A Bayesian Posterior Updating Algorithm in Reinforcement Learning

Fangzhou Xiong[1,2], Zhiyong Liu[1,2,3,5(✉)], Xu Yang[1], Biao Sun[4], Charles Chiu[6], and Hong Qiao[1,2,3,4,5]

[1] The State Key Lab of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Science, Beijing 100190, China
zhiyong.liu@ia.ac.cn
[2] School of Computer and Control, University of Chinese Academy of Sciences (UCAS), Beijing 100049, China
[3] CAS Centre for Excellence in Brain Science and Intelligence Technology (CEBSIT), Shanghai 200031, China
[4] University of Science and Technology Beijing, Beijing 100083, China
[5] Cloud Computing Center, Chinese Academy of Sciences, DongGuan 523808, Guangdong, China
[6] School for Higher and Professional Education, Chai Wan, Hong Kong, China

Abstract. Bayesian reinforcement learning (BRL) is an important approach to reinforcement learning (RL) that takes full advantage of methods from Bayesian inference to incorporate prior information into the learning process when the agent interacts directly with environment without depending on exemplary supervision or complete models of the environment. BRL tackles the problem by expressing prior information in a probabilistic distribution to quantify the uncertainty, and updates these distributions when the evidences are collected. However, the expected total discounted rewards cannot be obtained instantly to maintain these distributions after each transition the agent executes. In this paper, we propose a novel idea to adjust immediate rewards slightly in the process of Bayesian Q-learning updating by introducing a state pool technique which could improve total rewards that accrue over a period of time when this pool resets appropriately. We show experimentally on several fundamental BRL problems that the proposed method can perform substantial improvements over other traditional strategies.

Keywords: Bayesian reinforcement learning · Bayesian Q-learning · State pool technique

1 Introduction

As a rapidly growing branch in artificial intelligence, RL is a learning problem where an agent tries to behave optimally when interacting with the environment so as to finish a task through achieving its goal step by step [1,2]. One of the major challenges in RL is the trade-off between exploration of untested actions

D. Liu et al. (Eds.): ICONIP 2017, Part V, LNCS 10638, pp. 418–426, 2017.
https://doi.org/10.1007/978-3-319-70139-4_42

and exploitation of actions that are known to be good. Fortunately, BRL offers a solution to address this exploration-exploitation problem by maintaining an explicit distribution over unknown parameters to quantify the uncertainty [3].

Instead of focusing on learning point estimation of the parameters in traditional RL, BRL tries to transfer prior information encoded relevant domain knowledge into a form of probabilistic distribution to represent unknown parameters. When the agent interacts with the environment, rewards are obtained to update these distributions. After that, according to the latest distribution the agent makes a decision by selecting an action to land up in next state [4].

There are several techniques to select an action. Undirected approaches (e.g. *epsilon-greedy exploration* and *Boltzmann exploration* [2]) usually choose random actions occasionally with no exploration-specific knowledge. Dearden et al. [5] propose Q-value sampling technique by extending to solve bandit problems to multi-state RL problems, and use a myopic value of perfect information (VPI) criterion to offer another policy for action selection. Wang et al. [6] present an efficient "sparse sampling" technique for Bayes optimal decision. Brafman et al. [7] propose a *R-MAX* algorithm to explore under the assumption that unknown states provide maximal rewards.

In this paper, we mainly concentrate on the problem of how to update the estimation of distributions over Q-values. Roughly speaking, the agent maintains these distributions over random variables with a tuple of hyperparameters. Each action is selected based on prior distributions which are updated by the received rewards. However, these rewards usually require to be expected and total, which are totally different from the practical situation where available rewards are only local and instantaneous after each action execution [5]. In order to tackle this problem, some sampling techniques are introduced, such as the Thompson Sampling algorithm [8] which suggests a natural Bayesian approach to sample a parameter from the posterior.

From the aspect of the immediate reward, the paper proposes a new idea to refine its effect with almost no change in numerical values, so that the agent could explore state-action space more deep under the same condition, which leads to more total discounted rewards. More specifically, we introduce a state pool which records the "known" and "unknown" states based on whether they have been visited. In addition, in order to incorporate more state information between different sequential episodes, we maintain the state pool by resetting it to a empty collection every few episodes.

