Policy-Aware Model Learning for Policy Gradient Methods

Romina Abachi
Institution 1

Mohammad Ghavamzadeh
Institution 2

Amir-massoud Farahmand
Institution 3

Abstract

This paper considers the problem of learning a model in model-based reinforcement learning (MBRL). We examine how the planning module of an MBRL algorithm uses the model, and propose that model learning should incorporate the way the planner is going to use the model. This is in contrast to conventional model learning approaches, such as those based on maximum likelihood estimation, that learn a predictive model of the environment without explicitly considering the interaction of the model and the planner. We focus on policy gradient planning algorithms and derive new loss functions for model learning that incorporate how the planner uses the model. We call this approach Policy-Aware Model Learning (PAML). We theoretically analyze a model-based policy gradient algorithm and provide a convergence guarantee for the optimized policy. We also empirically evaluate PAML on some benchmark problems, showing promising results.

1 Introduction

A model-based reinforcement learning (MBRL) agent gradually learns a model of the environment as it interacts with it, and uses the learned model to plan and find a good policy. This can be done by planning with samples coming from the model, instead of or in addition to the samples from the environment, e.g., Sutton (1990); Peng and Williams (1993); Sutton et al. (2008); Deisenroth et al. (2015); Talvitie (2017); Ha and Schmidhuber (2018). If learning a model is easier than learning the policy or value function in a model-free manner, MBRL will lead to a reduction in the number of required interactions with the real-world and will improve the sample complexity of the agent. However, this is contingent on the ability of the agent to learn an accurate model of the real environment. Thus, the problem of learning a good model of the environment is of paramount importance in the success of MBRL. This paper addresses the question of how to approach the problem of learning a model of the environment, and proposes a method called *policy-aware model learning* (PAML).

The conventional approach to model learning in MBRL is to learn a model that is a good predictor of the environment. If the learned model is accurate enough, this leads to a value function or a policy that is close to the optimal one. Learning a good predictive model can be achieved by minimizing some form of a probabilistic loss. A common choice is to minimize the KL-divergence between the empirical data and the model, which leads to the Maximum Likelihood Estimator (MLE).

The often-unnoticed fact, however, is that no model can be completely accurate, and there are always differences between the model and the real-world. An important source of inaccuracy/error is the choice of model space, i.e., the space of predictors, such as a particular class of deep neural networks. We suffer an error if the model space does not contain the true model of the physical system.

The *decision-aware model learning* (DAML) viewpoint suggests that instead of trying to learn a model that is a good predictor of the environment, which may not be possible as just argued, one should learn only those aspects of the environment that are relevant to the decision problem. Trying to learn the complex dynamics that are irrelevant to the underlying decision problem is pointless, e.g., in a self-driving car, the agent does not need to model the movement of the leaves on trees when the decision problem is simply to decide whether or not to stop at a red light. The conventional model learning approach cannot distinguish between decision-relevant and irrelevant aspects of the environment, and may waste the "capacity" of

the model on unnecessary details. In order to focus the model on the decision-relevant aspects, we shall incorporate certain aspects of the decision problem into the model learning process.

There are several relatively recent works that can be interpreted as doing DAML, even though they do not always explicitly express their goal as such. Some methods such as Joseph et al. (2013); Silver et al. (2017); Oh et al. (2017); Farquhar et al. (2018) learn a model implicitly, in an end-to-end fashion. This can be interpreted as DAML because the model is learned in service of improving policy performance. Other methods incorporate the value function in model-learning. For example, Value-Aware Model Learning (VAML) is an instantiation of DAML that incorporates the information about the value function in learning the model of the environment (Farahmand et al., 2017; Farahmand, 2018). Recent similar works include Ayoub et al. (2020); Luo et al. (2019). In the latter, the loss is defined to only include the value function learned on the model, whereas Farahmand et al. (2017); Farahmand (2018); Ayoub et al. (2020) require the inclusion of the true value function in their loss as well.

Designing a decision-aware model learning approach, however, is not limited to methods that benefit from the structure of the value function. Policy is another main component of RL that can be exploited for learning a model. Using the policy in model learning is done concurrently by D'Oro et al. (2020) and Schrittwieser et al. (2019). We briefly compare to D'Oro et al. (2020) in Section 3. MuZero (Schrittwieser et al., 2019) takes into account both the value function and policy. The model in MuZero includes separate functions for predicting next "internal" states, policies and values. The policy prediction function learns to predict policies that would be obtained by the MCTS planner that is used to train it. On the other hand, in this work we consider policy gradient planners and instead of predicting policies directly, consider the interaction of the policy and value function in obtaining policy gradients. The high-level idea is simple: If we are using a policy gradient (PG) method to search for a good policy, we only need to learn a model that provides accurate estimates of the PG. All other details of the environment that do not contribute to estimating PG are irrelevant. Formalizing this intuition is the main algorithmic contribution of this paper (Section 3).

Our theoretical contribution is a global convergence guarantee for model-based policy gradient (MBPG). The result shows the effect of policy approximation error, the model error, the number of optimization steps, and a few properties of the MDP and sampling distribution. This is an extension of the recent work by Agarwal et al. (2019) to model-based PG. Moreover, our result introduces a new definition of the policy approximation error, which is perhaps a more accurate way to characterize this error. In the supplementary materials, we extend our results further and consider the error introduced by an imperfect critic. Our empirical contributions are demonstrating that PAML can easily be formulated for a commonly-used PG algorithm and showing its performance in benchmark environments (Section 5), for which the code is made available at `https://github.com/rabachi/paml`. In addition, our results in a finite-state environment show that PAML outperforms conventional methods when the model capacity is limited.

2 Background on Decision-Aware Model Learning (DAML)

A MBRL agent interacts with an environment, collects data, improves its internal model, and uses the internal model, perhaps alongside the real-data, to improve its policy. To formalize, we consider a (discounted) Markov Decision Process (MDP) $(\mathcal{X}, \mathcal{A}, \mathcal{R}^*, \mathcal{P}^*, \gamma)$ (Szepesvári, 2010). We denote the state space by $\mathcal{X}$, the action space by $\mathcal{A}$, the reward distribution by $\mathcal{R}^*$, the transition probability kernel by $\mathcal{P}^*$, and the discount factor by $0 \leq \gamma \leq 1$. In general, the true transition model $\mathcal{P}^*$ and the reward distribution $\mathcal{R}^*$ are not known to an RL agent. The agent instead can interact with the environment to collect samples from these distributions. The collected data is in the form of

$$\mathcal{D}_n = \{(X_i, A_i, R_i, X_i')\}_{i=1}^n, \tag{1}$$

with the current state-action being distributed according to $Z_i = (X_i, A_i) \sim \nu(\mathcal{X} \times \mathcal{A}) \in \bar{\mathcal{M}}(\mathcal{X} \times \mathcal{A})$, the reward $R_i \sim \mathcal{R}^*(\cdot|X_i, A_i)$, and the next-state $X_i' \sim \mathcal{P}^*(\cdot|X_i, A_i)$. Note that $\bar{\mathcal{M}}$ refers to the set of all probability distributions defined over $\mathcal{X}$ and $\mathcal{A}$. In many cases, an RL agent might follow a trajectory $X_1, X_2, \ldots$ in the state space (and similar for actions and rewards), that is, $X_{i+1} = X_i'$. We denote the expected reward by $r^*(x, a) = \mathbb{E}\left[\mathcal{R}^*(\cdot|x, a)\right]$.

