Provably Efficient Model-based Policy Adaptation

Yuda Song [1] **Aditi Mavalankar** [1] **Wen Sun** [2] **Sicun Gao** [1]

Abstract

The high sample complexity of reinforcement learning challenges its use in practice. A promising approach is to quickly adapt pre-trained policies to new environments. Existing methods for this policy adaptation problem typically rely on domain randomization and meta-learning, by sampling from some distribution of target environments during pre-training, and thus face difficulty on out-of-distribution target environments. We propose new model-based mechanisms that are able to make online adaptation in unseen target environments, by combining ideas from no-regret online learning and adaptive control. We prove that the approach learns policies in the target environment that can recover trajectories from the source environment, and establish the rate of convergence in general settings. We demonstrate the benefits of our approach for policy adaptation in a diverse set of continuous control tasks, achieving the performance of state-of-the-art methods with much lower sample complexity. Our project website, including code, can be found at `https://yudasong.github.io/PADA`.

1. Introduction

Deep Reinforcement Learning (RL) methods typically require a very large number of interactions with environments, making them difficult to be used on practical systems (Tan et al., 2018). A promising direction is to adapt policies trained in one environment to similar but unseen environments, such as from simulation to real robots. Existing approaches for policy adaptation mostly focus on pre-training the policies to be robust to predefined distributions of disturbances in the environment, by increasing the sample diversity during training (Peng et al., 2018; Tobin et al., 2017; Mordatch et al., 2015), or meta-learn policies or models that can be quickly adapted to in-distribution environments (Finn et al., 2017a; Nagabandi et al., 2019a;b; Yu et al., 2018a). A key assumption for these approaches is that the distribution of the target environments is known, and that it can be efficiently sampled during training. On out-of-distribution target environments, these methods typically do not deliver good performance, reflecting common challenges in generalization (Na et al., 2020). If we observe how humans and animals adapt to environment changes, clearly there is an online adaptation process in addition to memorization (Staddon, 2016). We can quickly learn to walk with a slightly injured leg even if we have not experienced the situation. We draw experiences from normal walking and adapt our actions online, based on how their effects differ from what we are familiar with in normal settings. Indeed, this intuition has recently led to practical approaches for policy adaptation. The work in (Christiano et al., 2016) uses a pre-trained policy and model of the training environment, and learns an inverse dynamics model from scratch in the new environment by imitating the behaviors of the pre-trained policy. However, it does not involve mechanisms for actively reducing the divergence between the state trajectories of the two environments, which leads to inefficiency and distribution drifting, and does not fully capture the intuition above. The work in (Zhu et al., 2018) uses Generative Adversarial Imitation Learning (GAIL) (Ho & Ermon, 2016) to imitate the source trajectories in the new environment, by adding the GAIL discriminator to the reward to reduce divergence, but relies on generic policy optimization methods with high sample complexity. In general, these recent approaches show the feasibility of policy adaptation, but are not designed for optimizing sample efficiency. There has been no theoretical analysis of whether policy adaptation methods can converge in general, or their benefits in terms of sample complexity.

In this paper, we propose a new model-based approach for the policy adaptation problem that focuses on efficiency with theoretical justification. In our approach, the agent attempts to predict the effects of its actions based on a model of the training environment, and then adapts the actions to minimize the divergence between the state trajectories in the new (target) environment and in the training (source) environment. This is achieved by iterating between two steps: a

[1]Department of Computer Science and Engineering, University of California, San Diego, La Jolla, USA [2]Department of Computer Science, Cornell University, Ithaca , USA. Correspondence to: Yuda Song <yus167@ucsd.edu>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

modeling step learns the divergence between the source environment and the target environment, and a planning step that uses the divergence model to plan actions to reduce the divergence over time. Under the assumption that the target environment is close to the source environment, the divergence modeling and policy adaption can both be done locally and efficiently. We give the first theoretical analysis of policy adaptation by establishing the rate of convergence of our approaches under general settings. Our methods combine techniques from model-based RL (Wang et al., 2019) and no-regret online learning (Ross et al., 2011). We demonstrate that the approach is empirically efficient in comparison to the state-of-the-art approaches (Christiano et al., 2016; Zhu et al., 2018). The idea of recovering state trajectories from the source environment in the target environment suggests a strong connection between policy adaptation and imitation learning, such as Learning from Observation (LfO) (Torabi et al., 2018; 2019a; Sun et al., 2019b; Yang et al., 2019). A key difference is that in policy adaptation, the connection between the source and target environments and their difference provide both new challenges and opportunities for more efficient learning. By actively modeling the divergence between the source and target environments, the agent can achieve good performance in new environments by only making local changes to the source policies and models. On the other hand, because of the difference in the dynamics and the action spaces, it is not enough to merely imitate the experts (Bain & Sommut, 1999; Ross et al., 2011; Sun et al., 2017). Traditionally, adaptive control theory (Åström, 1983) studies how to adapt to disturbances by stabilizing the error dynamics. Existing work in adaptive control assumes closed-form dynamics and does not apply to the deep RL setting (Nagabandi et al., 2019a). In comparison to domain randomization and meta-learning approaches, our proposed approach does not require sampling of source environments during pre-training, and makes it possible to adapt in out-of-distribution environments. Note that the two approaches are complementary, and we demonstrate in experiments that our methods can be used in conjunction with domain randomization and meta-learning to achieve the best results.

The paper is organized as follows. We review related work in Section 2 and the preliminaries in Section 3. In Section 4, we propose the theoretical version of the adaptation algorithm and prove its rate of convergence. In section 5 we describe the practical implementation using deviation models and practical optimization methods. We show detailed comparison with competing approaches in Section 6.

2. Related Work

Our work connects to existing works on imitation learning, online adaptation, domain randomization and meta-learning, and model-based reinforcement learning.

Imitation Learning. In imitation learning, there is typically no separation between training environments and test environments. Existing imitation learning approaches aim to learn a policy that generates state distributions (Tobin et al., 2017; Torabi et al., 2019b; Sun et al., 2019b; Yang et al., 2019) or state-action distributions (Ho & Ermon, 2016; Fu et al., 2017; Ke et al., 2019; Ghasemipour et al., 2019) that are similar to the ones given by the expert policy. The difference in the policy adaptation setting is that the expert actions do not work in the first place in the new environment, and we need to both model the divergence and find a new policy for the target environment. In light of this difference, the work (Zhu et al., 2018) considers a setting that is the closest to ours (which we will compare with in the experiments). It uses a state-action-imitation (GAIL(Ho & Ermon, 2016)) approach to learn a policy in the target environment, to generate trajectories that are similar to the trajectories of the expert from the source environments. It also relies on using the true reward signals in the target environment to train the policy besides state imitation. In recent work, (Liu et al., 2020) approaches a similar problem by using Wasserstein distance between the states as the reward. It uses adversarial training and model-free policy optimization methods. Our approach is model-based and relies on reduction to Data Aggregation (Ross et al., 2011) for efficiency. The reduction allows us to derive provable convergence guarantee with the rate of convergence. Experimentally we show that our approach is more sample efficient than model-free and minmax-based imitation approaches in general.

Online Adaptation. Online adaptation methods transfer policies by learning a mapping between the source and target domains (Daftry et al., 2016; Tzeng et al., 2015). Such methods have achieved success in vision-based robotics but require extra kernel functions or learning feature spaces. In contrast, we focus on control problems where the policy adaptation is completely autonomous. (Christiano et al., 2016) trains an inverse dynamics model (IDM) which can serve as the target policy by inquiring the source policy online. However, the approach does not focus on optimizing sample efficiency, which is crucial for the use of policy adaptation. In (Yu et al., 2018b), the agent selects from a range of pre-trained policies online, and does not perform further adaptation, and thus experiences problems similar to domain randomization approaches.

Domain Randomization and Meta-Learning. Domain randomization and meta-learning methods are popular ways of transferring pre-trained policies to new environments. These methods rely on the key assumption that the training environments and test environments are sampled from the same predefined distribution. Domain randomization methods train robust agents on diverse samples of target environments (Tobin et al., 2017; Mordatch et al., 2015; Antonova et al., 2017; Chebotar et al., 2019). When the configuration

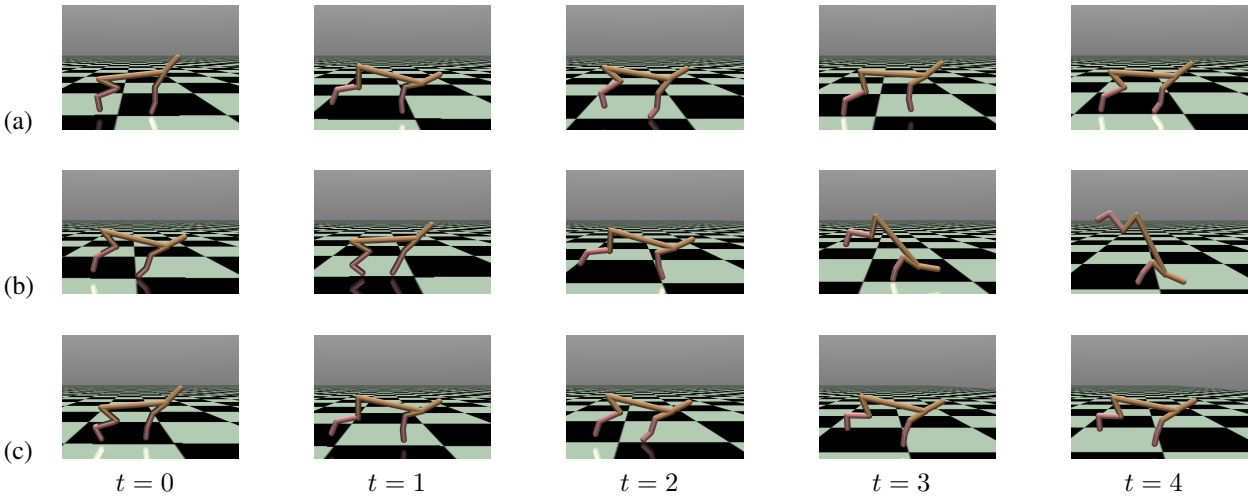


Figure 1. (a) Halfcheetah of mass m using the source policy $\pi^{(s)}$ in the source environment $\mathcal{M}^{(s)}$. (b) Halfcheetah of mass $1.5 \times m$ using the source policy $\pi^{(s)}$ in the target environment $\mathcal{M}^{(t)}$. (c) Halfcheetah of mass $1.5 \times m$ using the learned target policy $\pi^{(t)}$ in $\mathcal{M}^{(t)}$ with out method. Using the policy trained in the source environment without adapting it to the target environment yields suboptimal results. The adapted policy $\pi^{(t)}$ can recover gaits close to the source trajectory.

of the target environment lies outside the training distribution, there is no performance guarantee. In meta-learning, such as Model Agnostic Meta-Learning (MAML) (Finn et al., 2017a;b; Nagabandi et al., 2018), meta-learned dynamics policies and models can adapt to perturbed environments with notable success including in physical robots. However, similar to domain randomization based approaches, they experience difficulties on new environments that are not covered by the training distribution. Our proposed approach focuses on online adaption in unseen environments. It is orthogonal to domain randomization and meta-learning approaches. We show in experiments that these different approaches can be easily combined.

Model-based Reinforcement Learning. Model-based reinforcement learning (MBRL) provides a paradigm that learns the environment dynamics and optimizes the control actions at the same time. Recent work has shown that MBRL has much better sample efficiency compared to model-free approaches both theoretically and empirically (Tu & Recht, 2018; Chua et al., 2018; Sun et al., 2019a). Our setting is different from the traditional MBRL setting. We consider test environments that are different from the training environment, and adapt the policy from the training environment to the test environment.