2 Background

We consider a basic concept of Markov Decision Processes (MDPs) with infinite horizon represented by a 5-tuple $M = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where $\mathcal{S}$ is a set of possible states, $\mathcal{A}$ is a set of possible actions, $\mathcal{P}$ is a state transition function that captures the probability of reaching the next state s' after we select action a at state s according to the policy π which denotes a mapping from state s to action a, $\mathcal{R}$ is a reward function that maps state-action pairs to a bounded subset of $\mathcal{R}$, and

$\gamma \in (0,1)$ is the discount factor specifying the effect of the current decision on the future rewards. The agent's goal in RL is to maximize the total discounted reward R by

$$R = \sum_{k=0}^{\infty} \gamma^k r_{t+k} \tag{1}$$

where r_t is the reward received at time t, and $\gamma \in (0,1)$ is the discount factor. The state-action value function $Q^{\pi}(s,a) = \mathbb{E}[R_t | s_t = s, a_t = a]$ is the expected reward for selecting action a in state s and following policy π.

In this work, we mainly focus on the Bayesian Q-learning (BQL). As for traditional Q-learning, it works by keeping running estimates that are updated at each step, i.e., when action a is executed in state s and transferred to next state s' with the immediate reward r, the Q-value would be updated following Q-learning updating rule:

$$\hat{Q}(s,a) \leftarrow (1-\alpha)\hat{Q}(s,a) + \alpha(r + \gamma \max_{a'} \hat{Q}(s',a')). \tag{2}$$

BQL utilizes probability distributions to represent the uncertainty over the estimated Q-value at each state. The agent executes actions based on these distributions, and receives rewards to update these priors, which could be considered as a process of keeping and propagating distributions over Q-values [9]. In order to denote the total discounted reward R_t more explicit, we formally let $R_{s,a}$ be a random variable when action a is selected in state s and an optimal policy π is adopted thereafter. Obviously, we want to learn the value $\mathbb{E}[R_{s,a}]$ to achieve expected rewards, i.e. optimal state-action function $Q^*(s,a) = \mathbb{E}[R_{s,a}]$. According to [5], $R_{s,a}$ can be assumed to have a normal distribution with mean $\mu_{s,a}$ and precision $\tau_{s,a}$.

For comparisons of subsequent experiments, we consider the same normal-gamma distribution as prior distribution $p(\mu_{s,a}, \tau_{s,a})$:

$$p(\mu_{s,a}, \tau_{s,a}) \propto \tau^{\frac{1}{2}} e^{-\frac{1}{2}\lambda\tau(\mu-\mu_0)^2} \tau^{\alpha-1} e^{\beta\tau} \tag{3}$$

which could be represented by $p(\mu,\tau) \sim NG(\mu_0, \lambda, \alpha, \beta)$ with a tuple of hyper-parameters $\rho = \langle \mu_0, \lambda, \alpha, \beta \rangle$. Naturally, we only need to maintain these hyper-parameters to represent and update the prior distributions of $R_{s,a}$.

The policy based on the normal-gamma distribution for action selection we will adopt is called myopic value of perfect information algorithm (*VPI*). The idea of this policy is to balance the expected rewards from exploration in the form of improved policies against the expected cost of performing a potential suboptimal action [5].

Once the policy is executed according to the estimated distributions, the immediate rewards will be collected to calculate the posterior probability density. Now we review a updating rule for calculating posterior distribution over $R_{s,a}$ called moment updating method (*Mom*) [5] which will be used in our algorithm. The *Mom* method randomly samples values R^1_{t+1}, R^2_{t+1}, ..., R^n_{t+1} from the

prior distribution, and assums these n samples contribute equally to solve two moments as:

$$M_1 = \mathbb{E}[r + \gamma R_{t+1}] = r + \gamma \mathbb{E}[R_{t+1}] \tag{4}$$

$$\begin{aligned} M_2 &= \mathbb{E}[(r + \gamma R_{t+1})^2] \\ &= r^2 + 2\gamma r \mathbb{E}[R_{t+1}] + \gamma^2 \mathbb{E}[R_{t+1}^2]. \end{aligned} \tag{5}$$

According to the fact that the posterior distribution still is a normal-gamma distribution. We have $p(\mu, \tau | R^1, R^2, \ldots, R^n) \sim NG(\mu_0', \lambda', \alpha', \beta')$ where $\mu_0' = \frac{\lambda\mu_0 + nM_1}{\lambda + n}$, $\lambda' = \lambda + n$, $\alpha' = \alpha + \frac{1}{2}n$, and $\beta' = \beta + \frac{1}{2}n(M_2 - M_1^2) + \frac{n\lambda(M_1 - \mu_0)^2}{2(\lambda + n)}$.

