _{t \ 1}; 1\right\} \quad 2\ln\left(\det(\ _T)=\det(\ I)\right) = 2d\ln\left(1 + THkW^?k_2^2= \ ^2\right): \tag{4}$$

Finally recall the definition of I_T, we have:

$$\begin{aligned} I_T &= \sum_{t=0}^{T-1}\sum_{h=0}^{H-1}\min\left\{ \ _t^2(s_h^t; a_h^t); 1\right\} \quad \frac{2}{2}\sum_{t=0}^{T-1}\sum_{h=0}^{H-1}\min\left\{k \ (s_h^t; a_h^t)k^2 \ _{t \ 1}; 1\right\} \quad \frac{2}{2}2Hd\ln(1 + kW^?k_2^2TH= \ ^2) \\ &\quad 2Hd\left(2 + 8\left(d_s\ln(5) + 2\ln(T^2=) + \ln(4) + d\ln\left(1 + kW^?k_2^2TH= \ ^2\right)\right)\right)\ln\left(1 + kW^?k_2^2TH= \ ^2\right) \\ &= H\left(4d + 32dd_s + 32d\ln(T^2=) + 32d + 2d^2\ln\left(1 + kW^?k_2^2TH= \ ^2\right)\right)\ln\left(1 + kW^?k_2^2TH= \ ^2\right); \end{aligned}$$

which concludes the proof. □

Extension to Infinite Dimensional RKHS When $: S \ A \ 7! \ H$ where H is some infinite dimensional RKHS, we can bound our regret using the following intrinsic dimension:

$$\widetilde{d} = \max_{\{\{s_h^t; a_h^t\}_{h=0}^{H \ 1}\}_{t=0}^{T \ 1}} \ln\left(I + \frac{1}{}\sum_{t=0}^{T-1}\sum_{h=0}^{H-1} \ (s_h^t; a_h^t) \ (s_h^t; a_h^t)^\top\right):$$

In this case, recall Proposition 12, we have:

$$\begin{aligned} &_t \quad _T \quad \sqrt{2 \ kW^?k_2^2 + 8 \ ^2(d_s\ln(5) + 2\ln(t^2=) + \ln(4) + \ln\left(\det(\ _T)=\det(\ I)\right))} \\ &\sqrt{2 \ kW^?k_2^2 + 8 \ ^2\left(d_s\ln(5) + 2\ln(t^2=) + \ln(4) + \widetilde{d}\right)}: \end{aligned}$$

Also recall Eq. (4), we have:

$$\sum_{t=0}^{T-1}\min\left\{\sum_{h=0}^{H-1} k \ (s_h^t; a_h^t)k^2 \ _{t \ 1}; 1\right\} \quad 2\ln\left(\det(\ _T)=\det(\ I)\right) \quad 2\widetilde{d}:$$

Combine the above two, following similar derivation we had for finite dimensional setting, we have:

$$I_T = \widetilde{O}\left(H\widetilde{d}^2 + H\widetilde{d}d_s\right):$$

A.3 General Function Class G with Bounded Eluder dimension

Proposition 14. *Fix* $2 \ (0; 1)$. *Consider a general function class* G *where* G *is discrete, and* $\sup_{g\in\mathcal{G};s;a} kg(s; a)k_2 \quad G$. *At iteration* t, *denote* $\widehat{g}_t \ 2 \ \operatorname{argmin}_{g\in\mathcal{G}}\sum_{i=0}^{t-1}\sum_{h=0}^{H-1} kg(s_h^i; a_h^i)$ $s_{h+1}^i k_2^2$, *and denote a version space* G_t *as:*

$$G_t = \left\{g \ 2 \ G : \sum_{i=0}^{t-1}\sum_{h=0}^{H-1}\left\|g(s_h^i; a_h^i) \quad \widehat{g}_t(s_h^i; a_h^i)\right\|_2^2 \quad c_t\right\}; \ \textit{with} \ c_t = 2 \ ^2G^2\ln(2t^2jGj=):$$

The with probability at least $1 \quad$, *we have that for all* t, *and all* $s; a$:

$$\left\|\widehat{P}_t(\ js; a) \quad P^?(\ js; a)\right\|_1 \quad \min\left\{\frac{1}{}\max_{g_1\in\mathcal{G}_t; g_2\in\mathcal{G}_t} kg_1(s; a) \quad g_2(s; a)k_2 ; 2\right\}:$$

Proof. Consider a fixed function $g \in G$. Let us denote $z_h^t = \left\|g(s_h^t; a_h^t) - s_{h+1}^t\right\|_2^2 - \left\|g^?(s_h^t; a_h^t) - s_{h+1}^t\right\|_2^2$. We have:

$$\begin{aligned} z_h^t &= \left(g(s_h^t; a_h^t) - g^?(s_h^t; a_h^t)\right)^\top \left(g(s_h^t; a_h^t) + g^?(s_h^t; a_h^t) - 2g^?(s_h^t; a_h^t) - 2\epsilon_h^t\right) \\ &= \left\|g(s_h^t; a_h^t) - g^?(s_h^t; a_h^t)\right\|_2^2 - 2(g(s_h^t; a_h^t) - g^?(s_h^t; a_h^t))^\top \epsilon_h^t. \end{aligned}$$

Since $\epsilon_h^t \sim N(0; \sigma^2 I)$, we must have:

$$2(g(s_h^t; a_h^t) - g^?(s_h^t; a_h^t))^\top \epsilon_h^t \sim N\left(0; 4\sigma^2 \left\|g(s_h^t; a_h^t) - g^?(s_h^t; a_h^t)\right\|_2^2\right)$$

Since $\sup_{g;s;a} \|g(s; a)\|_2 \le G$, then $2(g(s_h^t; a_h^t) - g^?(s_h^t; a_h^t))^\top \epsilon_h^t$ is a $2\sigma G$ sub-Gaussian random variable.