A MBRL agent uses the interaction data to learn an estimate $\hat{\mathcal{P}}$ of the true model $\mathcal{P}^*$ and $\hat{\mathcal{R}}$ (or simply $\hat{r}$) of the true reward distribution $\mathcal{R}^*$ (or r^*). This is called *model learning*. These models are then used by a planning algorithm Planner to find a close-to-optimal policy. The policy may be used by the agent to collect more data and improve the estimates $\hat{\mathcal{P}}$ and $\hat{\mathcal{R}}$. This generic Dyna-style (Sutton, 1990) MBRL algorithm is shown in Algorithm 1.

How should we learn a model $\hat{\mathcal{P}}$ that is most suitable for a particular Planner? This is the fundamental question in model learning. The conventional approach in

Algorithm 1 Generic MBRL Algorithm

Initialize a policy π_0
for $k = 0, 1, \ldots, K$ **do**
Generate training set $\mathcal{D}_n^{(k)} = \{(X_i, A_i, R_i, X_i')\}_{i=1}^n$ by interacting with the true environment (potentially using π_k), i.e., $(X_i, A_i) \sim \nu_k$ with $X_i' \sim \mathcal{P}^*(\cdot|X_i, A_i)$ and $R_i \sim \mathcal{R}^*(\cdot|X_i, A_i)$.
$\hat{\mathcal{P}}^{(k+1)} \leftarrow \operatorname{argmin}_{\mathcal{P}\in\mathcal{M}} \text{Loss}_{\mathcal{P}}(\mathcal{P}; \cup_{i=0}^k \mathcal{D}_n^{(i)})$ {**PAML:** $\text{Loss}_{\mathcal{P}} = \|\nabla_\theta J(\mu_\theta^k) - \nabla_\theta \hat{J}(\mu_\theta^k)\|^2_{\cup_{i=0}^k \mathcal{D}_n^{(i)}}$ }
$\hat{r} \leftarrow \operatorname{argmin}_{r\in\mathcal{G}} \text{Loss}_{\mathcal{R}}(r; \cup_{i=0}^k \mathcal{D}_n^{(i)})$
$\pi_\theta^{k+1} \leftarrow \text{Planner}(\hat{\mathcal{P}}, \hat{\mathcal{R}})$ {**PAML:** PG-based (e.g., REINFORCE or DDPG); $\theta_{k+1} \leftarrow \theta_k + \eta \nabla_\theta \hat{J}(\pi_\theta^k)$.}
end for, {Return π_K}

model learning ignores how Planner is going to use the model and instead focuses on learning a good predictor of the environment. This can be realized by using a probabilistic loss, such as KL-divergence, which leads to the maximum likelihood estimate (MLE), or similar approaches. Ignoring how the planner uses the model, however, might not be a good idea, especially if the model class $\mathcal{M}$, from which we select our estimate $\hat{\mathcal{P}}$, does not contain the true model $\mathcal{P}^*$, i.e., $\mathcal{P}^* \notin \mathcal{M}$. This is the model approximation error and its consequence is that we cannot capture all aspects of the dynamics. The thesis behind DAML is that instead of being oblivious to how Planner uses the model, the model learner should pay more attention to those aspects of the model that affect the decision problem the most. A purely probabilistic loss ignores the underlying decision problem and how Planner uses the learned model, whereas a DAML method incorporates the decision problem and Planner.

Value-Aware Model Learning (VAML) is a class of DAML methods (Farahmand et al., 2016, 2017; Farahmand, 2018). It is a model learning approach that is designed for a value-based type of Planner, i.e., a planner that finds a good policy by approximating the optimal value function Q^* by $\hat{Q}^*$, and then computes the greedy policy w.r.t. $\hat{Q}^*$. In particular, the suggested formulations of VAML so far focus on value-based methods that use the Bellman optimality operator to find the optimal value function. More detail about VAML can be found in the supplementary material. The value function and our knowledge about it, however, are not the only extra information that we might have about the decision problem. Another source of information is the policy. The goal of the next section is to develop a model learning framework that benefits from the properties of the policy.

3 Policy-Aware Model Learning

The policy gradient (PG) algorithm and its several variants are important tools to solve RL problems (Williams, 1992; Sutton et al., 2000; Baxter and Bartlett, 2001; Marbach and Tsitsiklis, 2001; Kakade, 2001; Peters et al., 2003; Cao, 2005; Ghavamzadeh and Engel, 2007; Peters and Schaal, 2008; Bhatnagar et al., 2009; Deisenroth et al., 2013; Schulman et al., 2015). These algorithms parameterize the policy and compute the gradient of the performance (cf. (2)) w.r.t. the parameters. Model-free PG algorithms use the environment to estimate the gradient, but model-based ones use an estimated $\hat{\mathcal{P}}$ to generate "virtual" samples to estimate the gradient. In this section, we derive a loss function for model learning that is designed for model-based PG estimation. We specialize the derivation to discounted MDPs, but the changes for the episodic, finite-horizon, or average reward MDPs are straightforward.

A PG method relies on accurate estimation of the gradient. Intuitively, a model-based PG method would perform well if the gradient of the performance evaluated according to the model $\hat{\mathcal{P}}$ is close to the one computed from the true dynamics $\mathcal{P}^*$. In this case, one may use the learned model instead of the true environment to compute the PG. To formalize this intuition, we first introduce some notations.