3 The Proposed Method

Given the prior density $P(x)$ and the evidence D collected from the observations in the process of state transition, the posterior probability density $P(x|D)$ for model hyperparameters will be updated. Theoretically, we could apply the Bayes Theorem to solve the posterior distribution:

$$P(x|D) = \frac{p(D|x)P(x)}{p(D)}. \tag{6}$$

However, this updating is complicated by the fact that the available observations are local and immediate rewards, whereas the distribution over Q-value is a distribution over total discounted rewards. Hence we cannot use the Bayes Theorem directly.

In order to analyze the update for Q-value more conveniently, now we can rewrite the formula (1) as:

$$R_t = r + \gamma R_{t+1} \tag{7}$$

where R_t is a random variable denoting the total discounted reward from time t at state s. If the agent follows optimal policy with the best action a thereafter, then R_t is distributed as $R_{s,a}$. Since $Q^*(s, a) = \mathbb{E}[R_{s,a}] = \mu_{s,a}$, then the random variable $R_{s,a}$ can be utilized to update the posterior probability density. Now we employ $p(\mu, \tau)$ to generate samples randomly, which results in a moment updating for updating the estimate of the Q-value. Nevertheless, the best action a is not always performed, and there still enjoys a huge potential for improving effectiveness of updating. The existing updating rule only considers the rewards corresponded to the second term in the right part of formula (7), thus it is natural to take the first term into account, i.e., the immediate reward r.

We argue that the immediate reward should be relevant to the state that the agent resides. Therefore, there are two cases to be dealt with: (a) if the agent stays at next state s' that has not been visited before, the immediate reward r is encouraged to increase so that the posterior probability for executing an action a will increase, and (b) if s' has been stepped into previously, r should be reduced

a little contrasted to the normal immediate reward so that the corresponding probability density for action a will be decreased. Both of cases will lead to more exploration in the learning process, which is crucial for episode problems in RL. In principle, the immediate reward should be determined by the observation after every state transition. Now we actually modify this value only to assist in updating the posterior probability which benefits for action selection. Nonetheless, the calculation method for the total discounted reward still adopts the formula (1).

Specifically, the paper introduces a state pool P to record the "known" and "unknown" states based on whether they have been visited before so as to modify the immediate reward. After each distribution updating has finished, the "unknown" state will be added into the state pool which leads to the updating for the state pool, and this state becomes the "known" thereafter. Thus we can rewrite the immediate reward:

$$r_p = \begin{cases} r + \varepsilon & if\ s'\ not\ in\ P \\ r - \varepsilon & if\ s'\ in\ P \end{cases}$$

where r and ε denote the primitive reward and a small positive number, respectively.

Algorithm 1. Bayesian Q-learning with the state pool technique

Require: initial state $s \leftarrow s_0$, final state S_F,
episode $e \leftarrow 0$, discounted factor γ,
hyperparameters $\rho \leftarrow \langle \mu_0, \lambda_0, \alpha_0, \beta_0 \rangle$, policy π,
state pool $P \leftarrow \varnothing$, interval K, small positive number ε.
Ensure: total reward R

for step $i = 0$ to N **do**
  Execute action a at state s based on policy π.
  Generate next state s' and immediate reward r.
  Calculate new reward r_p:

$$r_p = \begin{cases} r + \varepsilon\ if\ s'\ not\ in\ P \\ r - \varepsilon\ if\ s'\ in\ P \end{cases}$$

  Update state pool P: $P = P \cup s'$
  Maintain policy π by updating ρ with Mom
  if $s' = S_F$ **then**
    Calculate $R_e = \sum \gamma r$, $s = s_0$, $e = e + 1$
    if $e\ mod\ K = 0$ **then**
      $P \leftarrow \varnothing$
    end if
  else
    $s \leftarrow s'$
  end if
end for
Calculate total reward $R = \sum_{j=0}^{e} R_j$.

