Bayesian-Adaptive Deep Reinforcement Learning using Model Ensembles

Gilwoo Lee gilwoo@cs.uw.edu
Jeongseok Lee jslee02@cs.uw.edu
Brian Hou bhou@cs.uw.edu
Aditya Mandalika adityavk@cs.uw.edu

1 Introduction

Although model-free deep reinforcement learning algorithms have shown tremendous success in a wide range of tasks, such as simulated control problems in OpenAI Gym [**?**] and games like Go [**?**], they face fundamental challenges in their application to physical control problems on robotic hardware. The high sample complexity of current methods results in long training periods if applied directly to hardware, even for simple grasping tasks [**?**]. Furthermore, the safety of the robot or surrounding environment when gathering data from the real world may be difficult to guarantee while the policy is training.

Model-based reinforcement learning techniques offer an opportunity to overcome these issues by taking advantage of simulations of the real systems, which can generate training data faster than real time. The primary challenge with model-based techniques is the discrepancy between simulation and the real world. Simplified models, inaccurate or uncertain parameters that govern the dynamics of the model, and other unmodeled disturbances and noise can render the policies learned on the simulated model ineffective on the real system.

We propose a model-based algorithm that learns a universal policy for Bayes-Adaptive Markov Decision Process (BAMDP) that is robust and optimal to model uncertainty and disturbances. Although the real system's exact physical parameters are unknown, we can maintain a belief distribution over those parameters and provide this belief as an additional input to the policy. This enables the learned policy to be bold when confident in its model and cautious when there is high model uncertainty.

2 Related Work

Our work is closely related to QMDP [**? ?**] which is a Q-value approximation method for Partially Observable Markov Decision Processes (POMDPs). QMDP assumes a fully-observable MDP after one action by approximating the Q-value at the current belief state $b(s)$ as a weighted average over the fully-observable MDP's Q-values $Q_a(b) = \sum_s b(s) Q_{\text{MDP}}(s,a)$. The algorithm requires access to the (approximate) Q-function for the MDP. QMDP performs surprisingly well for many problems, but determinizing the belief distribution after one timestep breaks Bayes-optimality.

The BAMDP formulation that we consider is also similar to the POMDP formulation used in POMDP-lite [**?**], which assumes that the hidden state variables remain constant or only change deterministically. In our case, the hidden state variables correspond to the physics parameters ϕ. Chen et al. show that this formulation is "equivalent to a set of fully observable Markov decision processes indexed by a hidden parameter", which, in our case, is a discretization of ϕ.

Robustness to model uncertainty has also been addressed by EPOpt [**?**], which learns a policy that maximizes the worst-case performance across multiple sampled MDPs. However, as is common with min-max techniques, EPOpt can be quite conservative (and therefore far from optimal) in its policy, especially when the model's possible parameter-space is large. Our work closes this gap in EPOpt by maintaining a belief over the training MDPs to balance between optimality and robustness.

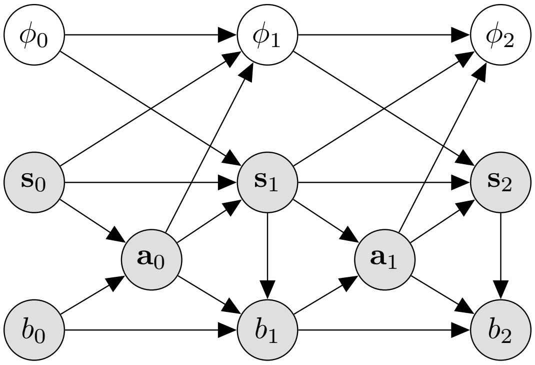


Figure 1: Graphical model for a Bayes-Adaptive Markov Decision Process.

Another recent approach to model uncertainty is UP-OSI [**?**], which utilizes online system identification to estimate the physical parameters of the system. However, UP-OSI only uses the mean estimate of those parameters as an additional input to the policy. Without a notion of belief and uncertainty in its parameter estimates, UP-OSI is prone to aggressively executing policies without being conservative toward remaining model uncertainty.

TODO: mention [**?**] and other bayesian work.

3 Bayes-Adaptive Reinforcement Learning

We follow the BAMDP framework (Figure 1), which assumes that a latent variable ϕ governs the transition function of the underlying Markov Decision Process [**? ? ?**]. A BAMDP is defined by a tuple $\langle \mathcal{S}', \mathcal{A}, P, P_0, R \rangle$, where

- $\mathcal{S}' = \mathcal{S} \times \Phi$ is the set of hyper-states (state s, latent variable ϕ),
- $\mathcal{A}$ is the set of actions,
- $P(s', \phi' | s, \phi, a)$ is the transition function between hyper-states, conditioned on action a being taken in hyper-state (s, ϕ),
- $P_0 \in \mathcal{P}(\mathcal{S} \times \Phi)$ combines the initial distribution over hyper-states,
- $R(s, \phi, a)$ represents the reward obtained when action a is taken in hyper-state (s, ϕ).