3. Preliminaries

We consider finite-horizon Markov Decision Processes (MDP) $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, f, H, R \rangle$ with the following components. $\mathcal{S}$ denotes the state space, and $\mathcal{A}$ the action space. The transition function $f \colon \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$ determines the probability $f(s'|s, a)$ of transitioning into state s' from state s after taking action a. The reward function $R : \mathcal{S} \to \mathbb{R}$ is defined *only* on states. We write π_θ to denote a stochastic policy $\pi_\theta \colon \mathcal{S} \times \mathcal{A} \to [0, 1]$ parameterized by θ. Each policy π_θ determines a distribution over trajectories $\{(s_i, a_i, r_i)\}_{i=1}^H$ under a fixed dynamics f. The goal of the agent is to maximize the expected cumulative reward $J(\theta) = \mathbb{E}_{\pi_\theta, f}\left[\sum_{h=1}^H R(s_h)\right]$ over all possible trajectories that can be generated by π_θ. Without loss of generality, in the theoretical analysis we always assume the normalized total reward is in the $[0, 1]$ range, i.e., $\max_{s_1, \ldots s_H} \sum_{h=1}^H R(s_h) \in [0, 1]$.

We write the set $\{1, 2, \ldots, N\}$ as $[N]$, and the uniform distribution over set A as $U(A)$ throughout the paper. For any two distributions d_1 and d_2, we use $\|d_1 - d_2\|$ to denote the total variation distance between the two distributions.

4. Policy Adaptation with Data Aggregation

4.1. Basic Definitions

In policy adapation, we consider a pair of MDPs and call them a source MDP and a target MDP. We define the source MDP as $\mathcal{M}^{(s)} := \{\mathcal{S}, \mathcal{A}^{(s)}, f^{(s)}, H, R\}$ and the target MDP as $\mathcal{M}^{(t)} := \{\mathcal{S}, \mathcal{A}^{(t)}, f^{(t)}, H, R\}$. Note that the two MDPs share the same state space and reward functions, but can have different action spaces and transition dynamics. Fig. 1 demonstrates the problem of adapting a policy from a source environment to a target environment. Because of the difference in the action space and dynamics, directly using good policies from the source environment (Fig.1(a))

does not work in the target environment (Fig.1(b)). The objective is to adapt the policy from the source to the target environment to achieve good performance (Fig.1(c)).

We focus on minimizing the samples needed for adaptation in the target MDP, by leveraging $\mathcal{M}^{(s)}$ to quickly learn a policy in $\mathcal{M}^{(t)}$. To achieve this, we assume that a pre-trained policy $\pi^{(s)}$ from $\mathcal{M}^{(s)}$ achieves high rewards in $\mathcal{M}^{(s)}$. We wish to adapt $\pi^{(s)}$ to a policy $\pi^{(t)}$ that works well in $\mathcal{M}^{(t)}$. For ease of presentation, we consider $\pi^{(s)}$ and $\pi^{(t)}$ as deterministic throughout the theoretical analysis.

Given a policy π, we write $\mathbb{E}_{\pi}^{(s)}(\cdot)$ for the expectation over random outcomes induced by π and $\mathcal{M}^{(s)}$. We write $d_{\pi;h}^{(s)}$ to denote the state distribution induced by π at time step h under $\mathcal{M}^{(s)}$, and $d_{\pi}^{(s)} = \sum_{h=1}^{H} d_{\pi;h}^{(s)}/H$ as the average state distribution of π under $\mathcal{M}^{(s)}$. We write $\rho_{\pi}^{(s)}$ to represent the distribution of the state trajectories from π: for $\tau = \{s_h\}_{h=0}^{H}$, $\rho_{\pi}^{(s)}(\tau) = \prod_{h=1}^{H} f^{(s)}(s_h|s_{h-1}, \pi(s_{h-1}))$. For the target MDP $\mathcal{M}^{(t)}$, we make the same definitions but drop the superscript (t) for ease of presentation. Namely, $\mathbb{E}_{\pi}(\cdot)$ denotes the expectation over the randomness from π and $\mathcal{M}^{(t)}$, d_{π} denotes the induced state distribution of π under $\mathcal{M}^{(t)}$, and ρ_{π} denotes the state trajectory distribution.

4.2. Algorithm

We now introduce the main algorithm Policy Adaptation with Data Aggregation (PADA). Note that this is the theoretical version of the algorithm, and the practical implementation will be described in detail in Section 5. To adapt a policy from a source environment to a target environment, PADA learns a model $\hat{f}$ to approximate the target environment dynamics $f^{(t)}$. Based on the learned model, the algorithm generates actions that attempt to minimize the divergence between the trajectories in the target environment and those in the source environment generated by $\pi^{(s)}$ at $\mathcal{M}^{(s)}$. Namely, the algorithm learns a policy $\pi^{(t)}$ that reproduces the behavior of $\pi^{(s)}$ on $\mathcal{M}^{(s)}$ in the target MDP $\mathcal{M}^{(t)}$. Since the state space $\mathcal{S}$ is often large, learning a model $\hat{f}$ that can accurately approximate $f^{(t)}$ globally is very costly. Instead, we only aim to iteratively learn a locally accurate model, i.e., a model that is accurate near the states that are generated by $\pi^{(t)}$. This is the key to efficient adaptation.

The detailed algorithm is summarized in Alg. 1. Given a model $\hat{f}_e$ at the e-th iteration, we define the ideal policy $\pi_e^{(t)}$

$$\pi_e^{(t)}(s) \triangleq \operatorname*{argmin}_{a \in \mathcal{A}^{(t)}} \|\hat{f}_e(\cdot|s,a) - f^{(s)}(\cdot|s, \pi^{(s)}(s))\|. \quad (1)$$

The intuition is that, assuming $\hat{f}_e$ is accurate in terms of modelling $f^{(t)}$ at state s, $\pi_e^{(t)}(s)$ aims to pick an action such that the resulting next state distribution under $\hat{f}_e$ is similar to the next state distribution resulting from $\pi^{(s)}$ under the source dynamics $f^{(s)}$. We then execute $\pi_e^{(t)}$ in the target environment $\mathcal{M}^{(t)}$ to generate a batch of data (s, a, s'). We further aggregate the newly generated data to the dataset $\mathcal{D}$ (i.e., data aggregation). We update model to $\hat{f}_{e+1}$ via Maximum Likelihood Estimation (MLE) on $\mathcal{D}$:

$$\hat{f}_{e+1} = \operatorname*{argmax}_{f \in \mathcal{F}} \sum_{s,a,s' \in \mathcal{D}} \log f(s'|s,a). \quad (2)$$

Algorithm 1 Policy Adaptation with Data Aggregation

Require: Source domain policy $\pi^{(s)}$, source dynamics $f^{(s)}$, model class $\mathcal{F}$
Initialize dataset $\mathcal{D} = \emptyset$
Initialize $\hat{f}_1$
for e = 1, ... T **do**
  Define policy $\pi_e^{(t)}$ as in Eq. 1
  for n = 1, ... N **do**
    Reset $\mathcal{M}^{(t)}$ to a random initial state
    Uniformly sample a time step $h \in [H]$
    Execute $\pi_e^{(t)}$ in $\mathcal{M}^{(t)}$ for h steps to get state s
    Sample exploration action $a \sim U(\mathcal{A}^{(t)})$
    Take a in $\mathcal{M}^{(t)}$ and get next state s'
    Add (s, a, s') into $\mathcal{D}$ (Data Aggregation)
  end for
  Update to $\hat{f}_{e+1}$ via MLE by Eq. 2
end for
Output: $\{\pi_e^{(t)}\}_{e=1}^{T}$

Note that Algorithm 1 relies on two black-box offline computation oracles: (1) a *one-step* minimization oracle (Eq. 1) and (2) a Maximum Likelihood Estimator (Eq. 2). In Section 5, we will introduce practical methods to implement these two oracles. We emphasize here that these two oracles are offline computation oracles and the computation itself does not require any fresh samples from the target environment $\mathcal{M}^{(t)}$.

4.3. Analysis

We now prove the performance guarantee of Alg.1 for policy adaptation and establish its rate of convergence. At a high level, our analysis of Alg. 1 is inspired from the analysis of DAgger (Ross et al., 2011; Ross & Bagnell, 2012) which leverages a reduction to no-regret online learning (Shalev-Shwartz et al., 2012). We will first make the connection with the Follow-the-Leader (FTL) algorithm, a classic no-regret online learning algorithm, on a sequence of loss functions. We then show that we can transfer the no-regret property of FTL to performance guarantee on the learned policy $\pi^{(t)}$. Our analysis uses the FTL regret bound $\tilde{O}(1/T)$ where T is the number of iterations (Shalev-Shwartz et al., 2012). Since our analysis is a reduction to general no-regret online learning, in theory we can also replace FTL by other no-regret

online learning algorithms as well (e.g., Online Gradient Descent (Zinkevich, 2003) and AdaGrad (Duchi et al., 2011)). Intuitively, for fast policy adaptation to succeed, one should expect that there is similarity between the source environment and the target environment. We formally introduce the following assumption to quantify this.

Assumption 4.1 (Adaptability)**.** *For any state action pair (s, a) with source action $a \in \mathcal{A}^{(s)}$, there exists a target action $a' \in \mathcal{A}^{(t)}$ in target environment, such that:*

$$\|f^{(s)}(\cdot|s, a) - f^{(t)}(\cdot|s, a')\| \leq \epsilon_{s,a},$$

for some small $\epsilon_{s,a} \in \mathbb{R}^+$.

Remark 4.2. *When $\epsilon_{s,a} \to 0$ in the above assumption, the target environment can perfectly recover the dynamics of the source domain at (s, a). However, $\epsilon_{s,a} = 0$ does not mean the two transitions are the same, i.e., $f^{(t)}(s, a) = f^{(s)}(s, a)$. First the two action spaces can be widely different. Secondly, there may exist states s, where one may need to take completely different target actions from $\mathcal{A}^{(t)}$ in order to match the source transition $f^{(s)}(\cdot|s, a)$, i.e., $\exists a' \in \mathcal{A}^{(t)}$ such that $f^{(t)}(\cdot|s, a') = f^{(s)}(\cdot|s, a)$, but $a \neq a'$.*

Assumption 4.3 (Realizability)**.** *Let the model class $\mathcal{F}$ be a subset of $\{f : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, 1]\}$. We assume $f^{(t)} \in \mathcal{F}$.*

Here we assume that our model class $\mathcal{F}$ is rich enough to include $f^{(t)}$. Note that the assumption on realizability is just for analysis simplicity. Agnostic results can be achieved with more refined analysis similar to (Ross et al., 2011; Ross & Bagnell, 2012).

We define the following loss function:

$$\ell_e(f) \triangleq \mathbb{E}_{s\sim d_{\pi_e^{(t)}}, a\sim U(\mathcal{A}^{(t)})} \left[D_{KL} \left(f^{(t)}(\cdot|s, a), f(\cdot|s, a) \right) \right],$$

for all $e \in [T]$. The loss function $\ell_e(f)$ measures the difference between $f^{(t)}$ and f under the state distribution induced by $\pi_e^{(t)}$ under $\mathcal{M}^{(t)}$ and the uniform distribution over the action space. This definition matches the way we collect data inside each episode. We generate (s, a, s') triples via sampling s from $d_{\pi_e^{(t)}}$, a from $U(\mathcal{A}^{(t)})$, and then $s' \sim f^{(t)}(\cdot|s, a)$. At the end of the iteration e, the learner uses FTL to compute $\hat{f}_{e+1}$ as:

$$\hat{f}_{e+1} = \operatorname*{argmin}_{f\in\mathcal{F}} \sum_{i=1}^{e} \ell_i(f).$$

Using the definition of KL-divergence, it is straightforward to show that the above optimization is equivalent to the following Maximum Likelihood Estimation:

$$\operatorname*{argmax}_{f\in\mathcal{F}} \sum_{i=1}^{e} \mathbb{E}_{s\sim d_{\pi_e^{(t)}}, a\sim U(\mathcal{A}^{(t)}), s'\sim f^{(t)}(\cdot|s,a)} \left[\log f(s'|s, a) \right].$$

At the end of the episode e, the aggregated dataset $\mathcal{D}$ contains triples that are sampled based on the above procedure from the first to the e-th episode.

