Isaac Han, Seungwon Oh, Hoyoun Jung,
Insik Chung, and Kyung-Joong Kim
Gwangju Institute of Science and Technology, SOUTH KOREA

Monte Carlo and Temporal Difference Methods in Reinforcement Learning

Abstract

Reinforcement learning (RL) is a subset of machine learning that allows intelligent agents to acquire the ability of executing desired actions through interactions with an environment. Its remarkable progress has achieved significant results in diverse domains, such as Go and StarCraft, and practical challenges like protein-folding. This short paper presents overviews of two common RL approaches: the Monte Carlo and temporal difference methods. To obtain a more comprehensive understanding of these concepts and gain practical experience, readers can access the full article on IEEE Xplore, which includes interactive materials and examples.

I. Introduction

Reinforcement learning (RL) [1] is a type of machine learning, in which an agent learns to make decisions by trial and error in a given environment. RL recently showed impressive results in various domains, such as Go [2], StarCraft [3], and protein-folding problems [4]. RL comprises the four main elements of policy, reward, value function, and environment model. Policy determines the agent's behavior. Reward serves as the reinforcement signal. Value function measures the long-term state quality. Lastly, model predicts the environment's behavior. Figure 1 provides an overview of RL. RL algorithms can either be model-free or model-based. Model-free algorithms update the policy by using value function estimates based on a trial and error feedback. In contrast, model-based algorithms evaluate the state values by considering possible future states and actions through the environment model. This study introduces two fundamental model-free algorithms, namely, the Monte Carlo (MC) and temporal difference (TD) methods, which are powerful tools for solving various RL problems.

II. Markov Model

The Markov decision process (MDP) provides a valuable framework for addressing sequential decision-making problems. It comprises a set of states, actions, transition probabilities, and rewards that define the problem space. RL algorithms leverage the MDP to determine the optimal policy or evaluate the action value within the MDP constraints. The MDP builds upon the concept of a Markov process, a stochastic process characterized by states and transition probabilities.

A. Markov Process

A Markov process is a stochastic process describing how a system changes over time. At each time step, the state may either remain the same or transition into a different state based on the transition probabilities. The system's transitioning state is determined only by the current state, not by any of the previous states. This property is known as the Markov Property, a defining characteristic of the Markov processes.

Definition 1 (Markov Property).

$$P[s_{t+1} \mid s_t] = P[s_{t+1} \mid s_1, s_2, \ldots, s_t]$$

A Markov process comprises two fundamental components: the state space (S) that represents all possible states a system can be in and the transition probability $P_{ss'}$ that represents the probability of transitioning from state s to state s' at step t. The transition probability is expressed as $P_{ss'} = P(S_{t+1} = s' \mid S_t = s)$.

B. Markov Decision Process

The MDP extends upon a Markov process by incorporating decision-making capabilities. Within the MDP, an agent selects an action at each time step based on its current state and associated reward. The next state is determined by the chosen action and the transition probability.

The MDP introduces additional components compared to a Markov process. The action space A includes all possible actions that can be taken given a state s. The transition probability $P^a_{ss'}$ models the impact of actions as $P^a_{ss'} = P(S_{t+1} = s' \mid S_t = s, A_t = a)$. The reward function R evaluates the desirability of different actions by providing a scalar value, called reward R_t, at step t. Lastly, the discount factor γ is a value between 0 and 1 that determines the importance of immediate rewards versus long-term rewards as a part of the agent's learning strategy.

The agent's goal is to learn a policy that maps states to actions maximizing the expected cumulative reward over time. Maximizing the sum of rewards by time in the MDP requires the evaluation of the value of each state and the determination of the optimal action. This is performed by calculating the expected sum of the discounted future rewards obtained by taking a particular action in each state.

$$G_t = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{T-1} R_T$$

The value function outputs the expected return value for each state and must be estimated from experience in a model-free environment, which is typically done using either MC or TD method.

$$\begin{aligned} V(s) &= \mathbb{E}[G_t \mid S_t = s] \\ &= \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s\right] \end{aligned}$$

C. SharkGame Environment

This study presents the MC and TD approaches using SharkGame, a grid-based game, as an example. Figure 2 illustrates the SharkGame environment. The agent, namely, the shark, aims to reach the treasure box at the bottom while avoiding dangerous obstacles, such as bombs and

Digital Object Identifier 10.1109/MCI.2023.3304145
Date of current version: 17 October 2023

Corresponding author: Kyung-Joong Kim (e-mail: kjkim@gist.ac.kr).

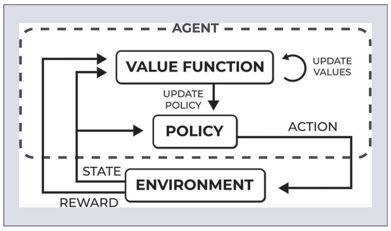

FIGURE 1 Overview of RL.

fishnets. SharkGame is an MDP with 36 states (each grid cell) and four actions (i.e., up, down, left, and right). The transition probability is always 1, making it a deterministic environment. The reward function provides a small negative reward at each step and a large negative reward when the agent encounters obstacles. Moreover, it encourages the agent to find the shortest path while avoiding obstacles.