Given a transition probability kernel $\mathcal{P}^\pi$, we denote by $\mathcal{P}^\pi(\cdot|x;k)$ the future-state distribution of following policy π from state x for k steps, i.e., $\mathcal{P}^\pi(\cdot|x;k) \triangleq (\mathcal{P}^\pi)^k(\cdot|x)$, with the understanding that $(\mathcal{P}^\pi)^0(\cdot|x) = \mathbf{I}$ is the identity map. For an initial probability distribution $\rho \in \bar{\mathcal{M}}(\mathcal{X})$, $\int \rho(\mathrm{d}x)\mathcal{P}^\pi(\cdot|x;k)$ is the distribution of selecting the initial distribution according to ρ and following $\mathcal{P}^\pi$ for k steps. We define a discounted future-state distribution of starting from ρ and following $\mathcal{P}^\pi$ as $\rho_\gamma^\pi(\cdot) = \rho_\gamma(\cdot;\mathcal{P}^\pi) \triangleq (1-\gamma)\sum_{k\geq 0}\gamma^k \int \mathrm{d}\rho(x)\mathcal{P}^\pi(\cdot|x;k)$. We may drop the dependence on π, if it is clear from the context. We use a shorthand notation $\hat{\rho}_\gamma^\pi = \rho_\gamma(\cdot;\hat{\mathcal{P}}^\pi)$, and a similar notation for other distributions, e.g., μ_γ^π and $\hat{\mu}_\gamma^\pi$.

For an MDP $(\mathcal{X}, \mathcal{A}, \mathcal{R}^*, \mathcal{P}, \gamma)$, we use the subscript $\mathcal{P}$ in the definition of value function $V_\mathcal{P}^\pi$ and $Q_\mathcal{P}^\pi$, if we want to emphasize its dependence on the transition probability kernel. We reserve the use of V^π and Q^π for $V_{\mathcal{P}^*}^\pi$ and $Q_{\mathcal{P}^*}^\pi$, the value functions of the true dynamics.

The performance of an agent starting from a user-defined initial probability distribution $\rho \in \bar{\mathcal{M}}(\mathcal{X})$, following policy π in the true MDP $(\mathcal{X}, \mathcal{A}, \mathcal{R}^*, \mathcal{P}^*, \gamma)$ is

$$J(\pi) = J_\rho(\pi) = \int \mathrm{d}\rho(x) V^\pi(x). \tag{2}$$

When the policy $\pi = \pi_\theta$ is parameterized by $\theta \in \Theta$, from the derivation of the PG theorem (cf. proof of Theorem 1 by Sutton et al. 2000), we have that

$$\frac{\partial V^{\pi_\theta}(x)}{\partial \theta} = \sum_{k\geq 0} \gamma^k \int \mathcal{P}^{*\pi_\theta}(\mathrm{d}x'|x;k) \sum_{a'\in\mathcal{A}} \frac{\partial \pi_\theta(a'|x')}{\partial \theta} Q^{\pi_\theta}_{\mathcal{P}^*}(x',a').$$

If the dependence of Q on the transition kernel is clear, we may omit it and simply use Q^{π_θ}. We also use $\mathcal{P}^{\pi_\theta}$ instead of $\mathcal{P}^{*\pi_\theta}$ to simplify the notation. The gradient of the performance $J(\pi_\theta)$ (2) w.r.t. θ is then

$$\nabla_\theta J(\pi_\theta) = \frac{\partial J(\pi_\theta)}{\partial \theta} = \tag{3}$$
$$\sum_{k\geq 0} \gamma^k \int \mathrm{d}\rho(x) \int \mathcal{P}^{\pi_\theta}(\mathrm{d}x'|x;k) \sum_{a'\in\mathcal{A}} \frac{\partial \pi_\theta(a'|x')}{\partial \theta} Q^{\pi_\theta}(x',a').$$

Let us expand the definition of $\nabla_\theta J$, which shall help us easily describe several ways a model-based PG method can be devised. For two transition probability kernels $\mathcal{P}_1$ and $\mathcal{P}_2$, and a policy π_θ, we define

$$\nabla_\theta J(\pi_\theta;\mathcal{P}_1,\mathcal{P}_2) = \sum_{k\geq 0} \gamma^k \int \mathrm{d}\rho(x) \int \mathcal{P}_1^{\pi_\theta}(\mathrm{d}x'|x;k) \sum_{a'\in\mathcal{A}} \frac{\partial \pi_\theta(a'|x')}{\partial \theta} Q^{\pi_\theta}_{\mathcal{P}_2}(x',a'). \tag{4}$$

This vector-valued function can be seen as the PG of following π_θ according to $\mathcal{P}_1$, and using a critic that is the value function in an MDP with $\mathcal{P}_2$ as the transition kernel.

We have several choices to design a model learning loss function that is suitable for a PG method. The overall goal is to match the true PG, i.e., $\frac{\partial J(\pi_\theta)}{\partial \theta} = \nabla_\theta J(\pi_\theta;\mathcal{P}^*,\mathcal{P}^*)$, or an empirical estimate thereof, with a PG that is somehow computed by the model $\hat{\mathcal{P}}$. Let us define $\frac{\partial \hat{J}(\pi)}{\partial \theta} \triangleq \nabla_\theta J(\pi_\theta;\hat{\mathcal{P}},\mathcal{P}^*)$ and set the goal of model learning to

$$\frac{\partial J(\pi_\theta)}{\partial \theta} \approx \frac{\partial \hat{J}(\pi_\theta)}{\partial \theta}. \tag{5}$$

This ensures that the gradient estimate based on following the learned model $\hat{\mathcal{P}}^{\pi_\theta}$ and computed using the true action-value function $Q^\pi_{\mathcal{P}^*}$ is close to the true gradient. There are various ways to quantify the error between the gradient vectors. We choose the ℓ_2-norm of their difference. When the gradient w.r.t. the model is close to the true gradient, we may use the model to perform PG.

Subtracting the gradient of the performances under two different transition probability kernels and taking the ℓ_2-norm, we get a loss function between the true and model PGs, i.e.,

$$c_\rho(\mathcal{P}^{\pi_\theta},\hat{\mathcal{P}}^{\pi_\theta}) = \left\| \frac{\partial J(\pi_\theta)}{\partial \theta} - \frac{\partial \hat{J}(\pi_\theta)}{\partial \theta} \right\|_2 =$$
$$\left\| \sum_{k\geq 0} \gamma^k \int \mathrm{d}\rho(x) \int \left(\mathcal{P}^{\pi_\theta}(\mathrm{d}x'|x;k) - \hat{\mathcal{P}}^{\pi_\theta}(\mathrm{d}x'|x;k) \right) \sum_{a'\in\mathcal{A}} \frac{\partial \pi_\theta(a|x')}{\partial \theta} Q^{\pi_\theta}(x',a') \right\|_2. \tag{6}$$

Note that the summation (or integral) over actions can be done using any of the commonly-known Monte Carlo gradient estimators such as the score function (REINFORCE) or pathwise gradient estimators (Mohamed et al., 2019), as we demonstrate in the Empirical Studies.