With no-regret learning on $\hat{f}_e$, assumptions 4.1, and 4.3, we can obtain the following main results. We first assume that the target environment $\mathcal{M}^{(t)}$ has a discrete action space, i.e., $\mathcal{A}^{(t)}$ is discrete, and then show that the result can be easily extended to continuous action spaces.

Theorem 4.4 (Main Theorem)**.** *Assume $\mathcal{M}^{(t)}$ has a discrete action space $\mathcal{A}^{(t)}$ and denote $A \triangleq |\mathcal{A}^{(t)}|$. Among the sequence of policies computed in Alg. 1, there exists a policy $\hat{\pi}$ such that:*

$$\mathbb{E}_{s\sim d_{\hat{\pi}}} \|f^{(t)}(\cdot|s, \hat{\pi}(s)) - f^{(s)}(\cdot|s, \pi^{(s)}(s))\| \leq O\left(AT^{-1/2} + \mathbb{E}_{s\sim d_{\hat{\pi}}} \left[\epsilon_{s,\pi^{(s)}(s)} \right] \right),$$

which implies that:

$$\left\| \rho_{\hat{\pi}} - \rho^{(s)}_{\pi^{(s)}} \right\| \leq O\left(HAT^{-1/2} + H\mathbb{E}_{s\sim d_{\hat{\pi}}} \left[\epsilon_{s,\pi^{(s)}(s)} \right] \right),$$

where we recall that ρ_π stands for the state-trajectory distribution of policy π under $\mathcal{M}^{(t)}$ and $\rho^{(s)}_{\pi^{(s)}}$ stands for the state-trajectory distribution of $\pi^{(s)}$ under $\mathcal{M}^{(s)}$.

The full proof is in the Appendix A. The theorem shows that our algorithm can provide a policy in the target environment that induces trajectories close to those induced by the experts in the source environment. For instance, if the target and source MDPs are completely adaptable (i.e., $\epsilon_{s,a} = 0$ in Assumption 4.1 for all (s, a)) and the number of iterations approach to infinity, then we can learn a policy $\hat{\pi}$ that generates state trajectories in $\mathcal{M}^{(t)}$ that match the state trajectories generated via the source policy $\pi^{(s)}$ at the source MDP $\mathcal{M}^{(s)}$.

Remark 4.5. *The error $\mathbb{E}_{s\sim d_{\hat{\pi}}} \left[\epsilon_{s,\pi^{(s)}(s)} \right]$ is averaged over the state distribution induced by the learned policy rather than in an ℓ_∞ form, i.e., $\max_{s,a} \epsilon_{s,a}$.*

Although the analysis is done on discrete action space, the algorithm can be naturally applied to compact continuous action space as follows. The proof of the following corollary and its extension to the d-dimensional continuous action spaces are in the Appendix.

Corollary 4.6 (Continuous Action Space)**.** *Assume $\mathcal{A}^{(t)} = [0, 1]$, $f^{(t)}$ and functions $f \in \mathcal{F}$ are Lipschitz continuous with (and only with) actions in $\mathcal{A}^{(t)}$. Among policies returned from Alg. 1, there exists a policy $\hat{\pi}$ such that:*

$$\|\rho_{\hat{\pi}} - \rho^{(s)}_{\pi^{(s)}}\| \leq O\left(HT^{-1/4} + H\mathbb{E}_{s\sim d_{\hat{\pi}}} \left[\epsilon_{s,\pi^{(s)}(s)} \right] \right).$$

Remark 4.7. *As we assume the reward function only depends on states, $\|\rho_{\hat{\pi}} - \rho^{(s)}_{\pi^{(s)}}\| \leq \delta$ implies $|J^{(t)}(\hat{\pi}) - J^{(s)}(\pi^{(s)})| \leq \|\rho_{\hat{\pi}} - \rho^{(s)}_{\pi^{(s)}}\| \left(\max_{s_1 \ldots s_H} \sum_h^H R(s_h) \right) \leq \delta$*

due to the normalization assumption on the rewards. Thus, though our algorithm runs without rewards, when $\pi^{(s)}$ achieves high reward in the source MDP $\mathcal{M}^{(s)}$, the algorithm is guaranteed to learn a policy $\hat{\pi}$ that achieves high rewards in the target environment $\mathcal{M}^{(t)}$.

5. Practical Implementation

In Algorithm 1 we showed the theoretical version of our approach, which takes an abstract model class $\mathcal{F}$ as input and relies on two offline computation oracles (Eq. 1 and Eq. 2). We now design the practical implementation by specifying the parameterization of the model class $\mathcal{F}$ and the optimization oracles. Algorithm 2 shows the practical algorithm, and we explain the details in this section.

5.1. Model Parameterization

In the continuous control environments, we focus on stochastic transitions with Gaussian noise, where $f^{(t)}(s,a) = \bar{f}^{(t)}(s,a) + \epsilon$, $f^{(s)}(s,a) = \bar{f}^{(s)}(s,a) + \epsilon'$, with ϵ and ϵ' from $\mathcal{N}(0,\Sigma)$ and $\bar{f}^{(t)}$ and $\bar{f}^{(s)}$ being nonlinear deterministic functions. In this case, we consider the following model class with parameterization θ:

$$\mathcal{F} = \{\delta_\theta(s,a) + \hat{f}^{(s)}(s,\pi^{(s)}(s)), \forall s,a : \theta \in \Theta\}.$$

where $\hat{f}^{(s)}$ is a pre-trained model of the source dynamics $f^{(s)}$ and we assume $\hat{f}^{(s)}$ well approximates $f^{(s)}$ (and one has full control to the source environment such as the ability to reset). Then for each state s, $\hat{f}^{(s)}(s,\pi^{(s)}(s))$ is a fixed distribution of the next state in the source environment by following the source policy. Define $\Delta^{\pi^{(s)}}(s,a) \triangleq \hat{f}^{(s)}(s,\pi^{(s)}(s)) - f^{(t)}(s,a)$, which captures the deviation from taking action a in the target environment to following $\pi^{(s)}$ in the source environment. So $\delta_\theta(s,a)$ is trained to approximate the deviation $\Delta^{\pi^{(s)}}(s,a)$. Note that learning $\Delta^{\pi^{(s)}}$ is just an alternative way to capture the target dynamics since we know $\hat{f}^{(s)}(s,\pi(s))$ foresight, thus it should be no harder than learning $f^{(t)}$ directly.

5.2. Model Predictive Control

For deterministic transition, Eq 1 reduces to one-step minimization $\operatorname{argmin}_{a\in\mathcal{A}^{(t)}} \|\hat{f}_e(s,a) - \hat{f}^{(s)}(s,\pi^{(s)}(s))\|_2$. Since $\hat{f}_e \in \mathcal{F}$, we have $\hat{f}_e(s,a) = \delta_{\theta_e}(s,a) + \hat{f}^{(s)}(s,\pi^{(s)}(s))$, and the optimization can be further simplfied to: $\operatorname{argmin}_{a\in\mathcal{A}^{(t)}} \|\delta_{\theta_e}(s,a)\|_2$. We use the Cross Entropy Method (CEM) (Botev et al., 2013) which iteratively repeats: randomly draw N actions, evaluate them in terms of the objective value $\|\delta_{\theta_e}(s,a)\|_2$, pick the top K actions in the increasing order of the objective values, and then refit a new Gaussian distribution using the empirical mean and covariance of the top K actions.

Algorithm 2 Policy Adaptation with Data Aggregation via Deviation Model

Require: π_s, $\hat{f}^{(s)}$, deviation model class $\{\delta_\theta : \theta \in \Theta\}$, explore probability ϵ, replay buffer $\mathcal{D}$, learning rate η

1: Randomly initialize divergence model δ_θ
2: **for** T Iterations **do**
3: **for** n steps **do**
4: $s \leftarrow$ Reset $\mathcal{M}^{(t)}$
5: **while** current episode does not terminate **do**
6: With probability ϵ: $a \sim U(\mathcal{A}^{(t)})$
7: Otherwise: $a \leftarrow \mathrm{CEM}(\mathcal{A}^{(t)}, s, \delta_\theta)$
8: Execute a in $\mathcal{M}^{(t)}$: $s' \leftarrow f^{(t)}(s,a)$
9: Update replay buffer: $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s,a,s')\}$
10: $s \leftarrow s'$
11: **end while**
12: **end for**
13: Update θ with Eq. 3
14: **end for**

We emphasize here we only need to solve a *one-step* optimization problem without unrolling the system for multiple steps. We write the CEM oracle as $\mathrm{CEM}(\mathcal{A}^{(t)}, s, \delta_\theta)$ which outputs an action a from $\mathcal{A}^{(t)}$ that approximately minimizes $\|\delta_\theta(s,a)\|_2$. Here, $\mathrm{CEM}(\mathcal{A}^{(t)}, s, \delta_\theta) : \mathcal{S} \to \mathcal{A}^{(t)}$ can be considered as a policy that maps state s to a target action a.

5.3. Experience Replay for Model Update

Note that Alg. 1 requires to solve a batch optimization problem (MLE in Eq. 2) in every iteration, which could be computationally expensive in practice. We use Experience Replay (Adam et al., 2011; Mnih et al., 2013), which is more suitable to optimize rich non-linear function approximators (δ_θ is a deep neural network in our experiments). Given the current divergence model δ_θ and the aggregated dataset $\mathcal{D} = \{s,a,s'\}$ (aka, replay buffer) with $s' = f^{(t)}(s,a)$, we randomly sample a mini-batch $B \subset \mathcal{D}$ and perform a stochastic gradient descent step with learning rate η:

$$\theta \leftarrow \theta - \tfrac{\eta}{|B|}\nabla_\theta \left(\textstyle\sum_{i=1}^{|B|} \|\hat{f}^{(s)}(s_i,\pi^{(s)}(s_i)) + \delta_\theta(s_i,a_i) - s_i'\|_2^2\right). \quad (3)$$

5.4. Policy Adaptation with Data Aggregation

As show in Algorithm 2, we maintain a reply buffer that stores all experiences from the target model $\mathcal{M}^{(t)}$ (Line 2) and constantly update the model δ_θ using mini-batch SGD (Eq. 3). Alg 2 performs local exploration in an ϵ-greedy way. We refer our method as Policy Adaptation with Data Aggregation via Deviation Model (**PADA-DM**).

Remark 5.1. *Even being one-step, $\mathrm{CEM}(\mathcal{A}^{(t)}, s, \delta_\theta)$ may be computationally expensive, we could obtain a MPC-free policy (target policy) by training an extra parameterized*

policy to mimic $\mathrm{CEM}(\mathcal{A}^{(t)}, s, \delta_\theta)$ *via techniques of Behavior Cloning (Bain & Sommut, 1999). When we train this extra parameterized policy, we name the method as* ***PADA-DM with target policy*** *and we will show it does not affect the performance of the overall algorithm during training. However, during test time, such parameterized policy runs faster than CEM and thus is more suitable to be potentially deployed on real-world systems.*

6. Experiments

In this section we compare our approach with the state-of-the-art methods for policy adaptation (Christiano et al., 2016; Zhu et al., 2018; Schulman et al., 2017; Finn et al., 2017a) and show that we achieve competitive results more efficiently. We also test the robustness of the approaches on multi-dimensional perturbations. We then compare to domain randomization and meta-learning approaches and show how they can be combined with our approach. We provide further experiments in Appendix D.

Following the same experiment setup as (Christiano et al., 2016), We focus on standard OpenAI Gym (Brockman et al., 2016) and Mujoco (Todorov et al., 2012) control environments such as HalfCheetah, Ant, and Reacher. We perturb the environments by changing their parameters such as mass, gravity, dimensions, motor noise, and friction. More details of task designs are in Appendix B.1.

6.1. Comparison with Existing Approaches

We compare our methods (PADA-DM, PADA-DM with target policy) with the following state-of-the-art methods for policy adaptation. The names correspond to the learning curves shown in Figure 2.