III. Monte Carlo Approach

The MC values are estimated by averaging the sample returns. During each episode, the agent takes actions and receives rewards, and the sequence of states, actions, and rewards is recorded. After each episode, the sample return G_t is computed by each step. The counter $N(s_t)$, which refers to the number of visits to state s_t, is then incremented for each visited state in the episode. The sample return G_t is then added to the total return $S(s_t)$. Finally, the state value $V(s_t)$ is obtained by dividing $S(s_t)$ by $N(s_t)$.

$$N(s_t) = N(s_t) + 1$$
$$S(s_t) = S(s_t) + G_t$$
$$V(s_t) = S(s_t)/N(s_t)$$

The MC approach has two limitations. First, it requires complete episodes of the interaction with the environment before updating the state values. This process can be time consuming and computationally expensive, particularly in long-episode scenarios. Its application also becomes unfeasible in the context of infinite episodes.

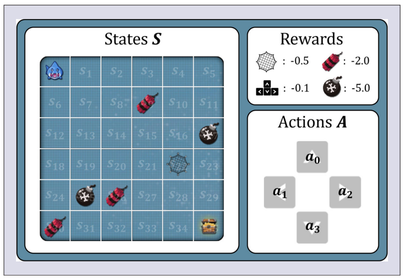
States S
s_1 s_2 s_3 s_4 s_5
s_6 s_7 s_8 s_{10} s_{11}
s_{12} s_{13} s_{14} s_{15} s_{16}
s_{18} s_{19} s_{20} s_{21} s_{23}
s_{24} s_{27} s_{28} s_{29}
s_{31} s_{32} s_{33} s_{34}
Rewards
: -0.5
: -2.0
: -0.1
: -5.0
Actions A
a_0
a_1
a_2
a_3

FIGURE 2 Illustration of the SharkGame environment.

Second, the MC method can be susceptible to a high variance in the value estimates, particularly when the episodes are short or few. This variance can impede the agent's ability to learn the true state values and lead to a slower convergence.

IV. Temporal Difference Learning

The TD approach is a method for estimating the value function in RL. Its main advantage over the MC method is that it updates the value at every transition instead of waiting until the end of an episode. The TD approach updates the value function for each step using the bootstrap method that updates an estimate using another estimate. More precisely, it updates the current value function ($V(s_t)$) with a more accurate estimate ($R_{t+1} + \gamma V(s_{t+1})$) that considers the sampled reward received by the agent. This incremental update enables the TD approach to estimate the value function at each step.

$$V(s_t) = \mathbb{E}[R_{t+1} + \gamma V(s_{t+1})]$$

Compared to the MC method, the TD method is more computationally efficient because of the frequent update. However, it has bias and may not converge to the true value function. It also uses an estimated function $V(s)$ for the update instead of a true value function. Despite its limitations, the TD approach demonstrates practical effectiveness in various real-world applications and is successfully applied to a wide range of RL problems.

V. Difference Between MC and TD

Both the MC and TD approaches are used in RL to estimate the value function; however, they differ in their update mechanism. The MC method updates values only at the end of each episode, while the TD approach updates values after each time step. The TD method has an inherent capability to swiftly pinpoint meaningful interactions due to its incremental update scheme, which enables the ongoing episode to be directly influenced by the changes resulting from previous interactions. This mechanism contributes to a faster convergence, which enhances the overall efficiency of the TD method compared to the MC approach. Therefore, the MC method can be more computationally expensive and time consuming, while the TD method is more efficient and takes faster to converge. Both methods estimate values in different ways and asymptotically converge to the true values.

The bias and the variance of these two algorithms can be considered from a machine learning perspective. MC learning has a low bias, but a high variance because it estimates the value function by averaging the returns from multiple episodes, which can lead to a high variance. This high variance makes the training unstable and may disturb it. However, the value function is updated only with the sample returns; hence, the estimation introduces no bias. By contrast, TD learning has a low variance, but a high bias because it updates the value function at each time step based on the estimated target value. This leads to a lower variance because each update is based on a single time step; however, it can introduce a bias because the value function is updated with the estimated target values.

In the full article on IEEE Xplore, we will delve into a comprehensive discussion regarding a hybrid approach that combines the strengths of the MC and TD learning methods. This advanced technique aims to optimize and balance the trade-offs inherent in each individual method, thereby enhancing the overall performance.

VI. Conclusion

In conclusion, this paper provides an overview of the two fundamental approaches in RL: MC and TD. The key distinctions between the MC and TD methods are outlined, highlighting the respective strengths and weaknesses of each approach. Further information and examples are available in the complete paper, which can be accessed online at IEEE Xplore.

Acknowledgment

This research was supported by the National Research Foundation of Korea (NRF) funded by the MSIT under Grant 2021R1A4A1030075.

References

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.
[2] D. Silver et al., "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.
[3] O. Vinyals et al., "Grandmaster level in StarCraft II using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.
[4] J. Jumper et al., "Highly accurate protein structure prediction with AlphaFold," *Nature*, vol. 596, no. 7873, pp. 583–589, 2021.