Call Lemma 3 in [52], we have that with probability at least $1 - \delta$:

$$\sum_t \sum_h \left\|g(s_h^t; a_h^t) - s_{h+1}^t\right\|_2^2 - \sum_t \sum_h \left\|g^?(s_h^t; a_h^t) - s_{h+1}^t\right\|_2^2 + 2\sum_t \sum_h \left\|g(s_h^t; a_h^t) - g^?(s_h^t; a_h^t)\right\|_2^2 \le 4\sigma^2 G^2 \ln(1/\delta).$$

Note that the above can also be derived directly using Azuma-Bernstein's inequality and the property of square loss. With a union bound over all $g \in G$, we have that with probability at least $1 - \delta$, for all $g \in G$.

$$\sum_t \sum_h \left\|g(s_h^t; a_h^t) - s_{h+1}^t\right\|_2^2 - \sum_t \sum_h \left\|g^?(s_h^t; a_h^t) - s_{h+1}^t\right\|_2^2 + 2\sum_t \sum_h \left\|g(s_h^t; a_h^t) - g^?(s_h^t; a_h^t)\right\|_2^2 \le 4\sigma^2 G^2 \ln(|G|/\delta).$$

Set $g = \widehat{g}_t$, and use the fact that g_t is the minimizer of $\sum_t \sum_h \|g(s_h^t; a_h^t) - s_{h+1}^t\|_2^2$, we must have:

$$\sum_t \sum_h \left\|\widehat{g}_t(s_h^t; a_h^t) - g^?(s_h^t; a_h^t)\right\|_2^2 \le 2\sigma^2 G^2 \ln(2t^2|G|/\delta).$$

Namely we prove that our version space G_t contains $g^?$ for all t. Thus, we have:

$$\left\|\widehat{P}_t(\cdot|s; a) - P^?(\cdot|s; a)\right\|_1 \le \frac{1}{n}\|\widehat{g}_t(s; a) - g^?(s; a)\|_2 \le \frac{1}{n} \sup_{g_1 \in \mathcal{G}_t; g_2 \in \mathcal{G}_t} \|g_1(s; a) - g_2(s; a)\|_2;$$

where the last inequality holds since both $g^?$ and $\widehat{g}_t$ belong to the version G_t.

□

Now we bound the information gain I_T below. The proof mainly follows from the proof in [42].

Lemma 15 (Lemma 1 in [42])**.** *Denote $\epsilon_t = 2\sigma^2 G^2 \ln(t^2|G|/\delta)$. Let us denote the uncertainty measure $w_{t;h} = \sup_{f_1; f_2 \in \mathcal{G}_t} \|f_1(s_h^t; a_h^t) - f_2(s_h^t; a_h^t)\|_2$ (note that $w_{t;h}$ is non-negative). We have:*

$$\sum_{i=0}^{t-1} \sum_{h=0}^{H-1} \mathbf{1}\{w_{t;h}^2 > \epsilon\} \left(\frac{4\epsilon_t}{\epsilon} + H\right) \le d_E(\epsilon).$$

Proposition 16 (Bounding I_T)**.** *Denote $d = d_E(1/TH)$. We have*

$$I_T \le \left(1/\sigma^2 + HdG^2/\sigma^2 + 8G^2 \ln(T^2|G|/\delta)d \ln(TH)\right).$$

Proof. Note that the uncertainty measures $w_{t;h}$ are non-negative. Let us reorder the sequence and denote the ordered one as $w_1 \ge w_2 \ge w_3 \ge \cdots \ge w_{TH-H}$. For notational simplicity, denote $M = TH - H$ We have:

$$\sum_{i=0}^{T-1} \sum_{h=0}^{H-1} w_{t;h}^2 = \sum_{i=0}^{M-1} w_i^2 \le 1 + \sum_i w_i^2 \mathbf{1}\{w_i^2 \ge \frac{1}{M}\epsilon\};$$

where the last inequality comes from the fact that $\sum_i w_i^2 \mathbf{1}\{w_i^2 < 1/M\epsilon\} \le M \frac{1}{M} = 1$. Consider any w_t where $w_t^2 \ge 1/M$. In this case, we know that $w_1^2 \ge w_2^2 \ge \cdots \ge w_t^2 \ge 1/M$. This means that:

$$t \ge \sum_i \sum_h \mathbf{1}\{w_{t;h}^2 > w_t^2\epsilon\} \left(\frac{4\epsilon_T}{w_t^2} + H\right) \le d_E(\epsilon w_t) \le \left(\frac{4\epsilon_T}{w_t^2} + H\right) d_E(1/M);$$

