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

Learning optimal behaviors and decision-making in the face of uncertainty and disturbances has been a major research area of interest to the robotics community. Although model-free deep reinforcement learning algorithms have shown tremendous success in a wide range of tasks such as simulated control problems, and games like Go and Poker, they face fundamental challenges in their application to physical control systems like robots. The need for large amounts of data and entailing huge learning times, high sample complexity and issues of safety in gathering data from the real world are some of the major barriers to break to directly apply these methods on real systems.

Model-based reinforcement learning techniques offer an opportunity to overcome these issues by taking advantage of simulations of the real systems. The primary challenge, however, with model-based techniques is the apparent discrepancy between the simulation models of the physical system. Simplified models, inaccurate or uncertain parameters that govern the dynamics of the model, and other unmodelled disturbances and noise can render the policies learned for the model, brittle. Identifying the need for robustness, in this work, we propose a model-based algorithm to learn an ensemble policy for Bayes-Adaptive Reinforcement Learning that elegantly handles model uncertainties and disturbances. <Mention the key idea and the contributions>

2 Related Work

Our work is closely related to QMDP [**?**, **?**] which is an approximation for POMDP. QMDP approximates POMDP by assuming fully-observable MDP after 1-step, and approximating the Q-value at the current belief state $b(s)$ as $Q_a(b) = \sum_s b(s) Q_{\text{MDP}}(s, a)$. The algorithm requires access to the (approximate) optimal Q-functions of the MDPs. Although it is assured to perform well amongst these MDPs for which the algorithm has access to the optimal Q-functions, the assumption of determinized belief after 1-step breaks Bayes-optimality.

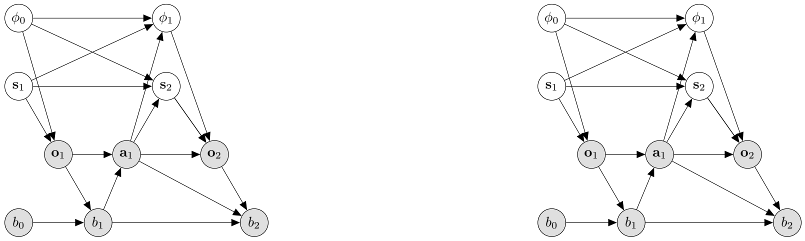
Robustness to model uncertainty has also been addressed by EPOpt [**?**] which learns a policy that maximized the worst-case performance across multiple MDPs. However, as common with min-max techniques, EPOpt can be quite conservative in its policy and therefore far from optimal, especially when the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

Another recent work, UP-OSI, trains on multple possible MDPs and utilizes online system identification to learn a policy that handles model uncertainty and sudden changes in the environment. However, without a notion of belief, the algorithm is prone to aggressively execute policies that might not be conservative under uncertainty.

The BAMDP formulation that we consider is also similar to POMDP formulation used in POMDP-lite [**?**] which assumes that the hidden state variables are constant or only change deterministically. In our case, the hidden state variables correspond to the physics parameters ϕ. The authors of POMDP-lite have shown that such 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.

3 Methods and Algorithm Description

3.1 Bayes-Adaptive Reinforcement Learning



(a) Graphical Model

(b) System structure

We assume that there exists a latent physics variable ϕ which determines the transition function of the underlying MDP, i.e., the transition function $P(s', \phi'|s, \phi, a)$ is now a function of state, action, and ϕ. We would like to learn a policy which maximizes the long term reward given ϕ. Formally, this is called a Bayes-Adaptive MDP [?, ?, ?], 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 (states, physics variable),
- $\mathcal{A}$ 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).

We would like to find the Bayes-optimal policy for the following Bellman equaton:

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

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

We make a simplification to the BARL formulation. We assume that the latent variable ϕ is either constant or the rate of change is slow enough that approximating the long-term value with a determinized ϕ is a reasonable short-term approximation for choosing one-step action, i.e. we can treat $V^*(s_t, \phi_t) \approx V^*(s_t, \phi_{t:\infty})$ for the purpose of one-step Bellman update.