Several comments are in order. This is a population loss function, in the sense that $\mathcal{P}$ appears in it. To make this loss practical, we need to use its empirical version. Moreover, this formulation requires us to know the action-value function $Q^{\pi_\theta} = Q^{\pi_\theta}_{\mathcal{P}^*}$, which is w.r.t. the true dynamics. This can be estimated using a model-free critic that only uses the real transition data (and not the data obtained by the model $\hat{\mathcal{P}}^\pi$) . We provide the empirical version of this loss function in the supplementary material.

Also note the loss function $c_\rho(\mathcal{P}^{\pi_\theta},\hat{\mathcal{P}}^{\pi_\theta})$ (6) is defined for a particular policy π_θ. However, policy π_θ gradually changes during the run of a PG algorithm. To deal with this change, we should regularly update $\hat{\mathcal{P}}^{\pi_\theta}$ based on data collected by the most recent π_θ. In the supplementary material, we show that under certain conditions, the error in the PG estimation of a new policy $\pi_{\theta_{\text{new}}}$ using a model that was learned for an old policy $\pi_{\theta_{\text{old}}}$ is $O(\|\theta_{\text{new}} - \theta_{\text{old}}\|)$. Our empirical studies (including in the Supp) show that the model does not expire quickly with policy changes.

Setting $\nabla_\theta J(\pi_\theta;\mathcal{P}^*,\mathcal{P}^*) \approx \nabla_\theta J(\pi_\theta;\hat{\mathcal{P}},\mathcal{P}^*)$ is only one way to define a model learning objective for a PG method. We can opt instead to find a $\hat{\mathcal{P}}$ such that

$$\nabla_\theta J(\pi_\theta;\mathcal{P}^*,\mathcal{P}^*) \approx \begin{cases} \nabla_\theta J(\pi_\theta;\hat{\mathcal{P}},\mathcal{P}^*), & \text{(7a)} \\ \nabla_\theta J(\pi_\theta;\mathcal{P}^*,\hat{\mathcal{P}}), & \text{(7b)} \\ \nabla_\theta J(\pi_\theta;\hat{\mathcal{P}},\hat{\mathcal{P}}). & \text{(7c)} \end{cases}$$

The difference between these cases is in whether the discounted future-state distribution is computed according to the true dynamics $\mathcal{P}^*$ or the learned dynamics $\hat{\mathcal{P}}$, and whether the critic $Q^{\pi_\theta}_{\mathcal{P}}$ is computed according to the true dynamics or the learned dynamics. Case (7a) is the same as (5). Case (7b) uses the model only to train the critic, but not to compute the future-state distribution $\hat{\rho}_\gamma$. Having $\hat{\mathcal{P}}$, the

critic can be estimated using Monte Carlo estimates or any other method for estimating the value function given a model. This is similar to how D'Oro et al. (2020) use their model, though their loss function is a weighted log-likelihood, and the model-learning step does not take the action-value function into account. Case (7c) corresponds to calculating the whole PG according to the model $\hat{\mathcal{P}}^{\pi}$. This requires us to estimate both the future-state distribution and the critic according to the model. In this paper, we theoretically analyze (7a) and provide empirical results for approximations of (7a) and (7b).

4 Theoretical Analysis of PAML

We theoretically study the convergence properties of a model-based PG (MBPG) method. We study how the model error affects the convergence behaviour. The result is generic for any MBPG method, but it further enlightens why PAML might be a good alternative to MLE.

Our result extends Agarwal et al. (2019) from a model-free setting, where the gradients are computed according to the true dynamics $\mathcal{P}^{\pi}$ of the policy, to the model-based setting where the PG is computed according to a learned model, incorporating the model error in the convergence result. We also remove the assumption of access to a perfect critic in our analysis, which we defer to the supplementary material due to space constraints. Additionally, we introduce a new notion of policy approximation error, which is perhaps a better characterization of the approximation error of the policy space. We would like to note that providing global convergence guarantees for PG methods and variants has only recently attracted attention (Agarwal et al., 2019; Shani et al., 2020; Bhandari and Russo, 2019; Liu et al., 2019). This section is a very brief summary of the theoretical results in the supplementary material, which not only include proofs, but also more discussion and intuition.

We analyze a projected PG with the assumption that the PGs are computed exactly. We consider a setup where the performance is evaluated according to a distribution $\rho \in \bar{\mathcal{M}}(\mathcal{X})$, but the PG is computed according to a possibly different distribution $\mu \in \bar{\mathcal{M}}(\mathcal{X})$. To be concrete, let us consider a policy space $\Pi = \{\pi_\theta : \theta \in \Theta\}$ with Θ being a convex subset of $\mathbb{R}^d$ and Proj_Θ be the projection operator onto Θ. Let us denote the best policy in the policy class Π according to the initial distribution ρ by $\bar{\pi}_\rho$ (or simply $\bar{\pi}$, if it is clear from the context), i.e., $\bar{\pi} \leftarrow \operatorname{argmax}_{\pi\in\Pi} J_\rho(\pi)$. We define a function that we call *Policy Approximation Error (PAE)*. Given a policy parameter θ and $w \in \mathbb{R}^d$, and for a probability distribution $\nu \in \bar{\mathcal{M}}(\mathcal{X})$, define

$$L_{\text{PAE}}(\theta, w; \nu) \triangleq \mathbb{E}_{X\sim\nu}\Big[\Big|\sum_{a\in\mathcal{A}}\big(\bar{\pi}(a|X) - \pi_\theta(a|X) - w^\top \nabla_\theta \pi_\theta(a|X)\big) Q^{\pi_\theta}(X,a)\Big|\Big].$$

This can be roughly interpreted as the error in approximating the improvement in the value from the current policy π_θ to the best policy, $\bar{\pi}$, in the class, i.e., $\sum_{a\in\mathcal{A}}(\bar{\pi}(a|X) - \pi_\theta(a|X))Q^{\pi_\theta}(X,a)$, by a linear model $\sum_{a\in\mathcal{A}} w^\top \nabla_\theta \pi_\theta(a|X) Q^{\pi_\theta}(X,a) = w^\top \mathbb{E}_{A\sim\pi_\theta(\cdot|X)}[\nabla_\theta \log \pi_\theta(a|X) Q^{\pi_\theta}(X,a)]$.