where the second inequality uses the lemma above, and the last inequality uses the fact that $d_E(\ )$ is non-decreasing when $\ $ gets smaller. Denote $d = d_E(1=M)$. The above inequality indicates that $w_t^2 \quad \frac{4 \ _T d}{t - Hd}$. This means that for any $w_t^2 \quad 1=M$, we must have $w_t^2 \quad 4 \ _T d=(t \quad Hd)$. Thus, we have:

$$\sum_{i=0}^{T-1}\sum_{h=0}^{H-1} w_{t;h}^2 \quad 1 + HdG^2 + \sum_{=Hd+1}^{M} w^2 \mathbf{1}fw^2 \quad 1=Mg \quad 1 + HdG^2 + 4 \ _T d\ln(M)$$
$$= 1 + HdG^2 + 4 \ _T d\ln(TH):$$

Finally, recall the definition of I_T, we have:

$$\sum_{t=0}^{T-1}\sum_{h=0}^{H-1} \min f \ _t^2(s_h^t; a_h^t); 1g \quad \sum_{t=0}^{T-1}\sum_{h=0}^{H-1} \ _t^2(s_h^t; a_h^t) \quad \frac{1}{2}\sum_{t=0}^{T-1}\sum_{h=0}^{H-1} w_{t;h}^2 \quad \frac{1}{2}\left(1 + HdG^2 + 4 \ _T d\ln(TH)\right):$$

This concludes the proof. □

A.4 Proof of Theorem 7

This section provides the proof of Theorem 7.

First we present the reduction from a bandit optimization problem to ILFO.

Consider a Multi-armed bandit (MAB) problem with A many actions $fa_i g_{i=1}^A$. Each action's ground truth reward r_i is sampled from a Gaussian with mean $\ _i$ and variance 1. Without loss of generality, assume a_1 is the optimal arm, i.e., $\ _1 \quad \ _i \ 8 \ i \ 6= 1$. We convert this MAB instance into an MDP. Specifically, set $H = 2$. Suppose we have a fixed initial state s_0 which has A many actions. For the one step transition, we have $P(\ js_0; a_i) = N(\ _i; 1)$, i.e., $g^*(s_0; a_i) = \ _i$. Here we denote the optimal expert policy $\ ^e$ as $\ ^e(s_0) = a_1$, i.e., expert policy picks the optimal arm in the MAB instance. Hence, when executing $\ ^e$, we note that the state s_1 generated from $\ ^e$ is simply the stochastic reward of a_1 in the original MAB instance. Assume that we have observed infinitely many such s_1 from the expert policy $\ ^e$, i.e., we have infinitely many samples of expert state data, i.e., $N \ ! \ 1$. Note, however, we do not have the actions taken by the expert (since this is the ILFO setting). This expert data is equivalent to revealing the optimal arm's mean reward $\ _1$ to the MAB learner a priori. Hence solving the ILFO problem on this MDP is no easier than solving the original MAB instance with additional information which is that optimal arm's mean reward is $\ _1$ (but the best arm's identity is unknown).

Below we show the lower bound for solving the MAB problem where the optimal arm's mean is known.

Theorem 17. *Consider best arm identification of Gaussian MAB with the additional information that the optimal arm's mean reward is $\ $. For any algorithm, there exists a MAB instance with number of arms $A \quad 2$, such that the expected cumulative regret is still $\ (\sqrt{AT})$, i.e., the additional information does not help improving the worst-case regret bound to solve the MAB instance.*

Proof of Theorem 17. Below, we will construct A many MAB instances where each instance has A many arms and each arm has a Gaussian reward distribution with the fixed variance $\ ^2$. Each of the A instances has the maximum mean reward equal to $\ $, i.e., all these A instances have the same maximum arm mean reward. Consider any algorithm Alg that maps $\ $ together with the history of the interactions $H_t = fa_0; r_0; a_1; r_1; : : : ; a_{t-1}; r_{t-1}g$ to a distribution over A actions. We will show for any such algorithm alg that knows $\ $, with constant probability, there must exist a MAB instance from the A many MAB instances, such that Alg suffers at least $\ (\sqrt{AT})$ regret where T is the number of iterations.

Now we construct the A instances as follows. Consider the i-th instance ($i = 1; : : : ; A$). For arm j in the i-th instance, we define its mean as $\ _j^i = \mathbf{1}fi = jg \ $. Namely, for MAB instance i, its arms have mean reward zero everywhere except that the i-th arm has reward mean $\ $. Note that all these MAB instances have the same maximum mean reward, i.e., $\ $. Hence, we cannot distinguish them by just revealing $\ $ to the learner.

We will construct an additional MAB instance (we name it as 0-th MAB instance) whose arms have reward mean zero. Note that this MAB instance has maximum mean reward 0 which is different from

the previous A MAB instances that we constructed. However, we will only look at the regret of Alg on the previously constructed A MAB instances. I.e., we do not care about the regret of Alg(; H_t) on the 0-th MAB instance.

