Milestone Report: Bayesian-Adaptive Deep Reinforcement Learning via Ensemble Learning

Gilwoo Lee gilwoo@cs.uw.edu

Jeongseok Lee jslee02@cs.uw.edu

Brian Hou bhou@cs.uw.edu

Aditya Vamsikrishna adityavk@cs.uw.edu

1 Introduction

While reinforcement learning is capable of controlling complex autonomous systems, RL algorithms typically require huge amounts of data, can overfit to a particular task, or may learn brittle policies that are prone to disturbances. One of the main challenges that needs to be addressed is to train a policy that is robust to various model uncertainties and disturbances. In this project, we aim to address this challenge via an ensemble policy for Bayes-Adaptive Reinforcement Learning [1].

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 [1, 2, 3], 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 optimal policy for the following Bellman equaton:

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

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.

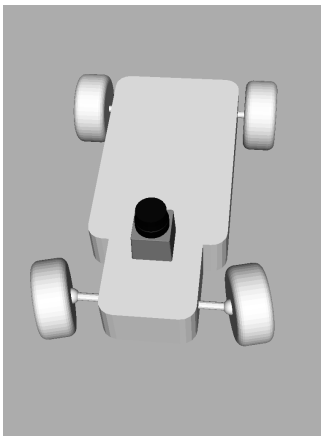
This assumption allows us to simplify BARL with an ensemble policy learning method. At a high level, we have a network that updates the *belief* of the physics parameters at time t,

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

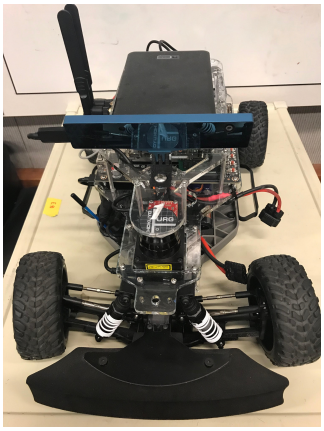
which is then used to compute the best policy from an ensemble of ϕ-dependent optimal policies, i.e., $\pi^*(\cdot; \phi)$ and $V^*(\cdot; \phi)$ are computed with typical RL algorithms for MDPs. Then the remaining task is to compute the one-step best action a:

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

We model b_ϕ as a network capable of modeling evolving state change, e.g., Recurrent Neural Networks, or as a Bayes filter. At the low level, we plan to discretize Φ and have one actor-critic



(a) Simulation in DART



(b) RACECAR

Figure 1: RACECAR

network per discretized value of ϕ: each critic estimates $V^*(\cdot;\phi)$ and each actor has an optimal policy for a particular discretized value of $\pi^*(\cdot;\phi)$. Given b_ϕ and the set of value-function approximators, it is straightforward to compute (2).

2 Background

Our work is closely related to QMDP [4, 5] 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 $Q_a(b) = \sum_s b(s) Q_{\text{MDP}}(s,a)$. In our problem setup, we have a belief over the physics parameters ϕ of the MDP, $b(\phi)$, and we compute the policy $Q_a(s;b) = \sum_\phi b(\phi) Q_{\text{MDP}}(s,a;\phi)$.

The BAMDP formulation is also similar to POMDP formulation used in POMDP-lite [6] 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" [6], which, in our case, is a discretization of ϕ.

3 Milestones

We have proposed the following milestones in our proposal:

- ✓ April 30 – Set up all simulation environments in DART [7] (including a kinematic simulation for the RACECAR)
- May 15 – Test algorithm on simulated OpenAI problems
- May 22 – Test algorithm on RACECAR in simulation
- May 31 – Run experiments on the real RACECAR

4 RACECAR

We are working on both simulation (Figure 1a)[1] and real-robot (Figure 1b) platform for the RACECAR.

The RACECAR is controlled by torque on the front wheels, and position and velocity inputs over the steering angle. The goal of the task is to track a given trajectory while minimizing the time of

[1] video: `https://youtu.be/9uoietve9S4`

traversal. The physics parameters we plan to change are mass of the car and friction coefficient between the car and the road, both of which affect the vehicle dynamics significantly.

In the real-robot experiment, the state of the RACECAR will be estimated by LIDAR observations, localized via SLAM techniques. While this does not provide exact state values, we take the most likely estimate as the current state, and only treat the physics parameters as hidden. A more theoretically appropriate approach would be to treat state as partially-observable, but such a treatment requires us to solve full POMDP instead of the simplified BARL we plan to explore in this project. We believe that given frequent updates and accurate estimates of the state, the most likely state will be a reasonable approximation for the purpose of this project.

5 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 [8]. We are utilizing TRPO [9], VPG, DDPG [10] provide by `rllab` [11] 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 [12]) would fall into the former, and EPOpt[13] and Ensemble-CIO [14] 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.

References

[1] 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.

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

[3] 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.

[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] J. Lee, M. X. Grey, S. Ha, T. Kunz, S. Jain, Y. Ye, S. S. Srinivasa, M. Stilman, and C. K. Liu, "DART: Dynamic Animation and Robotics Toolkit," *The Journal of Open Source Software*, vol. 3, p. 500, Feb 2018.

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

[9] 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.

[10] 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.

[11] 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.

[12] 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.

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

[14] 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.