A Game Theoretic Framework for Model Based Reinforcement Learning

Aravind Rajeswaran [1 2] **Igor Mordatch** [1] **Vikash Kumar** [1]

Abstract

Designing stable and efficient algorithms for model-based reinforcement learning (MBRL) with function approximation has remained challenging despite growing interest in the field. To help expose the practical challenges in MBRL and simplify algorithm design from the lens of abstraction, we develop a new framework that casts MBRL as a game between: (1) a policy player, which attempts to maximize rewards under the learned model; (2) a model player, which attempts to fit the real-world data collected by the policy player. We show that a near-optimal policy for the environment can be obtained by finding an approximate equilibrium for aforementioned game, and we develop two families of algorithms to find the game equilibrium by drawing upon ideas from Stackelberg games. Experimental studies suggest that the proposed algorithms achieve state of the art sample efficiency, match the asymptotic performance of model-free policy gradient, and scale gracefully to high-dimensional tasks like dexterous hand manipulation. Project page: https://sites.google.com/view/mbrl-game.

1. Introduction

We study the problem of model-based reinforcement learning (MBRL) where a world model is learned from data to aid policy search. Model-based algorithms can incorporate historical off-policy data and generic priors like knowledge of physics, making them highly sample efficient. In addition, the learned models can also be re-purposed to solve new tasks. As a result, there has been a recent surge of interest in MBRL. However, a clear algorithmic framework to understand MBRL and unify insights from recent works has been lacking. To bridge this gap, and to facilitate the design of stable and efficient algorithms, we develop a new framework that casts MBRL as a two-player game.

Classical frameworks for MBRL, adaptive control (Åström & Murray, 2004), and dynamic programming (Puterman, 1994), are often confined to simple linear models or tabular representations. They also rely on building global models through ideas like persistent excitation (Narendra & Annaswamy, 1987) or tabular generative models (Kearns & Singh, 1998a). Such settings and assumptions are often limiting for modern applications. To obtain a globally accurate model, we need the ability to collect data from all parts of the state space (Agarwal et al., 2019b), which is often impossible. Furthermore, learning globally accurate models may be unnecessary, unsafe, and inefficient. For example, to make an autonomous car drive on the road, we should not require accurate models in situations where it tumbles and crashes in different ways. This motivates a class of *incremental* methods for MBRL that interleave policy and model learning to gradually construct and refine models in the task-relevant parts of the state space. This is in sharp contrast to a two-stage approach of first building a model of the world, and subsequently planning in it.

A unifying framework for incremental MBRL can connect insights from different approaches and help simplify the algorithm design process from the lens of abstraction. As an example, *distribution* or *domain shift* is known to be a major challenge for incremental MBRL. When improving the policy using the learned model, the policy will attempt to shift the distribution over visited states. The learned model may be inaccurate for this modified distribution, resulting in a greatly biased policy update. A variety of approaches have been developed to mitigate this issue. One class of approaches (Levine & Abbeel, 2014; Sun et al., 2018a; Kakade & Langford, 2002), inspired by trust region methods, make conservative changes to the policy to constrain the distribution between successive iterates. In sharp contrast, an alternate set of approaches do not constrain the policy updates in any way, but instead rely on data aggregation to mitigate distribution shift (Ross & Bagnell, 2012; Chua et al., 2018; Nagabandi et al., 2019). Our game-theoretic framework for MBRL reveals that these two seemingly disparate approaches are essentially dual approaches to solve the same game.

[1] Google Brain, Mountain View, USA. [2] University of Washington, Seattle, USA. Work performed at Google Brain.. Correspondence to: Aravind Rajeswaran <aravraj@cs.washington.edu>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020. Copyright 2020 by the author(s).

Our Contributions:

1. We develop a novel framework that casts MBRL as a game between: (a) a *policy player*, which maximizes rewards in the learned model; and (b) a *model player*, which minimizes prediction error of data collected by policy player. Theoretically, we establish that at equilibrium, the policy is near-optimal for the environment.
2. Developing learning algorithms for general continuous games is well known to be challenging. To develop stable and convergent algorithms, we setup a *Stackelberg game* (von Stackelberg, 1934) between the two players, which can be solved efficiently through (approximate) bi-level optimization.
3. Stackelberg games are asymmetric games where players make decisions in a pre-specified order. The leader plays first and subsequently the follower. Due to the asymmetric nature, the MBRL game can take two forms depending on choice of leader player. This gives rise to two natural families of algorithms that have complementary strengths. Together, they unify and generalize many prior MBRL algorithms.
4. Experimentally, we show that our algorithms outperform prior model-based and model-free algorithms in sample efficiency; match the asymptotic performance of model-free policy gradient algorithms; and scale gracefully to high-dimensional tasks like dexterous manipulation.

2. Background and Notations

We treat the environment as an infinite horizon MDP characterized by: $\boldsymbol{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{R}, P, \gamma, \rho\}$. Per usual notation, $\mathcal{S} \subseteq \mathbb{R}^n$ and $\mathcal{A} \subseteq \mathbb{R}^m$ represent the continuous state and action spaces. The transition dynamics is described by $s' \sim P(\cdot|s,a)$. $\mathcal{R} : \mathcal{S} \to [0, R_{\max}]$, $\gamma \in [0,1)$, and ρ represents the reward, discount, and initial state distribution respectively. Policy is a mapping from states to a probability distribution over actions, i.e. $\boldsymbol{\pi} : \mathcal{S} \to P(\mathcal{A})$, and in practice we typically consider parameterized policies. The goal is to optimize the objective:

$$\max_{\boldsymbol{\pi}} \; J(\boldsymbol{\pi}, \boldsymbol{M}) := \mathbb{E}_{\boldsymbol{M},\boldsymbol{\pi}} \left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t) \right] \qquad (1)$$

Model-free methods solve this optimization by directly estimating a gradient using collected samples or through value functions. Model-based methods, in contrast, construct an explicit world model to aid policy optimization.

2.1. Model-Based Reinforcement Learning

We represent the world model with another tuple: $\widehat{\boldsymbol{M}} = \{\mathcal{S}, \mathcal{A}, \mathcal{R}, \widehat{P}, \gamma, \rho\}$. The model has the same state-action space, reward function, discount, and initial state distribution. We parameterize the transition dynamics of the model $\widehat{P}$ (as a neural network) and learn the parameters so that it approximates the environment transition dynamics, i.e. $\widehat{P} \approx P$. For simplicity, we assume that the reward function and initial state distribution are known. This is a benign assumption for many applications in control, robotics, and operations research. If required, these quantities can also be learned from data, and are typically easier to learn than $\widehat{P}$. Enormous quantities of experience can be cheaply generated by simulating the model, without interacting with the world, and can be used for policy optimization. Thus, model-based methods tend to be sample efficient.

Idealized Global Model Setting To motivate challenges in MBRL, we first consider the idealized setting of an approximate *global* model. This corresponds to the case where $\widehat{\boldsymbol{M}}$ is sufficiently expressive and approximates $\boldsymbol{M}$ everywhere. Lemma 1 relates the performance of a policy in the model and environment.

Lemma 1. *(Simulation Lemma) Suppose* $\widehat{\boldsymbol{M}}$ *is such that* $D_{TV}\left(P(\cdot|s,a), \widehat{P}(\cdot|s,a)\right) \le \epsilon_{\boldsymbol{M}} \; \forall (s,a)$. *Then, for any policy* $\boldsymbol{\pi}$, *we have*

$$\left| J(\boldsymbol{\pi}, \boldsymbol{M}) - J(\boldsymbol{\pi}, \widehat{\boldsymbol{M}}) \right| \le O\left(\frac{\epsilon_{\boldsymbol{M}}}{(1-\gamma)^2} \right) \;\; \forall \boldsymbol{\pi}. \qquad (2)$$

The proof is provided in the appendix. Since Lemma 1 provides a uniform bound applicable to all policies, we can expect good performance in the environment by optimizing the policy in the model, i.e. $\max_{\boldsymbol{\pi}} J(\boldsymbol{\pi}, \widehat{\boldsymbol{M}})$.

Beyond global models A global modeling approach as above is often impractical. To obtain a globally accurate model, we need the ability to collect data from all parts of the state space (Agarwal et al., 2019b; Pathak et al., 2017), which can be difficult. More importantly, learning globally accurate models may be unnecessary, unsafe, and inefficient. For example, to make a robot walk, we should not require accurate models in situations where it falls and crashes in different ways. This motivates the need for *incremental* MBRL, where models are gradually constructed and refined in the task-relevant parts of the state space. To formalize this intuition, we consider the below notion of model quality.

Definition 1. *(Model approximation loss) Given* $\widehat{\boldsymbol{M}}$ *and distribution* $\mu(s,a)$, *the model approximation loss is*

$$\ell(\widehat{\boldsymbol{M}}, \mu) = \mathbb{E}_{(s,a)\sim\mu} \left[D_{KL}\big(P(\cdot|s,a), \widehat{P}(\cdot|s,a)\big) \right]. \qquad (3)$$

We use D_{KL} to refer to the KL divergence which can be optimized using samples from $\boldsymbol{M}$, and is closely related to D_{TV} through Pinsker's inequality. In the case of isotropic Gaussian distributions, as typically considered in continuous control applications, D_{KL} reduces to the familiar ℓ_2 loss. Importantly, the loss is intimately tied to the sampling distribution μ. In general, models that are accurate in some parts of the state space need not generalize/transfer to other parts. As a result, a more conservative policy learning procedure is required, in contrast to the global model case.

3. Model Based RL as a Two Player Game

In order to capture the interactions between model and policy learning, we formulate MBRL as the following two-player general sum game (ref. as MBRL game)

$$\overbrace{\max_{\boldsymbol{\pi}} \; J(\boldsymbol{\pi}, \widehat{\boldsymbol{M}})}^{\text{policy}-\text{player}} \;,\; \overbrace{\min_{\widehat{\boldsymbol{M}}} \; \ell(\widehat{\boldsymbol{M}}, \mu_{\boldsymbol{M}}^{\boldsymbol{\pi}})}^{\text{model}-\text{player}} \tag{4}$$

We use $\mu_{\boldsymbol{M}}^{\boldsymbol{\pi}} = \frac{1}{T}\sum_{t=0}^{T} P(s_t = s, a_t = a)$ to denote the average state visitation distribution. The policy player maximizes performance in the learned model, while the model player minimizes prediction error under policy player's induced state distribution. This is a game since the objective of each player depends on the parameters of both players.

The above formulation separates MBRL into the constituent components of policy learning (planning) and generative model learning. At the same time, it exposes that the two components are closely intertwined and must be considered together in order to succeed in MBRL. We discuss algorithms for solving the game in Section 4, and first focus on the equilibrium properties of the MBRL game. Our results establish that at (approximate) Nash equilibrium of the MBRL game: (1) the model can accurately simulate and predict the performance of the policy; (2) the policy is near-optimal.

Theorem 1. *(Global perf. of equilibrium pair; informal) Suppose we have a pair of policy and model, $(\boldsymbol{\pi}, \widehat{\boldsymbol{M}})$, such that simultaneously*

$$\ell(\widehat{\boldsymbol{M}}, \mu_{\boldsymbol{M}}^{\boldsymbol{\pi}}) \leq \epsilon_{\boldsymbol{M}} \;\; \textit{and} \;\; J(\boldsymbol{\pi}, \widehat{\boldsymbol{M}}) \geq J(\boldsymbol{\pi}', \widehat{\boldsymbol{M}}) - \epsilon_{\boldsymbol{\pi}} \;\forall \boldsymbol{\pi}'.$$

For an optimal policy $\boldsymbol{\pi}^*$, we have

$$J(\boldsymbol{\pi}^*, \boldsymbol{M}) - J(\boldsymbol{\pi}, \boldsymbol{M}) \leq$$
$$O\left(\epsilon_{\boldsymbol{\pi}} + \frac{\sqrt{\epsilon_{\boldsymbol{M}}}}{(1-\gamma)^2} + \frac{1}{1-\gamma} D_{TV}\left(\mu_{\boldsymbol{M}}^{\boldsymbol{\pi}^*}, \mu_{\widehat{\boldsymbol{M}}}^{\boldsymbol{\pi}^*}\right)\right). \tag{5}$$

Proof. A more formal version of the theorem and proof is provided in appendix A. □

We now make some remarks about the above result.

1. The first two terms are related to sub-optimality in policy optimization (planning) and model learning, and can be made small with more compute and data, assuming sufficient capacity.
2. There may be multiple Nash equilibrium for the MBRL game, and the third *domain adaptation* or *transfer learning* term in the bound captures the quality of an equilibrium. It captures the idea that model is trained under distribution of $\boldsymbol{\pi}$, i.e. $\mu_{\boldsymbol{M}}^{\boldsymbol{\pi}}$, but evaluated under the distribution of $\boldsymbol{\pi}^*$, i.e. $\mu_{\boldsymbol{M}}^{\boldsymbol{\pi}^*}$. If the model can accurately simulate $\boldsymbol{\pi}^*$, we can expect to find it in the planning phase, since it would obtain high rewards. This domain adaptation term is a consequence of the exploration problem, and is unavoidable if we desire globally optimal policies. Indeed, even purely model-free algorithms suffer from an analogous divergence term (Kakade & Langford, 2002; Munos & Szepesvári, 2008). However, Theorem 1 also applies to locally optimal policies (see appendix A) for which we may expect better model transfer.
3. The domain adaptation term can be minimized by considering a wide initial state distribution (Kakade & Langford, 2002; Rajeswaran et al., 2017). This ensures the learned model is more broadly accurate. However, in some applications, the initial state distribution may not be under our control. In such a case, we may draw upon advances in domain adaptation (Ben-David et al., 2006; Sun et al., 2015) to learn state-action representations better suited for transfer across different policies.

4. Algorithms

So far, we have established how MBRL can be viewed as a game that couples policy and model learning. We now turn to developing algorithms for solving the game. Unlike common deep learning settings (e.g. supervised learning), there are no standard workhorses for continuous games. Direct extensions of optimization workhorses (e.g. SGD) are unstable for games due to non-stationarity (Wang et al., 2019b; Fiez et al., 2019). We first review some of these extensions before presenting our final algorithms.