Let us denote P_i (for $i = 0; : : : ; A$) as the distribution of the outcomes of algorithm Alg(; H_t) interacting with MAB instance i for n iterations, and $E_j[N_i(T)]$ as the expected number of times arm i is pulled by Alg(; H_t) in MAB instance j. Consider MAB instance i with i 1:

$$E_i[N_i(T)] \quad E_0[N_i(T)] \quad T\,kP_i \quad P_0k_1 \quad T\sqrt{KL(P_0;P_i)} \quad T\sqrt{\ ^2E_0[N_i(T)]};$$

where the last step uses the fact that we are running the same algorithm Alg(; H_t) on both instance 0 and instance i (i.e., same policy for generating actions), and thus, $KL(P_0;P_i) = \sum_{j=1}^{A} E_0[N_j(T)]KL\,(q_0(j);q_i(j))$ (Lemma 15.1 in [35]), where $q_i(j)$ is the reward distribution of arm j at instance i. Also recall that for instance 0 and instance i, their rewards only differ at arm i.

This implies that:

$$E_i[N_i(T)] \quad E_0[N_i(T)] + T\sqrt{\ ^2E_0[N_i(T)]}:$$

Sum over $i = 1; : : : ; A$ on both sides, we have:

$$\sum_{i=1}^{A} E_i[N_i(T)] \quad T + T\sum_{i=1}^{A}\sqrt{\ ^2E_0[N_i(T)]} \quad T + T\sqrt{A}\sqrt{\sum_{i=1}^{A}\ ^2E_0[N_i(T)]}$$
$$T + T\sqrt{A}\sqrt{\ ^2T}$$

Now let us calculate the regret of Alg(; H_t) on i-th instance, we have:

$$R_i = T \qquad E_i[N_i(T)] \ :$$

Sum over $i = 1; : : : ; A$, we have:

$$\sum_{i=1}^{A} R_i = \quad \left(AT \quad \sum_{i=1}^{A} E_i[N_i(T)]\right) \qquad \left(AT \quad T \quad T\sqrt{A\ ^2T}\right)$$

Set $= c\sqrt{A=T}$ for some c that we will specify later, we get:

$$\sum_{i=1}^{A} R_i \quad c\sqrt{AT}\,(AT \quad T \quad cAT):$$

Set $c = 1=4$, we get:

$$\sum_{i=1}^{A} R_i \quad c\sqrt{AT}\,(AT \quad T \quad cAT) \quad \frac{1}{4}\sqrt{AT}\,(A \quad 1 \quad A=4) = \frac{1}{4}\sqrt{AT}\,(3A=4 \quad 1) \quad \frac{1}{4}\sqrt{AT}\,(A=4);$$

assuming A 2.

Thus there must exist i 2 $f1; : : : ; Ag$, such that:

$$R_i \quad \frac{1}{16}\sqrt{AT}:$$

Note that the above construction considered any algorithm Alg(; H_t) that maps and history to action distributions. Thus it concludes the proof. □

The hardness result in Theorem 17 and the reduction from MAB to ILFO together implies the lower bound for ILFO in Theorem 7, namely solving ILFO with cumulative regret smaller then $O(\sqrt{AT})$ will contradict the MAB lower bound in Theorem 17.

B Auxiliary Lemmas

Lemma 18 (Simulation Lemma)**.** *Consider any two functions* $f : S \quad A \; 7! \; [0; 1]$ *and* $\widehat{f} : S \quad A \; 7! \; [0; 1]$*, any two transitions* P *and* $\widehat{P}$*, and any policy* $\quad : S \; 7! \quad (A)$*. We have:*

$$V_{P;f} \quad V_{\widehat{P};\widehat{f}} = \sum_{h=0}^{H-1} \mathrm{E}_{s;a\sim d_P} \left[f(s;a) \quad \widehat{f}(s;a) + \mathrm{E}_{s'\sim P(\cdot|s;a)} V_{\widehat{P};\widehat{f};h}(s') \quad \mathrm{E}_{s'\sim \widehat{P}(\cdot|s;a)} V_{\widehat{P};\widehat{f};h}(s') \right]$$
$$\sum_{h=0}^{H-1} \mathrm{E}_{s;a\sim d_P} \left[f(s;a) \quad \widehat{f}(s;a) + kV_{\widehat{P};\widehat{f};h}k_\infty kP(\,js;a) \quad \widehat{P}(\,js;a)k_1 \right] :$$

where $V_{P;f;h}$ *denotes the value function at time step* h*, under* $\quad ;P;f$*.*

Such simulation lemma is standard in model-based RL literature and can be found, for instance, in the proof of Lemma 10 from [58].

Lemma 19. *Consider two Gaussian distribution* $P_1 := N(\ _1;\ ^2I)$ *and* $P_2 := N(\ _2;\ ^2I)$*. We have:*

$$kP_1 \quad P_2k_1 \quad \frac{1}{} k \ _1 \quad \ _2k_2 :$$

The above lemma can be proved by Pinsker's inequality and the closed-form of the KL divergence between P_1 and P_2.