In our setting of robot control, the latent variables ϕ are physics parameters such as mass and length of different robot links (TODO: better word than links) (JS: robot bodies?).

Our goal is to find the Bayes-optimal policy for the following Bellman equaton:

$$V^*(b, s) = \max_a \left\{ R(s, b, a) + \gamma \sum_{s'} P(s', b' | s, b, a) V^*(b', s') \right\} \tag{1}$$

where $b(\phi)$ is the belief over the set of latent physics parameters $\phi \in \Phi$,

$$\begin{aligned} R(s, b, a) &= \sum_{\phi \in \Phi} b(\phi) R(s, \phi, a) \\ P(s', b' | s, b, a) &= \sum_{\phi \in \Phi} b(\phi) P(s', b' | s, \phi, a) \\ &= \sum_{\phi \in \Phi} b(\phi) P(s' | s, \phi, a) \sum_{\phi' \in \Phi} P(\phi' | s, \phi, a). \end{aligned}$$

TODO(?): mention PAC-MDP?

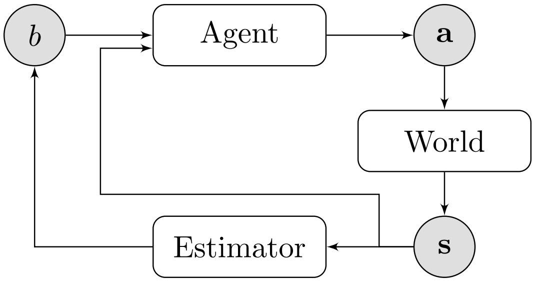


Figure 2: System structure

3.1 Belief Update

We make a simplification to the BAMDP formulation by assuming that the latent variable ϕ is constant or changing deterministically according to a known transition function. As in POMDP-lite[**?**], we can then view the simplified BAMDP model as a set of MDPs that are indexed by the latent variable ϕ.

The posterior probability of an MDP governed by latent physics variable ϕ given a trajectory $\tau_{0:H} = (s_0, a_0, \cdots, s_H)$ is given by:

$$\begin{aligned} P(\phi_H|\tau_{0:H}) &= \frac{1}{Z} P(\phi_0) P(\tau_{0:H}|\phi_0) \\ &= \frac{1}{Z} P(\phi_0) \prod_{t=0}^{H-1} P(s_{t+1}, \phi_{t+1}|s_t, \phi_t, a_t) \\ &= \frac{1}{Z} P(\phi_0) \prod_{t=0}^{H-1} P(s_{t+1}|s_t, \phi_t, a_t) P(\phi_{t+1}|s_t, \phi_t, a_t) \end{aligned} \tag{2}$$

where Z is a normalizing constant and $P(\phi_{t+1}|\cdot) = 1$ is the deterministic transition of ϕ.

4 Bayes-Adaptive Policy Gradient

The main contribution of our work is a policy-gradient algorithm which produces a universal policy that maintains different strategies for different beliefs. The key idea is to augment the state with the belief and provide this augmented feature as input to the policy. The policy is a function of both state and belief: $\pi : \mathcal{S} \times \mathcal{B} \to P(\mathcal{A})$, where $\mathcal{B}$ is the belief space. This allows the policy to differentiate between the beliefs while sharing globally effective strategies.

In order for this algorithm to work in practice, we need an effective representation of the belief space. If the space of MDPs $\mathcal{M}$ is finite, the belief b can be a vector of size $|\mathcal{M}|$. If $\mathcal{M}$ is infinite, we propose two methods to approximate the belief space:

- Sample K MDPs from the prior belief distribution b_0 and use belief over these K MDPs
- Discretize the parameter space and use belief over the parameter space

In the first case, we use a vector of size K as b; K should be large enough to approximate all MDPs with the K sampled ones. In the second case, assuming the latent parameters are independent, the belief is represented as a vector of size $(N_1 + ... + N_M)$ where M is the total number of parameters and N_m is the discretization of the m-th parameter. In this project, we have implemented the first one, and leave the second one for future work.

The algorithm is shown in Algorithm 1. At each iteration, we sample a number of MDPs from the prior distribution b_0, and sample a trajectory for each MDP using the current policy. During the rollout of the trajectory, the estimator provides the updated belief estimate as an additional feature to the policy. The estimator resets to b_0 at the beginning of each trajectory.

