Improving Bayesian Reinforcement Learning Using Transition Abstraction

Daniel Acuna ACUNA002@UMN.EDU
Department of Computer Science & Engineering
University of Minnesota

Paul Schrater SCHRATER@UMN.EDU
Departments of Computer Science & Engineering and Psychology
University of Minnesota

1. Introduction

Bayesian Reinforcement learning (BRL) is a model-based approach to reinforcement learning that provides a principled solution to the problem of simultaneous planning and learning under uncertainty, including partially observed states and unknown or variable state dynamics. While in theory, traditional reinforcement learning (RL) can be used to find an optimal policy for problems with partially observed states and unknown transition matrices, in the online setting, RL methods suffer from slow convergence, require too much data, and must implement methods to handle the exploration/exploitation trade-off. In contrast, BRL is better suited to online learning because it can use the prior knowledge embedded in the model to speed-up convergence, decrease the amount of data required, and it can optimally trade-off exploration/exploitation.

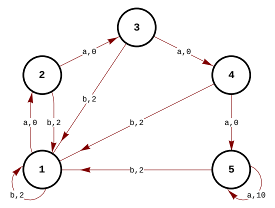
But BRL is computationally intractable for all but the simplest problems: at each decision time, an agent should weigh all possible courses of action by beliefs about future outcomes constructed over long time horizons. Recent efforts to find good approximations for BRL have generated a variety of approaches (Dearden et al., 1998; Strens, 2000; Wang et al., 2005; Castro & Precup, 2007). In particular, sparsely sampling possible courses of action that are most relevant to computing value has shown to be an effective and straightforward way to perform Bayesian action selection (Wang et al., 2005; Castro & Precup, 2007). However, sampling alone does not scale well to larger environments. Concurrently, new advances in abstraction in traditional RL have shown to be very promising at scaling RL by providing and/or discovering additional structure about the environment. But it is unclear how to incorporate abstraction into BRL and whether it can be used to speed up learning.

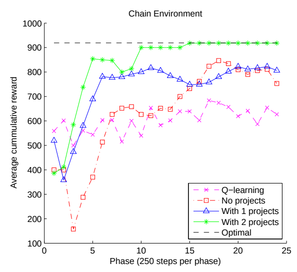
In this work, we answer these questions using an abstraction called projects—parts of the transition dynamics that bias the look ahead to areas of the environment that are promising. In a sense, a project is similar to the *option* framework (Precup, 2000), but we prefer the term project in a BRL context because of our inspiration on the Multi-armed Bandit literature. We propose a sparse sampler that can incorporate projects into Bayesian action selection. By testing on standard problems that require effective exploration–exploration balance, we show that our algorithm is able to use projects to speed up learning, outperforming BRL without projects and the classic Q-learning.

2. Adding Projects to Bayesian Reinforcement Learning

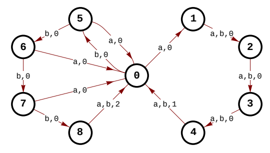
Sparse sampling consists on sampling a set of courses of actions until a given horizon and keeping an estimate of the state-action values associated to the states of the MDP. Several criteria to select which states and actions to sample next have been proposed (Wang et al., 2005; Castro & Precup, 2007). The samples generated form a tree of *hyper*-states—states of the MDP together with the beliefs about environment dynamics. The state-action values are used whenever the final set of courses of actions misses a subtree—an unsampled action or next-state. The state-action value serves as an estimate of the value of that subtree. And finally, once the values on the tree are computed, the old state-action value estimates are updated.

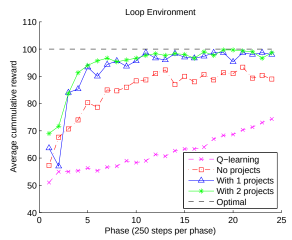
The intuition behind our idea on using projects comes from Multi-armed Bandit Problems (Gittins, 1989), which have an elegant structure for high-level planning: projects are independent, and the planning prob-





Chain environment





Loop environment

Figure 1. Test problems and performance of algorithms

lem separates into scheduling projects and learning them. At this stage of research, we assume that the projects have been given to us, and, hence, we do not need to learn them.

We modify a sampler based on linear programming that samples actions uniformly (Castro & Precup, 2007). The heart of our algorithm is an online Bayesian action selection procedure that, while considering courses of actions using the general beliefs on transition dynamics, it monitors whether some project might be engaged from the current state. If so, we expand an additional tree of future outcomes using the project's reduced transition matrix. To make the projects more important, we allow them to have a larger horizon. The size of the sample will still be controlled by a general parameter that restricts the number of samples of the final tree.

3. Results

We test our algorithm on two problems with known reward function but unknown transitions. The chain problem consists of five states. There are two actions a and b available for the agent, but, with probability 0.2, the action will have resulted in an opposite effect. The optimal policy for this problem is to perform action a at every state and would generate a total reward of 919 after 250 steps. The Loop problem consists of two 5-state loops, which are connected at a single start state. Two deterministic actions are available. Since taking action b at every state in the left loop yields a reward of 2, the optimal policy is to perform action b everywhere and would yield a total reward of 100 after 250 steps.

We test our algorithm with no-projects, with 1 project, and with 2 projects, and show the Q-learning algorithm with ϵ-greedy action selection strategy for comparison. Project 1 in the Chain environment is to take action a with probability 1 everywhere. Project 2 would take action b on state 1 with probability 1—the project doesn't exist on states 2 through 5. On the Loop environment, project 1 is to take action b at state 1, 6, 7, 8, and 9 with probability 1. And project 2 is to take action a on states 1, 2, 3, 4, and 5 with probability 1. For the Chain and Loop environments, we use a maximum of 150 samples, a horizon of 3 for the general hyper-tree, and a horizon of 5 for projects. For ϵ-greedy Q-learning, we use $\epsilon = 0.05$ and $\alpha = 0.3$.

In Figure 1, we show how much reward each algorithm accumulates at every 250 steps for a total of 6000 steps. We average the performance on 10 runs for each algorithm. As we can see, the no-project BRL gets a boost in learning if we add the hand-coded projects. It is important to notice that the project does not necessarily need to lead to high rewards, but be a bias in the transition beliefs towards important parts of the environment. This is the case when we tested the algorithm with two projects. In the Loop environment, for example, the second project allowed the agent to compare a greedy policy—right-hand loop; project 2—to the optimal one—left-hand loop; project 1.

In the future, our hope is to incorporate automatic discovery of projects, which is closely related to automatic discovery of options, hierarchies, or factorizations (Barto & Mahadevan, 2003).

References

Barto, A. G., & Mahadevan, S. (2003). Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems*, *13*, 41–77.

Castro, P. S., & Precup, D. (2007). Using linear programming for bayesian exploration in markov decision processes. *IJCAI* (pp. 2437–2442).

Dearden, R., Friedman, N., & Russell, S. J. (1998). Bayesian q-learning. *AAAI/IAAI* (pp. 761–768).

Gittins, J. C. (1989). *Multi-armed bandit allocation indices*. Chichester [West Sussex] ; New York: Wiley.

Precup, D. (2000). *Temporal abstraction in reinforcement learning*. Doctoral dissertation, University of Massachusetts Amherst.

Strens, M. J. A. (2000). A bayesian framework for reinforcement learning. *Proceedings of the Seventeenth International Conference on Machine Learning* (pp. 943–950). Morgan Kaufmann Publishers Inc.

Wang, T., Lizotte, D., Bowling, M., & Schuurmans, D. (2005). Bayesian sparse sampling for on-line reward optimization. *International Conference on Machine Learning*. Bonn, Germany.