4.1. Independent simultaneous learners

We first consider a class of algorithms where each player individually optimize their own objectives using gradient descent. Thus, each player treats the setting as stochastic optimization unaware of potential drifts in their objectives due to the two-player nature. These algorithms are sometimes called independent learners, simultaneous learners, or naive learners (Wang et al., 2019b; Foerster et al., 2017).

Gradient Descent Ascent (GDA) In GDA, each player performs an improvement step holding the parameters of the other player fixed. The resulting updates are given below.

$$\boldsymbol{\pi}_{k+1} = \boldsymbol{\pi}_k + \alpha_k \nabla_{\boldsymbol{\pi}} J(\boldsymbol{\pi}_k, \widehat{\boldsymbol{M}}_k) \tag{6}$$
$$\widehat{\boldsymbol{M}}_{k+1} = \widehat{\boldsymbol{M}}_k - \beta_k \nabla_{\widehat{\boldsymbol{M}}} \ell(\widehat{\boldsymbol{M}}_k, \mu_{\boldsymbol{M}}^{\boldsymbol{\pi}_k}) \tag{7}$$

Both the players update their parameters simultaneously from iteration k to $k+1$. For simplicity, we consider standard gradient descent, which can be equivalently replaced with momentum, Adam, natural gradient etc. Variants of GDA have been used to solve min-max games arising in deep learning such as GANs. However, for certain problems, it can exhibit poor convergence and require very small learning rates (Schäfer & Anandkumar, 2019) or domain-specific heuristics. Furthermore, it makes sub-optimal use of data, since it is desirable to take multiple policy improvement steps to fully reap the benefits of model learning.

Best Response (BR) The BR algorithm aims to mititage the above drawback, where each player computes the best response while fixing the parameters of other players. The best response can be approximated in practice using a large number of gradient steps.

$$\boldsymbol{\pi}_{k+1} = \arg\max_{\boldsymbol{\pi}} \; J(\boldsymbol{\pi}, \widehat{\boldsymbol{M}}_k) \tag{8}$$

$$\widehat{\boldsymbol{M}}_{k+1} = \arg\min_{\widehat{\boldsymbol{M}}} \; \ell(\widehat{\boldsymbol{M}}, \mu_{\boldsymbol{M}}^{\boldsymbol{\pi}_k}) \tag{9}$$

Again, both players simultaneously update their parameters. It is known from a large body of work in online learning that aggressive changes can destabilize learning in non-stationary settings (Cesa-Bianchi & Lugosi, 2006). Large changes to the policy can dramatically alter the sampling distribution, which renders the model incompetent. Similarly, large changes in the model can bias policy learning. In Section 5 we experimentally study the performance of GDA and BR on a suite of control tasks and verify that they inefficient (slow) or unstable.

4.2. Stackelberg formulation and algorithms

To achieve stable and sample efficient learning, we require algorithms that take the game structure into account. While good workhorses are lacking for general games, Stackelberg games (von Stackelberg, 1934) are an exception. They are asymmetric games where we impose a specific playing order and are a generalization of min-max games. We cast the MBRL game in the Stackelberg form, and derive gradient based algorithms to solve the resulting game.

First, we briefly review continuous Stackelberg games. Consider a two player game with players A and B. Let $\boldsymbol{\theta}_A, \boldsymbol{\theta}_B$ be their parameters, and $\mathcal{L}_A(\boldsymbol{\theta}_A, \boldsymbol{\theta}_B)$, $\mathcal{L}_B(\boldsymbol{\theta}_A, \boldsymbol{\theta}_B)$ be their losses. Each player would like their losses minimized. With player A as the leader, the Stackelberg game corresponds to the following nested optimization:

$$\begin{aligned} &\min_{\boldsymbol{\theta}_A} \mathcal{L}_A\big(\boldsymbol{\theta}_A, \boldsymbol{\theta}_B^*(\boldsymbol{\theta}_A)\big) \\ &\text{subject to } \; \boldsymbol{\theta}_B^*(\boldsymbol{\theta}_A) = \arg\min_{\tilde{\boldsymbol{\theta}}} \; \mathcal{L}_B(\boldsymbol{\theta}_A, \tilde{\boldsymbol{\theta}}) \end{aligned} \tag{10}$$

Since the follower chooses the best response, the follower's parameters are implicitly a function of the leader's parameters. The leader is aware of this, and can utilize this information when updating its parameters. The Stackelberg formulation has a number of appealing properties.

- **Algorithm design based on optimization:** From the leader's viewpoint, the Stackelberg formulation transforms a game with complex interactions into a more familiar albeit complex bi-level optimization, for which we have gradient based workhorses (Colson et al., 2007).
- **Notion of stability and progress:** In general games, there exists no single function that can be used to check if an iterative algorithm makes progress towards the equilibrium. This makes algorithm design and diagnosis difficult. By reducing the game to an optimization, the leader's loss $\mathcal{L}_A(\boldsymbol{\theta}_A, \boldsymbol{\theta}_B)$ can be used to track progress.

For simplicity of exposition, we assume that the best-response is unique for the follower. We later remark on the possibility of multiple minimizers. To solve the nested optimization, it suffices to focus on $\boldsymbol{\theta}_A$ since the follower parameters $\boldsymbol{\theta}_B^*(\boldsymbol{\theta}_A)$ are implicitly a function of $\boldsymbol{\theta}_A$. We can iteratively optimize $\boldsymbol{\theta}_A$ as: $\boldsymbol{\theta}_A \leftarrow \boldsymbol{\theta}_A - \alpha_A \left(\mathrm{d}\mathcal{L}_A(\boldsymbol{\theta}_A, \boldsymbol{\theta}_B^*(\boldsymbol{\theta}_A))/\mathrm{d}\boldsymbol{\theta}_A\right)$, where the gradient is described in Eq. 11. The key to solving a Stackelberg game is to make the follower learn very quickly to approximate the best response, while the leader learns slowly.

$$\begin{aligned} \frac{\mathrm{d}\mathcal{L}_A\left(\boldsymbol{\theta}_A, \boldsymbol{\theta}_B^*(\boldsymbol{\theta}_A)\right)}{\mathrm{d}\boldsymbol{\theta}_A} &= \frac{\mathrm{d}\boldsymbol{\theta}_B^*}{\mathrm{d}\boldsymbol{\theta}_A} \frac{\partial \mathcal{L}_A(\boldsymbol{\theta}_A, \boldsymbol{\theta}_B)}{\partial \boldsymbol{\theta}_B}\bigg|_{\boldsymbol{\theta}_B = \boldsymbol{\theta}_B^*} \\ &\quad + \frac{\partial \mathcal{L}_A(\boldsymbol{\theta}_A, \boldsymbol{\theta}_B)}{\partial \boldsymbol{\theta}_A}\bigg|_{\boldsymbol{\theta}_B = \boldsymbol{\theta}_B^*} \end{aligned} \tag{11}$$

The implicit Jacobian term $(\mathrm{d}\boldsymbol{\theta}_B^*/\mathrm{d}\boldsymbol{\theta}_A)$ can be obtained using the implicit function theorem (Krantz & Parks, 2002; Rajeswaran et al., 2019). Thus, in principle, we can compute the gradient with respect to the leader parameters and solve the nested optimization (to at least a local minimizer). To develop a practical algorithm based on these ideas, we use a few relaxations and approximations. First, we approximate the best response with multiple steps of an iterative optimization algorithm. Secondly, we drop the implicit Jacobian term and use a "first-order" approximation of the gradient. Such an approximation has proven effective in applications like meta-learning (Nichol et al., 2018), GANs (Heusel et al., 2017; Metz et al., 2017), and multiple timescale actor-critic methods (Konda & Borkar, 1999). Finally, since the Stackelberg game is asymmetric, we can cast the MBRL game in two forms based on which player we choose as the leader.

Policy As Leader (PAL): Choosing the policy player as leader results in the following optimization:

$$\max_{\boldsymbol{\pi}} \left\{ J(\boldsymbol{\pi}, \widehat{\boldsymbol{M}}^{\boldsymbol{\pi}}) \;\; s.t. \;\; \widehat{\boldsymbol{M}}^{\boldsymbol{\pi}} \in \arg\min_{\widehat{\boldsymbol{M}}} \; \ell(\widehat{\boldsymbol{M}}, \mu_{\boldsymbol{M}}^{\boldsymbol{\pi}}) \right\}.$$

We solve this nested optimization using the first order gradient approximation, resulting in updates:

$$\widehat{\boldsymbol{M}}_{k+1} \approx \arg\min_{\widehat{\boldsymbol{M}}} \ell(\widehat{\boldsymbol{M}}, \mu_{\boldsymbol{M}}^{\boldsymbol{\pi}_k}) \tag{12}$$

$$\boldsymbol{\pi}_{k+1} = \boldsymbol{\pi}_k + \alpha_k \nabla_{\boldsymbol{\pi}} J(\boldsymbol{\pi}, \widehat{\boldsymbol{M}}_{k+1}) \tag{13}$$

We first aggressively improve the model to minimize the loss under current visitation distribution. Subsequently we take a conservative policy. The algorithmic template is described further in Algorithm 1. Note that the PAL updates are different from GDA even if a single gradient step is used to approximate the $\arg\min$. In PAL, the model is first updated using the current visitation distribution from $\widehat{\boldsymbol{M}}_k$ to $\widehat{\boldsymbol{M}}_{k+1}$. The policy subsequently uses $\widehat{\boldsymbol{M}}_{k+1}$ for improvement. In contrast, GDA uses $\widehat{\boldsymbol{M}}_k$ for improving the policy. Finally, suppose we find an approximate solution to the PAL optimization such that $J(\boldsymbol{\pi}, \widehat{\boldsymbol{M}}^{\boldsymbol{\pi}}) \geq \sup_{\tilde{\boldsymbol{\pi}}} J(\tilde{\boldsymbol{\pi}}, \widehat{\boldsymbol{M}}^{\tilde{\boldsymbol{\pi}}}) - \epsilon_{\boldsymbol{\pi}}$. Since the model is optimal for the policy by constriction, we inherit the guarantees of Theorem 1.

Algorithm 1 Policy as Leader (PAL) meta-algorithm

```
Initialize: policy $\boldsymbol{\pi}_0$, model $\widehat{\boldsymbol{M}}_0$, data buffer $\mathcal{D} = \{\}$
for $k = 0, 1, 2, \ldots$ forever do
    Collect data $\mathcal{D}_k$ by executing $\boldsymbol{\pi}_k$ in the environment
    Build local (policy-specific) dynamics model: $\widehat{\boldsymbol{M}}_{k+1} = \arg\min \ \ell(\widehat{\boldsymbol{M}}, \mathcal{D}_k)$
    Improve policy: $\boldsymbol{\pi}_{k+1} = \boldsymbol{\pi}_k + \alpha \nabla_{\boldsymbol{\pi}} J(\boldsymbol{\pi}_k, \widehat{\boldsymbol{M}}_{k+1})$ with a conservative algorithm like NPG or TRPO.
end for
```

Model as Leader (MAL): Conversely, choosing model as the leader results in the optimization

$$\min_{\widehat{\boldsymbol{M}}} \left\{ \ell(\widehat{\boldsymbol{M}}, \mu_{\widehat{\boldsymbol{M}}}^{\boldsymbol{\pi}_{\widehat{\boldsymbol{M}}}}) \ \ s.t. \ \ \boldsymbol{\pi}_{\widehat{\boldsymbol{M}}} \in \arg\max_{\boldsymbol{\pi}} J(\boldsymbol{\pi}, \widehat{\boldsymbol{M}}) \right\}. \tag{14}$$

Similar to the PAL formulation, using first order approximation to the bi-level gradient results in:

$$\boldsymbol{\pi}_{k+1} \approx \arg\max_{\boldsymbol{\pi}} J(\boldsymbol{\pi}, \widehat{\boldsymbol{M}}_k) \tag{15}$$

$$\widehat{\boldsymbol{M}}_{k+1} = \widehat{\boldsymbol{M}}_k - \beta_k \nabla_{\widehat{\boldsymbol{M}}} \ell(\widehat{\boldsymbol{M}}, \mu_{\widehat{\boldsymbol{M}}}^{\boldsymbol{\pi}_{k+1}}) \tag{16}$$

We first optimize a policy for the current model. Subsequently, we conservatively improve the model using the data collected with the optimized policy. In practice, instead of a single conservative model improvement step, we aggregate all the historical data and perform a few epochs of training. This has an effect similar to conservative model improvement in a follow the regularized leader interpretation (Shalev-Shwartz, 2012; Ross & Bagnell, 2012; McMahan, 2011). The algorithmic template is described in Algorithm 2. Similar to the PAL case, we again inherit the guarantees from Theorem 1.