Christiano et al., 2016: (Christiano et al., 2016) uses a pre-trained policy $\pi^{(s)}$ and source dynamics $f^{(s)}$, to learn an inverse dynamics model $\phi\colon \mathcal{A} \times \mathcal{S} \times \mathcal{S} \to [0, 1]$, where $\phi(a|s, s')$ is the probability of taking action a that leads to s' from s.[1] That is, $\pi^{(t)}(s) = \phi(s, f^{(s)}(s, \pi^{(s)}(s)))$.

Zhu et al., 2018: (Zhu et al., 2018) proposed an approach for training policies in the target domain with a new reward $\lambda R(s_h) + (1 - \lambda) R_{gail}(s_h, a_h)$, $\lambda \in [0, 1]$. Here R_{gail} is from the discriminator from GAIL. Note that this baseline has access to true reward signals while ours do not.

For additional baselines, we also show the performance of directly running Proximal Policy Optimization (**PPO**) (Schulman et al., 2017) in the target environment, as well as directly using the **source policy** in the perturbed environment without adapation.

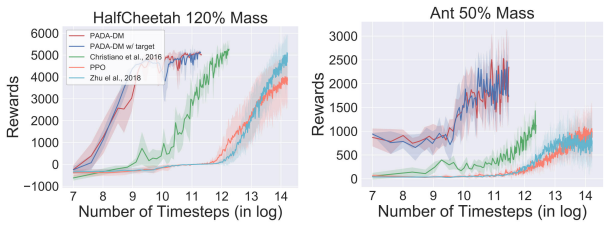


Figure 3. The long-term learning curves. The x-axis is the number of timesteps in natural logarithm scale.

Results. Figure 2 demonstrates the sample efficiency of our methods compared to the other methods and baselines. Both PADA-DM and PADA-DM with target policy converge within 10k to 50k training samples in the target environments. In contrast, (Christiano et al., 2016) requires 5 times more samples than our methods on average, and (Zhu et al., 2018) and PPO require about 30 times more. At convergence, our methods obtain the highest episodic rewards in 7 out of 8 tasks above among the policy adaptation methods. The baseline performance of PPO is better than the policy adaptation methods in HalfCheetah and Reacher (recall PPO uses true reward signals), but it takes significantly longer as shown in Fig. 2. Note that in the Ant environment, even at convergence our methods outperform PPO as well.

The only task where our methods failed to achieve top performance is Ant-v2 0.6 std motor noise. In this environment, the action noise causes high divergence between the target and source environments, making it hard to efficiently model the domain divergence. All the adaptation methods deliver bad performance in this case, indicating the difficulty of the task.

We observe that the learning curves of PADA-DM and PADA-DM with target policy are similar across all tasks without sacrificing efficiency or performance. The target policy can be directly used without any MPC step.

To further illustrate the sample efficiency of our method, we compare the long-term learning curves in Fig. 3. We plot the learning curves up to convergence of each method. We further include a long-term version of Fig 2 and the hyperparameters in the Appendix.

6.2. Performance on Multi-Dimensional Perturbations

We further evaluate the robustness of our methods by perturbing multiple dimensions of the target environment (Fig. 4). Note that online adaptation is particularly useful for multiple-dimension perturbations, because they generate an exponentially large space of source environments that are hard to sample offline. In Fig. 4(b), we show that even when perturbing 15 different degrees of freedom of the tar-

[1] In general an inverse model is ill-defined without first specifying a policy, i.e., via Bayes rule, $\phi(a|s, s') \propto P(a, s, s') = f^{(t)}(s'|s, a)\pi(a|s)$. Hence one needs to first specify π in order to justify the existence of an inverse model $\phi(a|s, s')$.

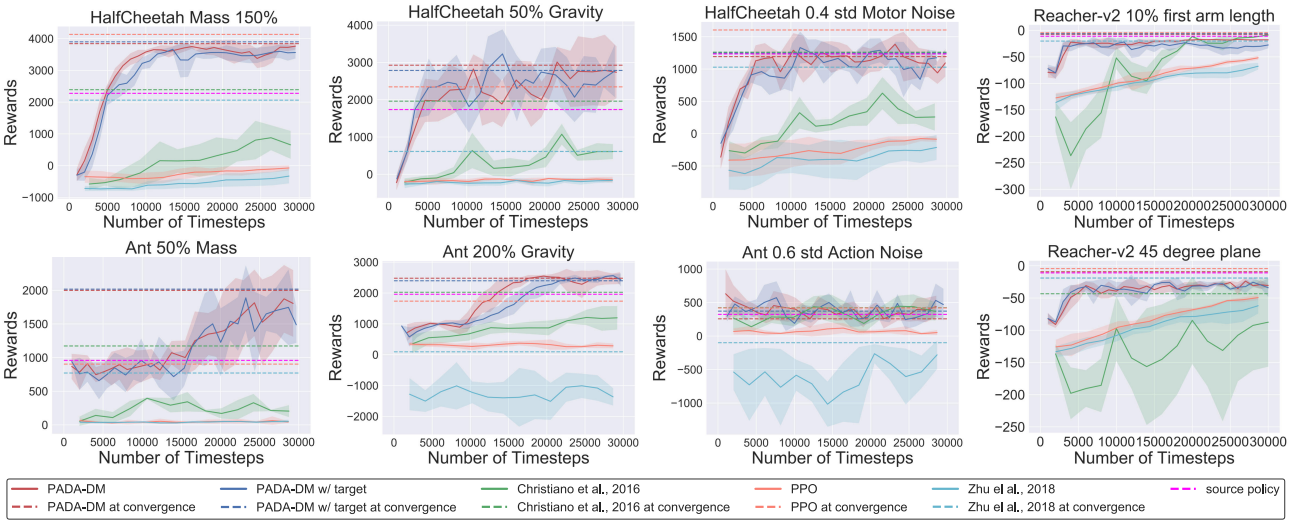


Figure 2. We plot the learning curves across 5 random seeds on a number of tasks. The title of each plot correspounds to the perturbation in the target domain, e.g., HalfCheetah Mass 150% means in mass of the agent in the target domain is 150% of that in the source domain.

get environment, our adaptation method can still achieve competitive performance at a much faster rate than all the other methods. We record the details of the configurations of the target environments in Appendix B.2.7.

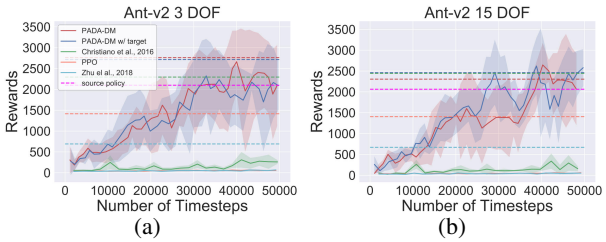


Figure 4. Learning curves for: changing 3 (a) and 15 (b) configurations in the target environment.

6.3. Comparison with Domain Randomization and Meta-Learning

We now compare with domain randomization and meta-learning approaches and show how they can be combined with our methods.

	single source domain	randomized source domains
no adaptation in target domain	source policy	DR
adaptation in target domain	PADA-DM	PADA-DM w/ DR; MAML

Table 1. Relationship of 5 methods in our ablation study.

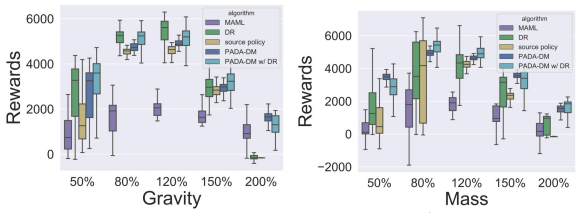
Domain randomization (DR) (Peng et al., 2018): During the training in the source domain, at the beginning of each episode, we randomly sample a new configuration for the source domain within a specified distribution. The total number of training samples here is the same as that for training the source policy. The policy outputed from DR is used in the target environment without further adaptation.

MAML: We adopt the RL version of MAML (Finn et al., 2017a) that meta-learns a model-free policy over a distribution of source environments and performs few-shot adaptation on the target environment.

We compare the following methods: (1) source policy trained in a fixed source environment, (2) domain randomization, (3) PADA-DM, (4) PADA-DM with DR (using domain randomization's output as $\pi^{(s)}$ for PADA), and (5) MAML. Table 1 shows how these methods relate to each other.

For the first four methods, we train the source policy with 2m samples and perform adaptation with 80k samples. For MAML, we use 500 batches of meta-training (400m samples), and adapt 10 times with 80k samples in the target domain. We perform 100 trajectories across 5 random seeds in the target domain for each method and compare the episodic reward in Figure 5. We first observe that when the target domains lie in the distribution of domain randomization ($70\% - 130\%$), domain randomization outperforms source policy significantly, but does not help when the target lies far away from the distribution, which is the most notable shortcoming of domain randomization. Note that using domain randomization in conjunction with our adaptation method usually yields the best results. Domain randomization often

provides robustness within the environments' distribution, and online adaptation in real target environment using our approach further ensures robustness to out-of-distribution environments. We also observe that our method provides the most stable performance given the smallest test variances. We include additional experiments and detailed numbers of the performances of all methods (mean and standard deviations) in Appendix D.4.



(a) (b)

Figure 5. Ablation experiments using domain randomization and meta-learning. (a) Varying gravity. (b) Varying mass.

7. Conclusion

We proposed a novel policy adaptation algorithm that combines techniques from model-based RL and no-regret online learning. We theoretically proved that our methods generate trajectories in the target environment that converge to those in the source environment. We established the rate of convergence of the algorithms. We have shown that our algorithm achieves competitive performance across a diverse set of continuous control tasks with better sample efficiency. A natural extension is to use our approach on simulation-to-real problems in combination with domain randomization and meta-learning.

As our experiments indicated that the combination of domain randomization and our online adaptation approach together often yields good results, for future work, we plan to investigate general theoretical framework for combining domain randomization and online adaptive control techniques.

Acknowledgement

This material is based upon work supported by the US Air Force and DARPA under Contract No. FA8750-18-C-0092 and FA9550-19-1-0041, and the National Science Foundation under NRI CNS-1830399. Part of the work was done while WS was at Microsoft Research NYC.

References

Adam, S., Busoniu, L., and Babuska, R. Experience replay for real-time reinforcement learning control. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(2):201–212, 2011.

Antonova, R., Cruciani, S., Smith, C., and Kragic, D. Reinforcement learning for pivoting task. *arXiv preprint arXiv:1703.00472*, 2017.

Åström, K. J. Theory and applications of adaptive controla survey. *Automatica*, 19(5):471–486, 1983.

Bain, M. and Sommut, C. A framework for behavioural claning. *Machine intelligence*, 15(15):103, 1999.

Botev, Z. I., Kroese, D. P., Rubinstein, R. Y., and LEcuyer, P. The cross-entropy method for optimization. In *Handbook of statistics*, volume 31, pp. 35–59. Elsevier, 2013.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym. *CoRR*, abs/1606.01540, 2016. URL http://arxiv.org/abs/1606.01540.

Chebotar, Y., Handa, A., Makoviychuk, V., Macklin, M., Issac, J., Ratliff, N., and Fox, D. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8973–8979. IEEE, 2019.

Christiano, P., Shah, Z., Mordatch, I., Schneider, J., Blackwell, T., Tobin, J., Abbeel, P., and Zaremba, W. Transfer from simulation to real world through learning deep inverse dynamics model. *arXiv preprint arXiv:1610.03518*, 2016.

Chua, K., Calandra, R., McAllister, R., and Levine, S. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*, pp. 4754–4765, 2018.

Daftry, S., Bagnell, J. A., and Hebert, M. Learning transferable policies for monocular reactive mav control. In *International Symposium on Experimental Robotics*, pp. 3–11. Springer, 2016.

Duchi, J., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1126–1135. JMLR. org, 2017a.