C Implementation Details

C.1 Environment Setup and Benchmarks

This section sketches the details of how we setup the environments. We utilize the standard environment horizon of 500; 50; 200 for `Cartpole-v1`, `Reacher-v2`, `Cartpole-v0`. For `Swimmer-v2`, `Hopper-v2` and `Walker2d-v2`, we work with the environment horizon set to 400 [33, 39, 38, 48, 32]. Furthermore, for `Hopper-v2`, `Walker2d-v2`, we add the velocity of the center of mass to the state parameterization [48, 38, 32]. As noted in the main text, the expert policy is trained using NPG/TRPO [26, 54] until it hits a value of (approximately) 460; 10; 38; 3000; 2000; 170 for `Cartpole-v1`, `Reacher-v2`, `Swimmer-v2`, `Hopper-v2`, `Walker2d-v2`, `Cartpole-v0` respectively. Furthermore, for `Walker2d-v2` we utilized pairs of states $(s; s')$ for defining the feature representation used for parameterizing the discriminator. All the results presented in the experiments section are averaged over five seeds. Furthermore, in terms of baselines, we compare `MobILE` to BC, BC-O, ILPO, GAIL and GAIFO. Note that BC/GAIL has access to expert actions whereas our algorithm does not have access to the expert actions. We report the average of the best performance offered by BC/BC-O when run with five seeds, even if this occurs at different epochs for each of the runs - this gives an upper hand to BC/BC-O. Moreover, note that for BC, we run the supervised learning algorithm for 500 passes. Furthermore, we run BC-O/GAIL with same number of online samples as `MobILE` in order to present our results. Furthermore, we used 2 CPUs with 16-32 GB of RAM usage to perform all our benchmarking runs implemented in Pytorch. Finally, our codebase utilizes Open-AI's implementation of TRPO [15] for environments with discrete actions, and the MJRL repository [47] for working with continuous action environments. With regards to results in the main paper, our bar graph presenting normalized results was obtained by dividing every algorithm's performance (mean/standard deviation) by the expert mean; for `Reacher-v2` because the rewards themselves are negative, we first added a constant offset to make all the algorithm's performance to become positive, then, divided by the mean of expert policy.

C.2 Practical Implementation of `MobILE`

We will begin with presenting the implementation details of `MobILE` (refer to Algorithm 2):

Algorithm 2 `MobILE`: Model-based Imitation Learning and Exploring for ILFO (used in practical implementation)

Require: Expert Dataset D_e, Access to dynamics of the true environment i.e. $P^?$.
Initialize Policy $\quad_0$, Discriminator w_0, Replay Buffer of pre-determined size D, Dynamics Model $\widehat{P}_{-1}$, Bonus b_{-1}.
for $t = 0; \quad ; T \quad 1$ **do**
  Online Interaction: Execute $\quad_t$ in true environment $P^?$ to get samples S_t.
  Update replay buffer: $D =$ Replay-Buffer-Update$(D; S_t)$ (refer to section C.2.2).
  Update dynamics model: Obtain $\widehat{P}_t$ by starting at $\widehat{P}_{t-1}$ and update using replay buffer D (refer to section C.2.1).
  Bonus Update: Update bonus $b_t : S \quad A \; ! \; \mathbb{R}^+$ using replay buffer D (refer to section C.2.3).
  Discriminator Update: Update discriminator as $w_t \quad \arg\max_w L(w; \quad_t; \widehat{P}_t; b_t; D_e)$ (refer to section C.2.4).
  Policy Update: Perform incremental policy update through approximate minimization of $L(\,)$,
    i.e.: $\quad_t \quad \arg\min \; L(\quad ; w_t; \widehat{P}_t; b_t; D_e)$ by running K_{PG} steps of TRPO (refer to section C.2.5).
end for
Return $\quad_T$.

C.2.1 Dynamics Model Training

As detailed in the main paper, we utilize a class of Gaussian Dynamics Models parameterized by an MLP [48], i.e. $\widehat{P}(s; a) := N(h\;(s; a); \; ^2 I)$, where, $h\;(s; a) = s + \quad_s$ MLP $(s_c; a_c)$, where, are MLP's trainable parameters, $s_c = (s \quad _s)= \; _s$, $a_c = (a \quad _a)= \; _a$ with $\; _s; \; _a$ (and $\; _s; \; _a$) being the mean of states, actions (and standard deviation of states and actions) in the replay buffer D. Note that we predict normalized state differences instead of the next state directly.

In practice, we fine tune our estimate of dynamics models based on the new contents of the replay buffer as opposed to re-training the models from scratch, which is computationally more expensive. In particular, we start from the estimate $\widehat{P}_{t-1}$ in the $t \quad 1$ epoch and perform multiple updates gradient updates using the contents of the replay buffer D. We utilize constant stepsize SGD with momentum [60] for updating our dynamics models. Since the distribution of $(s; a; s')$ pairs continually drift as the algorithm progresses (for instance, because we observe a new state), we utilize gradient clipping to ensure our model does not diverge due to the aggressive nature of our updates.

C.2.2 Replay Buffer

Since we perform incremental training of our dynamics model, we utilize a replay buffer of a fixed size rather than training our dynamics model on all previously collected online $(s; a; s')$ samples. Note that the replay buffer could contain data from all prior online interactions should we re-train our dynamics model from scratch at every epoch.