Algorithm 2 Model as Leader (MAL) meta-algorithm

```
Initialize: policy $\boldsymbol{\pi}_0$, model $\widehat{\boldsymbol{M}}_0$, data buffer $\mathcal{D} = \{\}$
for $k = 0, 1, 2, \ldots$ forever do
    Optimize $\boldsymbol{\pi}_{k+1} = \arg\max_{\boldsymbol{\pi}} J(\boldsymbol{\pi}, \widehat{\boldsymbol{M}}_k)$ using any algorithm (RL, MPC, planning etc.)
    Collect environment data $\mathcal{D}_{k+1}$ using $\boldsymbol{\pi}_{k+1}$
    Improve model $\widehat{\boldsymbol{M}}_{k+1} = \widehat{\boldsymbol{M}}_k - \beta \nabla_{\widehat{\boldsymbol{M}}} \ell(\widehat{\boldsymbol{M}}, \mathcal{D}_{k+1})$ using any conservative algorithm like mirror descent, data aggregation etc.
end for
```

On distributionally robust models and policies Finally, we illustrate how the Stackelberg framework is consistent with commonly used robustification heuristics. We now consider the case where there could be multiple best responses to the leader eq. 10. For instance, in PAL, there could be multiple models that achieve low error for the policy. Similarly, in MAL, there could be multiple policies that achieve high rewards for the specified model. In such cases, the standard notion of Stackelberg equilibrium is to optimize under the worst case realization (Fiez et al., 2019), which results in:

$$\begin{gathered} \min_{\boldsymbol{\theta}_A} \max_{\boldsymbol{\theta}_B \in R(\boldsymbol{\theta}_A)} \ \mathcal{L}_A(\boldsymbol{\theta}_A, \boldsymbol{\theta}_B), \ \text{ where} \\ R(\boldsymbol{\theta}_A) \overset{\text{def}}{=} \left\{ \tilde{\boldsymbol{\theta}} \mid \mathcal{L}_B(\boldsymbol{\theta}_A, \tilde{\boldsymbol{\theta}}) \leq \mathcal{L}_B(\boldsymbol{\theta}_A, \boldsymbol{\theta}_B) \ \forall \boldsymbol{\theta}_B \right\}. \end{gathered} \tag{17}$$

In PAL, model ensemble approaches correspond to approximating the best response set with a finite collection (ensemble) of models. Algorithms inspired by robust or risk-averse control (Zhou et al., 1996; Garcia & Fernández, 2015; Rajeswaran et al., 2016) explicitly improve against the adversarial choice in the ensemble, consistent with the Stackelberg setting. Similarly, in the MAL formulation, entropy regularization (Haarnoja et al., 2018; Hazan et al., 2018) and disagreement based reward bonuses (Pathak et al., 2017; 2019) lead to adversarial best response by encouraging the policy to visit parts of the state space where the model is likely to be inaccurate. Our Stackelberg formulation provides a principled foundation for these important components, which have thus far been viewed as heuristics.

5. Experiments

In our experiemental evaluation, we aim to primarily answer the following questions:

1. Do independent learning algorithms (GDA and BR) learn slowly or suffer from instabilities?
2. Do the Stackelberg-style algorithms (PAL and MAL) enable stable and sample efficient learning?
3. Do MAL and PAL exhibit different learning characteristics and strengths? Can we characterize the situations where one is more preferable than the other?

Task Suite We study the behavior of algorithms on a suite of continuous control tasks consisting of: `DClaw-Turn`, `DKitty-Orient`, `7DOF-Reacher`, and `InHand-Pen`. The tasks are illustrated in Figure 2

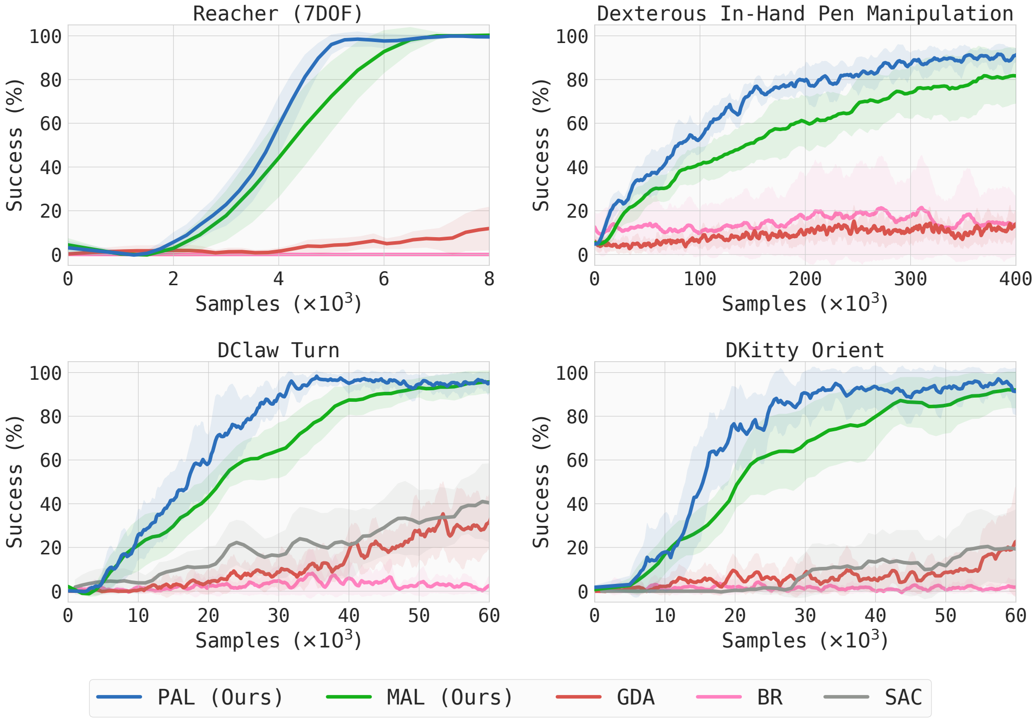


Figure 1. Comparison of the learning algorithms. We report results based on 5 random seeds, with solid lines representing the average performance, and shaded regions indicate standard deviation across seeds. PAL and MAL exhibit stable and sample efficient learning. GDA learns very slowly due to sub-optimal use of data. BR does not lead to stable learning due to aggressive changes to both policy and model. For the ROBEL tasks, as a point of comparison, we also include results of SAC a state of the art model-free algorithm.

and further details are provided in Appendix B.1. The DClaw and DKitty tasks use physically accurate models of robots (Zhu et al., 2018; Ahn et al., 2019). The Reacher task is a representative whole arm manipulation task, while the in-hand dexterous manipulation task (Rajeswaran et al., 2018) serves as a representative high-dimensional control task. In addition, we also present results with our algorithms in the OpenAI gym tasks in Appendix B.2.

Algorithm Details For all the algorithms of interest (GDA, BR, PAL, MAL), we represent the policy as well as the dynamics model with fully connected neural networks. We instantiate all of these algorithm families with model-based natural policy gradient. Details about the implementation are provided in Appendix B. We use ensembles of dynamics models and entropy regularization to encourage robustness.

Comparison of learning algorithms We first study the performance of Stackelberg-style algorithms (PAL, MAL) and compare against the performance of independent algorithms (GDA and BR). Our results, summarized in Figure 1, suggest that PAL and MAL can learn all the tasks efficiently. We observe near monotonic improvement, suggesting that the Stackelberg formulation enables stable learning. We also observe that PAL learns faster than MAL for the tasks we study. While GDA eventually achieves near-100% success rate, it leads to considerably slower learning. As outlined in

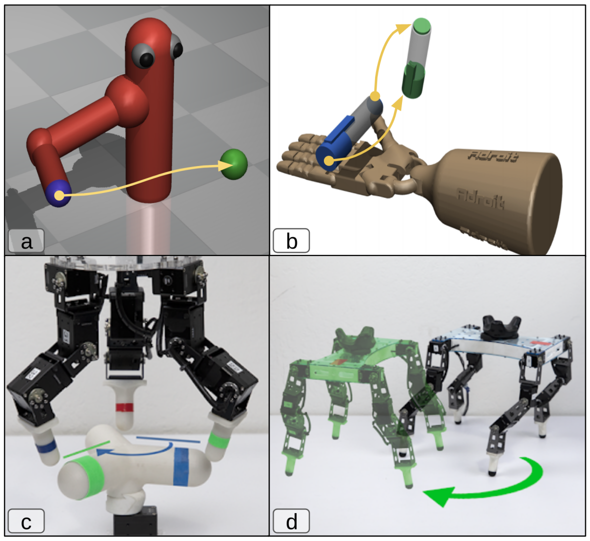


Figure 2. (a) Reacher task with a 7DOF arm. (b) In-hand manipulation task with a 24DOF dexterous hand. (c) DClaw-Turn task with a 3 fingered "claw". (d) DKitty-Orient task with a quadrupedal robot. In all the tasks, the desired goal configurations are randomized every episode, which forces the RL agent to learn generalizable policies. We measure and use success rate for our experimental evaluations.

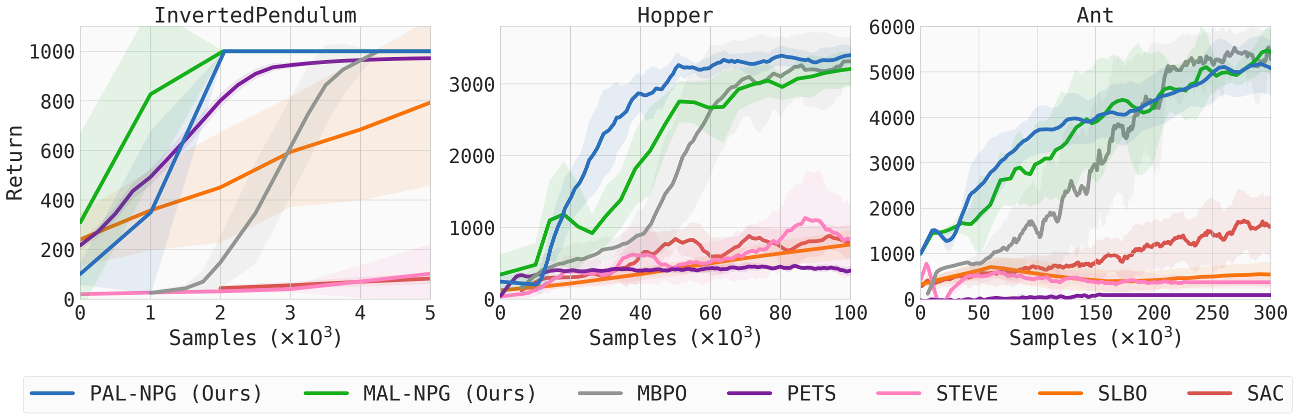


Figure 3. Comparison of results on the OpenAI gym benchmark tasks. Results for the baselines are reproduced from Janner et al. (2019). Solid lines are the average performance curves over 5 random seeds, while shaded region represents the standard deviation over these 5 runs. We observe that PAL and MAL show near-monotonic improvement, and substantially outperform the baselines.

Section 4, this is likely due to conservative nature of updates for both the policy and the model. Furthermore, the performance fluctuates rapidly during course of learning, since it does not correspond to stable optimization of any objective. Finally, we observe that BR is unable to make consistent progress. As suggested earlier in Section 4, BR makes rapid changes to both model and policy which exacerbates the challenge of distribution mismatch.

As a point of comparison, we also plot results of SAC (Haarnoja et al., 2018), a leading model-free algorithm for the ROBEL tasks (results taken from Ahn et al. (2019)). Although SAC is able to solve these tasks, its sample efficiency is comparable to GDA, and substantially slower than PAL and MAL. To compare against other model-based algorithms, we turn to published results from prior work on OpenAI gym tasks. In Figure 3, we show that PAL and MAL significantly outperforms prior algorithms. In particular, PAL and MAL are 10 times as efficient as other model-based and model-free methods. PAL is also twice as efficient as MBPO (Janner et al., 2019), a state of the art hybrid model-based and model-free algorithm. Further details about this comparison are provided in Appendix B.2.

Overall our results indicate that PAL and MAL: (a) are substantially more sample efficient than prior model-based and model-free algorithms; (b) achieve the asymptotic performance of their model-free counterparts; (c) can scale to high-dimensional tasks with complex dynamics like dexterous manipulation; (d) can scale to tasks requiring extended rollout horizons (e.g. the OpenAI gym tasks).

Choosing between PAL and MAL Finally, we turn to studying relative strengths of PAL and MAL. For this, we consider two variations of the 7DOF reacher task (from Figure 2) corresponding to environment perturbations at an intermediate point of training. In the first case, we perturb the dynamics by changing the length of the forearm. In the second case, halfway through the training, we change the goal distribution to a different region of 3D space. Training curves are presented in Figure 4. Note that there is a performance drop at the time of introducing the perturbation.

For the first case of dynamics perturbation, we observe that PAL recovers faster. Since PAL learns the model aggressively using recent data, it can forget old inconsistent data and improve the policy using an accurate model. In contrast, MAL adapts the model conservatively, taking longer to forget old inconsistent data, ultimately biasing and slowing the policy learning. In the second experiment, the dynamics is stationary but the goal distribution changes midway. Note that the policy does not generalize zero-shot to the new goal distribution, and requires additional learning or fine-tuning. Since MAL learns a more broadly accurate model, it quickly adapts to the new goal distribution. In contrast, PAL conservatively changes the policy and takes longer to adapt to the new goal distribution.

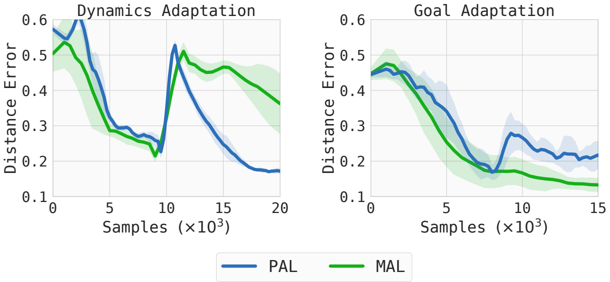


Figure 4. PAL vs MAL in non-stationary learning environments. Y axis is the distance between end effector and goal, averaged over the trajectory (lower is better). The left plot corresponds to the case where the dynamics of M is changed after 10^4 samples, while the right plot corresponds to the case where we change the goal distribution after 8×10^3 samples. We observe that PAL recovers quickly from dynamics perturbations, while MAL recovers quickly from goal perturbations.

Thus, in summary, we find that PAL is better suited for situations where the dynamics of the world can drift over time. In contrast, MAL is better suited for situations where the task or goal distribution can change over time, and related settings like multi-task learning.

6. Related Work

MBRL and the closely related fields of adaptive control and system identification have a long and rich history (see Åström & Murray (2004); Ljung (1987) for overview). Early works in MBRL primarily focused on tabular reinforcement learning in a known *generative model* setting (Kearns & Singh, 1998a; Agarwal et al., 2019a). However, this setting assumes access to a highly exploratory policy to collect data, which is often not available in practice. Subsequent works like E3 (Kearns & Singh, 1998b) and R-MAX (Brafman & Tennenholtz, 2001) attempt to lift this limitation, but rely heavily on tabular representations which are inadequate for modern applications like robotics. Coupled with advances in deep learning, there has been a surge of interest in incremental MBRL algorithms with rich function approximation. They generally fall into two sets of approaches, as we outline below.

