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 [1] and games like Go [2], 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 [3]. 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 MDP 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 [4, 5] 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 [6], 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 [7], 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.

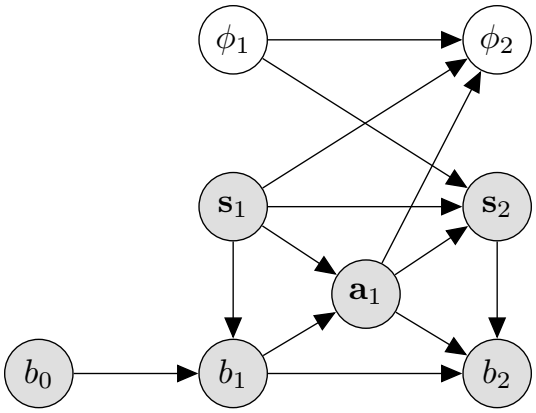


Figure 1: Graphical model for a Bayes-Adaptive Markov Decision Process.

Another recent approach to model uncertainty is UP-OSI [8], 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 [9] and other bayesian work.

3 Bayes-Adaptive Reinforcement Learning

We follow the Bayes-Adaptive Markov Decision Process framework (Figure 1), which assumes that a latent variable ϕ governs the transition function of the underlying Markov Decision Process [10–12]. A Bayes-Adaptive MDP 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).

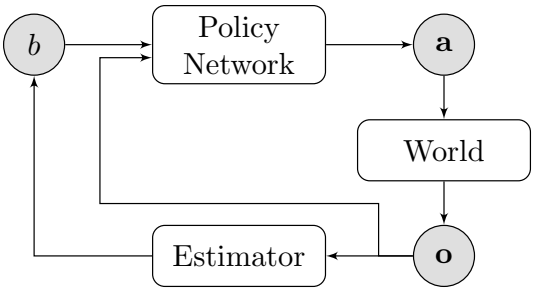
Our goal is to find the Bayes-optimal policy for the following Bellman equaton:

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

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?



(a) 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[6], 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 $|\mathcal{M}|$. In case where $\mathcal{M}$ 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 as 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 latest 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`

Bayes-Estimator ES, b_0, θ_0, n_{itr}, n_{sample}, $\mathcal{M}$ ▷ Initialization

for i = 1, 2, ..., n_{itr} **do**
  for n = 1, 2 , ..., n_{sample} **do**
    Reset ES with b_0
    Sample model parameter $\phi \sim b_0$ to get M_ϕ
    Sample a trajectory τ_n from M_ϕ using policy $\pi(\theta_i)$ and estimator ES
Update policy: $\pi(\theta_{i+1}) = BatchPolOpt(\theta_i, \{\tau_1, ..., \tau_{n_{sample}}\})$

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 [8], in which the policy has no notion of belief or uncertainty.

5 Analysis

- By augmenting the observation with a belief over MDPs, policy networks can learn to be robust against model uncertainty while maintaining some of the "optimal" actions w.r.t. each MDP.
- When the optimal policies across MDPs have a lot in common (e.g., Swimmer), simple "interpolation" of the deterministic policies provide good action proposals, suggesting that a mixture of policies (with a large number of policies that cover the space), may reduce sample complexity and offer even better performance.

6 Results

- MuJoCo environments: Ant, Swimmer, HalfCheetah
  - Uniformly sampled 20 MDPs across a range of parameters (e.g., body link length, mass, geometry size, joint damping, friction)
- Algorithm Implementations:
  - All policy networks were Gaussian MLPs with two layers, (64, 64), except for Bayes ones which had (128, 64) to account for the larger input space
  - TRPO implementation from `rllab`
  - EPOpt implemented based on the paper
  - Bayes-Mixture Policy uses TRPO policies trained on the 20 MDPs to mix actions based on the belief
  - QMDP uses TRPO policies trained on the 20 MDPs to rollout trajectories and approximate Q-functions