Finn, C., Yu, T., Zhang, T., Abbeel, P., and Levine, S. One-shot visual imitation learning via meta-learning. *arXiv preprint arXiv:1709.04905*, 2017b.

Fu, J., Luo, K., and Levine, S. Learning robust rewards with adversarial inverse reinforcement learning. *arXiv preprint arXiv:1710.11248*, 2017.

Ghasemipour, S. K. S., Zemel, R., and Gu, S. A divergence minimization perspective on imitation learning methods. *arXiv preprint arXiv:1911.02256*, 2019.

Ho, J. and Ermon, S. Generative adversarial imitation learning. In *Advances in neural information processing systems*, pp. 4565–4573, 2016.

Ke, L., Barnes, M., Sun, W., Lee, G., Choudhury, S., and Srinivasa, S. Imitation learning as f-divergence minimization. *arXiv preprint arXiv:1905.12888*, 2019.

Liu, F., Ling, Z., Mu, T., and Su, H. State alignment-based imitation learning. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rylrdxHFDr.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Mordatch, I., Lowrey, K., and Todorov, E. Ensemble-cio: Full-body dynamic motion planning that transfers to physical humanoids. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5307–5314. IEEE, 2015.

Na, D., Lee, H. B., Lee, H., Kim, S., Park, M., Yang, E., and Hwang, S. J. Learning to balance: Bayesian meta-learning for imbalanced and out-of-distribution tasks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rkeZIJBYvr.

Nagabandi, A., Clavera, I., Liu, S., Fearing, R. S., Abbeel, P., Levine, S., and Finn, C. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. *arXiv preprint arXiv:1803.11347*, 2018.

Nagabandi, A., Clavera, I., Liu, S., Fearing, R. S., Abbeel, P., Levine, S., and Finn, C. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. *ICLR*, 2019a.

Nagabandi, A., Finn, C., and Levine, S. Deep online learning via meta-learning: Continual adaptation for model-based rl. *ICLR*, 2019b.

Nair, V. and Hinton, G. E. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814, 2010.

Peng, X. B., Andrychowicz, M., Zaremba, W., and Abbeel, P. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–8. IEEE, 2018.

Ross, S. and Bagnell, J. A. Agnostic system identification for model-based reinforcement learning. *arXiv preprint arXiv:1203.1007*, 2012.

Ross, S., Gordon, G., and Bagnell, D. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 627–635, 2011.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Shalev-Shwartz, S. et al. Online learning and online convex optimization. *Foundations and Trends® in Machine Learning*, 4(2):107–194, 2012.

Staddon, J. E. *Adaptive behavior and learning*. Cambridge University Press, 2016.

Sun, W., Venkatraman, A., Gordon, G. J., Boots, B., and Bagnell, J. A. Deeply aggrevated: Differentiable imitation learning for sequential prediction. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 3309–3318. JMLR. org, 2017.

Sun, W., Jiang, N., Krishnamurthy, A., Agarwal, A., and Langford, J. Model-based rl in contextual decision processes: Pac bounds and exponential improvements over model-free approaches. In *Conference on Learning Theory*, pp. 2898–2933, 2019a.

Sun, W., Vemula, A., Boots, B., and Bagnell, J. A. Provably efficient imitation learning from observation alone. *arXiv preprint arXiv:1905.10948*, 2019b.

Tan, J., Zhang, T., Coumans, E., Iscen, A., Bai, Y., Hafner, D., Bohez, S., and Vanhoucke, V. Sim-to-real: Learning agile locomotion for quadruped robots. *arXiv preprint arXiv:1804.10332*, 2018.

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., and Abbeel, P. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 23–30. IEEE, 2017.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *IROS 2012*, pp. 5026–5033. IEEE, 2012.

Torabi, F., Warnell, G., and Stone, P. Behavioral cloning from observation. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pp. 4950–4957, 2018.

Torabi, F., Warnell, G., and Stone, P. Imitation learning from video by leveraging proprioception. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pp. 3585–3591. AAAI Press, 2019a.

Torabi, F., Warnell, G., and Stone, P. Recent advances in imitation learning from observation. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pp. 6325–6331. AAAI Press, 2019b.

Tu, S. and Recht, B. The gap between model-based and model-free methods on the linear quadratic regulator: An asymptotic viewpoint. *arXiv preprint arXiv:1812.03565*, 2018.

Tzeng, E., Devin, C., Hoffman, J., Finn, C., Abbeel, P., Levine, S., Saenko, K., and Darrell, T. Adapting deep visuomotor representations with weak pairwise constraints. *arXiv preprint arXiv:1511.07111*, 2015.

Wang, T., Bao, X., Clavera, I., Hoang, J., Wen, Y., Langlois, E., Zhang, S., Zhang, G., Abbeel, P., and Ba, J. Benchmarking model-based reinforcement learning. *CoRR*, abs/1907.02057, 2019. URL http://arxiv.org/abs/1907.02057.

Yang, C., Ma, X., Huang, W., Sun, F., Liu, H., Huang, J., and Gan, C. Imitation learning from observations by minimizing inverse dynamics disagreement. In *Advances in Neural Information Processing Systems*, pp. 239–249, 2019.

Yu, T., Finn, C., Xie, A., Dasari, S., Zhang, T., Abbeel, P., and Levine, S. One-shot imitation from observing humans via domain-adaptive meta-learning. *arXiv preprint arXiv:1802.01557*, 2018a.

Yu, W., Liu, C. K., and Turk, G. Policy transfer with strategy optimization. *arXiv preprint arXiv:1810.05751*, 2018b.

Zhu, Y., Wang, Z., Merel, J., Rusu, A., Erez, T., Cabi, S., Tunyasuvunakool, S., Kramr, J., Hadsell, R., de Freitas, N., and Heess, N. Reinforcement and imitation learning for diverse visuomotor skills. In *Proceedings of Robotics: Science and Systems*, Pittsburgh, Pennsylvania, June 2018. doi: 10.15607/RSS.2018.XIV.009.

Zinkevich, M. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pp. 928–936, 2003.

A. Detailed Analysis of Algorithm 1 (Proof of Theorem 4.4)

In this section, we provide detailed proof for Theorem 4.4.

Consider a Markov Chain $p : \mathcal{S} \to \Delta(\mathcal{S})$ over horizon H. Denote d_p as the induced state distribution under p and ρ_p as the induced state trajectory distribution under p.

Lemma A.1. *Consider two Markov Chains $p_i : \mathcal{S} \to \Delta(\mathcal{S})$ with $i \in \{1, 2\}$. If*

$$\mathbb{E}_{s\sim d_{p_1}} \left[\|p_1(\cdot|s) - p_2(\cdot|s)\|\right] \le \delta,$$

then for trajectory distributions, we have:

$$\|\rho_{p_1} - \rho_{p_2}\| \le O(H\delta).$$

The above lemma implies that if the Markov chain p_2 can predict the Markov chain p_1 *under the state distribution induced by* p_1, then we can guarantee that the state-wise trajectory distributions from p_1 and p_2 are close as well.

Proof. Denote $\rho_{p,s_1,\dots s_h}$ as the trajectory distribution induced by p conditioned on the first h many states are equal to $s_1, \dots, s_h$. Denote $p(\cdot|s_0)$ as the initial state distribution for s_1 with s_0 being a faked state. By definition, we have:

$$\begin{aligned}
&\|\rho_{p_1} - \rho_{p_2}\| = \sum_{\tau} |\rho_{p_1}(\tau) - \rho_{p_2}(\tau)| \\
&= \sum_{s_1,\dots s_H} \left| \prod_{h=1}^{H} p_1(s_h|s_{h-1}) - \prod_{h=1}^{H} p_2(s_h|s_{h-1}) \right| \\
&= \sum_{s_1,\dots s_H} \left| \prod_{h=1}^{H} p_1(s_h|s_{h-1}) - p_1(s_1|s_0)\prod_{h=2}^{H} p_2(s_h|s_{h-1}) + p_1(s_1|s_0)\prod_{h=2}^{H} p_2(s_h|s_{h-1}) - \prod_{h=1}^{H} p_2(s_h|s_{h-1}) \right| \\
&\le \sum_{s_1} p_1(s_1|s_0) \sum_{s_2,\dots,s_H} \left| \prod_{h=2}^{H} p_1(s_h|s_{h-1}) - \prod_{h=2}^{H} p_2(s_h|s_{h-1}) \right| + \sum_{s_1} |p_1(s_1|s_0) - p_2(s_1|s_0)| \left(\sum_{s_2,\dots,s_H} \prod_{h=2}^{H} p_2(s_h|s_{h-1}) \right) \\
&= \mathbb{E}_{s_1\sim p_1} \left[\|\rho_{p_1,s_1} - \rho_{p_2,s_1}\|\right] + \|p_1(\cdot|s_0) - p_2(\cdot|s_0)\| \\
&\le \mathbb{E}_{s_1,s_2\sim p_1} \left[\|\rho_{p_1,s_1,s_2} - \rho_{p_2,s_1,s_2}\|\right] + \|p_1(\cdot|s_0) - p_2(\cdot|s_0)\| + \mathbb{E}_{s_1\sim d_{\pi_1;1}}[\|p_1(\cdot|s_1) - p_2(\cdot|s_1)\|].
\end{aligned}$$

Recursively applying the same operation on $\|\rho_{p_1,s_1} - \rho_{p_2,s_1}\|$ to time step H, we have:

$$\|\rho_{p_1} - \rho_{p_2}\| \le \sum_{h=1}^{H} \mathbb{E}_{s_h\sim d_{p_1;h}} \left[\|p_1(\cdot|s_h) - p_2(\cdot|s_h)\|\right] \le H\delta,$$

where we recall $d_\pi = \sum_{h=1}^{H} d_{\pi;h}/H$ by definition. Extension to continuous state can be achieved by simply replaying summation by integration.

□

The next lemma shows that by leveraging the no-regret property of FTL, we can learn a locally accurate model.

Lemma A.2 (Local Accuracy of the Learned Model)**.** *Denote the sequence of models learned in Alg. 1 as $\{\hat{f}_1, \dots, \hat{f}_T\}$, there exists a model $\hat{f} \in \{\hat{f}_1, \dots, \hat{f}_T\}$ such that:*

$$\mathbb{E}_{s\sim d_{\pi_{\hat{f}}}} \left[\mathbb{E}_{a\sim U(\mathcal{A}^{(t)})} \left[D_{KL}(f^{(t)}(\cdot|s,a), \hat{f}(\cdot|s,a)) \right] \right] \le O(1/T),$$

where $\pi_f(s) \triangleq \operatorname{argmin}_{a\in\mathcal{A}^{(t)}} \|f(\cdot|s,a) - f^{(s)}(\cdot|s,\pi^{(s)}(s))$ for all $s \in \mathcal{S}$ for any f.

Proof. Denote loss function $\ell_e(f)$ as:

$$\ell_e(f) \triangleq \mathbb{E}_{s\sim d_{\pi_e^{(t)}}, a\sim U(\mathcal{A}^{(t)})}\left[\mathbb{E}_{s'\sim f^{(t)}_{s,a}}\left[-\log(f(s'|s,a))\right]\right].$$

Since Alg. 1 is equivalent to running FTL on the sequence of strongly convex loss functions $\{\ell_e(f)\}_{e=1}^T$, we have (Shalev-Shwartz et al., 2012):

$$\sum_{e=1}^{T}\ell_e(\hat{f}_e) \le \min_{f\in\mathcal{F}}\sum_{e=1}^{T}\ell_e(f) + O\left(\log T\right).$$

Add $\sum_{e=1}^T \mathbb{E}_{s\sim d_{\pi_e^{(t)}}, a\sim U(\mathcal{A}^{(t)})}[\mathbb{E}_{s'\sim f^{(t)}_{s,a}} \log f^{(t)}(s'|s,a)]$ on both sides of the above inequality and using the definition of KL divergence, we have:

$$\begin{aligned}
&\sum_{e=1}^{T}\mathbb{E}_{s\sim d_{\pi_e^{(t)}}, a\sim U(\mathcal{A}^{(t)})}\left[D_{KL}(f^{(t)}(\cdot|s,a), \hat{f}_e(\cdot|s,a))\right] \\
&\le \min_{f\in\mathcal{F}}\sum_{e=1}^{T}\mathbb{E}_{s\sim d_{\pi_e^{(t)}}, a\sim U(\mathcal{A}^{(t)})}\left[D_{KL}(f^{(t)}(\cdot|s,a), f(\cdot|s,a))\right] + O(\log(T)) = O(\log(T)),
\end{aligned}$$

where the last equality comes from the realizability assumption $f^{(t)} \in \mathcal{F}$.