The first set of approaches are largely inspired by trust region methods, and are similar to the PAL family from our work. A highly accurate "local" model is constructed around the visitation distribution of the current policy, which is subsequently used to conservatively improve the policy. The trust region is intended to ensure that the model is accurate for all policies within it, thereby enabling monotonic performance improvement. GPS (Levine & Abbeel, 2014; Mordatch & Todorov, 2014), DPI (Sun et al., 2018b), and related approaches (Kumar et al., 2016) learn a time varying linear model and perform a KL-constrained policy improvement step. Such a model representation is convenient for an iLQG based policy update (Todorov & Li, 2005), but might be restrictive for complex dynamics beyond trajectory-centric RL. To remove these limitations, recent works have started to consider neural networks to represent both policy and the dynamics model. However, somewhat surprisingly, a clean version from the PAL family has not been studied with neural network models. The motivations presented by Xu et al. (2018) and Kurutach et al. (2018) resemble PAL, however their practical implementations do not strongly enforce the conservative nature of the policy update.

An alternate set of MBRL approaches take a view similar to MAL. Models are updated conservatively through data aggregation, while policies are aggressively optimized. Ross & Bagnell (2012) explicitly studied the role of data aggregation in MBRL. They presented an agnostic online learning view of MBRL and showed that data aggregation can lead to a no-regret algorithm for learning the model, even with aggressive policy optimization. Subsequent works have used data augmentation and proposed additional components to enhance efficiency and stability, such as the use of model predictive control (Williams et al., 2017; Lowrey et al., 2019; Nagabandi et al., 2019), uncertainty quantification through Bayesian models (Deisenroth & Rasmussen, 2011), and ensembles of dynamics models (Rajeswaran et al., 2016; Chua et al., 2018; Nagabandi et al., 2019). We refer readers to Wang et al. (2019a) for overview of recent MBRL advances.

While specific instances of PAL and MAL have been studied in the past, an overarching framework around them has been lacking. Our descriptions of the PAL and MAL families generalize and unify core insights from prior work and simplify them from the lens of abstraction. Furthermore, the game theoretic formulation enables us to form a connection between the PAL and MAL frameworks. We also note that the PAL and MAL families have similarities to multiple timescale algorithms (Konda & Borkar, 1999; Konda & Tsitsiklis, 2004; Karmakar & Bhatnagar, 2015) studied for actor-critic temporal difference learning. These ideas have also been extended to study min-max games like GANs (Heusel et al., 2017). However, they have not been extended to study model-based RL.

We presented a model-based setting where the model is used to directly improve the policy through rollout based optimization. However, models can be utilized in other ways too. Dyna (Sutton, 1990) and MBPO (Janner et al., 2019) use a learned model to provide additional learning targets for an actor-critic algorithm through short-horizon synthetic trajectories. MBVE (Feinberg et al., 2018), STEVE (Buckman et al., 2018), and doubly-robust methods (Jiang & Li, 2015; Thomas & Brunskill, 2016; Farajtabar et al., 2018) use model-based rollouts to obtain more favorable bias-variance trade-offs for off-policy evaluation. Some of these works have noted that long horizon rollouts can exacerbate model bias. However, in our experiments, we were able to successfully perform rollouts of hundreds of steps. This is likely due to our practical implementation closely following the game theoretic algorithms designed explicitly to mitigate distribution shift and enable effective simulation. It is straightforward to extend PAL and MAL to a hybrid model-based and model-free algorithm, which is likely to provide further performance gains. Similarly, approaches that bootstrap from the model's predictions can improve multi-step simulation (Venkatraman et al., 2015; Bengio et al., 2015). We leave exploration of these directions for future work.

7. Summary and Conclusion

In this work, we developed a new framework for MBRL that casts it as a game between a policy player and a model player. We established that at equilibrium: (1) the model accurately simulates the policy and predicts its performance; (2) the policy is near-optimal. We derived sub-optimality bounds and made a connection to domain adaptation to characterize the equilibrium quality.

In order to solve the MBRL game, we constructed the Stackelberg version of the game. This has the advantage of: (1) effective gradient based workhorses to solve the Stackelberg optimization problem; (2) an effective objective function to track learning progress towards equilibrium. General continuous games possess neither of these characteristics. The Stackelberg game can take two forms based on which player we choose as the leader, resulting in two natural algorithm families, which we named PAL and MAL. Together they encompass, generalize, and unify a large collection of prior MBRL works. This greatly simplifies MBRL and particularly algorithm design from the lens of abstraction.

We developed practical versions of PAL and MAL using model-based natural policy gradient. We demonstrated stable and sample efficient learning on a suite of control tasks, including state of the art results on OpenAI gym benchmarks. These results suggest that our practical variants of PAL and MAL:

- are substantially more sample efficient compared to prior model-based and model-free algorithms,
- can achieve the same asymptotic performance as model-free counterparts,
- can scale to high-dimensional tasks with complex dynamics like dexterous manipulation,
- can scale to tasks requiring long horizon rollouts (e.g. OpenAI gym tasks which have a 1000 timestep horizon).

More broadly, our work adds to a growing body of recent work which suggests that MBRL can be stable, sample efficient, and more generalizable or adaptable to new tasks and non-stationary settings. For future work, we hope to study alternate ways to solve the Stackelberg optimization; such as using the full implicit gradient term and unrolled optimization. Finally, although we presented our game theoretic framework in the context of MBRL, it is more broadly applicable for any surrogate based optimization including actor-critic methods. It would make for interesting future work to study broader extensions and implications.

Acknowledgements

We thank Profs. Emo Todorov, Sham Kakade, Drew Bagnell, and Sergey Levine for valuable feedback and discussions. We thank Michael Ahn and Michael Janner for sharing the baseline learning curves; and Ben Eysenbach and Anirudh Vemula for feedback on the paper draft. The work was done by Aravind Rajeswaran during internships at Google Brain, MTV. Aravind thanks Google for providing a highly encouraging research environment.

References

Abbeel, P. and Ng, A. Y. Exploration and apprenticeship learning in reinforcement learning. In *ICML*, 2005.

Agarwal, A., Kakade, S. M., Lee, J. D., and Mahajan, G. Optimality and approximation with policy gradient methods in markov decision processes. *ArXiv*, abs/1908.00261, 2019a.

Agarwal, A., Kakade, S. M., and Yang, L. F. On the optimality of sparse model-based planning for markov decision processes. *ArXiv*, abs/1906.03804, 2019b.

Ahn, M., Zhu, H., Hartikainen, K., Ponte, H., Gupta, A., Levine, S., and Kumar, V. ROBEL: RObotics BEnchmarks for Learning with low-cost robots. In *Conference on Robot Learning (CoRL)*, 2019.

Åström, K. J. and Murray, R. M. Feedback systems an introduction for scientists and engineers. 2004.

Ben-David, S., Blitzer, J., Crammer, K., and Pereira, F. C. Analysis of representations for domain adaptation. In *NIPS*, 2006.

Bengio, S., Vinyals, O., Jaitly, N., and Shazeer, N. Scheduled sampling for sequence prediction with recurrent neural networks. *ArXiv*, abs/1506.03099, 2015.

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

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym, 2016.

Buckman, J., Hafner, D., Tucker, G., Brevdo, E., and Lee, H. Sample-efficient reinforcement learning with stochastic ensemble value expansion. *arXiv preprint arXiv:1807.01675*, 2018.

Cesa-Bianchi, N. and Lugosi, G. Prediction, learning, and games. 2006.

Chua, K., Calandra, R., McAllister, R., and Levine, S. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *NeurIPS*, 2018.

Colson, B., Marcotte, P., and Savard, G. An overview of bilevel optimization. *Annals of Operations Research*, 153: 235–256, 2007.

Deisenroth, M. P. and Rasmussen, C. E. Pilco: A model-based and data-efficient approach to policy search. In *ICML*, 2011.

Farajtabar, M., Chow, Y., and Ghavamzadeh, M. More robust doubly robust off-policy evaluation. In *ICML*, 2018.

Feinberg, V., Wan, A., Stoica, I., Jordan, M. I., Gonzalez, J., and Levine, S. Model-based value estimation for efficient model-free reinforcement learning. *CoRR*, abs/1803.00101, 2018.

Fiez, T., Chasnov, B., and Ratliff, L. J. Convergence of learning dynamics in stackelberg games. *ArXiv*, abs/1906.01217, 2019.

Foerster, J. N., Chen, R. Y., Al-Shedivat, M., Whiteson, S., Abbeel, P., and Mordatch, I. Learning with opponent-learning awareness. In *AAMAS*, 2017.

Garcia, J. and Fernández, F. A comprehensive survey on safe reinforcement learning. *J. Mach. Learn. Res.*, 16: 1437–1480, 2015.

Greensmith, E., Bartlett, P. L., and Baxter, J. Variance reduction techniques for gradient estimates in reinforcement learning. *JMLR*, 2001.

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., and Levine, S. Soft actor-critic algorithms and applications. *ArXiv*, abs/1812.05905, 2018.

Hazan, E., Kakade, S. M., Singh, K., and Soest, A. V. Provably efficient maximum entropy exploration. In *ICML*, 2018.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NIPS*, 2017.

Janner, M., Fu, J., Zhang, M., and Levine, S. When to trust your model: Model-based policy optimization. *ArXiv*, abs/1906.08253, 2019.

Jiang, N. and Li, L. Doubly robust off-policy value evaluation for reinforcement learning. In *ICML*, 2015.

Kakade, S. M. and Langford, J. Approximately optimal approximate reinforcement learning. In *ICML*, 2002.

Kakade, S. M., Kearns, M., and Langford, J. Exploration in metric state spaces. In *ICML*, 2003.

Karmakar, P. and Bhatnagar, S. Two time-scale stochastic approximation with controlled markov noise and off-policy temporal-difference learning. *Mathematics of Operations Research*, 43:130–151, 2015.

Kearns, M. and Singh, S. P. Finite-sample convergence rates for q-learning and indirect algorithms. In *NIPS*, 1998a.

Kearns, M. and Singh, S. P. Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49:209–232, 1998b.

Konda, V. and Borkar, V. S. Actor-critic - type learning algorithms for markov decision processes. *SIAM J. Control and Optimization*, 38:94–123, 1999.

Konda, V. R. and Tsitsiklis, J. N. Convergence rate of linear two-time-scale stochastic approximation. In *The Annals of Applied Probability*, 2004.

Krantz, S. G. and Parks, H. R. The implicit function theorem: History, theory, and applications. 2002.

Kumar, V., Todorov, E., and Levine, S. Optimal control with learned local models: Application to dexterous manipulation. *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 378–383, 2016.

Kurutach, T., Clavera, I., Duan, Y., Tamar, A., and Abbeel, P. Model-ensemble trust-region policy optimization. *ArXiv*, abs/1802.10592, 2018.

Levine, S. and Abbeel, P. Learning neural network policies with guided policy search under unknown dynamics. In *NIPS*, 2014.

Ljung, L. System identification: Theory for the user. 1987.

Lowrey, K., Rajeswaran, A., Kakade, S., Todorov, E., and Mordatch, I. Plan Online, Learn Offline: Efficient Learning and Exploration via Model-Based Control. In *International Conference on Learning Representations (ICLR)*, 2019.

McMahan, H. B. Follow-the-regularized-leader and mirror descent: Equivalence theorems and l1 regularization. In *AISTATS*, 2011.

Metz, L., Poole, B., Pfau, D., and Sohl-Dickstein, J. Unrolled generative adversarial networks. *ArXiv*, abs/1611.02163, 2017.

Mordatch, I. and Todorov, E. Combining the benefits of function approximation and trajectory optimization. In *RSS*, 2014.

Munos, R. and Szepesvári, C. Finite-time bounds for fitted value iteration. *Journal of Machine Learning Research*, 2008.

Nagabandi, A., Konoglie, K., Levine, S., and Kumar, V. Deep dynamics models for learning dexterous manipulation. *ArXiv*, abs/1909.11652, 2019.

Narendra, K. S. and Annaswamy, A. M. Persistent excitation in adaptive systems. *International Journal of Control*, 1987.

Nichol, A., Achiam, J., and Schulman, J. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. Curiosity-driven exploration by self-supervised prediction. In *ICML*, 2017.

Pathak, D., Gandhi, D., and Gupta, A. Self-supervised exploration via disagreement. *ArXiv*, abs/1906.04161, 2019.

Puterman, M. L. Markov decision processes: Discrete stochastic dynamic programming. In *Wiley Series in Probability and Statistics*, 1994.

Rajeswaran, A., Ghotra, S., Ravindran, B., and Levine, S. Epopt: Learning robust neural network policies using model ensembles. In *ICLR*, 2016.

Rajeswaran, A., Lowrey, K., Todorov, E., and Kakade, S. Towards Generalization and Simplicity in Continuous Control. In *NIPS*, 2017.

Rajeswaran, A., Kumar, V., Gupta, A., Vezzani, G., Schulman, J., Todorov, E., and Levine, S. Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations. In *Proceedings of Robotics: Science and Systems (RSS)*, 2018.

Rajeswaran, A., Finn, C., Kakade, S. M., and Levine, S. Meta-learning with implicit gradients. In *NeurIPS*, 2019.

Ross, S. and Bagnell, J. A. Agnostic system identification for model-based reinforcement learning. In *ICML*, 2012.

Schäfer, F. and Anandkumar, A. Competitive gradient descent. In *NeurIPS*, 2019.

Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. High-dimensional continuous control using generalized advantage estimation. In *ICLR*, 2016.

Shalev-Shwartz, S. Online learning and online convex optimization. *"Foundations and Trends in Machine Learning"*, 2012.

Sun, B., Feng, J., and Saenko, K. Return of frustratingly easy domain adaptation. In *AAAI*, 2015.

Sun, W., Gordon, G. J., Boots, B., and Bagnell, J. A. Dual policy iteration. In *NeurIPS*, 2018a.