Note that the policy observes the *evolution* of belief along each trajectory. This allows the policy to learn to be optimal with respect to the evolution of belief, and to take actions which affect the

Algorithm 1 Bayes-Adaptive Policy Gradient

```
Require: Bayes-Estimator $ES$, $b_0$, $\mathcal{M}$, initial policy $\theta_0$, $n_{\text{itr}}$, $n_{\text{sample}}$
for $i = 1, 2, \ldots, n_{\text{itr}}$ do
    for $n = 1, 2, \ldots, n_{\text{sample}}$ do
        Reset $ES$ with $b_0$
        Sample model parameter $\phi \sim b_0$ to get $M_\phi$
        Sample a trajectory $\tau_n$ from $M_\phi$ using policy $\pi(\theta_i)$ and estimator $ES$
    Update policy: $\theta_{i+1} = \texttt{BatchPolicyOptimization}(\theta_i, \{\tau_1, ..., \tau_{n_{sample}}\})$
```

evolution. For example, given prior belief b_0 with high entropy (i.e., high uncertainty), our algorithm policy would produce conservative or informative actions until b becomes informative enough along the course of the trajectory. This is a key difference between our algorithm and UP-OSI [**?**], in which the universal policy has no notion of belief or uncertainty.

5 Experiments

Our algorithm falls under the class of model-based algorithms that consider multiple MDPs to train a policy. We therefore compare our algorithm against (1) QMDP and (2) EPOpt. QMDP makes the assumption that ϕ is revealed after one step, and takes the best action given $b(\cdot)$, i.e. $a^* = \max R(s, b, a) + \sum_{\phi \in \Phi} b(\phi) Q(s, a, \phi)$. EPOpt is a minimax-style algorithm which trains the policy to be robust across multiple MDPs, without utilizing the belief over MDPs in learning the policy.

We evaluate the three algorithms on a set of simulated benchmark examples: **ant** (TODO cite), **swimmer** (TODO cite) and the **half-cheetah** (TODO cite) using the MuJoCo physics simulator (TODO cite). Owing to the numerous model parameters that can be varied to simulate multiple possible MDPs, these benchmarks prove to be ideal to test robustness against model discrepancies 1. Each of the agents is provided a pre-defined reward function as in OpenAI Gym [**?**].

We implemented EPOpt as illustrated in [**?**] with TRPO [**?**] provided by `rllab` [**?**] as the underlying batch policy optimization sub-routine.

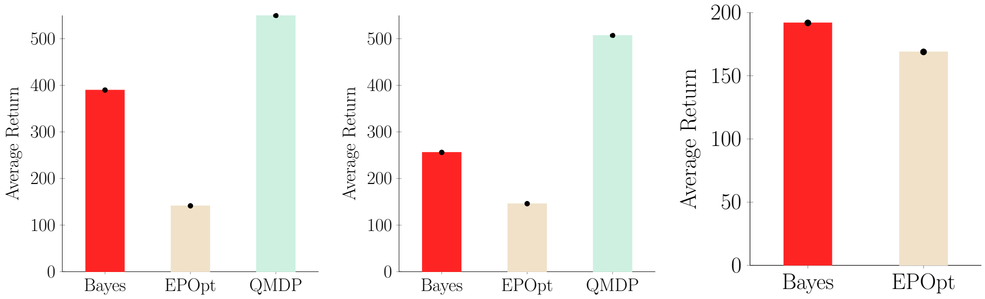
To implement QMDP and BAPG, for each agent, we uniformly sampled 20 MDPs across a range of parameters as in 1 (e.g., body link length, mass, geometry size, joint damping, friction). QMDP used TRPO policies trained on the chosen 20 MDPs to rollout trajectories and approximate Q-functions. BAPG used these as the set of MDPs with which it approximates the belief space.

All the policy networks involved in the implementation of EPOpt and QMDP were Gaussian MLPs with two hidden layers $(64, 64)$. To account for a larger input space involving beliefs, the policy network for Bayes-Adaptive Policy Gradient was architect as a Gaussian MLP with hidden layers $(128, 64)$.

5.1 Results

Hypothesis: Policies learned via BAPG balance robustness and optimality.

We successfully verify our hypothesis by comparing the three algorithms on two factors, namely, robustness to model discrepancy and performance. In Figure 3a we plot the average (undiscounted) return computed over 10 rollouts of the learned policy on an MDP uniformly sampled from the parameter space of half-cheetah. We observe that our algorithm outperforms EPOpt significantly and close behind QMDP. We attribute this the fact that the belief converges fairly soon in the trajectories and therefore QMDP , with access to optimal Q-functions outperforms our algorithm. We leave the experiments that involve sudden changes in the MDP to future work, where we believe QMDP would fail to perform as well. In Figures 3b and 3c, we evaluate the performance of the algorithms on one of the 20 MDPs used for training. QMDP clearly outperforms the other two algorithms as it has access to the optimal Q-functions for each of these MDPs. We note that our algorithm again outperforms EPOpt on these select MDPs, which is expected since EPOpt does not take advantage of these MDPs during training.