Using the fact that the minimum among a sequence is less than the average of the sequence, we arrive:

$$\min_{\hat{f}\in\{\hat{f}_e\}_{e=1}^T}\mathbb{E}_{s\sim d_{\pi_{\hat{f}}}, a\sim U(\mathcal{A}^{(t)})}\left[D_{KL}(f^{(t)}(\cdot|s,a), \hat{f}(\cdot|s,a))\right] \le \tilde{O}(1/T).$$

□

The above lemma indicates that as $T\to\infty$, we will learn a model $\hat{f}$ which is close to the target true model $f^{(t)}$ under the state distribution induced by $\pi_{\hat{f}}$. But it does not state the difference between the behavior generated by $\pi_{\hat{f}}$ at the target domain and the behavior generated by $\pi^{(s)}$ at the source domain. The next lemma uses the definition $\pi_{\hat{f}}$ to show that when executing $\pi_{\hat{f}}$ in the target domain $\mathcal{M}^{(t)}$, $\pi_{\hat{f}}$ can actually generates behavior that is similar to the behavior generated by $\pi^{(s)}$ in the source domain $\mathcal{M}^{(s)}$.

Lemma A.3 (The Behavior of $\pi_{\hat{f}}$)**.** *Denote $d_{\pi_{\hat{f}}}$ as the state distribution induced by $\pi_{\hat{f}}$ induced at $\mathcal{M}^{(t)}$ (target domain).*

$$\mathbb{E}_{s\sim d_{\pi_{\hat{f}}}}\|f^{(t)}(\cdot|s,\pi_{\hat{f}}(s)) - f^{(s)}(\cdot|s,\pi^{(s)}(s))\| \le O(|\mathcal{A}^{(t)}|/\sqrt{T}) + \mathbb{E}_{s\sim d_{\pi_{\hat{f}}}}\left[\epsilon_{s,\pi^{(s)}(s)}\right],$$

where we recall the definition of ϵ in Assumption 4.1.

Proof. Consider the Markov chain that is defined with respect to $f^{(t)}$ and $\pi_{\hat{f}}$, i.e., $f^{(t)}(s'|s,\pi_{\hat{f}}(s))$. Denote the state distri-

bution induced by $f^{(t)}(s'|s, \pi_{\hat{f}}(s))$ at the target domain as $d_{\pi_{\hat{f}}}$. Let us bound $\mathbb{E}_{s\sim d_{\pi_{\hat{f}}}}\|f^{(t)}(\cdot|s, \pi_{\hat{f}}(s)) - f^{(s)}(\cdot|s, \pi^{(s)}(s))\|$.

$$
\begin{aligned}
&\mathbb{E}_{s\sim d_{\pi_{\hat{f}}}}\|f^{(t)}(\cdot|s, \pi_{\hat{f}}(s)) - f^{(s)}(\cdot|s, \pi^{(s)}(s))\| \\
&\leq \mathbb{E}_{s\sim d_{\pi_{\hat{f}}}}\|f^{(t)}(\cdot|s, \pi_{\hat{f}}(s)) - \hat{f}(\cdot|s, \pi_{\hat{f}}(s))\| + \mathbb{E}_{s\sim d_{\pi_{\hat{f}}}}\|\hat{f}(\cdot|s, \pi_{\hat{f}}(s)) - f^{(s)}(\cdot|s, \pi^{(s)}(s))\| \\
&\leq \sqrt{\mathbb{E}_{s\sim d_{\pi_{\hat{f}}}} D_{KL}(f^{(t)}(\cdot|s, \pi_{\hat{f}}(s)), \hat{f}(\cdot|s, \pi_{\hat{f}}(s)))} + \mathbb{E}_{s\sim d_{\pi_{\hat{f}}}}\|\hat{f}(\cdot|s, \pi_{\hat{f}}(s)) - f^{(s)}(\cdot|s, \pi^{(s)}(s))\| \\
&\leq O(|\mathcal{A}^{(t)}|/\sqrt{T}) + \mathbb{E}_{s\sim d_{\pi_{\hat{f}}}}\|\hat{f}(\cdot|s, \pi_{\hat{f}}(s)) - f^{(s)}(\cdot|s, \pi^{(s)}(s))\| \\
&\leq O(|\mathcal{A}^{(t)}|/\sqrt{T}) + \mathbb{E}_{s\sim d_{\pi_{\hat{f}}}}\|\hat{f}(\cdot|s, \pi_{f^{(t)}}(s)) - f^{(s)}(\cdot|s, \pi^{(s)}(s))\| \\
&\leq O(|\mathcal{A}^{(t)}|/\sqrt{T}) + \mathbb{E}_{s\sim d_{\pi_{\hat{f}}}}\|\hat{f}(\cdot|s, \pi_{f^{(t)}}(s)) - f^{(t)}(\cdot|s, \pi_{f^{(t)}}(s))\| \\
&\quad + \mathbb{E}_{s\sim d_{\pi_{\hat{f}}}}\|f^{(t)}(\cdot|s, \pi_{f^{(t)}}(s)) - f^{(s)}(\cdot|s, \pi^{(s)}(s))\| \\
&\leq O(|\mathcal{A}^{(t)}|/\sqrt{T}) + \sqrt{\mathbb{E}_{s\sim d_{\pi_{\hat{f}}}} D_{KL}(f^{(t)}(\cdot|s, \pi_{f^{(t)}}(s)), \hat{f}(\cdot|s, \pi_{f^{(t)}}(s)))} + \mathbb{E}_{s\sim d_{\pi_{\hat{f}}}}\left[\epsilon_{s,\pi^{(s)}(s)}\right] \\
&= O(|\mathcal{A}^{(t)}|/\sqrt{T}) + \mathbb{E}_{s\sim d_{\pi_{\hat{f}}}}\left[\epsilon_{s,\pi^{(s)}(s)}\right],
\end{aligned}
$$

where the first inequality uses triangle inequality, the second inequality uses Pinsker's inequality, the third inequality uses the local accuracy of the learned model $\hat{f}$ (Lemma A.2), the fourth inequality uses the definition that $\pi_{\hat{f}}$, and the fifth inequality uses triangle inequality again, and the sixth inequality uses Pinsker's inequality again with the definition of adaptive ability together with the definition of $\pi_{f^{(t)}}(s) \triangleq \text{argmin}_{a\sim\mathcal{A}^{(t)}}\|f^{(t)}(\cdot|s, a) - f^{(s)}(\cdot|s, \pi^{(s)}(s))\|$. □

Proof of Theorem 4.4. Use Lemma A.1 and Lemma A.3 and consider $f^{(s)}(\cdot|s, \pi^{(s)}(s))$ as a Markov chain, we have that:

$$\|\rho^t_{\pi_{\hat{f}}} - \rho^s_{\pi_s}\| \leq O(H|\mathcal{A}^{(t)}|/\sqrt{T}) + H\epsilon,$$

where we denote $\epsilon := \mathbb{E}_{s\sim d_{\pi_{\hat{f}}}}\left[\epsilon_{s,\pi^{(s)}(s)}\right]$ □

This concludes the proof of Theorem 4.4.

A.1. Extension to Continuous Action Space (Proof of Corollary 4.6)

For simplicity, we consider $\mathcal{A}^{(t)} = [0, 1]$.[2] We consider Lipschitz continuous transition dynamics with and only with respect to actions, i.e.,

$$\|f(\cdot|s, a) - f(\cdot|s, a')\| \leq L|a - a'|, \tag{4}$$

where L is a Lipschitz constant. We emphasize here that we only assume Lipschitz continuity with respect to action in $\mathcal{M}^{(t)}$. Hence this is a much weaker assumption than the common Lipschitz continuity assumption used in RL community, which requires Lipschitz continuity in both action and state spaces. We also assume that our function class $\mathcal{F}$ only contains function approximators that are Lipschitz continuous with respect to action a (e.g., feedforward fully connected ReLu network is Lipschitz continuous).

Proof of Corollary 4.6. For analysis purpose, let us discretize the action space into bins with size $\delta \in (0, 1)$. Denote the discrete action set $\bar{\mathcal{A}}^{(t)} = \{0.5\delta, 1.5\delta, 2.5\delta, \ldots, 1 - 0.5\delta\}$ (here we assume $1/\delta = \mathbb{N}^+$). Here $|\bar{\mathcal{A}}^{(t)}| = 1/\delta$.

Now consider the following quantity:

$$\mathbb{E}_{s\sim d_{\pi_{\hat{f}}}}\|f^{(t)}(\cdot|s, \hat{a}) - \hat{f}(\cdot|s, \hat{a})\|,$$

for any $\hat{a}$. Without loss of generality, we assume $\hat{a} \in [0, \delta]$. Via Pinsker's inequality and Lemma A.2, we have:

$$\mathbb{E}_{s\sim d_{\pi_{\hat{f}}}}\mathbb{E}_{a\sim U([0,1])}\|f^{(t)}(\cdot|s, a) - \hat{f}(\cdot|s, a)\| \leq O(1/\sqrt{T}),$$

[2] We can always normalize action to $[0, 1]$.

which implies that:

$$\mathbb{E}_{s\sim d_{\pi_{\hat{f}}}}\mathbb{E}_{a\sim U([0,\delta])}\|f^{(t)}(\cdot|s,a)-\hat{f}(\cdot|s,a)\|\leq O(1/(\delta\sqrt{T})).$$

We proceed as follows:

$$\begin{aligned}
&\mathbb{E}_{s\sim d_{\pi_{\hat{f}}}}\mathbb{E}_{a\sim U([0,\delta])}\|f^{(t)}(\cdot|s,a)-\hat{f}(\cdot|s,a)\|\\
&=\mathbb{E}_{s\sim d_{\pi_{\hat{f}}}}\mathbb{E}_{a\sim U([0,\delta])}\|f^{(t)}(\cdot|s,\hat{a}+a-\hat{a})-\hat{f}(\cdot|s,\hat{a}+a-\hat{a})\|\\
&\geq\mathbb{E}_{s\sim d_{\pi_{\hat{f}}}}\mathbb{E}_{a\sim U([0,\delta])}\left(\|f^{(t)}(\cdot|s,\hat{a})-\hat{f}(\cdot|s,\hat{a})\|-2L|a-\hat{a}|\right)\\
&=\mathbb{E}_{s\sim d_{\pi_{\hat{f}}}}\|f^{(t)}(\cdot|s,\hat{a})-\hat{f}(\cdot|s,\hat{a})\|-2L\mathbb{E}_{a\sim U([0,\delta])}|a-\hat{a}|\\
&\geq\mathbb{E}_{s\sim d_{\pi_{\hat{f}}}}\|f^{(t)}(\cdot|s,\hat{a})-\hat{f}(\cdot|s,\hat{a})\|-2L\mathbb{E}_{a\sim U([0,\delta])}\delta\\
&=\mathbb{E}_{s\sim d_{\pi_{\hat{f}}}}\|f^{(t)}(\cdot|s,\hat{a})-\hat{f}(\cdot|s,\hat{a})\|-2L\delta,
\end{aligned}$$

where the first inequality uses the fact that $\hat{a}\in[0,\delta]$ and the Lipschitz conditions on both $f^{(t)}$ and $\hat{f}\in\mathcal{F}$, the second inequality uses the fact that $|a-\hat{a}|\leq\delta$ for any $a\in[0,\delta]$ as $\hat{a}\in[0,\delta]$.