Nonetheless, there is a problem that how long we should keep updating for the state pool. On the one hand, if we reset the state pool at one or more steps in one episode, there is no need for resetting since the executed action during one episode would not always behave optimal and it is necessary to bring in more episodes to support the posterior updating. On the other hand, if the state pool is reset at the end of total step, repetitive states will be visited continuously which possibly causes that the state pool can be increased to a full status only after several episodes, then subsequent updating for state pool has no significance to conduct. As a consequence of these two aspects, the paper proposes to maintain the state pool on every K episode by resetting it to a empty collection, which integrates state information between different sequential episodes and considers them as a unit with state pool technique to improve total discounted rewards. Thus, the proposed algorithm is summarized in Algorithm 1.

4 Experimental Illustration

4.1 Experiment Setting

There are 3 different episode experiments [10] conducted to evaluate the proposed algorithm.

Chain. Figure 1 presents the 5-state "Chain" problem that aims at achieving as many rewards as possible over fixed steps. Two available actions are labeled on the arcs followed by the immediate rewards. With probability 0.2, the agent slips and executes the opposite action. Once the final state 5 is visited, the agent starts at state 1 again.

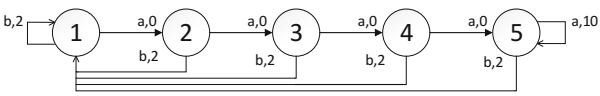

Fig. 1. The "Chain" problem.

Loop. Figure 2 illustrates the 9-state "Loop" problem with the purpose of maximizing total rewards. Similarly, the arcs are labeled with the actions and associated rewards. Performing action a will lead to the traversal of the right loop, and choosing action b repeatedly causes traversal of left loop. State 0 is set both for start state and final state. Hence once it is visited, the next state resets to itself again.

Maze. Figure 3 shows the 264-state "Maze" problem. The reward is measured by the number of flags collected before the agent reaches the goal. In this figure, S stands for the start state, G marks the goal, and F illustrates the location of flag. The agent can move up, down, left and right in the maze except staying at

current state when it hits the wall or moves out of the maze. With probability 0.1, the agent slips and conducts an action that goes in a perpendicular direction. Once the agent arrives at the goal, it will immediately return to the start.

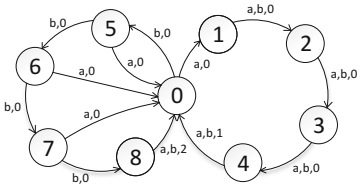

Fig. 2. The "Loop" problem.

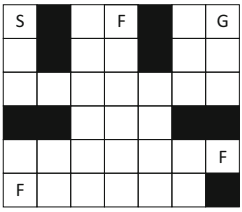

Fig. 3. The "Maze" problem.

The algorithms we have utilized are as follows:

ϵ-greedy. Q-learning with epsilon-greedy exploration policy.
VPI+Mom. Bayesian Q-learning with VPI policy and moment updating.
VPI+SP. Bayesian Q-learning with VPI policy and state pool technique (SP).

4.2 Experimental Results and Discussion

The experiments are evaluated by accumulated total rewards obtained during the learning process which comprises 1000 steps for Chain and Loop, and 20000 steps for Maze. Table 1 represents correlative results by running 10 times for Chain, Loop, and Maze respectively.

Table 1. Results of accumulated rewards with average and standard deviation.

Chain	Avg	Dev	Loop	Avg	Dev	Maze	Avg	Dev.
ϵ-greedy	1288.0	46.3	ϵ-greedy	180.1	8.7	ϵ-greedy	540.4	105.3
VPI+Mom	1530.4	153.6	VPI+Mom	198.7	0.9	VPI+Mom	153.9	7.3
VPI+SP0[1]	1527.2	96.5	VPI+SP0[1]	272.2	70.0	VPI+SP0[1]	455.9	23.5
VPI+SP	1560.6	52.1	VPI+SP	359.2	18.5	VPI+SP	845.6	45.9

[1]This mark stands for using state pool technique without reset operation.

The first two experiments are designed to state the significance of existence for state pool technique so that the subsequent experiment can be performed reasonably to evaluate the improvements for total rewards received. Table 1 presents the relevant results.