C.2.3 Design of Bonus Function

We utilize an ensemble of two transition dynamics models incrementally learned using the contents of the replay buffer. Specifically, given the models $h\;_1(\,)$ and $h\;_2(\,)$, we compute the discrepancy as: $(s; a) = jjh\;_1(s; a) \quad h\;_2(s; a)jj_2$: Moreover, given a replay buffer D, we compute the maximum discrepancy as $\quad_{\mathcal{D}} = \max_{(s;a;s^0)\sim\mathcal{D}} \; (s; a)$. We then set the bonus as $b(s; a) = \min(1; \; (s; a)= \; _{\mathcal{D}})$, thus ensuring the magnitude of our bonus remains bounded between $[0; \quad]$ roughly.

C.2.4 Discriminator Update

Recall that $f_w(s) = w^\top \; (s)$, where w are the parameters of the discriminator. Given a policy , the update for the parameters w take the following form:

$$\max_{w:||w||_2^2 \leq} \; L(w; \quad ; \widehat{P}; b; D_e) := \mathbb{E}_{(s;a)\sim d_{\widehat{P}}}[f_w(s) \quad b(s; a)] \quad \mathbb{E}_{s\sim\mathcal{D}_e}[f_w(s)]$$

$$\max_{w} L(w; \pi; \widehat{P}; b; D_e) = \mathbb{E}_{(s;a)\sim d_{\widehat{P}}}[f_w(s) \quad b(s;a)] \quad \mathbb{E}_{s\sim\mathcal{D}_e}[f_w(s)] \quad \frac{1}{2}(\|w\|_2^2 \quad);$$

$$=) \quad @_w L(w; \pi; \widehat{P}; b; D_e) = \mathbb{E}_{s\sim d_{\widehat{P}}}[\quad (s)] \quad \mathbb{E}_{s\sim\mathcal{D}_e}[\quad (s)] \quad w \; 2 \; 0;$$

where, $@_w L(w; \pi; \widehat{P}; b; D_e)$ denotes the sub-differential of $L(\quad)$ wrt w. This in particular implies the following:

1. **Exact Update:** $w^* = P_{\mathcal{B}(\quad)}\left(\mathbb{E}_{s\sim d_{\widehat{P}}}[\quad (s)] \quad \mathbb{E}_{s\sim\mathcal{D}_e}[\quad (s)]\right)$, $P_{\cdot}$ is the projection operator, and $B(\quad)$ is the norm ball.
2. **Gradient Ascent Update:** $w_{t+1} = P_{\mathcal{B}(\quad)}\left((1 \quad _w)w_t + \quad _w \left(\mathbb{E}_{s\sim d_{\widehat{P}}}[\quad (s)] \quad \mathbb{E}_{s\sim\mathcal{D}_e}[\quad (s)]\right)\right)$, $_w > 0$ is the step-size.

We found empirically either of the updates to work reasonably well. In the `Swimmer-v2` task, we use the gradient ascent update with $_w = 0{:}67$, and, in the other tasks, we utilize the exact update. Furthermore, we empirically observe the gradient ascent update to yield more stability compared to the exact updates. In the case of `Walker2d-v2`, we found it useful to parameterize the discriminator based on pairs of states $(s; s')$.

C.2.5 Model-Based Policy Update

Once the maximization of the discriminator parameters w is performed, consider the policy optimization problem, i.e.,

$$\min L(\quad; w; \widehat{P}; b; D_e) := \mathbb{E}_{(s;a)\sim d_{\widehat{P}}}[f_w(s) \quad b(s;a)] \quad \mathbb{E}_{s\sim\mathcal{D}_e}[f_w(s)]$$

$$\min L(\quad; w; \widehat{P}; b; D_e) = \mathbb{E}_{(s;a)\sim d_{\widehat{P}}}[f_w(s) \quad b(s;a)]$$

Hence we perform model-based policy optimization under $\widehat{P}$ and cost function $f_w(s) \quad b(s;a)$. In practice, we perform approximate minimization of $L(\quad)$ by incrementally updating the policy using K_{PG}-steps of policy gradient, where, K_{PG} is a tunable hyper-parameter. In our experiments, we find that setting K_{PG} to be around 10 to generally be a reasonable choice (for precise values, refer to Table 2). This paper utilizes TRPO [54] as our choice of policy gradient method; note that this can be replaced by other alternatives including PPO [55], SAC [20] *etc.* Similar to practical implementations of existing policy gradient methods, we implement a reward filter by clipping the IPM reward $f(s)$ by truncating it between $c_{\min}$ and $c_{\max}$ as this leads to stability of the policy gradient updates. Note that the minimization is done with access to $\widehat{P}$, which implies we perform *model-based* planning. Empirically, for purposes of tuning the exploration-imitation parameter , we minimize a surrogate namely: $\mathbb{E}_{(s;a)\sim d_{\widehat{P}}}[(1 \quad) \quad f_w(s) \quad b(s;a)]$ (recall that $b(s;a)$ has a factor of associated with it). This ensures that we can precisely control the magnitude of the bonuses against the IPM costs, which, in our experience is empirically easier to work with.

C.3 Hyper-parameter Details

This section presents an overview of the list of hyper-parameters necessary to implement Algorithm 1 in practice, as described in Algorithm 2. The list of hyper-parameters is precisely listed out in Table 2. The hyper-parameters are broadly categorized into ones corresponding to various components of `MobILE`, namely, (a) environment specifications, (b) dynamics model, (c) ensemble based bonus, (d) IPM parameterization, (e) Policy parameterization, (f) Planning algorithm parameters, (g) Critic parameterization. Note that if there a hyper-parameter that has not been listed, for instance, say, the value of momentum for the ADAM optimizer in the critic, this has been left as is the default value defined in Pytorch.