This quantity can be compared to the Bellman Policy Error of Agarwal et al. (2019), which is

$$L_{\text{BPE}}(\theta, w; \nu) \triangleq \mathbb{E}_{X\sim\nu}\Big[\Big|\sum_{a\in\mathcal{A}}\Big|\operatorname*{argmax}_{a\in\mathcal{A}} Q^{\pi_\theta}(X,a) - \pi_\theta(a|X) - w^\top \nabla_\theta \pi_\theta(a|X)\Big|\Big|\Big].$$

There are two differences between these two notions of policy approximation errors. The first is that L_{PAE} considers how well one can approximate the best policy in the policy class Π, instead of the greedy policy w.r.t. the action-value function of the current policy as in L_{BPE}. Moreover, while L_{BPE} ignores the value function and its interaction with the policy error, L_{PAE} explicitly considers it. For example, if the reward function is constant everywhere, the action-value function Q^π for any policy would be constant too. In this case, $L_{\text{PAE}}(\theta;\nu)$ is zero (simply choose $w = 0$ as the minimizer), but L_{BPE} may not be. We leave further study of these two policy approximation errors to a future work.

For any $\theta \in \Theta$, we can define the best $w^*(\theta) = w^*(\theta;\nu)$ that minimizes $L_{\text{PAE}}(\theta, w; \nu)$ as

$$w^*(\theta;\nu) \leftarrow \operatorname*{argmin}_{w+\theta\in\Theta} L_{\text{PAE}}(\theta, w; \nu). \tag{8}$$

We use $L_{\text{PAE}}(\theta;\nu)$ to represent $L_{\text{PAE}}(\theta, w^*(\theta);\nu)$. We may drop the distribution ν whenever it is clear from the context. We consider a projected PG algorithm that uses the model $\hat{\mathcal{P}}^{\pi_{\theta_k}}$ and the exact value function Q^{π_k} to compute the gradient, i.e.,

$$\theta_{t+1} \leftarrow \mathsf{Proj}_\Theta\left[\theta_t + \eta \nabla_\theta J_\mu(\pi_{\theta_k}, \hat{\mathcal{P}}^{\pi_{\theta_k}}, Q^{\pi_k})\right], \tag{9}$$

with a learning rate $\eta > 0$ to be specified. Refer to the supplementary materials for results on when an estimated, and thus inexact, model-free critic is used.

We require the following smoothness assumption on the policy space.

Assumption A1 (Assumption 6.12 of Agarwal et al. (2019)) Assume that there exist finite constants $\beta_1, \beta_2 \geq 0$ such that for all $\theta_1, \theta_2 \in \Theta$, and for

all $(x,a) \in \mathcal{X} \times \mathcal{A}$, we have $|\pi_{\theta_1}(a|x) - \pi_{\theta_2}(a|x)| \leq \beta_1 \|\theta_1 - \theta_2\|_2$ and $\|\nabla_\theta \pi_{\theta_1}(a|x) - \nabla_\theta \pi_{\theta_1}(a|x)\|_2 \leq \beta_2 \|\theta_1 - \theta_2\|_2$.

As an example, this assumption holds for an exponential family $\pi_\theta(a|x) \propto \exp\left(\phi^\top(a|x)\theta\right)$ with bounded features $\|\phi(a|x)\|_2 \leq B$. In that case, $\beta_1 = 2B$ and $\beta_2 = 6B^2$.

We are now ready to state the convergence guarantee for the projected PG method for a general policy class.

Theorem 1. *Consider any initial distributions $\rho, \mu \in \bar{\mathcal{M}}(\mathcal{X})$ and a policy space Π parameterized by $\theta \in \Theta$ with Θ being a convex subset of $\mathbb{R}^d$. Assume that all policies $\pi_\theta \in \Pi$ satisfy Assumption A1 and $0 \leq \gamma < 1$. Let T be an integer number. Starting from a $\pi_{\theta_0} \in \Pi$, consider the sequence of policies $\pi_{\theta_1}, \ldots, \pi_{\theta_T}$ generated by the projected model-based PG algorithm (9) with step-size as defined in the Supplementary. Assume that for any policy $\pi_\theta \in \{\pi_{\theta_0}, \ldots, \pi_{\theta_{T-1}}\}$, there exist constants ε_{PAE} and ε_{model} such that $L_{PAE}(\theta; \rho_\gamma^{\bar{\pi}}) \leq \varepsilon_{PAE}$ (policy approximation error) and $\|\nabla_\theta J_\mu(\pi_\theta) - \nabla_\theta \hat{J}_\mu(\pi_\theta)\|_2 \leq \varepsilon_{model}$ (model error). We then have*

$$\mathbb{E}_{t \sim Unif(1,\ldots,T)} \left[J_\rho(\bar{\pi}) - J_\rho(\pi_{\theta_t})\right] \leq \mathcal{O}\left(\frac{\varepsilon_{PAE}}{1-\gamma} + \frac{\varepsilon_{model}}{1-\gamma} + \frac{1}{(1-\gamma)\sqrt{T}}\right).$$

This result shows the effect of the policy approximation error, the model error, and the number of iterations, and order notation is used to drop constants. We observe that the error due to optimization decreases as $\mathcal{O}(\frac{1}{\sqrt{T}})$. The policy approximation error, ε_{PAE}, is similar to the function approximation term (or bias) in supervised learning, and depends on how expressive the policy space is. This term may not go to zero, which means that the projected PG method may not find the best policy in the class.

The model error, $\varepsilon_{\text{model}}$, captures how well one can replace the PG computed according to the true dynamics $\mathcal{P}^\pi$ with the learned dynamics $\hat{\mathcal{P}}^\pi$. Its magnitude depends on how expressive the model class is, the number of samples used in minimizing the loss, etc. Note that this error is the same as the population loss of PAML (6). The PAML objective appears naturally as a factor in the convergence result of a generic MBPG method.

How does PAML compare to MLE as an objective for model learning? Recall that the MLE is the minimizer of the KL-divergence between the empirical distribution of samples generated from $\mathcal{P}^\pi$ and $\hat{\mathcal{P}}^\pi$. On the other hand, the population version of PAML's loss (6) is exactly the error in the PG estimates that we care about. We theoretically analyze the error between $\nabla_\theta J(\pi_\theta)$ obtained following the true model $\mathcal{P}^*$ and $\nabla_\theta \hat{J}(\pi_\theta)$ obtained following $\hat{\mathcal{P}}$, and relate it to the error between the models. We only briefly report this result here, and defer its detailed description to the supplementary material. The result states that for an exponential family parametrization of a policy (see the supplementary for a general policy parameterization), the PG error $\|\nabla_\theta J(\pi_\theta) - \nabla_\theta \hat{J}(\pi_\theta)\|_2$ can be upper bounded by