Sun, W., Gordon, G. J., Boots, B., and Bagnell, J. A. Dual policy iteration. *CoRR*, abs/1805.10755, 2018b.

Sutton, R. S. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *ICML*, 1990.

Thomas, P. S. and Brunskill, E. Data-efficient off-policy policy evaluation for reinforcement learning. *ArXiv*, abs/1604.00923, 2016.

Todorov, E. and Li, W. A generalized iterative lqg method for locally-optimal feedback control of constrained nonlinear stochastic systems. In *ACC*, 2005.

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

Venkatraman, A., Hebert, M., and Bagnell, J. A. Improving multi-step prediction of learned time series models. In *AAAI*, 2015.

von Stackelberg, H. Market structure and equilibrium. 1934.

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. *ArXiv*, abs/1907.02057, 2019a.

Wang, Y., Zhang, G., and Ba, J. On solving minimax optimization locally: A follow-the-ridge approach. *ArXiv*, abs/1910.07512, 2019b.

Williams, G., Wagener, N., Goldfain, B., Drews, P., Rehg, J. M., Boots, B., and Theodorou, E. Information theoretic mpc for model-based reinforcement learning. *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1714–1721, 2017.

Wu, C., Rajeswaran, A., Duan, Y., Kumar, V., Bayen, A. M., Kakade, S. M., Mordatch, I., and Abbeel, P. Variance reduction for policy gradient with action-dependent factorized baselines. In *ICLR*, 2018.

Xu, H., Li, Y., Tian, Y., Darrell, T., and Ma, T. Algorithmic framework for model-based reinforcement learning with theoretical guarantees. *ArXiv*, abs/1807.03858, 2018.

Zhou, K., Doyle, J. C., and Glover, K. *Robust and Optimal Control*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1996. ISBN 0-13-456567-3.

Zhu, H., Gupta, A., Rajeswaran, A., Levine, S., and Kumar, V. Dexterous manipulation with deep reinforcement learning: Efficient, general, and low-cost. *2019 International Conference on Robotics and Automation (ICRA)*, pp. 3651–3657, 2018.

A. Theory

We provide the formal statements and proofs for theoretical results in the paper.

A.1. Performance with Global Models

Lemma 1 restated. *(Simulation lemma) Suppose we have a model* $\widehat{\boldsymbol{M}}$ *such that*

$$D_{TV}(P_{\boldsymbol{M}}(\cdot|s,a), P_{\widehat{\boldsymbol{M}}}(\cdot|s,a)) \leq \epsilon_{\boldsymbol{M}} \ \ \forall (s,a),$$

and the reward function is such that $|\mathcal{R}(s)| \leq R_{\max} \, \forall s \in \mathcal{S}$. *Then, we have*

$$\left| J(\boldsymbol{\pi}, \boldsymbol{M}) - J(\boldsymbol{\pi}, \widehat{\boldsymbol{M}}) \right| \leq \frac{2\gamma \epsilon_{\boldsymbol{M}} R_{\max}}{(1-\gamma)^2} \ \ \forall \boldsymbol{\pi}$$

Proof. Let $V^{\pi}(s, \boldsymbol{M})$ and $V^{\pi}(s, \widehat{\boldsymbol{M}})$ denote the value of policy $\boldsymbol{\pi}$ starting from an arbitrary state $s \in \mathcal{S}$ in $\boldsymbol{M}$ and $\widehat{\boldsymbol{M}}$ respectively. For simplicity of notation, we also define

$$P^{\pi}_{\boldsymbol{M}}(s'|s) := \mathbb{E}_{a \sim \boldsymbol{\pi}(\cdot|s)} \left[P_{\boldsymbol{M}}(s'|s,a) \right] \quad \text{and} \quad P^{\pi}_{\widehat{\boldsymbol{M}}}(s'|s) := \mathbb{E}_{a \sim \boldsymbol{\pi}(\cdot|s)} \left[P_{\widehat{\boldsymbol{M}}}(s'|s,a) \right].$$

Before the proof, we note the following useful observations.

1. Since $D_{TV}(P_{\boldsymbol{M}}(\cdot|s,a), P_{\widehat{\boldsymbol{M}}}(\cdot|s,a)) \leq \epsilon_{\boldsymbol{M}} \ \forall (s,a)$, the inequality also holds for an average over actions, i.e. $D_{TV}(P^{\pi}_{\boldsymbol{M}}(\cdot|s), P^{\pi}_{\widehat{\boldsymbol{M}}}(\cdot|s)) \leq \epsilon_{\boldsymbol{M}} \ \forall s$.
2. Since the rewards are bounded, we can achieve a maximum reward of $R_{\max}$ in each time step. Using a geometric summation with discounting γ, we have
$$\max_{s \in \mathcal{S}} \ V^{\pi}(s, \boldsymbol{M}) \leq \frac{R_{\max}}{1-\gamma} \ \ \forall \boldsymbol{\pi}, s$$
3. Let $f(x) : x \in \mathcal{X} \rightarrow [-f_{\max}, f_{\max}]$ be a real-valued function with bounded range, i.e. $0 \leq f_{\max} < \infty$. Let $P_1(x)$ and $P_2(x)$ be two probability distribution (density) over the space $\mathcal{X}$. Then, we have
$$\left| \mathbb{E}_{x \sim P_1(\cdot)}[f(x)] - \mathbb{E}_{x \sim P_2(\cdot)}[f(x)] \right| \leq 2 f_{\max} \, D_{TV}(P_1, P_2)$$

Using the above observations, we have the following inequalities:

$$\begin{aligned}
&\left| V^{\pi}(s, \boldsymbol{M}) - V^{\pi}(s, \widehat{\boldsymbol{M}}) \right| \\
&= \left| \mathcal{R}(s) + \gamma \mathbb{E}_{s' \sim P^{\pi}_{\boldsymbol{M}}(\cdot|s)} \left[V^{\pi}(s', \boldsymbol{M}) \right] - \mathcal{R}(s) - \gamma \mathbb{E}_{s' \sim P^{\pi}_{\widehat{\boldsymbol{M}}}(\cdot|s)} \left[V^{\pi}(s', \widehat{\boldsymbol{M}}) \right] \right| \\
&\leq \gamma \left| \mathbb{E}_{s' \sim P^{\pi}_{\boldsymbol{M}}(\cdot|s)} \left[V^{\pi}(s', \boldsymbol{M}) \right] - \mathbb{E}_{s' \sim P^{\pi}_{\widehat{\boldsymbol{M}}}(\cdot|s)} \left[V^{\pi}(s', \boldsymbol{M}) \right] \right| + \\
&\quad \gamma \left| \mathbb{E}_{s' \sim P^{\pi}_{\widehat{\boldsymbol{M}}}(\cdot|s)} \left[V^{\pi}(s', \boldsymbol{M}) - V^{\pi}(s', \widehat{\boldsymbol{M}}) \right] \right| \\
&\leq 2\gamma \left(\max_{s' \in \mathcal{S}} \ V^{\pi}(s', \boldsymbol{M}) \right) D_{TV}(P^{\pi}_{\boldsymbol{M}}(\cdot|s), P^{\pi}_{\widehat{\boldsymbol{M}}}(\cdot|s)) + \gamma \max_{s' \in \mathcal{S}} \left| V^{\pi}(s', \boldsymbol{M}) - V^{\pi}(s', \widehat{\boldsymbol{M}}) \right|
\end{aligned}$$

Since the above bound holds for all states, we have that $\forall \boldsymbol{\pi}$

$$\begin{aligned}
(1-\gamma) \max_{s' \in \mathcal{S}} \left| V^{\pi}(s', \boldsymbol{M}) - V^{\pi}(s', \widehat{\boldsymbol{M}}) \right| &\leq 2\gamma \left(\max_{s' in \mathcal{S}} \ V^{\pi}(s', \boldsymbol{M}) \right) D_{TV}(P^{\pi}_{\boldsymbol{M}}(\cdot|s), P^{\pi}_{\widehat{\boldsymbol{M}}}(\cdot|s)) \\
&\leq \frac{2\gamma R_{\max}}{1-\gamma} D_{TV}(P^{\pi}_{\boldsymbol{M}}(\cdot|s), P^{\pi}_{\widehat{\boldsymbol{M}}}(\cdot|s)) \\
&\leq \frac{2\gamma \epsilon_{\boldsymbol{M}} R_{\max}}{1-\gamma}
\end{aligned}$$

Stated alternatively, the above inequality implies

$$\left| V^{\pi}(s, \boldsymbol{M}) - V^{\pi}(s, \widehat{\boldsymbol{M}}) \right| \leq \frac{2\gamma\epsilon_{\boldsymbol{M}} R_{\max}}{(1-\gamma)^2} \;\; \forall s, \boldsymbol{\pi}$$

Finally, note that the performance criteria $J(\boldsymbol{\pi}, \widehat{\boldsymbol{M}})$ and $J(\boldsymbol{\pi}, \boldsymbol{M})$ are simply the average of the value function over the initial state distribution. Since the above inequality holds for all states, it also holds for the average over initial state distribution. □

We note that the above simulation lemma (or closely related forms) have been proposed and proved several times in prior literature (e.g. see Kearns & Singh (1998b); Kakade et al. (2003); Abbeel & Ng (2005)). We present the proof largely for completeness and also to motivate the proof techniques we will use for our main theoretical result (Theorem 1).

A.2. Performance with Task-Driven Local Models

In this section, we relax the global model requirement and consider the case where we have more local models, as well as the case of a policy-model equilibrium pair. We first provide a lemma that characterizes error amplification in local simulation.

Lemma 2. *(Error amplification in local simulation) Let $P_1(\cdot|s)$ and $P_2(\cdot|s)$ be two Markov chains with the same initial state distribution. Let $P_1^t(s)$ and $P_2^t(s)$ be the marginal distributions over states at time t when following P_1 and P_2 respectively. Suppose*

$$\mathbb{E}_{s \sim P_1^t}\left[D_{TV}(P_1(\cdot|s), P_2(\cdot|s)) \right] \leq \epsilon \;\; \forall \, t$$

then, the marginal distributions are bounded as:

$$D_{TV}(P_1^t, P_2^t) \leq \epsilon t \;\; \forall \, t$$

Proof. Let us fix a state $s \in \mathcal{S}$, and let $\bar{s} \in \mathcal{S}$ denote a "dummy" state variable. Then,

$$\begin{aligned}
\left| P_1^t(s) - P_2^t(s) \right| &= \left| \sum_{\bar{s} \in \mathcal{S}} P_1(s|\bar{s}) P_1^{t-1}(\bar{s}) - \sum_{\bar{s} \in \mathcal{S}} P_2(s|\bar{s}) P_2^{t-1}(\bar{s}) \right| \\
&\leq \sum_{\bar{s} \in \mathcal{S}} \left| P_1(s|\bar{s}) P_1^{t-1}(\bar{s}) - P_2(s|\bar{s}) P_2^{t-1}(\bar{s}) \right| \\
&\leq \sum_{\bar{s} \in \mathcal{S}} \left| P_1^{t-1}(\bar{s}) \big(P_1(s|\bar{s}) - P_2(s|\bar{s}) \big) \right| + \left| P_2(s|\bar{s}) \big(P_1^{t-1}(\bar{s}) - P_2^{t-1}(\bar{s}) \big) \right|
\end{aligned}$$

Using the above inequality, we have

$$\begin{aligned}
2 D_{TV}(P_1^t, P_2^t) &= \sum_{s \in \mathcal{S}} \left| P_1^t(s) - P_2^t(s) \right| \\
&\leq \sum_{\bar{s} \sim \mathcal{S}} P_1^{t-1}(\bar{s}) \sum_{s \in \mathcal{S}} \left| P_1(s|\bar{s}) - P_2(s|\bar{s}) \right| + \sum_{\bar{s} \in \mathcal{S}} \left| P_1^{t-1}(\bar{s}) - P_2^{t-1}(\bar{s}) \right| \\
&\leq 2\epsilon + 2 D_{TV}(P_1^{t-1}, P_2^{t-1}) \\
&\leq 2t\epsilon
\end{aligned}$$

where the last step uses the previous inequality recursively till $t = 0$, where the Markov chains have the same (initial) state distribution. □

The above lemma considers the error between two Markov chains. Note that fixing a policy in an MDP results in a Markov chain transition dynamics. Thus, fixing the policy, we can use the above lemma to compare the resulting Markov chains in $\boldsymbol{M}$ and $\widehat{\boldsymbol{M}}$. Consider the following definitions:

$$\mu_{\widehat{\boldsymbol{M}}}^{\boldsymbol{\pi}}(s, a) = \frac{1}{T_\infty} \sum_{t=0}^{T_\infty} P(s_t = s, a_t = a)$$

$$\tilde{\mu}_{\widehat{\boldsymbol{M}}}^{\boldsymbol{\pi}}(s, a) = (1-\gamma) \sum_{t=0}^{\infty} \gamma^t P(s_t = s, a_t = a)$$

The first distribution $\mu^{\boldsymbol{\pi}}_{\widehat{\boldsymbol{M}}}$ is the average state visitation distribution when executing $\boldsymbol{\pi}$ in $\widehat{\boldsymbol{M}}$, and T_∞ is the episode duration (could tend to ∞ in the non-episodic case). The second distribution $\tilde{\mu}^{\boldsymbol{\pi}}_{\widehat{\boldsymbol{M}}}$ is the discounted state visitation distribution when executing $\boldsymbol{\pi}$ in $\widehat{\boldsymbol{M}}$. Let $\mu^{\boldsymbol{\pi}}_{\boldsymbol{M}}$ and $\tilde{\mu}^{\boldsymbol{\pi}}_{\boldsymbol{M}}$ be their analogues in $\boldsymbol{M}$. When learning the dynamics model, we would minimize the prediction error under $\mu^{\boldsymbol{\pi}}_{\boldsymbol{M}}$, while $J(\boldsymbol{\pi}, \boldsymbol{M})$ is dependent on rewards under $\tilde{\mu}^{\boldsymbol{\pi}}_{\boldsymbol{M}}$. Let

$$\mu^{\boldsymbol{\pi},t}_{\boldsymbol{M}}(s,a) = P(s_t = s, a_t = a)$$

be the marginal distribution at time t when following $\boldsymbol{\pi}$ in $\boldsymbol{M}$. Let $\mu^{\boldsymbol{\pi},t}_{\widehat{\boldsymbol{M}}}(s,a)$ be analogously defined when following $\boldsymbol{\pi}$ in $\widehat{\boldsymbol{M}}$. Using these definitions, we first characterize the difference in performance of the same policy $\boldsymbol{\pi}$ under $\boldsymbol{M}$ and $\widehat{\boldsymbol{M}}$.