7 Future Work

- Extend to a continuous version which has Gaussian belief distribution as input
- Bootstrap a set of stochastic/deterministic policies, each trained for one MDP or a small range of parameters
- Train with additional reward bonus for information gain which helps distinguishing policies (not just beliefs)

8 Related Works

9 Experiments

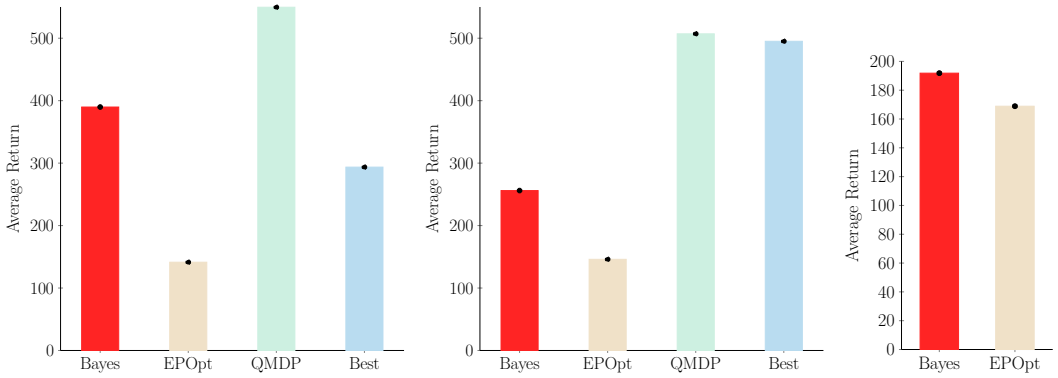
We have setup a set of simulated examples and a set of RL algorithms to be utilized in our ensemble approach. For simulated examples, we have the following agents: **ant, reacher, swimmer, half-cheetah**, each with a predefined reward function defined similar to those given by OpenAI Gym [13]. We are utilizing TRPO [14], VPG, DDPG [15] provide by `rllab` [16] 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 [17]) would fall into the former, and EPOpt[7] and Ensemble-CIO [18] 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.

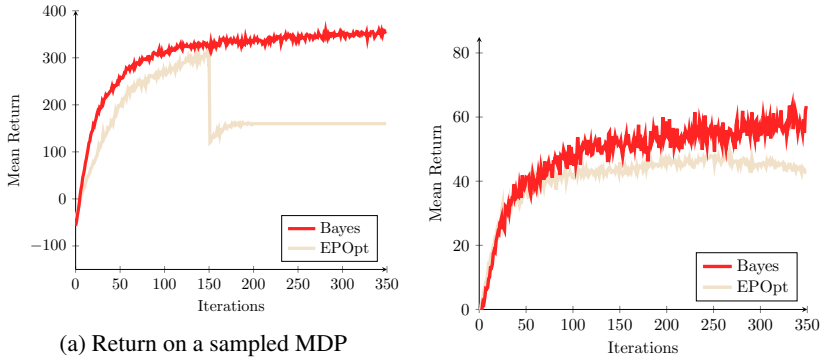
(a) Nominal (b) Key frames (c) Key frames

Figure 3: 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 4: 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