Hence, we have:

$$\mathbb{E}_{s\sim d_{\pi_{\hat{f}}}}\|f^{(t)}(\cdot|s,\hat{a})-\hat{f}(\cdot|s,\hat{a})\|\leq 2L\delta+O(1/(\delta\sqrt{T}))=O(T^{-1/4}),\forall\hat{a}\in\mathcal{A}^{(t)},$$

where in the last step we set $\delta=\Theta(T^{-1/4})$.

Now, we can simply repeat the process we have for proving Lemma A.3, we will have:

$$\mathbb{E}_{s\sim d_{\pi_{\hat{f}}}}\|f^{(t)}(\cdot|s,\pi_{\hat{f}}(s))-f^{(s)}(\cdot|s,\pi^{(s)}(s))\|\leq O(T^{-1/4})+\epsilon.$$

Combine with Lemma A.1, we prove the corollary for continuous action setting. □

For general n-dim action space, our bound will scale in the order $O(\sqrt{n}T^{-1/(2n+2)})+\epsilon$. The proof of the n-dim result is similar to the proof of the 1-d case and is included below for completeness:

Proof for n-dim Action Space. For n-dimensional action space, we have:

$$\mathbb{E}_{s\sim d_{\pi_{\hat{f}}}}\mathbb{E}_{a\sim U([0,\delta]^n)}\|f^{(t)}(\cdot|s,a)-\hat{f}(\cdot|s,a)\|\leq O(1/(\delta^n\sqrt{T})).$$

Using the Lipschitz property:

$$\begin{aligned}
&\mathbb{E}_{s\sim d_{\pi_{\hat{f}}}}\mathbb{E}_{a\sim U([0,\delta]^n)}\|f^{(t)}(\cdot|s,a)-\hat{f}(\cdot|s,a)\|\\
&=\mathbb{E}_{s\sim d_{\pi_{\hat{f}}}}\mathbb{E}_{a\sim U([0,\delta]^n)}\|f^{(t)}(\cdot|s,\hat{a}+a-\hat{a})-\hat{f}(\cdot|s,\hat{a}+a-\hat{a})\|\\
&\geq\mathbb{E}_{s\sim d_{\pi_{\hat{f}}}}\mathbb{E}_{a\sim U([0,\delta]^n)}\left(\|f^{(t)}(\cdot|s,\hat{a})-\hat{f}(\cdot|s,\hat{a})\|-2L\|a-\hat{a}\|\right)\\
&=\mathbb{E}_{s\sim d_{\pi_{\hat{f}}}}\|f^{(t)}(\cdot|s,\hat{a})-\hat{f}(\cdot|s,\hat{a})\|-2L\mathbb{E}_{a\sim U([0,\delta]^n)}\|a-\hat{a}\|\\
&\geq\mathbb{E}_{s\sim d_{\pi_{\hat{f}}}}\|f^{(t)}(\cdot|s,\hat{a})-\hat{f}(\cdot|s,\hat{a})\|-2L\mathbb{E}_{a\sim U([0,\delta]^n)}\sqrt{n}\delta\\
&=\mathbb{E}_{s\sim d_{\pi_{\hat{f}}}}\|f^{(t)}(\cdot|s,\hat{a})-\hat{f}(\cdot|s,\hat{a})\|-2L\sqrt{n}\delta,
\end{aligned}$$

Combining with above leads to

$$\mathbb{E}_{s\sim d_{\pi_{\hat{f}}}}\|f^{(t)}(\cdot|s,\hat{a})-\hat{f}(\cdot|s,\hat{a})\|\leq 2\sqrt{n}L\delta+O(1/(\delta^n\sqrt{T}))=O(\sqrt{n}T^{-1/(2n+2)})$$

where in the last step we set $\delta=\Theta((\frac{T}{n})^{\frac{-1}{2(n+1)}})$. Finally we have:

$$\mathbb{E}_{s\sim d_{\pi_{\hat{f}}}}\|f^{(t)}(\cdot|s,\pi_{\hat{f}}(s))-f^{(s)}(\cdot|s,\pi^{(s)}(s))\|\leq O(\sqrt{n}T^{-1/(2n+2)})+\epsilon.$$

□

B. Detailed description of our experiments

B.1. Environment descriptions.

For each of the four environments (HalfCheetah, Ant, Reacher) we tested on, we include a detailed numerical description of the environments in table 2.

B.2. Descriptions of the perturbations

B.2.1. CHANGING GRAVITY.

We change the gravity in the whole Mujoco locomotion task by a max range from 50% to 200% of the normal gravity. The normal gravity is set to 0 on x and y axis and -9.81 on z axis.

B.2.2. CHANGING MASS.

We change the mass of the agent in the Mujoco locomotion task by a max range from 50% to 200% of its original mass. Note that most agents are composed of links with independent masses, so besides changing the agent's mass as a whole, we also design experiments that change the mass of each individual links of the agent respectively.

B.2.3. CHANGING PLANE ORIENTATION.

In the Reacher task, we tilt the platform of the Reacher so that it forms an angle of 45 degree of the original plane.

B.2.4. CHANGING ARM LENGTH.

In the Reacher task, we change the first link of the Reacher arm (which is composed of two parts) into one tenth of its original length.

B.2.5. CHANGING FRICTION.

We change the frictional coefficient in the environment on an uniform scale. We found if friction is the only change in the target domain, then the adaptation task is relatively simple, so we incorporate it as one of the changes in the Multi-dimensional perturbation task.

B.2.6. MOTOR NOISE.

This task tries to mimic the motor malfunction or inaccuracy in the real world. After the agent outputs an action, we add on a noise from a normal distribution with mean 0 and a fixed standard deviation from 0.2 to 1.

B.2.7. MULTIPLE DOFS OF CHANGES.

In the 3DOF task, we set the gravity to 0.8, mass to 1.2 and friction coefficient to 0.9. In the 15DOF task, we uniformly sample a coefficient from 0.9 to 1.1 for each of the following configuration: gravity, friction coefficient and the mass of each joint of the agent. We record these changes and apply for all comparing methods.

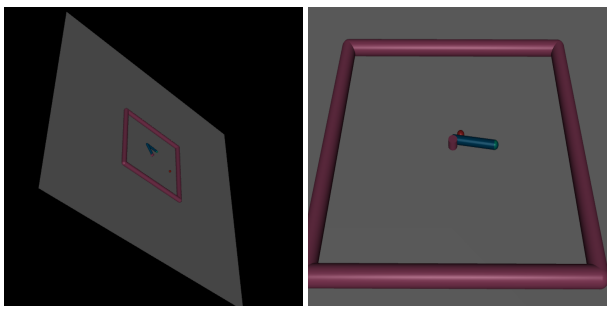


Figure 6. Visual illustration of modified Reacher environment. **left:** 45 degrees of tilted plane. **right:** Reacher with 10% length of its first arm.

Environment name	Full state space size	Model agnostic state size[3]	Action space size	Reward function
HalfCheetah	18	1	6	forward reward - 0.1 $\times$ control cost
Ant	29	2	8	forward reward - 0.5 $\times$ control cost - 0.0005 $\times$ contact cost + survive reward
Reacher	16	4	2	forward distance - control cost

Table 2. Description of the OpenAI gym environments. Note that to enforce safety of the agent in the target environment, we make a modification on HalfCheetah environment such that the episode will be terminated when the cheetah falls off.

B.3. Hyperparameters

	source	PADA-DM	PADA-DM with target	Christiano et al., 2016	Zhu et al., 2018	PPO
# timesteps	2e6	8e4 (5e4,8e4,12e4,15e4)	8e4 (5e4,8e4,1.2e5,1.5e5)	2e5 (1e5,2e5,4e5)	2e6	2e6
learning rate (with linear decay)	7e-4	5e-3	5e-3	5e-3	7e-4	7e-4
soft update rate			every 3000 timesteps (3000,5000,10000)			
explore rate ϵ		0.01 (0.01,0.02)	0.01 (0.01,0.02)			
reward tradeoff λ					0.5	

Table 3. Final hyperparameters we use for our methods and baselines in the experiments. The values in the brackets are the value we considered.

C. Implementation details

C.1. Pretraining of the source dynamics model

In section 5.1, one assumption we make is that we have a pretrained model $\hat{f}^{(s)}$ that well approximates the source dynamics $f^{(s)}$. In practice, we pretrain the model $\hat{f}^{(s)}$ with the (s, a, s') triplets generated during the training of $\pi^{(s)}$. The model $\hat{f}^{(s)}$ is a two-layer neural network with hidden size 128 and ReLU nonlinearity, trained with MSE loss since we assume deterministic transitions. Using these existing samples has two major advantages: first is that we don't need further interaction with the source environment and second is that the trained model $\hat{f}^{(s)}$ especially well approximates the transitions around the actions taken by $\pi^{(s)}$, which is important to our algorithm.

Remark that if we already have the ground truth source dynamics $f^{(s)}$, which is a mild assumption while using a simulator, we can also directly use $f^{(s)}$ to replace $\hat{f}^{(s)}$. During our experiments, we observe that whether using $f^{(s)}$ or $\hat{f}^{(s)}$ won't affect the performance of our method.

C.2. Cross Entropy Method

Here we provide a pseudocode of the Cross Entropy Method that we used in our method, as in Alg. 3. In our implementation, we use $T = 10$ iterations, $N = 200$ actions sampled per iteration, $K = 40$ elites (elite ratio 0.4) and $\sigma_0 = 0.5$. We use the output of target policy as the initial mean, and when we don't have the target policy, we use $\pi^{(s)}(s)$ as the initial mean. To avoid invalid actions, for each action a_i we sample, we clip the action if certain dimension is out of the bound of $\mathcal{A}^{(t)}$ (usually [-1,1] on each dimension).

[3]This means the number of states that won't be passed as inputs to models or policies, e.g., the current coordinate or location of the agent.

Algorithm 3 Cross Entropy Method

Require: Initial mean μ_0, initial standard deviation σ_0, action space $\mathcal{A}^{(t)}$, current model δ_θ, current state s, number of iterations T, sample size N, elite size K.

$\Sigma_0 \leftarrow I_{|\mathcal{A}^{(t)}|}(\sigma_0^2)$
for t = 1, ... T **do**
  Sample $\{a_i\}_{i=1}^N \sim \mathcal{N}(\mu_{t-1}, \Sigma_{t-1})$
  $\{a_i\}_{i=1}^N \leftarrow clip(a_i, \min(\mathcal{A}^{(t)}), \max(\mathcal{A}^{(t)}))$
  Sort $\{a_i\}$ with descending $\|\delta_\theta(s, a_i)\|_2^2$ and pick the first K actions $\{a_j\}_{j=1}^K$
  $\mu_t \leftarrow \frac{1}{K}\Sigma_{j=1}^K a_j$
  $\Sigma_t \leftarrow \frac{1}{K}\Sigma_{j=1}^K (a_j - \mu_t)(a_j - \mu_t)^T$
end for
Output: μ_T

D. Supplemental experiments

D.1. Accuracy of Deviation Model

We evaluate the performance of the deviation model by comparing its output with the actual deviation (i.e., $\Delta^{\pi^{(s)}}$) in the target and source environment states. We compare the performance between linear and nonlinear deviation models. We include linear models as they are convenient if we want to use optimal control algorithms. The nonlinear model is the same model we use for our PADA-DM algorithm, which has two 128-neuron layers with ReLU (Nair & Hinton, 2010) nonlinearity. Both of the deviation models are tested on the same initial state distribution after training on 80k samples. In 7(left), we plot the output of the deviation model against the ground truth deviation. We test on 50 trajectories and each data point refers to the average L2 state distance along one trajectory. In 7(right), we plot the ground truth deviation, and the outputs of the linear and nonlinear deviation models over time, on 10 test trajectories.