$$\frac{\gamma Q_{\max} B_2}{(1-\gamma)^2} \times \begin{cases} c_{\text{PG}}(\rho, \nu; \pi_\theta) \left\|\Delta \mathcal{P}^{\pi_\theta}\right\|_{1,1(\nu)}, \\ 2 \left\|\Delta \mathcal{P}^{\pi_\theta}\right\|_{1,\infty}, \end{cases} \tag{10}$$

where B_2 is the ℓ_2-norm of features used in the definition of the policy, the value function is bounded by $Q_{\max}$, $\|\Delta \mathcal{P}^{\pi_\theta}\|_{1,1(\nu)}$ and $\|\Delta \mathcal{P}^{\pi_\theta}\|_{1,\infty}$ are total variation-based norms of the model error $\Delta \mathcal{P}^\pi = \mathcal{P}^\pi - \hat{\mathcal{P}}^\pi$, and $c_{\text{PG}}(\rho, \nu; \pi_\theta) \triangleq \|\frac{\mathrm{d}\rho_\gamma^{\pi_\theta}}{\mathrm{d}\nu}\|_\infty$ is the supremum of the Radon-Nikodym derivative of distribution $\rho_\gamma^{\pi_\theta}$ w.r.t. ν.

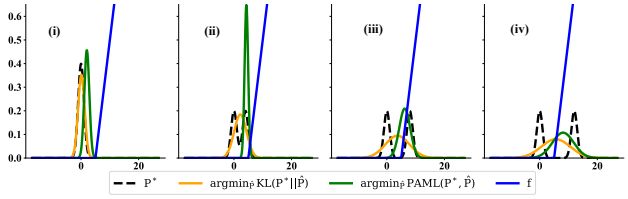
As we show in the supplementary, the R.H.S. of (10) can be further upper-bound by a function of the KL-divergence between the distributions through an application of Pinsker's inequality. This upper bound suggests why PAML might be a more suitable approach in learning a model. An MLE-based approach tries to minimize an upper bound of an upper bound for the quantity that we care about (PG error). This consecutive upper bounding might be quite loose. On the other hand, the population version of PAML's loss (6) is exactly the error in the PG estimates that we care about.

A question that may arise is that although these two losses are different, are their minimizers the same? In Figures 1a and 1b we show through a simple visualization that the minimizers of PAML and KL could indeed be different. We illustrate that the minimizers of PAML and KL are different when we seek to estimate the expectation of a function f, whose weight lies mostly on a specific part of the underlying distribution.

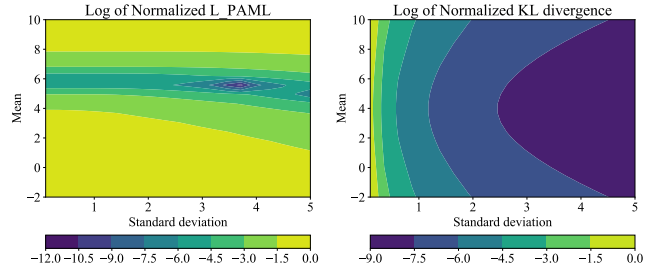
5 Empirical Studies

We compare the performances of PAML and MLE in the framework of Algorithm 1. We first present an illustration of PAML and MLE for a finite-state MDP. We then discuss how the loss introduced in Section 3 can be formulated for PG-based planner DDPG (Lillicrap et al., 2015). Details for reproducing these results and additional experiments can be found in the supplementary materials.

We illustrate the difference between PAML and MLE on a finite 3-state MDP. In this setting, we can calculate exact PGs with no estimation error, and thus the



(a) Visualization of minimizers for PAML and MLE. $\mathcal{P}^*$ is a Gaussian mixture model with 2 modes and the learned model $\hat{\mathcal{P}}$ is a single Gaussian. The loss minimized by PAML for this simple case is: $|\sum_x (\mathcal{P}^* - \hat{\mathcal{P}}) f(x)|^2$, where f is an arbitrary function.



(b) Contours of the two loss surfaces for Figure 1a(iii). Note that the minimizers of PAML and KL-divergence are at different points. (The losses were log-normalized for better visual contrast in this figure.)

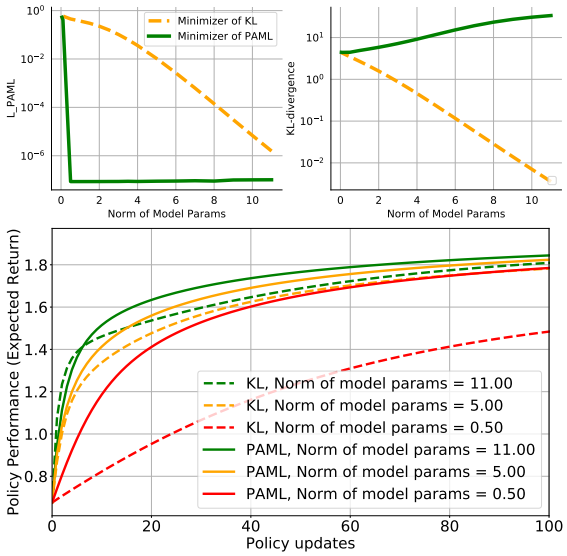


Figure 2: (Top) Comparison of the minimizers of the PAML loss and the KL-divergence as a function of the maximum allowable norm of model parameters. Note that the minimizer of PAML does not necessarily correspond to small KL-divergence. (Bottom) Policy performance as a function of model loss and maximum norm of model parameters. Note that there is no estimation error in this setting. Performance of the PAML agents degrades much less than the performance of the KL agent when the norm of model parameters is constrained.

exact PAML loss and KL-divergence. In these experiments, we use Projected Gradient Descent to update the model parameters and constrain their L_2 norm, in order to limit model capacity. In Figure 2 (Left two), we compare the PAML loss and KL-divergence of models trained to minimize each for a fixed policy. We see that the PAML loss of a model trained to minimize PAML is (expectedly) much lower than that of a model trained to minimize KL. Note that the PAML loss of the KL minimizer decreases as the constraint on model parameters is relaxed, whereas the PAML minimizer is much less dependent on model capacity.

We also evaluate the performances of policies learned using these models, in a process similar to Algorithm 1, but with exact values rather than sampled ones. Referring to Figure 2 (Right), as the norm of the model parameters becomes smaller, the performance of the KL agent drops much more than the PAML agent. However, when the constraint is relaxed (i.e. increased), the KL agent performs similarly to the PAML one. This example provides justification for the use of PAML: when the model space is constrained, such that it does not contain $\mathcal{P}^*$, PAML is able to learn a model that is more useful for planning.