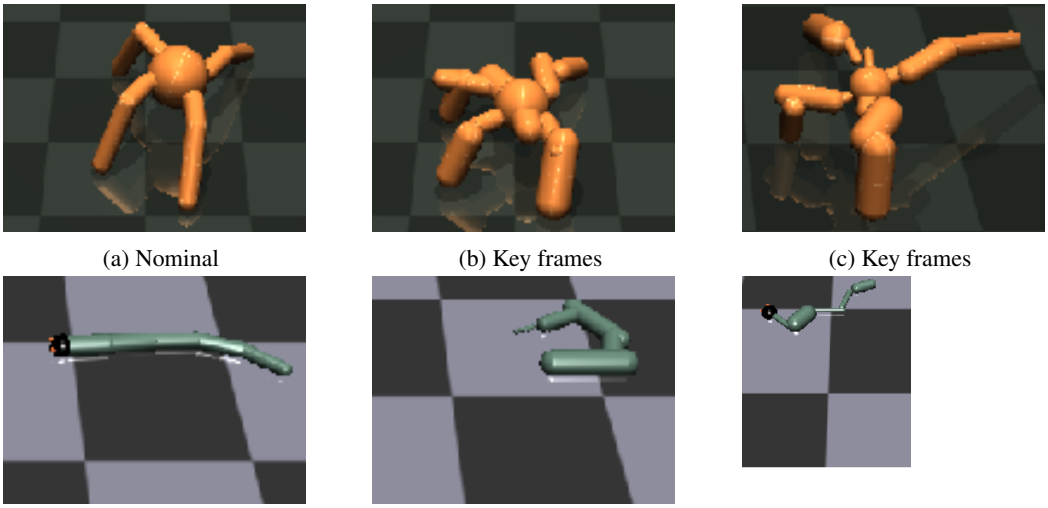
Figure 5: Training curves

References

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

[2] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, p. 354, 2017.

[3] S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen, "Learning hand-eye coordination for robotic grasping with large-scale data collection," in *International Symposium on Experimental Robotics*, pp. 173–184, Springer, 2016.



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

(d) Nominal (e) Key frames (f) Key frames

Figure 6: Keyframes from rollouts on various MDPs. Some of the MDPs are gemeotrically significatnly differnt that they require drastically different policies to achieve optimal returns.

[4] M. L. Littman, A. R. Cassandra, and L. P. Kaelbling, "Learning policies for partially observable environments: Scaling up," in *Machine Learning Proceedings 1995*, pp. 362–370, Elsevier, 1995.

[5] P. Karkus, D. Hsu, and W. S. Lee, "QMDP-Net: Deep Learning for Planning under Partial Observability," in *Advances in Neural Information Processing Systems*, pp. 4697–4707, 2017.

[6] M. Chen, E. Frazzoli, D. Hsu, and W. S. Lee, " POMDP-lite for Robust Robot Planning under Uncertainty," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pp. 5427–5433, IEEE, 2016.

[7] A. Rajeswaran, S. Ghotra, B. Ravindran, and S. Levine, "EPOpt: Learning Robust Neural Network Policies Using Model Ensembles," *arXiv preprint arXiv:1610.01283*, 2016.

[8] W. Yu, J. Tan, C. K. Liu, and G. Turk, "Preparing for the unknown: Learning a universal policy with online system identification," *arXiv preprint arXiv:1702.02453*, 2017.

[9] A. Guez, N. Heess, D. Silver, and P. Dayan, "Bayes-adaptive simulation-based search with value function approximation," in *Advances in Neural Information Processing Systems*, pp. 451–459, 2014.

[10] M. Ghavamzadeh, S. Mannor, J. Pineau, A. Tamar, *et al.*, "Bayesian Reinforcement Learning: A Survey," *Foundations and Trends® in Machine Learning*, vol. 8, no. 5-6, pp. 359–483, 2015.

[11] S. Ross, B. Chaib-draa, and J. Pineau, "Bayes-Adaptive POMDPs," in *Advances in Neural Information Processing Systems*, pp. 1225–1232, 2008.

[12] A. Guez, D. Silver, and P. Dayan, "Efficient Bayes-Adaptive Reinforcement Learning using Sample-Based Search," in *Advances in Neural Information Processing Systems*, pp. 1025–1033, 2012.

[13] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI Gym," 2016.

[14] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International Conference on Machine Learning*, pp. 1889–1897, 2015.

[15] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[16] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking Deep Reinforcement Learning for Continuous Control," in *International Conference on Machine Learning*, pp. 1329–1338, 2016.

[17] I. Osband, D. Russo, and B. Van Roy, "(More) Efficient Reinforcement Learning via Posterior Sampling," in *Advances in Neural Information Processing Systems*, pp. 3003–3011, 2013.

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