D Additional Experimental Results

D.1 Modified `Cartpole-v0` environment with noise added to transition dynamics

Parameter	Cartpole-v1	Reacher-v2	Swimmer-v2	Cartpole-v0	Hopper-v2	Walker2d-v2
Environment Specifications						
Horizon H	500	50	400	200	400	400
Expert Performance ( )	460	10	38	181	3000	2000
# online samples per outer loop	2 H	2 H	2 H	2 H	8 H	3 H
Dynamics Model						
Architecture/Non-linearity	MLP(64; 64)/ReLU	MLP(64; 64)/ReLU	MLP(512; 512)/ReLU	MLP(64; 64)/ReLU	MLP(512; 512)/ReLU	MLP(512; 512)/ReLU
Optimizer(LR, Momentum, Batch Size)	SGD(0:005; 0:99; 256)	SGD(0:005; 0:99; 256)	SGD(0:005; 0:99; 256)	SGD(0:005; 0:99; 256)	SGD(0:005; 0:99; 256)	SGD(0:005; 0:99; 256)
# train passes per outer loop	20	100	100	20	50	200
Grad Clipping	2:0	2:0	1:0	2:0	4:0	1:0
Replay Buffer Size	10 H	10 H	10 H	10 H	16 H	15 H
Ensemble based bonus						
# models/bonus range	2/[0; 1]	2/[0; 1]	2/[0; 1]	2/[0; 1]	2/[0; 1]	2/[0; 1]
IPM parameters						
Step size for w update ($_w$)	Exact	Exact	0:33	Exact	Exact	Exact
# RFFs/BW Heuristic	128/0:1 quantile	128 / 0:1 quantile	128 / 0:1 quantile	128 / 0:1 quantile	128 / 0:1 quantile	128 / 0:1 quantile
Policy parameterization						
Architecture/Non-linearity	MLP(64; 64)/TanH	MLP(64; 64)/TanH	MLP(64; 64)/TanH	MLP(32; 32)/TanH	MLP(32; 32)/TanH	MLP(32; 32)/TanH
Policy Constraints	None	None	None	None	log $_{\min}$ = 1:0	log $_{\min}$ = 2:0
Planning Algorithm						
# model samples per TRPO step	2 H	10 H	4 H	4 H	8 H	20 H
# TRPO steps per outer loop (K_{PG})	3	10	20	5	10	15
TRPO Parameters	(50; 0:001; 0:01;	(100; 0:001; 0:01;	(100; 0:001; 0:01;	(100; 0:001; 0:01;	(10; 0:0001; 0:025;	(10; 0:0001; 0:025;
(CG iters, dampening, kl, gae ,)	0:97; 0:995)	0:97; 0:995)	0:97; 0:995)	0:97; 0:995)	0:97; 0:995)	0:97; 0:995)
Critic parameterization						
Architecture/Non-linearity	MLP(128; 128)/ReLU	MLP(128; 128)/ReLU	MLP(128; 128)/ReLU	MLP(32; 32)/ReLU	MLP(128; 128)/ReLU	MLP(128; 128)/ReLU
Optimizer	Adam(0:001; 64; 1e 5; 0)	Adam(0:001; 64; 1e 5; 0)	Adam(0:001; 64; 1e 5; 0)	Adam(0:001; 64; 1e 5; 0)	Adam(0:001; 64; 1e 8; 1e 3)	Adam(0:001; 64; 1e 8; 1e 3)
(LR, Batch Size, , Regularization)						
# train passes per TRPO update	1	1	1	1	2	2

Table 2: List of various Hyper-parameters employed in MobILE's implementation.

We consider a stochastic variant of Cartpole-v0, wherein, we add additive Gaussian noise of variance unknown to the learner in order to make the transition dynamics of the environment to be stochastic. Specifically, we train an expert of value 170 in Cartpole-v0 with stochastic dynamics using TRPO. Now, using 20 trajectories drawn from this expert, we wish to consider solving the ILFO problem using MobILE as well as other baselines including BC, BC-O, ILPO, GAIL and GAIFO. Figure 4 presents the result of this comparison. Note that MobILE compares favorably against other baseline methods - in particular, BC tends suffer in environments like Cartpole-v0 with stochastic dynamics because of increased generalization error of the supervised learning algorithm used for learning a policy. Our algorithm is competitive with both BC-O, GAIL, GAIFO and ILPO. Note that BC-O tends to outperform BC both in Cartpole-v1 and in Cartpole-v0 (with stochastic dynamics).

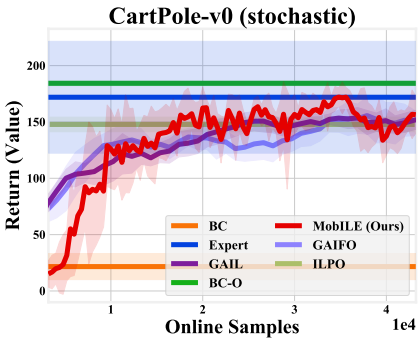

Figure 4: Learning curves for Cartpole-v0 with stochastic dynamics with 20 expert trajectories comparing MobILE with BC, BC-O, GAIL, GAIFO and ILPO.