A question that arises is how updates on the policy affect model error since PAML is policy-aware. The top of Figure 3 shows the change in L_{PAML} as a result of a number of policy updates in an epoch (i.e. an epoch refers to each iteration k in 1), while keeping the model fixed in that epoch. The bottom of Figure 3 shows the policy performance over the same epochs. We observe in the 3-State MDP experiments that the change in L_{PAML} decreases as the policy performance improves. This is expected as the policy, and therefore model, converge. We also observe that for higher numbers of policy updates, the performance of the PAML agent does not always show consistent improvement over the KL agent, especially at the beginning of training. This is also expected as the PAML model is only accurate for policies similar to the policy it was trained on. We observe a similar trend for the continuous control experiments in the supplementary. We see that the change in L_{PAML} for more virtual episodes is higher. This is expected as the gradients in this case cannot be exactly computed and so the policy is not necessarily converging to the optimal policy at each timestep, which can also be seen in the performance plots. Thus, the optimal number of policy updates should be tuned according to the dynamics. Another option is to use an objective closer to KL at the beginning of training and fine-tune with PAML as the policy improves. We leave exploration of this option to future work.

We next test PAML on several continuous control environments. We use the actor-critic algorithm

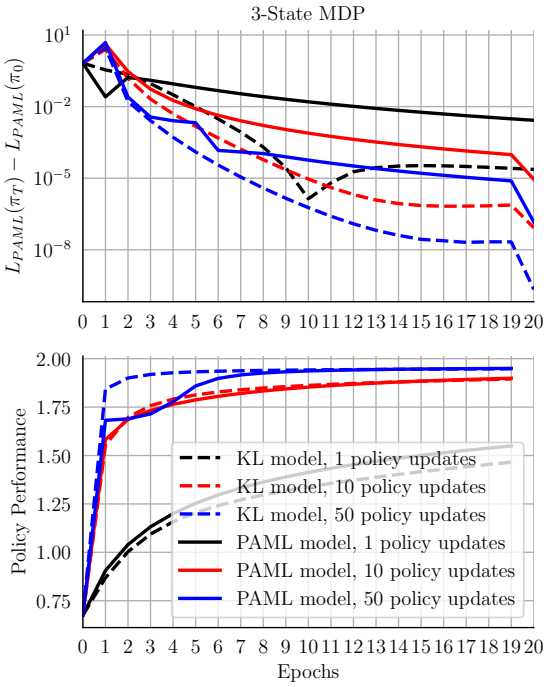


Figure 3: (Top) Difference in PAML loss after a number of policy updates for a fixed model. Note that in this case π_0 would be the policy the model is trained for, and π_T would be the policy with the most number of updates. The model is updated for 200 steps after each set of policy updates (with learning rates in the supplementary). (Bottom) The true policy performance corresponding to each timestep in the Top diagram. The lines for 1 policy update (black) correspond to the plots in Figure 2. Note that there is no source of randomness in these experiments.

Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al., 2015) as the planner for models learned with PAML and MLE.We also evaluate the performance of the model-free method for reference. Although this is a relatively old algorithm, we defer to it due to its simplicity and reasonably good performance on benchmark environments.Our goal with these experiments is not to show state-of-the-art results but rather to demonstrate the feasibility of PAML on high-dimensional problems, and show an example of how the loss in (6) could be formulated.

(additional types in the supplementary):

1. **Random irrel dims:** $(x_t, \eta) \in \mathbb{R}^{d+n}$, $\eta \sim \mathcal{N}_n(0, 1)$

2. **Correlated irrel dims:** $(x_t, \eta_t) \in \mathbb{R}^{d+n}$, where $\eta_t = \eta_0^t$ and $\eta_0 \sim \text{Unif}(0, 1)$. n can be chosen by the user. We show results for a few cases.
3. **Linear redundant dims:** $(x_t, W^T x_t) \in \mathbb{R}^{2d}$, where $W \sim \text{Unif}_{d\times d}(0, 1)$
4. **Nonlinear and linear redundant dims:** $(x_t, \cos(x_t), \sin(x_t), W^T x_t) \in \mathbb{R}^{4d}$, W as in type 3.

In this way, the agent's observation vector is higher-dimensional than the underlying state, and it contains information that would not be useful for a model to learn. In the most general case, this may be replaced by the full-pixel observations, which contain more information than is necessary for solving the problem. To illustrate the differences between model learning methods, we choose to forgo evaluations over pixel inputs for the scope of this work. Although differentiating between useful state variables and irrelevant variables generated by concatenating noise may be overly simplistic (for example, a certain set of pixels could convey both useful and unnecessary information that the model may not know are unnecessary), it is an approximation that can highlight the weakness of purely predictive model learning. The DDPG planner uses a deterministic policy and explores using correlated noise (Lillicrap et al., 2015). The PAML formulation we use for this planner corresponds to case (7a): the model PG is calculated using the future-state distribution of $\hat{\mathcal{P}}^\pi$, and the true PG using samples from $\mathcal{P}^{*\pi}$. The critic for both the model PG and true PG is learned using samples collected from $\mathcal{P}^{*\pi}$. Although here we demonstrate formulating the PAML loss with an actor-critic algorithm as the planner, other PG-based algorithms can also be used.

To train a model using MLE, we minimize the squared ℓ_2 distance between predicted and true next states for all time-steps in a trajectory. Our model in all experiments is deterministic and directly predicts the change in state from the previous to the current timestep.

The results are shown in Figure **??**. In general, PAML performs similarly to MLE in these domains. It seems that the gains that were observed in the tabular domain do not transfer to these domains. This could be due to several factors. For example, it is not clear how to limit the capacity of neural networks as we did for the experiments in Figure 2. Another reason could be that for the planning horizon used (1), the MLE model performs sufficiently well to hide any differences between the models. Additional experiments are in the supplementary materials.

6 Discussion and Future Work

We introduced Policy-Aware Model Learning, a decision-aware MBRL framework that incorporates the policy in the way the model is learned. PAML encourages the model to learn about aspects of the environment that are relevant to planning by a PG method, instead of trying to build an accurate predictive model. We proved a convergence guarantee for a generic model-based PG algorithm, and introduced a new notion of policy approximation error. We em-

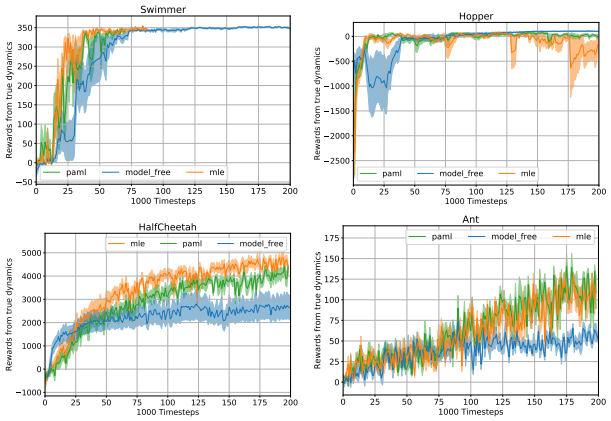


Figure 4: Comparison of policies trained using a model-free method (DDPG, Lillicrap et al. 2015), or by planning using a model learned by PAML, or MLE. Experimental details in the supplementary. Solid lines indicate mean of 5 runs and shaded regions the standard error.

pirically evaluated PAML and compared it with MLE on some benchmark domains. A fruitful direction is deriving PAML loss for other PG methods, especially the state of the art ones.