Lemma 3. *(Performance difference due to model error) Let $\boldsymbol{M}$ and $\widehat{\boldsymbol{M}}$ be two different MDPs differing only in their transition dynamics – $P_{\boldsymbol{M}}$ and $P_{\widehat{\boldsymbol{M}}}$. Let the absolute value of rewards be bounded by $R_{\max}$. Fix a policy $\boldsymbol{\pi}$ for both $\boldsymbol{M}$ and $\widehat{\boldsymbol{M}}$, and let $P^t_{\boldsymbol{M}}$ and $P^t_{\widehat{\boldsymbol{M}}}$ be the resulting marginal state distributions at time t. If the MDPs are such that*

$$\mathbb{E}_{(s,a)\sim\mu^{\boldsymbol{\pi},t}_{\boldsymbol{M}}}\left[D_{TV}\left(P_{\boldsymbol{M}}(\cdot|s,a), P_{\widehat{\boldsymbol{M}}}(\cdot|s,a)\right)\right] \leq \epsilon \ \ \forall t$$

then, the performance difference is bounded as:

$$\left|J(\boldsymbol{\pi}, \boldsymbol{M}) - J(\boldsymbol{\pi}, \widehat{\boldsymbol{M}})\right| \leq \frac{2\gamma\epsilon R_{\max}}{(1-\gamma)^2}$$

Proof. Recall that the performance of a policy can be written as:

$$J(\boldsymbol{\pi}, \widehat{\boldsymbol{M}}) = \frac{1}{1-\gamma}\mathbb{E}_{\tilde{\mu}^{\boldsymbol{\pi}}_{\boldsymbol{M}}}\left[\mathcal{R}(s)\right] = \mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t\mathcal{R}(s_t)\right]$$

where the randomness for the second term is due to $\widehat{\boldsymbol{M}}$ and $\boldsymbol{\pi}$. We can analogously write $J(\boldsymbol{\pi}, \boldsymbol{M})$ as well. Thus, the performance difference can be bounded as:

$$\begin{aligned}\left|J(\boldsymbol{\pi}, \boldsymbol{M}) - J(\boldsymbol{\pi}, \widehat{\boldsymbol{M}})\right| &= \left|\frac{1}{1-\gamma}\mathbb{E}_{\tilde{\mu}^{\boldsymbol{\pi}}_{\boldsymbol{M}}}\left[\mathcal{R}(s)\right] - \frac{1}{1-\gamma}\mathbb{E}_{\tilde{\mu}^{\boldsymbol{\pi}}_{\widehat{\boldsymbol{M}}}}\left[\mathcal{R}(s)\right]\right| \\ &\leq \frac{2R_{\max}}{1-\gamma}D_{TV}\left(\tilde{\mu}^{\boldsymbol{\pi}}_{\boldsymbol{M}}, \tilde{\mu}^{\boldsymbol{\pi}}_{\widehat{\boldsymbol{M}}}\right)\end{aligned}$$

Also recall that we have

$$\mu^{\boldsymbol{\pi},t}_{\widehat{\boldsymbol{M}}}(s,a) = P(s_t = s, a_t = a) = P^t_{\widehat{\boldsymbol{M}}}(s)\boldsymbol{\pi}(a|s)$$

We can bound the discounted state visitation distribution as

$$\begin{aligned}2D_{TV}\left(\tilde{\mu}^{\boldsymbol{\pi}}_{\boldsymbol{M}}, \tilde{\mu}^{\boldsymbol{\pi}}_{\widehat{\boldsymbol{M}}}\right) &= \sum_{s,a}\left|\tilde{\mu}^{\boldsymbol{\pi}}_{\boldsymbol{M}}(s,a) - \tilde{\mu}^{\boldsymbol{\pi}}_{\widehat{\boldsymbol{M}}}(s,a)\right| \\ &= (1-\gamma)\sum_{s,a}\left|\sum_t \gamma^t\mu^{\boldsymbol{\pi},t}_{\boldsymbol{M}}(s,a) - \gamma^t\mu^{\boldsymbol{\pi},t}_{\widehat{\boldsymbol{M}}}(s,a)\right| \\ &\leq (1-\gamma)\sum_{s,a}\sum_t \gamma^t\left|\mu^{\boldsymbol{\pi},t}_{\boldsymbol{M}}(s,a) - \mu^{\boldsymbol{\pi},t}_{\widehat{\boldsymbol{M}}}(s,a)\right| \\ &= (1-\gamma)\sum_{s}\sum_t \gamma^t\left|P^t_{\boldsymbol{M}}(s) - P^t_{\widehat{\boldsymbol{M}}}(s)\right| \\ &\leq (1-\gamma)\sum_{t=0}^{\infty}\gamma^t\left(2t\epsilon\right)\end{aligned}$$

where the last inequality uses Lemma 2. Notice that the final summation is an arithmetico-geometric series. When simplified, this results in

$$D_{TV}\left(\tilde{\mu}^{\boldsymbol{\pi}}_{\boldsymbol{M}}, \tilde{\mu}^{\boldsymbol{\pi}}_{\widehat{\boldsymbol{M}}}\right) \leq (1-\gamma)\frac{\epsilon\gamma}{(1-\gamma)^2} \leq \frac{\epsilon\gamma}{1-\gamma}$$

Using this bound for the performance difference yields the desired result. □

Remarks: The performance difference (due to model error) lemma we present is quite distinct and different from the performance difference lemma from Kakade & Langford (2002). Specifically, our lemma bounds the performance difference between the *same policy* in two *different models.* In contrast, the lemma from Kakade & Langford (2002) characterizes the performance difference between two *different policies* in the *same model.*

Finally, we study the global performance guarantee when we have a policy-model pair close to equilibrium.

Theorem 1 restated. *(Global performance of equilibrium pair) Suppose we have policy-model pair* $(\boldsymbol{\pi}, \widehat{\boldsymbol{M}})$ *such that the following conditions hold simultaneously:*

$$\ell(\widehat{\boldsymbol{M}}, \mu_{\boldsymbol{M}}^{\boldsymbol{\pi},t}) \leq \epsilon_{\boldsymbol{M}} \ \forall t \ \textit{ and } \ J(\boldsymbol{\pi}, \widehat{\boldsymbol{M}}) \geq \sup_{\boldsymbol{\pi}'} J(\boldsymbol{\pi}', \widehat{\boldsymbol{M}}) - \epsilon_{\boldsymbol{\pi}}.$$

Let $\boldsymbol{\pi}^*$ *be an optimal policy so that* $J(\boldsymbol{\pi}^*, \boldsymbol{M}) \geq J(\boldsymbol{\pi}', \boldsymbol{M}) \ \forall \boldsymbol{\pi}'$*. The performance is bounded as*

$$J(\boldsymbol{\pi}^*, \boldsymbol{M}) - J(\boldsymbol{\pi}, \boldsymbol{M}) \leq \frac{2\gamma\sqrt{\epsilon_{\boldsymbol{M}}}R_{\max}}{(1-\gamma)^2} + \epsilon_{\boldsymbol{\pi}} + \frac{2R_{\max}}{1-\gamma} D_{TV}\left(\tilde{\mu}_{\boldsymbol{M}}^{\boldsymbol{\pi}^*}, \tilde{\mu}_{\widehat{\boldsymbol{M}}}^{\boldsymbol{\pi}^*}\right).$$

Proof. We first simplify the performance difference, and subsequently bound the different terms. Let $\boldsymbol{\pi}^*_{\widehat{\boldsymbol{M}}}$ to be an optimal policy in the model, so that $J(\boldsymbol{\pi}^*_{\widehat{\boldsymbol{M}}}, \widehat{\boldsymbol{M}}) \geq J(\boldsymbol{\pi}', \widehat{\boldsymbol{M}}) \ \forall \boldsymbol{\pi}'$. We can decompose the performance difference due to various contributions as:

$$\begin{aligned} J(\boldsymbol{\pi}^*, \boldsymbol{M}) - J(\boldsymbol{\pi}, \boldsymbol{M}) &= J(\boldsymbol{\pi}^*, \boldsymbol{M}) - J(\boldsymbol{\pi}^*, \widehat{\boldsymbol{M}}) + J(\boldsymbol{\pi}^*, \widehat{\boldsymbol{M}}) - J(\boldsymbol{\pi}, \boldsymbol{M}) \\ &= \underbrace{J(\boldsymbol{\pi}^*, \boldsymbol{M}) - J(\boldsymbol{\pi}^*, \widehat{\boldsymbol{M}})}_{\text{Term}-\text{I}} + \underbrace{J(\boldsymbol{\pi}^*, \widehat{\boldsymbol{M}}) - J(\boldsymbol{\pi}, \widehat{\boldsymbol{M}})}_{\text{Term}-\text{II}} + \underbrace{J(\boldsymbol{\pi}, \widehat{\boldsymbol{M}}) - J(\boldsymbol{\pi}, \boldsymbol{M})}_{\text{Term}-\text{III}} \end{aligned}$$

Let us first consider **Term-II**, which is related to the sub-optimality in the planning problem. Notice that we have:

$$J(\boldsymbol{\pi}^*, \widehat{\boldsymbol{M}}) - J(\boldsymbol{\pi}, \widehat{\boldsymbol{M}}) = \left(J(\boldsymbol{\pi}^*, \widehat{\boldsymbol{M}}) - J(\boldsymbol{\pi}^*_{\widehat{\boldsymbol{M}}}, \widehat{\boldsymbol{M}})\right) + \left(J(\boldsymbol{\pi}^*_{\widehat{\boldsymbol{M}}}, \widehat{\boldsymbol{M}}) - J(\boldsymbol{\pi}, \widehat{\boldsymbol{M}})\right) \leq 0 + \epsilon_{\boldsymbol{\pi}}$$

We have $J(\boldsymbol{\pi}^*, \widehat{\boldsymbol{M}}) - J(\boldsymbol{\pi}^*_{\widehat{\boldsymbol{M}}}, \widehat{\boldsymbol{M}}) \leq 0$ since $\boldsymbol{\pi}^*_{\widehat{\boldsymbol{M}}}$ is the optimal policy in the model, and we have $J(\boldsymbol{\pi}^*_{\widehat{\boldsymbol{M}}}, \widehat{\boldsymbol{M}}) - J(\boldsymbol{\pi}, \widehat{\boldsymbol{M}}) \leq \epsilon_{\boldsymbol{\pi}}$ due to the approximate equilibrium condition.

For **Term-III**, we will draw upon the model error performance difference lemma (Lemma 3). Note that the equilibrium condition of low error along with Pinsker's inequality implies

$$\mathbb{E}_{s \sim \mu_{\boldsymbol{M}}^{\boldsymbol{\pi},t}} \left[D_{TV}\left(P_{\boldsymbol{M}}(\cdot|s,a), P_{\widehat{\boldsymbol{M}}}(\cdot|s,a)\right)\right] \leq \sqrt{\epsilon_{\boldsymbol{M}}}$$

Using this and Lemma 3, we have

$$J(\boldsymbol{\pi}, \widehat{\boldsymbol{M}}) - J(\boldsymbol{\pi}, \boldsymbol{M}) \leq \frac{2\gamma\sqrt{\epsilon_{\boldsymbol{M}}}R_{\max}}{(1-\gamma)^2}$$

Finally, **Term-I** is a transfer learning term that measures the error of $\widehat{\boldsymbol{M}}$ (which has low error under $\boldsymbol{\pi}$) under the distribution of $\boldsymbol{\pi}^*$. The performance difference can be written as

$$\begin{aligned} J(\boldsymbol{\pi}^*, \boldsymbol{M}) - J(\boldsymbol{\pi}^*, \widehat{\boldsymbol{M}}) &= \frac{1}{1-\gamma}\mathbb{E}_{(s,a)\sim\tilde{\mu}_{\boldsymbol{M}}^{\boldsymbol{\pi}^*}}\left[\mathcal{R}(s)\right] - \frac{1}{1-\gamma}\mathbb{E}_{(s,a)\sim\tilde{\mu}_{\widehat{\boldsymbol{M}}}^{\boldsymbol{\pi}^*}}\left[\mathcal{R}(s)\right] \\ &\leq \frac{2R_{\max}}{1-\gamma} D_{TV}\left(\tilde{\mu}_{\boldsymbol{M}}^{\boldsymbol{\pi}^*}, \tilde{\mu}_{\widehat{\boldsymbol{M}}}^{\boldsymbol{\pi}^*}\right) \end{aligned}$$

Putting all the terms together, we have

$$J(\boldsymbol{\pi}^*, \boldsymbol{M}) - J(\boldsymbol{\pi}, \boldsymbol{M}) \leq \frac{2R_{\max}}{1-\gamma} D_{TV}\left(\tilde{\mu}_{\boldsymbol{M}}^{\boldsymbol{\pi}^*}, \tilde{\mu}_{\widehat{\boldsymbol{M}}}^{\boldsymbol{\pi}^*}\right) + \epsilon_{\boldsymbol{\pi}} + \frac{2\gamma\sqrt{\epsilon_{\boldsymbol{M}}}R_{\max}}{(1-\gamma)^2}$$

□

Remarks: Tighter bounds on the transfer learning term is not possible without additional assumptions. However, the spirit of the transfer learning issue is captured by the term.

1. It suggests that there is a preference hierarchy between models that achieve similar low error under $\mu_{\boldsymbol{M}}^{\boldsymbol{\pi}}$. The models that can simulate a wider class of policies (i.e. have better transfer) are preferable for MBRL. This establishes a concrete connection between MBRL and domain adaptation, and we hope that various ideas from transfer learning and domain adaptation (Ben-David et al., 2006) can benefit MBRL.