This assumption allows us to simplify BARL with an ensemble policy learning method. At a high level, we have a network that updates the *belief* of the physics parameters at time t,

$$b(\phi_t) = P(\phi_t | s_{t-1}, \phi_{t-1}, a_{t-1})$$

which is then used to compute the best policy from an ensemble of ϕ-dependent optimal policies, i.e., $\pi^*(\cdot; \phi)$ and $V^*(\cdot; \phi)$ are computed with typical RL algorithms for MDPs. Then the remaining task is to compute the one-step best action a:

$$a^* = \arg\max_a \mathbb{E}_{\phi \sim b(\phi)} \left[R(s, a, \phi) + \gamma \sum_{s', \phi'} P(s', \phi'|s, \phi, a) V^{*'}(s', \phi') \right]. \tag{2}$$

The probability of an MDP $\mathcal{M}$, governed by latent physics variable $\phi_{\mathcal{M}}$, generating a trajectory τ is given by:

$$\begin{aligned} P(\mathcal{M}|\tau) &= \frac{1}{Z} P(\tau|\mathcal{M}) \times P(\phi_{\mathcal{M}}) \\ &= \frac{1}{Z} \prod_{t=0}^{T-1} P(s = s_{t+1}|s_t, a_t, \phi_{\mathcal{M}}) \times P(\phi_{\mathcal{M}}) \end{aligned} \tag{3}$$

where Z is the normalizing constant that allows the sum of probabilities over all the MDPs to be unity. These probabilities are used to update the belief.

Algorithm 1 `Bayesian-DRL`

```
Statement and a comment.                                  ▷ Initialization
for each statement in this for loop do                    ▷ Hi there
    down a can of beer. :P
okie-dokey.
while JS is drawing his error bars do
    add text.
    if I mess up then
        Gilwoo will correct it ha!
    else
        we are screwed
return brian's awesome results.
```

3.2 Bayes-Adaptive Policy Network

TODO: Explain how the system works (input to the network, etc.)

4 Analysis

5 Results

6 Future Work

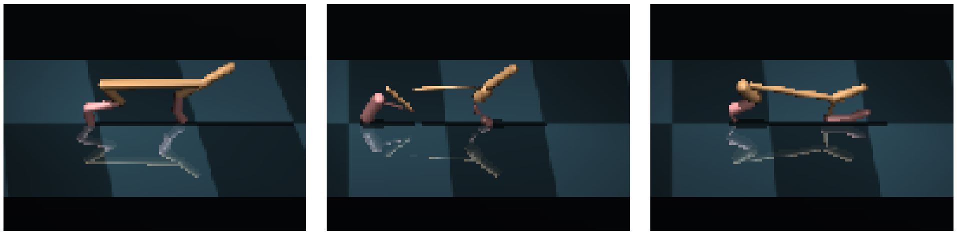
7 Related Works

8 Experiments

We have setup a set of simulated examples and a set of RL algorithms to be utilized in our ensemble approach. For simulated examples, we have the following agents: **ant, reacher, swimmer, half-cheetah**, each with a predefined reward function defined similar to those given by OpenAI Gym [**?**]. We are utilizing TRPO [**?**], VPG, DDPG [**?**] provide by `rllab` [**?**] as a set of algorithms to be utilized in our ensemble approach. In addition, we plan to implement PPO and a PID controller for RACECAR.

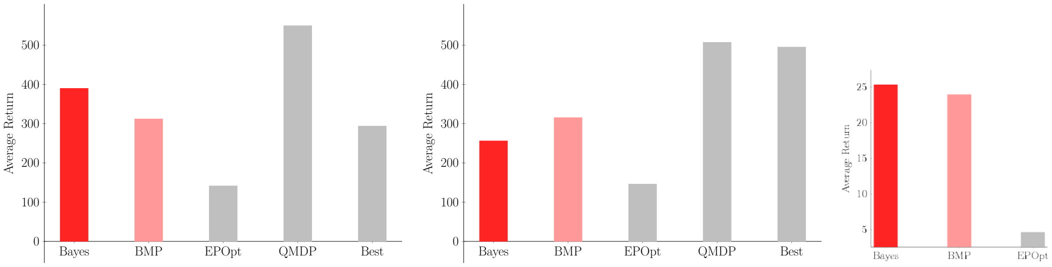
Our algorithm will be compared against two classes of algorithms: (1) sample-based algorithms which chooses an MDP and commit to this policy for a fixed horizon, and (2) ensemble algorithms which train a policy over an ensemble of MDP models. A greedy algorithm which chooses the maximum-likely MDP, or one that samples from a posterior distribution of MDPs given previous observations (e.g. Posterior Sampling Reinforcement Learning [**?**]) would fall into the former, and EPOpt[**?**] and Ensemble-CIO [**?**] would fall into the latter.