References

Alekh Agarwal, Sham M. Kakade, Jason D. Lee, and Gaurav Mahajan. Optimality and approximation with policy gradient methods in Markov Decision Processes. *arXiv:1908.00261v2*, August 2019. 2, 5

Alex Ayoub, Zeyu Jia, Csaba Szepesvari, Mengdi Wang, and Lin F Yang. Model-based reinforcement learning with value-targeted regression. In *International Conference on Machine Learning (ICML)*, 2020. 2

Jonathan Baxter and Peter L. Bartlett. Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research (JAIR)*, 15:319–350, 2001. 3

Jalaj Bhandari and Daniel Russo. Global optimality guarantees for policy gradient methods. *CoRR*, abs/1906.01786, 2019. 5

Shalabh Bhatnagar, Richard S. Sutton, Mohammad Ghavamzadeh, and Mark Lee. Natural actor–critic algorithms. *Automatica*, 45(11):2471–2482, 2009. 3

Xi-Ren Cao. A basic formula for online policy gradient algorithms. *Automatic Control, IEEE Transactions on*, 50(5):696–699, 2005. 3

Marc Peter Deisenroth, Gerhard Neumann, and Jan Peters. A survey on policy search for robotics. *Foundations and Trends in Robotics*, 2(1-2):1–142, 2013. 3

Marc Peter Deisenroth, Dieter Fox, and Carl Edward Rasmussen. Gaussian processes for data-efficient learning in robotics and control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(2): 408–423, 2015. 1

Pierluca D'Oro, Alberto Maria Metelli, Andrea Tirinzoni, Matteo Papini, and Marcello Restelli. Gradient-aware model-based policy search. In *Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*, 2020. 2, 5

Amir-massoud Farahmand. Iterative value-aware model learning. In *Advances in Neural Information Processing Systems (NeurIPS - 31)*, pages 9072–9083, 2018. 2, 3

Amir-massoud Farahmand, André M.S. Barreto, and Daniel N. Nikovski. Value-aware loss function for model learning in reinforcement learning. In *13th European Workshop on Reinforcement Learning (EWRL)*, December 2016. 3

Amir-massoud Farahmand, André M.S. Barreto, and Daniel N. Nikovski. Value-aware loss function for model-based reinforcement learning. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1486–1494, April 2017. 2, 3

Gregory Farquhar, Tim Rocktaeschel, Maximilian Igl, and Shimon Whiteson. TreeQN and ATreeC: Differentiable tree planning for deep reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2018. 2

Mohammad Ghavamzadeh and Yaakov Engel. Bayesian policy gradient algorithms. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems (NIPS - 19)*, pages 457–464. MIT Press, Cambridge, MA, 2007. 3

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems (NeurIPS - 31)*, pages 2455–2467, 2018. 1

Joshua Joseph, Alborz Geramifard, John W. Roberts, Jonathan P. How, and Nicholas Roy. Reinforcement learning with misspecified model classes. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 939–946. IEEE, 2013. 2

Sham Kakade. A natural policy gradient. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1531–1538, 2001. 3

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015. 6, 8, 9

Boyi Liu, Qi Cai, Zhuoran Yang, and Zhaoran Wang. Neural proximal/trust region policy optimization attains globally optimal policy. In *Advances in Neural Information Processing Systems (NeurIPS - 33)*, pages 10564–10575. 2019. 5

Yuping Luo, Huazhe Xu, Yuanzhi Li, Yuandong Tian, Trevor Darrell, and Tengyu Ma. Algorithmic framework for model-based deep reinforcement learning with theoretical guarantees. In *International Conference on Learning Representations (ICLR)*, 2019. 2

Peter Marbach and John N. Tsitsiklis. Simulation-based optimization of Markov reward processes. *Automatic Control, IEEE Transactions on*, 46(2):191–209, February 2001. 3

Shakir Mohamed, Mihaela Rosca, Michael Figurnov, and Andriy Mnih. Monte carlo gradient estimation in machine learning. *arXiv preprint arXiv:1906.10652*, 2019. 4

Junhyuk Oh, Satinder Singh, and Honglak Lee. Value prediction network. In *Advances in Neural Information Processing Systems (NIPS - 30)*, pages 6118–6128. Curran Associates, Inc., 2017. 2

Jing Peng and Ronald J. Williams. Efficient learning and planning within the Dyna framework. *Adaptive Behavior*, 1(4):437–454, 1993. 1

Jan Peters and Stefan Schaal. Natural actor-critic. *Neurocomputing*, 71(7):1180–1190, 2008. 3

Jan Peters, Vijayakumar Sethu, and Stefan Schaal. Reinforcement learning for humanoid robotics. In *Humanoids2003, Third IEEE-RAS International Conference on Humanoid Robots*, 2003. 3

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *arXiv preprint arXiv:1911.08265*, 2019. 2

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning (ICML)*, 2015. 3

Lior Shani, Yonathan Efroni, and Shie Mannor. Adaptive trust region policy optimization: Global convergence and faster rates for regularized MDPs. In *AAAI Conference on Artificial Intelligence*, 2020. 5

David Silver, Hado van Hasselt, Matteo Hessel, Tom Schaul, Arthur Guez, Tim Harley, Gabriel Dulac-Arnold, David Reichert, Neil Rabinowitz, André M.S. Barreto, and Thomas Degris. The predictron: End-to-end learning and planning. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 3191–3199, 2017. 2

Richard S. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the 7th International Conference on Machine Learning (ICML)*, 1990. 1, 2

Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems (NIPS - 12)*, 2000. 3, 4

Richard S. Sutton, Csaba Szepesvári, Alborz Geramifard, and Michael Bowling. Dyna-style planning with linear function approximation and prioritized sweeping. In *Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence (UAI)*, 2008. 1

Csaba Szepesvári. *Algorithms for Reinforcement Learning*. Morgan Claypool Publishers, 2010. 2

Erik Talvitie. Self-correcting models for model-based reinforcement learning. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 2597–2603, 2017. 1

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256, 1992. 3