2. The structure of $\mu_{\boldsymbol{M}}^{\boldsymbol{\pi}^*}$ provides avenues to achieve better transfer. Note that the start state distribution is the same for $\boldsymbol{M}$ and $\widehat{\boldsymbol{M}}$, and is also shared by all policies. Thus, if we could design or choose the start state distribution to be wide, it automatically ensures good mixing between $\mu_{\boldsymbol{M}}^{\boldsymbol{\pi}^*}$ and $\mu_{\widehat{\boldsymbol{M}}}^{\boldsymbol{\pi}^*}$ by virtue of them sharing the start state distribution. We could obtain such a distribution by training an exploratory policy (Pathak et al., 2017; Hazan et al., 2018), and executing it for a few steps to construct a starting state distribution.

The theorem assumes that the errors are small at each timestep: $\ell(\widehat{\boldsymbol{M}}, \mu_{\boldsymbol{M}}^{\boldsymbol{\pi},t}) \leq \epsilon_{\boldsymbol{M}} \; \forall t$. This is only a slightly stronger assumption than the average error being small. In practice, by executing the policy, we would have a dataset drawn from $\mu_{\boldsymbol{M}}^{\boldsymbol{\pi}}$. Thus, it should be possible to make the error small under $\mu_{\boldsymbol{M}}^{\boldsymbol{\pi}}$. Recall that $\mu_{\boldsymbol{M}}^{\boldsymbol{\pi}} = (1/T_\infty) \sum_{t=0}^{T_\infty} \mu_{\boldsymbol{M}}^{\boldsymbol{\pi},t}$. If we use an expressive function approximator, and if there is sufficient concentration of measure, small error over $\mu_{\boldsymbol{M}}^{\boldsymbol{\pi}}$ would lead to small error at each timestep. Furthermore, since we typically store time-indexed trajectories, we can check in practice that the error is small at each timestep.

B. Algorithm Implementation Details and Experiments

Our implementation builds on top of MJRL (`https://github.com/aravindr93/mjrl`) for NPG (Rajeswaran et al., 2017; 2018) and interfacing with MuJoCo/OpenAI-gym (Todorov et al., 2012; Brockman et al., 2016). We adapt the NPG implementation to work with learned models. Our model learning minimizes one-step prediction error using Adam. We first describe the details of these subroutines before describing the full algorithms.

Policy Details We represent the policy as a neural network, and use the learned model for performing synthetic rollouts as specified in Subroutine 1. For the set of initial states, we can either sample from the initial state distribution of MDP (if it is known) or keep track of initial states from the environment in a separate initial state replay buffer. We found both to perform near-identically. Furthermore, the synthetic rollouts can be started from either the initial state distribution of the MDP, or from intermediate states in real rollouts. We found starting 50% of synthetic rollouts from intermediate (real-world) rollout states leads to better asymptotic results for longer horizon gym tasks. This is consistent with prior works that suggest sampling from a wide initial state distribution is beneficial for policy gradient methods (Kakade & Langford, 2002; Rajeswaran et al., 2017).

The subroutine is written assuming a reward oracle, which can either be a known function, or can be learned from data. We found both settings to work near-identically, since rewards are often simple functions of the state-action and substantially easier to learn than dynamics. We consider a maximum rollout horizon of 500, which can become shorter if the maximum environment horizon is smaller, or if termination conditions kick in for the rollouts. If the environments have termination conditions, we enforce these for the synthetic rollouts as well. Finally, we use a baseline/value network for the purposes of variance reduction (Greensmith et al., 2001; Wu et al., 2018) – specifically GAE (Schulman et al., 2016). We use the default values for most parameters as summarized in Table 1, and do not tune them.

Subroutine 1 Model-Based Natural Policy Gradient Update Step

```
**Require:** Policy (stochastic) network $\boldsymbol{\pi}_\theta$, value/baseline network $V_\psi$, ensemble of MDP dynamics models $\{\widehat{\boldsymbol{M}}_\phi\}$, reward function $\mathcal{R}$, initial state distribution or buffer.
**Hyperparameters:** Discount factor $\gamma$, GAE $\lambda$, number of trajectories $N_\tau$, rollout horizon $H$, normalized NPG step size $\delta$
Initialize trajectory buffer $\mathcal{D}_\tau = \{\}$
**for** $k = 1, 2, \dots, N_\tau$ **do**
    Sample initial state $s_0^k$ from initial state distribution/buffer
    Perform $H$ step rollout from $s_0^k$ with $\boldsymbol{\pi}_\theta$ to get $\tau_j^k = (s_0^k, a_0^k, s_1^k, a_2^k, \dots s_H^k, a_H^k)$, one for each model $\widehat{\boldsymbol{M}}_\phi^j$ in the ensemble.
    Query reward function to obtain rewards for each step of the trajectories
    Truncate trajectories if termination/truncation conditions are part of the environment
    Aggregate the trajectories in trajectory buffer, $\mathcal{D}_\tau = \mathcal{D}_\tau \cup \{\tau\}$
**end for**
Compute advantages for each trajectory using $V_\psi$ and GAE (Schulman et al., 2016).
Compute vanilla policy gradient using the dataset
    $g = \mathbb{E}_{(s,a)\sim\mathcal{D}_\tau}\left[\nabla_\theta \log \boldsymbol{\pi}_\theta(a|s) A^{\boldsymbol{\pi}}(s,a)\right]$
Perform normalized NPG update ($F$ denotes the Fisher matrix)
    $\theta = \theta + \sqrt{\frac{\delta}{g^T F^{-1} g}}\, F^{-1} g$
Update value/baseline network $V_\psi$ to fit the computed returns in $\mathcal{D}_\tau$.
**Return** Policy network $\boldsymbol{\pi}_\theta$, value network $V_\psi$
```

Model details We model the MDP dynamics with ensembles of neural network dynamics models. Ensembles capture epistemic uncertainty (Chua et al., 2018) and provide robustness for policy optimization (Rajeswaran et al., 2016). We are provided with a dataset of tuples $\mathcal{D} = \{(s_t, a_t, s_{t+1})\}$, and we parameterize the model as:

$$\widehat{\boldsymbol{M}}_\phi(s_t, a_t) = s_t + \sigma_\Delta \; MLP_\phi \left(\frac{s_t - \mu_s}{\sigma_s}, \frac{a_t - \mu_a}{\sigma_a} \right)$$

where we $\Delta_t = s_{t+1} - s_t$ are the state differences, and mean centering and scaling are performed based on the dataset. We solve the following optimization problem to learn the parameters:

$$\min_\phi \;\; \mathbb{E}_{(s_t, a_t, s_{t+1}) \sim \mathcal{D}} \left[\left\| (s_{t+1} - s_t) - \sigma_\Delta \; MLP_\phi \left(\frac{s_t - \mu_s}{\sigma_s}, \frac{a_t - \mu_a}{\sigma_a} \right) \right\|^2 \right].$$

We specify the important hyperparameters along with PAL and MAL descriptions. When training, we also ensure that at-least 10^2 gradient steps and at-most 10^5 gradient steps are used, to avoid boundary issues when the buffer size is too small or large.

Table 1. Hyperparameters used for policy improvement with NPG

Parameter	**Value**
Policy network	MLP (64, 64)
Value/baseline network	MLP (128, 128)
Discount γ	0.995
GAE λ	0.97
# synthetic trajectories (N_τ)	200
Rollout horizon (H)	min (env-horizon, 500, termination)
normalized step size δ	0.05

Policy As Leader: The practical version of the PAL-NPG algorithm is provided in Algorithm 4. The algorithm alternates between collecting a small amount of data in each iteration, learning a dynamics model, and conservatively improving the policy. We use a small replay buffer to aggregate data from the past few iterations, but the replay buffer is kept small in size to ensure that the model is primarily trained to be accurate under current state visitation.

Algorithm 3 Policy As Leader (PAL) – Practical Version

```
Initialize: Policy network $\boldsymbol{\pi}_0$, model network(s) $\widehat{\boldsymbol{M}}_0$, value network $V_0$.
Hyperparameters: Initial samples $N_{init}$, samples per update $N$, buffer size $B \approx N$, number of NPG steps $K \approx 1$
Initial Data: Collect $N_{init}$ samples from the environment by interacting with initial policy. Store data in buffer $\mathcal{D}$.
for $k = 0, 1, 2, \dots$ do
    Learn dynamics model(s) $\widehat{\boldsymbol{M}}_{k+1}$ using data in the buffer.
    Policy updates: $\boldsymbol{\pi}_{k+1}, V_{k+1} =$ Model-Based NPG($\boldsymbol{\pi}_k, V_k, \widehat{\boldsymbol{M}}_{k+1}$) // call K times
    Collect dataset of $N$ samples from world by interacting with $\boldsymbol{\pi}_{k+1}$. Add data to replay buffer $\mathcal{D}$, discarding old data if size is larger than $B$.
end for
```

For hyperparameter selection, we performed a coarse search on DClaw task and used the same parameters for the remaining tasks with minor changes. The main parameters we focused on were the number of NPG updates per iteration, for which we tried $K = \{1, 2, 4, 8\}$ and found 4 to be best. Similarly, we studied number of samples per iteration $N = \{1, 5, 10, 20\} \times$env-horizon, and found 5 to be ideal for DClaw, DKitty, Reacher, and Hand tasks. For the hand task, $N = 10 \times$horizon produced more stable results, and we report this in the paper. The OpenAI gym tasks are longer horizon and we found fewer samples are sufficient. For the gym tasks, we use $N = 1000$ samples per iteration, which towards the later half of training often amounts to only one trajectory. We use a buffer of size $B = 2500$, which often amounts to using data from the past 2-5 iterations. We also use ensembles of dynamics models and we tried ensemble sizes of $\{1, 2, 4, 8\}$ and found 4 to be a good trade-off between performance and computation. We also found it important to initialize the policy with sufficient small amount of exploratory noise to avoid stability and divergence issues. We consider Gaussian policies with diagonal covariance where the neural network parameterizes the mean, and the diagonal co-variance is also learned. We initialize the standard deviation as $\sigma = \exp(-1)$. We do not add any additional exploratory noise when collecting data, but simply use the learned covariance in the Gaussian policy. We summarize the details in Table 2.

Table 2. Hyperparameters used for the PAL-NPG algorithm

Parameter	**Value**
Model network	MLP (512, 512)
Learning algorithm	Adam (default parameters)
No. of epochs	100
Mini-batch size	200
Ensemble size	4
Buffer size B	2500
Initial samples (N_{init})	2500
Samples per iteration (N)	min(5×env-horizon, 1000)
NPG updates (K)	4

Model As Leader: The practical version of the MAL-NPG algorithm is provided in Algorithm 4. The algorithm alternates between optimizing a policy using current model, collecting additional data which is aggregated into a data buffer, and finally improving the model using the aggregated data.

Algorithm 4 Model As Leader (MAL) – Practical Version

```
Initialize: Policy network $\boldsymbol{\pi}_0$, model network(s) $\widehat{\boldsymbol{M}}_0$, value network $V_0$.
Hyperparameters: Initial samples $N_{init}$, samples per update $N$, number of NPG steps $K \gg 1$
Initial Data: Collect $N_{init}$ samples from the environment by interacting with initial policy. Store data in buffer $\mathcal{D}$.
Initial Model: Learn model(s) $\widehat{\boldsymbol{M}}_0$ using data in $\mathcal{D}$.
for $k = 0, 1, 2, \ldots$ do
    Optimize $\boldsymbol{\pi}_{k+1}$ using $\widehat{\boldsymbol{M}}_k$ by running $K \gg 1$ steps of model-based NPG (Subroutine 1).
    Collect dataset $\mathcal{D}_{k+1}$ of $N$ samples from world using $\boldsymbol{\pi}_{k+1}$. Aggregate data $\mathcal{D} = \mathcal{D} \cup \mathcal{D}_{k+1}$.
    Learn dynamics model(s) $\widehat{\boldsymbol{M}}_{k+1}$ using data in $\mathcal{D}$.
end for
```

For hyperparameter selection, we follow the same overall approach as described in MAL. Compared to PAL, the main differences are that a larger number of initial samples are required, since the policy is optimized aggressively. We tried $K = \{10, 25, 40, 60\}$ and found $K = 25$ to be a good trade-off between performance and computation. We tried $N = \{1, 5, 10, 20\}$×env-horizon and found $N = 20$×horizon to provide the best results. For the OpenAI gym tasks, we used $N = 3000$ samples. For the simpler Pendulum task, we use fewer samples which still leads to stable results. We again use an ensemble of 4 models. The hyperparameter details are summarized in Table 3.