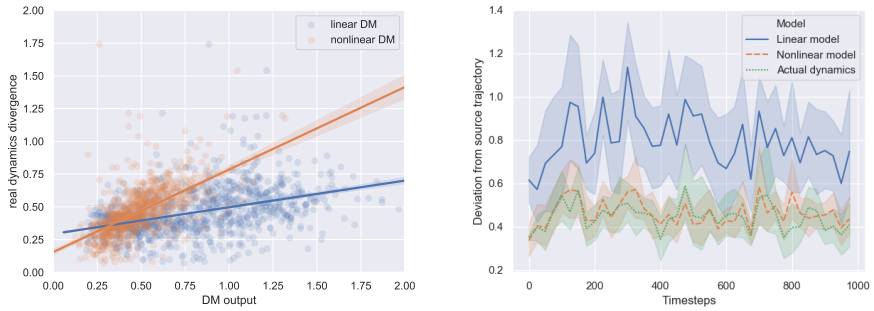


Figure 7. Comparing the performances of the linear and nonlinear deviation models. **left:** This plot depicts the correlation between the predicted deviation and the actual deviation. The nonlinear deviation model is more accurate since its slope is closer to 1. **right:** This plot shows the predicted and actual deviation over the course of 10 trajectories. Here again, the nonlinear model (orange) curve lies very close to the actual deviation curve (green).

D.2. Long-term learning curves

In this section we show a more comprehensive long-term learning curve in Fig. 8. Each task here corresponds to the task in Fig. 2. Note that here again the x-axis is in natural logarithm scale.

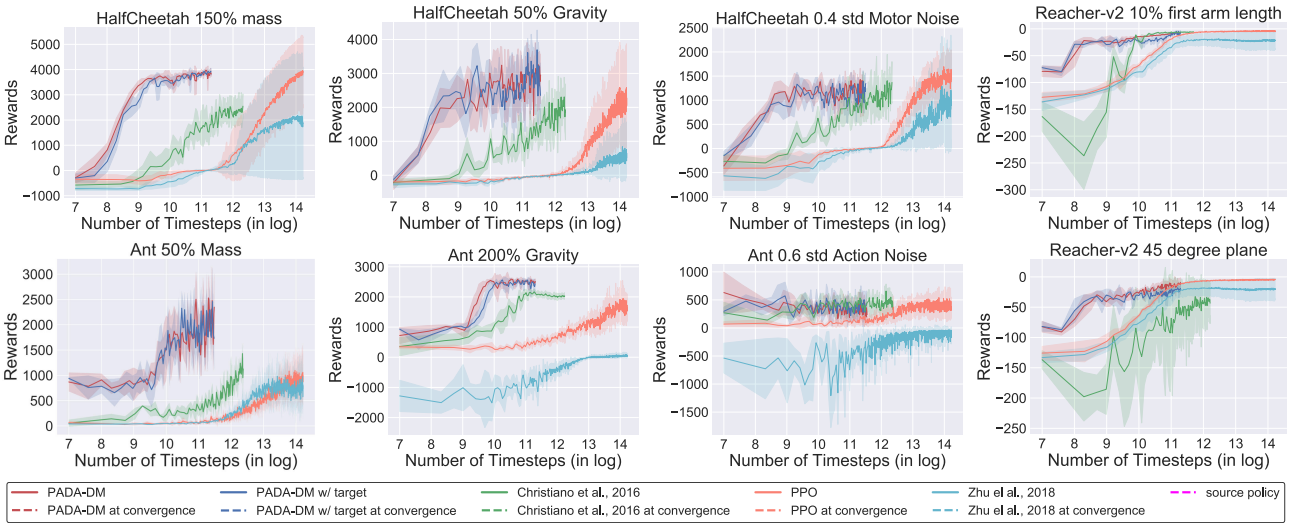


Figure 8. We plot the learning curves across 5 random seeds of our adaptation method (using PADA-DM and PADA-DM with target), and the baseline methods on a number of tasks including perturbation of the mass, gravity, motor noise for the locomotion environments (HalfCheetah and Ant), and the plane angle and arm lengths for the navigation environment (Reacher). The title of each plot corresponds to the perturbation in the target domain, e.g., HalfCheetah Mass 150% means in mass of the agent in the target domain is 150% of that in the source domain. The shaded area denotes 1 standard deviation range from the mean.

D.3. Comparing using true reward

To further verify the efficiency of our choice of reward (the deviations between two environments), we conduct an additional experiments where we use the ground truth reward for CEM. We fix all the hyperparameters and train an additional model to learn to ground truth reward. To ensure the fairness of the comparison, when we use the ground truth reward, we keep doing 1 step look-ahead during CEM. The results verify that the deviation serves as a better reward for conducting policy adaptation, where the ground truth reward leads to a local minimum that results in suboptimal performance.

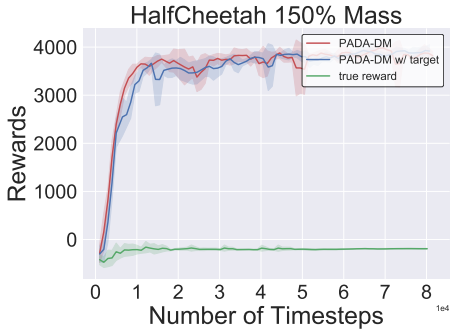


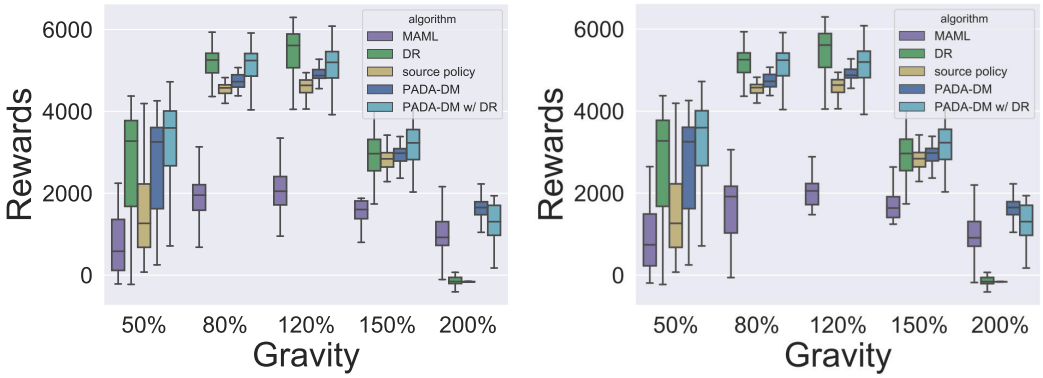
Figure 9. Comparing learning ground truth reward, learning deviations quickly adapts the policy in the target domain, where using ground truth reward may not necessarily leads to optimal performance in the target domain.

D.4. Additional Experiments for Meta-Learning MAML

In this section we conduct an additional experiment to Section 6.3. In section 6.3, we use 80k samples of adaptation for our method and MAML to conduct a fair comparison. However, using so many samples for adaptation contradicts MAML's claim of few-shot adaptation and we also observe that MAMLs test performance does not improve too much as we change the sample size in adaptation phase. Thus here we report the additional experiments to support this claim: we adopt the same experiment setting in Section 6.3, and this time we use 20k samples for MAML during adaptation. The test performance

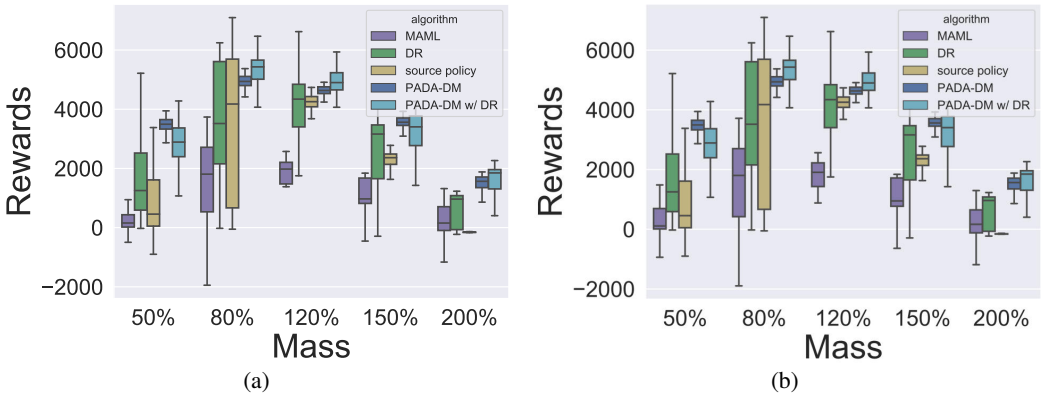
is recorded in Fig. 10(a) and Fig. 11(a). Comparing with the original performance (Fig. 10(b) and Fig. 11(b)), the test performance of MAML does not change that much as the number of adaptation samples decreases and our approach still outperforms MAML consistently.

In addition, we record the mean and the standard deviation of the test performance of each method to deliver a more direct comparison in Table 4 and Table 5. As we can see, our approach outperforms other baselines most of the time. When the perturbation is small (e.g., the 120% columns in both tables), DR also delivers very strong performances. However when perturbation is large (e.g., 200% columns in both tables), DR fails to adapt completely, which indicates that DR has troubles to adapt to out-of-distribution new environments.



(a) (b)

Figure 10. Ablation experiments using domain randomization and meta-learning. (a) MAML with 20k adaptation samples. (b) MAML with 80k adaptation samples. The boxplots show the median of the data while more statistics such as mean and standard deviation are shown in the following tables.



(a) (b)

Figure 11. Ablation experiments using domain randomization and meta-learning. (a) MAML with 20k adaptation samples. (b) MAML with 80k adaptation samples.

	Gravity Perturbation				
	50%	80%	120%	150%	200%
Source policy	1549.74 (1090.73)	4444.86 (637.77)	4592.18 (201.30)	2543.55 (916.16)	-156.51 (25.93)
Domain Randomization	2282.74 (1563.70)	4838.87 (1134.98)	**5236.46 (1179.85)**	2896.03 (554.00)	-43.97 (423.47)
PADA-DM	2694.78 (1166.88)	4739.32 (279.06)	4889.02 164.38)	2998.32 (266.75)	**1531.23 (400.96)**
PADA-DM w/ DR	**3230.29 (1280.54)**	**5036.59 (657.98)**	4934.04 (720.34)	**3200.73 (521.50)**	1431.53 (496.11)
MAML (20k)	854.24 (692.50)	1810.51 (663.72)	1895.86 (650.76)	1575.06 (653.09)	831.07 (717.78)
MAML (80k)	876.37 (711.98)	1778.67 (669.86)	1894.28 (644.55)	1568.60 (646.79)	823.13 (721.15)

Table 4. Mean and standard deviation (in the brackets) of the episodic rewards of each method in the target environment with perturbed gravity across 100 trajectories of 5 random seeds (500 trajectories in total).

	Mass Perturbation				
	50%	80%	120%	150%	200%
Source policy	921.13 (1192.43)	3343.05 (2317.32)	4166.10 (494.94)	2045.26 (665.30)	-149.92 (27.28)
Domain Randomization	1665.05 (1357.31)	3823.45 (1944.70)	3932.86 (1791.18)	2635.72 (1105.15)	944.50 (1134.32)
PADA-DM	**3271.52 (752.62)**	4914.67 (379.73)	4584.95 375.51)	**3557.25 (183.46)**	1398.88 (500.09)
PADA-DM w/ DR	2673.87 (1009.75)	**5348.98 (556.19)**	**4854.30 (591.50)**	3276.70 (874.54)	**1616.75 (490.53)**
MAML (20k)	854.24 (692.50)	1810.51 (663.72)	1895.86 (650.76)	1575.06 (653.09)	831.07 (717.78)
MAML (80k)	876.37 (711.98)	1778.67 (669.86)	1894.28 (644.55)	1568.60 (646.79)	823.13 (721.15)

Table 5. Mean and standard deviation (in the brackets) of the episodic rewards of each method in the target environment with perturbed mass across 100 trajectories of 5 random seeds (500 trajectories in total).