Reinforcement Learning Algorithms for Regret Minimization in Structured Markov Decision Processes

(Extended Abstract)

Prabuchandran K. J.
Indian Institute of Science
Bangalore 560012, India
prabu.kj@csa.iisc.ernet.in

Tejas Bodas
Indian Institute of Technology
Mumbai, 400076, India
tejaspbodas@ee.iitb.ac.in

Theja Tulabandhula
Xerox Research Centre India
Bangalore 560103, India
Theja.Tulabandhula@xerox.com

ABSTRACT

A recent goal in the Reinforcement Learning (RL) framework is to choose a sequence of policy to minimize the regret incurred in *a finite time horizon*. For several RL problems in operation research and optimal control, the optimal policy of the underlying Markov Decision Process (MDP) is characterized by a known structure. The state of the art algorithms do not utilize this known structure of the optimal policy while minimizing regret. In this work, we develop new RL algorithms that exploit the structure of the optimal policy to minimize regret. Numerical experiments on MDPs with structured optimal policies show that our algorithms have better performance and are easy to implement.

Categories and Subject Descriptors

G.3 [**Probability & Statistics**]: Markov processes, Renewal theory

General Terms

Algorithms, Performance, Experimentation, Theory

Keywords

Multi-arm Bandit, Markov Decision Processes, UCB, Thompson Sampling, Renewal Reward Processes, Optimal threshold Policy.

1. INTRODUCTION

Optimal sequential decision problems are common in artificial intelligence, operation research and various other scientific disciplines. These problems are modeled using the framework of MDPs where a goal is to find a policy that maximizes the long run average reward collected. In an important subclass of these problems, one can characterize the structure of the optimal policy that maximizes this long run average reward. For example, the multi-period (s, S) policy in inventory control management [5] and the multi-threshold policies in queueing systems [3] are MDP problems where the optimal policy is characterized by a special structure. For such problems, one can develop efficient algorithms to obtain the optimal policy by restricting the search over such structured policies.

In this paper, our aim is to minimize the cumulative regret in a finite time horizon for the class of RL problems where the structure of the optimal policy is known. We provide two algorithms, *pUCB and pThompson*, that treat 'structured policies' as arms of a corresponding multi-arm bandit (MAB) problem [2] and then compare their performances with state of the art PSRL [4] algorithm.

Appears in: *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2016), J. Thangarajah, K. Tuyls, S. Marsella, C. Jonker (eds.), May 9–13, 2016, Singapore.*

2. PRELIMINARIES

Let us consider a finite unichain MDP [5] setting. Let P and R correspond to the transition probabilities and reward functions of the MDP. In our setting, we assume reward function is bounded by 1. A stationary deterministic policy (SDP) π specifies the action $(\pi(i))$ that needs to be chosen in state i. For each policy π, the long-run average reward is $\rho(\pi) = \lim_{T\to\infty} \frac{\mathbb{E}\left[\sum_{t=1}^{T} R(s_t, \pi(s_t), s_{t+1})\right]}{T}$. Let π^* correspond to the optimal policy that maximizes the long-run average reward among the finite set of SDPs.

In many modern RL settings, the goal is to find a sequence of policy $\{\pi_t\}$ that minimizes the expected *regret* in a given horizon T. Here, the expected regret when started from state s_{start} is $\mathcal{R}_{s_{start}}(T) = \rho(\pi^*)T - \sum_{t=1}^{T}\mathbb{E}[R(s_t, \pi_t(s_t), s_{t+1})]$, where π_t represents the policy chosen at time t.

In this paper we are concerned with minimizing regret when there exists some structure for π^*. Such structure exists in a wide collection of MDPs. Now using this structure in the optimal policy, one can create a smaller subset of policies that satisfies this structure. We can search for a policy that minimizes regret using this set as compared to searching in the complete policy space.

In our algorithms, we directly maintain a belief on the average reward $\rho(\pi)$ for each policy. Note that this has the advantage that we can reuse ideas from the UCB [2] and the Thompson sampling algorithms [1], which are the state-of-the-art algorithms in the multi-armed bandit setting. Another particularly attractive feature is that we can work directly with average reward of policies rather than the underlying transition probabilities and reward function.

One of the *key innovations* that we provide is *the way the beliefs on the average rewards $\rho(\pi)$ will be updated*. For this we use the Renewal-Reward Theorem ([6]). Under a given policy π, by fixing the return times to a fixed state s_{start} as renewal times, the MDP becomes a renewal reward process. Now one can estimate $\rho(\pi)$ by estimating the reward collected in a cycle (starting from s_{start} and returning to s_{start}) and the duration of the cycle.

3. ALGORITHMS

In our algorithms, we consider policies that has same structure as of the optimal policy. Let K denote the number of such policies.

In pUCB, we maintain an estimate of the long-run average reward obtained under each policy π_k as $\hat{\rho}(k)$. At the end of a cycle,

Algorithm 1: pUCB algorithm (pUCB)