To be specific, in the first experiment of Chain, the agent has doubtlessly traversed all states when it lands up in final state. There is no need to consider

the exploration about whether the next state is an unknown state since all states will be labeled as "known" when it arrives at the goal. In addition, to reset the problem every few episodes is also in vain as the state pool actually is maintained to make full use of the information about unknown states between different sequential episodes. Therefore, it turns out that the agent achieves similar accumulated rewards, i.e., 1530.4 for MVPI+ Mom policy and 1560.6 with proposed technique.

In the second experiment of Loop, even if the start and the goal share the same state, there are commonly some unexplored states during one episode. Naturally, it leaves the space for improvement with state pool technique. As the Table 1 shows, the agent obtains the average reward of 359.2 in 10 runs when we set interval $K = 10$, which outperforms than ϵ-greedy policy and moment updating rule.

In the last experiment, a maze problem is designed to show that Bayesian Q-learning with proposed technique outperforms the traditional methods. In the maze, the agent tries to conduct sufficient exploration to carry 3 flags to the goal, which pushes it to make more attempts to visit unknown states. Once the agent has been stuck in a corner of the maze, more steps will be executed. Furthermore, the prior for each action shares the same distribution at the initialization phase, thus the agent treats them equally so that it will cost more steps when struggled with a corner, which tends to achieve less accumulated rewards when given the same total steps. Therefore, apart from the action, the state information has to be considered, i.e. the proposed state pool technique. Obviously, the results in Table 1 testify the effectiveness of the proposed algorithm.

Moreover, in the case of Loop and Maze, if we adopt state pool method without reset operation, the total discounted reward will be largely affected. These results manifest that it is crucial to integrate the state information between sequential episodes by state pool technique with a specific form, such as reset operation, which can result in more total discounted rewards.

5 Conclusions

In this paper, the proposed state pool technique enables the agent to obtain more rewards in Bayesian Q-learning domain. It has been evaluated over 3 experiments that the agent with the proposed technique has a potential ability to update the posterior distribution by model hyperparameters towards high rewards so that it will achieve substantial improvements in accumulated rewards. In the future work, we will pay more attention to bridge the relationship between the action selections and state information, and hope to make some improvements for the action initializations by borrowing ideas from previous episodes, thus the agent will not easily visit useless states when performs some explorations, which ultimately leads to more accumulated rewards.

Acknowledgments. This work is partly supported by NSFC grants 61375005, U1613213, 61210009, MOST grants 2015BAK35B00, 2015BAK35B01, Guangdong Science and Technology Department grant 2016B090910001.

References

1. Kaelbling, L.P., Littman, M.L., Moore, A.W.: Reinforcement learning: a survey. J. Artif. Intell. Res. **4**, 237–285 (1996)
2. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT press, Cambridge (1998)
3. Ghavamzadeh, M., Mannor, S., Pineau, J., Tamar, A.: Bayesian reinforcement learning: a survey. Found. Trends? Mach. Learn. **8**(5–6), 359–483 (2015)
4. Vlassis, N., Ghavamzadeh, M., Mannor, S., Poupart, P.: Bayesian reinforcement learning. Reinforcement Learning **12**, 359–386 (2012)
5. Dearden, R., Friedman, N., Russell, S.: Bayesian Q-learning. In: The Association for the Advancement of Artificial Intelligence, pp. 761–768 (1998)
6. Wang, T., Lizotte, D., Bowling, M., Schuurmans, D.: Bayesian sparse sampling for on-line reward optimization. In: Proceedings of the 22nd international conference on Machine learning, pp. 956–963 (2005)
7. Brafman, R.I., Tennenholtz, M.: R-max-a general polynomial time algorithm for near-optimal reinforcement learning. J. Mach. Learn. Res. **3**(Oct), 213–231 (2002)
8. Chapelle, O., Li, L.: An empirical evaluation of Thompson sampling. In: Advances in neural information processing systems, pp. 2249–2257 (2011)
9. Strens, M.: A Bayesian framework for reinforcement learning. In: International Conference on Machine Learning, pp. 943–950 (2000)
10. Castronovo, M., Ernst, D., Couëtoux, A., Fonteneau, R.: Benchmarking for Bayesian reinforcement learning. PloS One **11**(6), e0157088 (2016)