(a) Nominal MDP

(b) One of 20 MDPs

(c) Over 20 MDPs

Figure 3: Return on a uniformly sampled and prechosen MDPs.

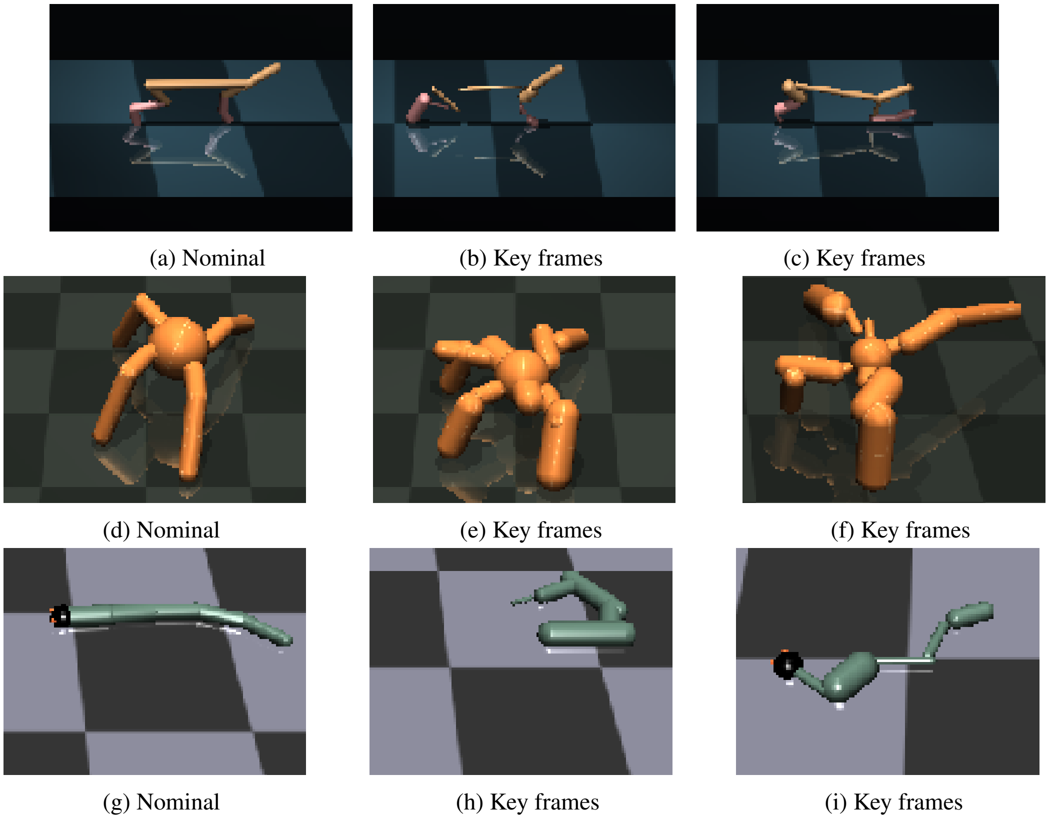
Our experiments demonstrate the robustness of our algorithm in handling model discrepancy over a wide range of parameter space. We learn one universal policy that adapts to different MDPs while maintaining a belief over the sampled MDPs. In Figure 4, we note the MDPs to be geometrically significantly different that result in different dynamics. Our learned policy network is capable of generating different policies for each of these MDPs to achieve the optimal returns. See `http://shorturl.at/efxBI` for videos.

Table 1: Model Parameter Variation

Parameter	Variation
Damping	±50%
Friction	±50%
Body link size	±50%

6 Conclusion and Future Work

We presented a policy-gradient algorithm that, by maintaining a belief over the MDPs, achieves robustness to model variations and disturbances without compromising significantly on the optimality. While in this work our belief was over discrete MDPs, there is a natural extension to maintaining a continuous belief distribution either over the MDPs or over the parameter space directly. A possible extension to enable exploration and distinguish policies (and not just beliefs) is to augment the reward to encourage actions that achieve an information gain. An immediate interesting avenue worth exploring is to exploit the framework of mixture of experts to consider the action proposals from the MDPs along with the belief.


(a) Nominal

(b) Key frames

(c) Key frames

(d) Nominal

(e) Key frames

(f) Key frames

(g) Nominal

(h) Key frames

(i) Key frames

Figure 4: Keyframes from rollouts on various MDPs. Some of the MDPs are gemeotrically significatnly differnt that they require drastically different policies to achieve optimal returns. See `http://shorturl.at/efxBI` for videos.