1: **Input:** $T, \Pi = \{\pi_k : k = 1, \ldots, K\}, s_{start}, \tau$
2: **Output:** CR_T (Cumulative reward in T rounds)
3: **for** $k = 1$ to K **do**
4: $\hat{\rho}(k) = 1,\ n(k) = 0$
5: $R_{arm}(k) = 0$ (Running sum of rewards for policy π_k)
6: $T_{arm}(k) = 0$ (Running sum of number of rounds for π_k)
7: **end for**
8: $s = s_{start},\ t' = 0, r = 0,\ k \sim \text{rand}(1, K)$
9: **for** $t = 1$ to T **do**
10: **if** $(t \neq 1$ **and** $s = s_{start})$ **or** $t' \geq \tau$ **then**
11: $R_{arm}(k) = R_{arm}(k) + r,\ T_{arm}(k) = T_{arm}(k) + t'$
12: $\hat{\rho}(k) = \frac{R_{arm}(k)}{T_{arm}(k)},\ c(k) = \sqrt{\frac{2 \log t}{n(k)}}$
13: $n(k) = n(k) + 1,\ k = \underset{j=1,\ldots,K}{\arg\max} \Big\{\hat{\rho}(j) + c(j)\Big\}$
14: $t' = 0,\ r = 0$
15: **end if**
16: $s_{next} \sim P(\cdot \mid s, \pi_k(s)), r = R(s, a, s_{next})$
17: $t' = t' + 1,\ s = s_{next}$
18: **end for**
19: $CR_T = \sum_{j=1}^{K} R_{arm}(j)$

we update $\hat{\rho}(k)$ using r and t' as shown in line number 11 in Algorithm 1. This is justified using the renewal-reward theorem as discussed in Section 2. In the next cycle, we follow the policy that has the highest values of the sum of the average reward estimate $\hat{\rho}(k)$ and the confidence bonus $\sqrt{\frac{2 \log t}{n(k)}}$. Here $n(k)$ tracks the count of the number of times policy k has been picked by round t.

Algorithm 2: pThompson algorithm (pThompson)

1: **Input:** $T, \Pi = \{\pi_k : k = 1, \ldots, K\}, s_{start}, \tau$
2: **Output:** CR_T (Cumulative reward in T rounds)
3: **for** $k = 1$ to K **do**
4: $CR_T = 0, S(k) = 0,\ F(K) = 0$
5: **end for**
6: $s = s_{start},\ t' = 0,\ r = 0,\ k \sim \text{rand}(1, K)$
7: **for** $t = 1$ to T **do**
8: **if** $(t \neq 1$ **and** $s = s_{start})$ **or** $t' \geq \tau$ **then**
9: $CR_T = CR_T + r$
10: $S(k) = S(k) + r,\ F(k) = F(k) + t' - r$
11: **for** $k = 1$ to K **do**
12: $\theta(k) \sim Beta(S(k), F(k))$
13: **end for**
14: $k = \underset{j=1,\ldots,K}{\arg\max} \Big\{\theta(j)\Big\},\ t' = 0, r = 0$
15: **end if**
16: $s_{next} \sim P(\cdot \mid s, \pi_k(s)),\ r = R(s, a, s_{next})$
17: $t' = t' + 1,\ s = s_{next}$
18: **end for**

In pThompson, we maintain a different set of internal estimates. In particular, for each policy π_k, it maintains two estimates $S(k)$ and $F(k)$. These two estimates parameterize a Beta distribution that encodes our beliefs on the average reward of policy π_k. The cumulative reward in a cycle is added to the running estimate $S(k)$ (line 10) of the current policy k and an update of the form $t - r$ is added to $F(k)$. This update step is critical and ensure that the mean of the Beta distribution is an unbiased estimate of average reward $\rho(k)$. Then for the new policy selection, we draw a realization for each of the K Beta distributions and pick that policy whose realization value is the highest.

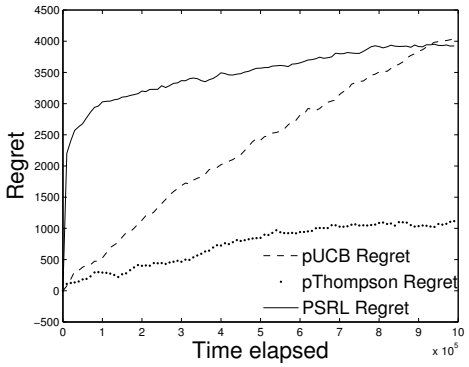

Figure 1: Regret as a function of number of rounds for the slow server problem.

Note that the PSRL algorithm maintain $O(M^2 N)$ estimates. This is significantly higher compared to our algorithms, that typically maintain $O(M)$ estimates (as seen in many problem instances).

4. NUMERICAL RESULTS

We consider the slow server problem [3] and assume that we do not know the service rate parameters themselves in this setting but know that $\mu_1 > \mu_2$. Let the buffer size be $B < \infty$. Note that for this setting, the optimal policy is of the threshold type. For our experiment, we chose $\lambda = \frac{12}{31}$, $\mu_1 = \frac{18}{31}$ and $\mu_2 = \frac{1}{31}$. The buffer length of the queue was chosen to be 20 (thus leading to a number of states equal to 80). We ran 10 Monte Carlo simulations. As shown in Figure 1, pUCB and pThompson perform much better than PSRL from the very outset. This clearly illustrates the advantage of using knowledge of the optimal policy structure.

5. CONCLUDING REMARKS

We have built on the well established, easy to use UCB and Thompson sampling algorithms to provide competitive regret minimizing RL algorithms (making novel modifications for getting unbiased estimated of the long run average reward $\rho(k)$ and the number of rounds that each policy π_k needs to be applied). Such direct use of structural information is not easily possible for the state of the art PSRL algorithm. The algorithms presented are simple and better than state-of-the-art methods for cumulative regret minimization in RL settings. They have significant advantage in terms of computation and sampling costs.

REFERENCES

[1] S. Agrawal and N. Goyal. Analysis of thompson sampling for the multi-armed bandit problem. *arXiv preprint arXiv:1111.1797*, 2011.
[2] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.
[3] W. Lin and P. Kumar. Optimal control of a queuing system with two heterogeneous servers. *IEEE Trans. on Automatic Control*, 29:696–703, 1984.
[4] I. Osband, D. Russo, and B. Van Roy. (more) efficient reinforcement learning via posterior sampling. pages 3003–3011, 2013.
[5] M. L. Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.
[6] S. M. Ross. *Stochastic processes*, volume 2. John Wiley & Sons New York, 1996.