D.2 Swimmer Learning Curves

We supplement the learning curves for Swimmer-v2 (with 40 expert trajectories) with the learning curves for Swimmer-v2 with 10 expert trajectories in figure 5. As can be seen, MobILE outperforms baseline algorithms such as BC, BC-O, ILPO, GAIL and GAIFO in Swimmer-v2 with both 40 and 10 expert trajectories. The caveat is that for 10 expert trajectories, all algorithms tend to show a lot more variance in their behavior and this reduces as we move to the 40 expert trajectory case.

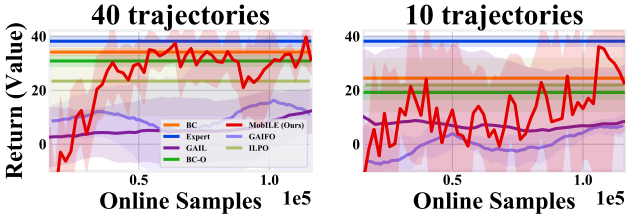

Figure 5: Learning curves for Swimmer-v2 with 40 (left) and 10 (right) expert trajectories comparing MobILE with BC, BC-O, ILPO, GAIL and GAIFO. MobILE continues to perform well relative to all other benchmarks with both 10 and 40 expert trajectories. The variance of the algorithm as well as the benchmarks is notably higher with lesser number of expert trajectories.

D.3 Additional Results

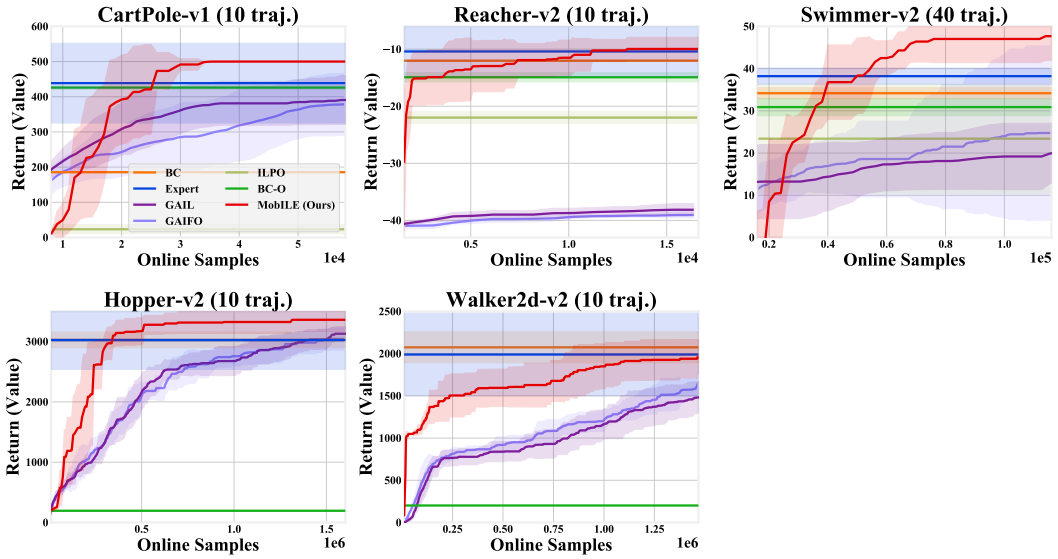

Figure 6: Learning curves tracking the running maximum averaged across seeds comparing `MobILE` against BC, BC-O, ILPO, GAIL and GAIFO. `MobILE` tends to reach expert performance consistently and in a more sample efficient manner.

In this section, we give another view of our results for `MobILE` compared against the baselines (BC/BC-O/ILPO/GAIL/GAIFO) by tracking the running maximum of each policy's value averaged across seeds. Specifically, for every iteration t, we plot the best policy performance obtained by the algorithm so far averaged across seeds (note that this quantity is monotonic, since the best policy obtained so far can never be worse at a later point of time when running the algorithm). For BC/BC-O/ILPO, we present a simplified view by picking the best policy obtained through the course of running the algorithm and averaging it across seeds (so the curves are flat lines). As figure 6 shows, `MobILE` reliably hits expert performance faster than GAIL and GAIFO while often matching/outperforming ILPO/BC/BC-O.

D.4 Ablation Study on Number of Models used for Strategic Exploration Bonus

In this experiment, we present an ablation study on using more number of models in the ensemble for setting the strategic exploration bonus. Figure 7 suggests that even utilizing two models for purposes of setting the bonus is effective from a practical perspective.

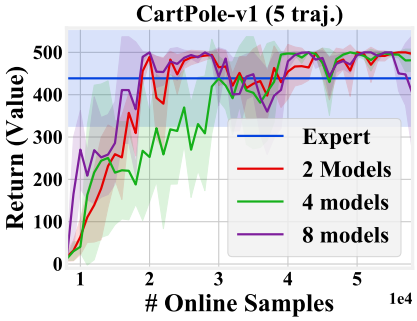

Figure 7: Learning curves for `Cartpole-v1` with varying number of dynamics models for assigning bonuses for strategic exploration.