We are currently in the process of setting up baseline algorithms which may be used for direct comparison or as internal policy update algorithms for each of MDP in our algorithm.



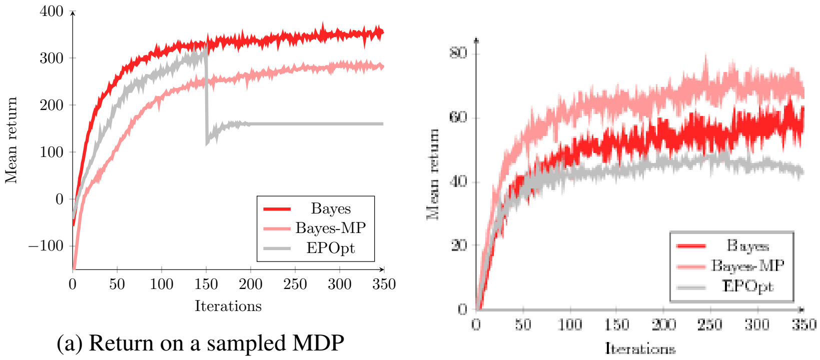
(a) Nominal (b) Key frames (c) Key frames

Figure 2: Keyframes from rollouts on various MDPs. The universal policy network learns to use different policies on different MDPs.



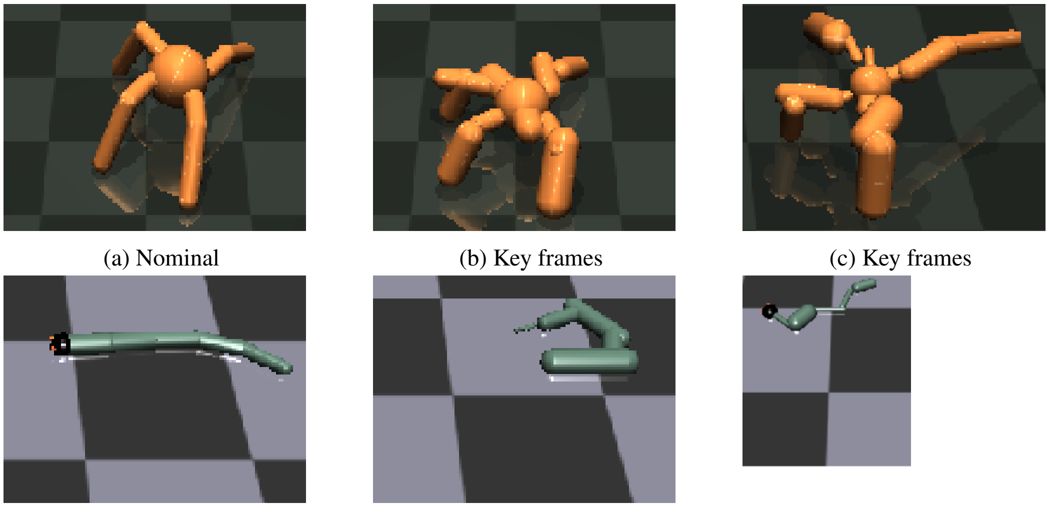
(a) Return on a sampled MDP (b) Return on one of the chosen MDPs (c) Average return

Figure 3: Return on sampled and prechosen MDPs. BARL is better than EPOpt. On average across K pre-chosen MDPs, it outperforms EPOpt by a large margin. TODO: remove BMP.



(a) Return on a sampled MDP

Figure 4: Training curves



(a) Nominal

(b) Key frames

(c) Key frames

(d) Nominal

(e) Key frames

(f) Key frames

Figure 5: Keyframes from rollouts on various MDPs. Some of the MDPs are gemeotrically significatnly differnt that they require drastically different policies to achieve optimal returns.