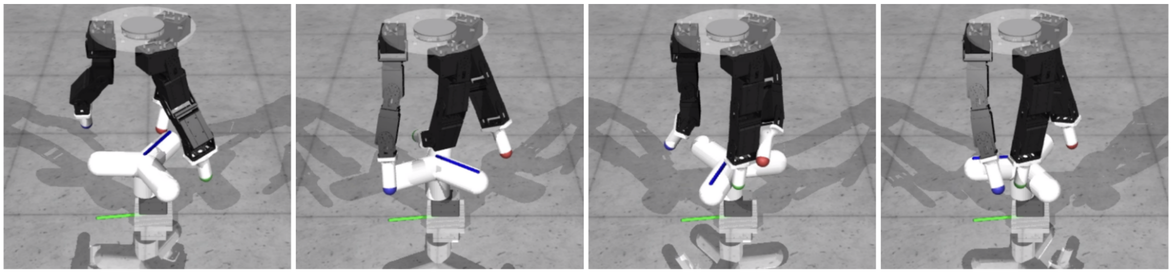
Table 3. Hyperparameters used for the MAL-NPG algorithm

Parameter	**Value**
Model network	MLP (512, 512)
Learning algorithm	Adam (default parameters)
No. of epochs	10
Mini-batch size	200
Ensemble size	4
Buffer size B	∞
Initial samples (N_{init})	5000
Samples per iteration (N)	min(20×env-horizon, 3000)
NPG updates (K)	25

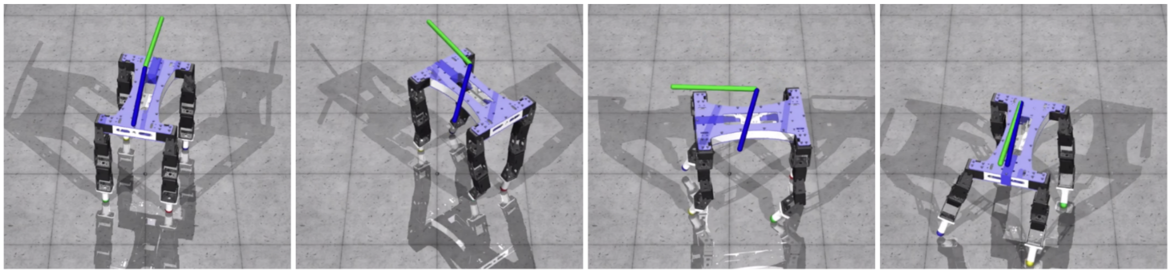
B.1. Task Suite

The main tasks we study are `DClaw-Turn`, `DKitty-Orient`, `7DOF-Reacher`, and `InHand-Pen`.

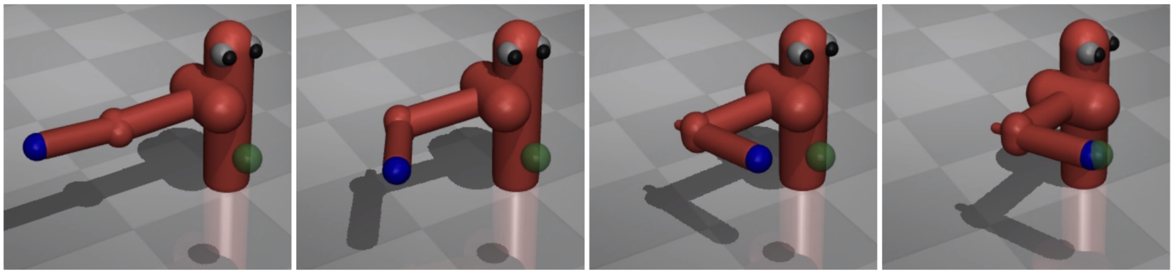
1. The `DClaw-Turn` task requires a 3 fingered "DClaw" to rotate a faucet to a desired orientation (see illustration below). The observations consist of the joint positions and velocities of the claw as well as the faucet; in addition to the desired valve orientation. The reward measures the closeness between the current faucet configuration and desired configuration. For further details about the task, see Ahn et al. (Ahn et al., 2019) (task `DClawTurnRandom-v0`)..



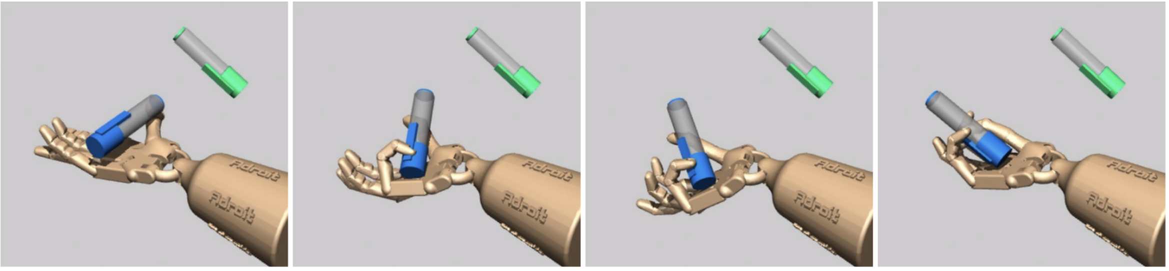
2. The `DKitty-Orient` task requires a quadruped (DKitty) to change its orientation in order to face in a desired direction (as illustrated below). The observations consist of the pose and velocity of various joints in the robot, and the desired orientation. The reward measures the difference between current pose of the robot and the desired pose. For further details about the task, see Ahn et al. (Ahn et al., 2019) (task `DKittyOrientRandom-v0`).



3. The `7DOF-Reacher` reacher task requires a 7DOF robot arm (corresponding to a Sawyer robot) to reach various spatial goals with its end effector (finger tip). The observations consist of the joint pose and velocities of the arm and the desired location for the end effector. The reward measures the distance between the end effector and the goal.



4. Finally, the `InHand-Pen` task requires a 24DOF dexterous hand to manipulate a pen in-hand to point in a desired orientation. The observations consist of the joint pose and velocity for the hand and pen, and the desired pose for the pen. The reward measures the difference between the pose of the pen and the desired pose. For additional details about the task, see Rajeswaran et al. (Rajeswaran et al., 2018) (task `pen-v0`).



B.2. Results on OpenAI gym benchmarks

We also benchmark the performance of PAL-NPG and MAL-NPG in the OpenAI gym benchmarks (Brockman et al., 2016). Specifically, we consider three tasks: `InvertedPendulum-v2`, `Hopper-v2`, and `Ant-v2`. For baselines, we consider MBPO (Janner et al., 2019), PETS (Chua et al., 2018), STEVE (Buckman et al., 2018), SLBO (Xu et al., 2018), and SAC (Haarnoja et al., 2018). A subset of these algorithms were considered for benchmarking deep RL in the recent work of Wang et al. (Wang et al., 2019a). MBPO is the current state of the art model-based method, and SAC is a state of the art model-free algorithm. The hyperparameters for PAL and MAL are specified in Tables 1-3 and related discussion. The results are presented in Figure 5. We find that our methods substantially outperform the baselines. In particular, compared to state of the art MBPO, our method is nearly twice as efficient in InvertedPendulum and Hopper. Our methods are nearly $10\times$ as efficient as other baselines.

In the hopper and ant tasks, we include the velocity of center of mass in the observation space in order to be able to compute the rewards for synthetic rollouts. Similar approaches are followed in prior works as well, e.g. in SLBO (Xu et al., 2018).

Finally, we note that MBPO is a hybrid model-based and model-free method, while our PAL and MAL implementations are entirely model-based. In MBPO, it was noted that long horizon model-based rollouts were unstable and combining with an off-policy critic was important. We find that through our Stackelberg formulation, which is intended to carefully control the effects of distribution shift, we are able to perform rollouts of hundreds of steps without error amplification. As a result, even though our algorithms are purely model based, they can achieve sample efficient learning without loss in asymptotic performance. It is straightforward to extend our PAL and MAL approaches to the hybrid model-based and model-free setting, and could likely lead to a further increase in efficiency for some tasks. We leave exploration of this to future work.

B.3. Model Error Amplification

While the 1-step (prediction) generalization error is easy to measure, it does not provide direct intuitions about the model quality for the purpose of policy improvement. We study error amplification over lookahead horizon to better understand the quality of model for purposes of policy improvement. Let s_0 be the initial state for both M and $\widehat{M}$. We wish to measure

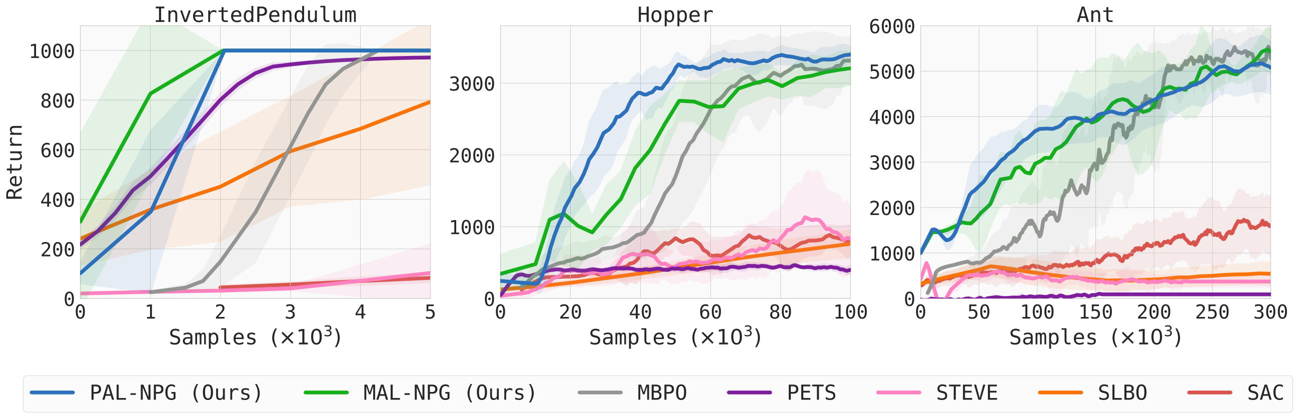


Figure 5. Comparison of results on the OpenAI gym benchmark tasks. Results for the baselines are reproduced from Janner et al. (2019). We observe that PAL and MAL have stable near-monotonic improvement, and substantially outperform the baselines.

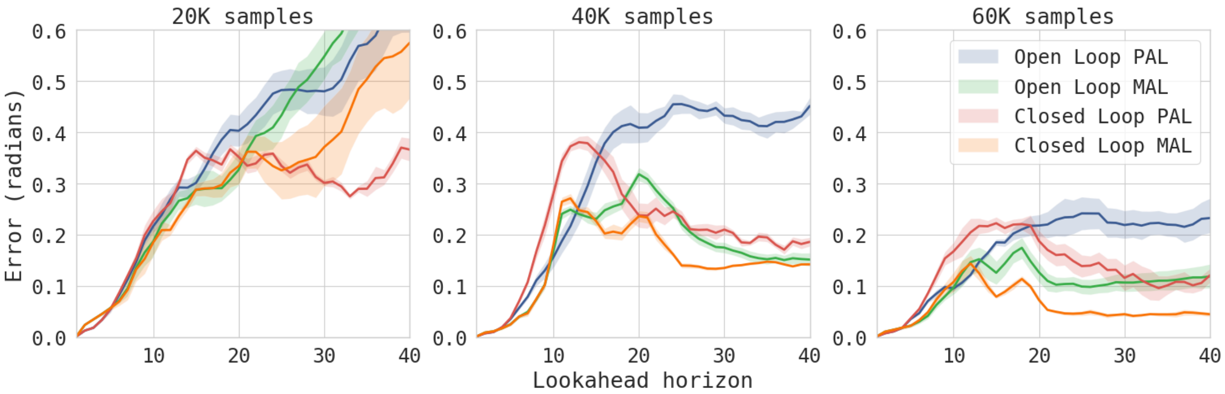


Figure 6. Error amplification over look-ahead horizon in the DClawTurnRandom task. In this experiment, we measure the error of policy $\boldsymbol{\pi}_k$ under model $\widehat{\boldsymbol{M}}_k$ for various stages of training (20K, 40K, and 60K samples). The x-axis is the look-ahead horizon, and the y-axis is the error in prediction of the valve orientation (in radians). See main text for explanation of open loop and closed loop. We observe that: (a) errors decrease with more training; (b) closed loop error is smaller than open loop error; (c) initially PAL has lower error, but finally MAL achieves lower error; (d) long term error is small indicating that the policy-model combination captures the semantics of the task. Note that prediction error of ≈ 0.1 radians in the long term is small.

$L(t) = \mathbb{E}[\|s_t^{\boldsymbol{M}} - s_t^{\widehat{\boldsymbol{M}}}\|]$ where $s_t^{\boldsymbol{M}}$ and $s_t^{\widehat{\boldsymbol{M}}}$ are obtained by following the dynamics of $\boldsymbol{M}$ and $\widehat{\boldsymbol{M}}$ respectively. The state evolution depends on actions, and for this we consider two modes: open loop and closed loop.

In **open loop** mode, we first sample an initial state and set it for both $\boldsymbol{M}$ and $\widehat{\boldsymbol{M}}$, i.e. $s_0^{\boldsymbol{M}} = s_0^{\widehat{\boldsymbol{M}}}$. Subsequently, we execute $\boldsymbol{\pi}$ in $\boldsymbol{M}$ in obtain a trajectory. The action sequence is then executed in open-loop in $\widehat{\boldsymbol{M}}$. Specifically, this makes $a_t^{\widehat{\boldsymbol{M}}} = a_t^{\boldsymbol{M}} = \boldsymbol{\pi}(s_t^{\boldsymbol{M}})$. In **closed loop** mode, we again sample the initial state and set $s_0^{\boldsymbol{M}} = s_0^{\widehat{\boldsymbol{M}}}$. Subsequently, we collect trajectory by indipendetly executing the policy, so that we have: $a_t^{\boldsymbol{M}} = \boldsymbol{\pi}(s_t^{\boldsymbol{M}})$ and $a_t^{\widehat{\boldsymbol{M}}} = \boldsymbol{\pi}(s_t^{\widehat{\boldsymbol{M}}})$.

We study the error for the DClaw task, and plot the error in prediction of faucet angle. This is the primary quantity of interest, since the task involves turning the faucet to the desired orientation. The results are presented in Figure 6. We make the following observations:

1. With more training, the entire profile of errors reduce. This is encouraging, since it suggests that the model quality improves with more training.
2. Closed loop prediction errors are smaller than open loop errors. This suggests that the policy shapes the state visitation to regions where the model is more accurate. Thus, the algorithms we consider ensure that the policy and model remain compatible.
3. During initial stages of training, PAL has lower error. However, towards the end of training, MAL learns more accurate models, by improving the model quality at a faster rate. This is likely due to MAL maintaining a larger replay buffer with more diverse set of transitions obtained by executing very different policies over the course of training. This further underscores why MAL can handle non-stationarities in task and goal distribution as outlined in the main paper.
4. The error does not strictly increase with time. In particular, we observe profiles where the error shrinks towards the end of the horizon. As the policy improves, it turns the faucet to the desired configuration with greater probability. Thus, the long term consequences of the policy are in fact more easily predictable than the intermediate transient effects. This further suggests that the policy-model pair together capture the semantics of the task.