YatRL SYSU CSE 2025-1 Homework2

Cliff Walk with TD Learning

Course Number：DCS245

Student's Name：傅祉珏

Student's Number：21307210

Advisor's Name / Title：Prof. Chen Xu

Date Due：19 November, 2025

12 November, 2025

Copyright 傅祉珏

HOMEWORK2: Cliff Walk with TD Learning

傅祉珏 21307210

Sun Yat-sen University, School of Computer Science and Engineering

Abstract

This report aims to deeply investigate and implement a series of Temporal Difference (TD) learning algorithms to solve the classic Cliff Walk path planning problem. In this work, we first rigorously formalize the Cliff Walk environment as a Markov Decision Process (MDP), with a focus on model-free reinforcement learning methods. Subsequently, we implement and compare four core TD algorithms: the on-policy methods SARSA, Expected SARSA, and N-step SARSA, as well as the off-policy method Q-learning.The experimental results clearly demonstrate the fundamental divergence between on-policy and off-policy approaches when solving risk-aversion problems. A core finding reveals that although Q-learning's reward curve performs poorly during the training process due to persistent risky exploration, its final learned policy is theoretically optimal (i.e., the shortest path). Conversely, SARSA and its variants achieve more stable and higher immediate returns during training by learning a longer "safe path" to mitigate risk. This research not only validates the effectiveness of different TD algorithms but also, through an empirical analysis of this classic "optimality versus safety" trade-off, provides profound and practical insights for selecting and optimizing decision-making algorithms in real-world problems, such as robotics navigation and game AI. The complete source code is available at `https://github.com/Billiefu/YatRL2025.git`.

Key words: Reinforcement Learning; Temporal Difference Learning; Cliff Walk; On-Policy; Off-Policy; SARSA; Q-learning; Risk-Reward Trade-off.

Copyright © 2025 傅祉珏

1 Introduction

As a continuation of our prior work, we successfully applied Dynamic Programming (DP) methods to solve the optimal path problem in grid mazes under the idealized condition of a known environment model, where state transitions and reward functions are fully specified. While the power of DP algorithms lies in their theoretical completeness, their core premise —a fully known model of the environment —is often difficult to satisfy in many real-world scenarios. When an agent is placed in an unknown environment, it must learn through direct exploration rather than relying on predefined rules.

To address this more general and challenging problem, Temporal Difference (TD) learning provides a powerful framework. Unlike DP methods, TD learning is a model-free approach to reinforcement learning. It does not require a pre-built model of the environment; instead, it learns directly from sampled experience gained through interaction. This characteristic allows TD methods to be widely applied to real-world problems where the environment's rules are unknown or too complex to model. Therefore, adopting the TD approach is a critical step in advancing our application of reinforcement learning theory from idealized models to more realistic scenarios.

This project aims to systematically implement and conduct a comparative analysis of a series of classic TD learning algorithms. We will use the ”Cliff Walk” problem as our core experimental platform, as it serves as an excellent testbed for examining and contrasting how different learning strategies handle the trade-off between risk and reward. Specifically, this project implements the on-policy algorithms SARSA, Expected SARSA, and N-step SARSA, as well as the off-policy algorithm Q-learning. Through an in-depth comparison of their convergence, stability, and the ”safety” versus ”optimality” of their final learned policies, this project seeks to reveal the inherent differences and core mechanisms of these different TD learning paradigms in solving sequential decision-making problems.

2 Related Work

Serving as a theoretical bridge between Dynamic Programming (DP) and Monte Carlo (MC) methods, Temporal Difference (TD) learning has been one of the core research directions in reinforcement learning since its inception. One of its earliest and most celebrated successes was Tesauro’s TD-Gammon, which achieved world-class backgammon performance through TD learning, demonstrating for the first time the immense potential of TD algorithms in solving complex problems. The theoretical framework of TD learning is not confined to game theory; its principles have been widely applied in broader domains. For instance, Kurth-Nelson et al. explored its application with distributed representations, revealing the profound impact of TD learning as a general learning mechanism.

The numerous algorithms within TD learning can be broadly categorized into two main branches based on whether the learning policy is the same as the behavior policy: on-policy and off-policy. The SARSA algorithm is a direct embodiment of on-policy control, learning the value of the agent’s current behavior policy. Due to its conceptual simplicity and robustness, SARSA continues to be a fundamental building block in modern research. For example, Zhao et al. combined SARSA with deep learning and experience replay to explore its application in more complex state spaces. To address the high variance in SARSA’s updates, Van Seijen et al. conducted a thorough theoretical and empirical analysis of Expected SARSA, demonstrating that using expectation calculations can effectively enhance learning stability. Furthermore, to strike a more flexible balance between bias and variance, N-step TD methods were proposed as a more general framework. The work by De Asis et al. further unified this into a generalized algorithm, showcasing the advantage of multi-step learning in accelerating credit assignment.

Developing in parallel to on-policy methods is the off-policy approach, epitomized by Q-learning. The Q-learning algorithm, introduced by Watkins and Dayan, is a landmark contribution to the field of reinforcement learning. Its core max operator allows it to break free from the constraints of the behavior policy and directly learn the optimal value function, a breakthrough idea that made the direct pursuit of an optimal policy possible. The theoretical elegance and power of Q-learning have made it the cornerstone of numerous subsequent algorithms. However, its off-policy nature can also lead to stability issues in certain scenarios, particularly when learning from offline datasets. To address these challenges, modern research continues to evolve; for example, Conservative Q-learning, proposed by Kumar et al., aims to improve the safety and reliability of the algorithm in offline reinforcement learning settings. This also reflects that the trade-off between ”optimality”

Copyright © 2025 傅祉珏

and "safety" initiated by Q-learning remains a central theme in the field today.

This research is built upon this solid theoretical foundation. Through an empirical analysis of these foundational algorithms in the classic "Cliff Walk" environment, we aim to intuitively and systematically reproduce and dissect the classic trade-off between the robustness of on-policy methods (represented by SARSA and its variants) and the optimality of off-policy methods (represented by Q-learning).

3 Method

This project aims to solve the classic Cliff Walk problem, the core of which is to find an optimal path in an environment that contains significant risk (the cliff). Since the environment model for this problem —that is, the state transition probabilities and reward function —is unknown to the agent, traditional Dynamic Programming methods are not applicable. Consequently, we employ a model-free reinforcement learning approach: Temporal Difference (TD) learning. TD learning ingeniously combines the bootstrapping concept from Dynamic Programming with the sampling-based approach of Monte Carlo methods, enabling the agent to learn from direct interaction with the environment and from incomplete episodes.

The core principle of TD learning is that it does not wait for an episode to conclude before updating value estimates. Instead, after each step, it uses the current estimate of a subsequent state's value to update the value of the current state—a process known as updating an estimate with an estimate. This project will build upon this core idea to implement and compare a series of classic TD algorithms, including the foundational SARSA algorithm, its variants Expected SARSA and N-step SARSA, and the contrasting Q-learning algorithm, to comprehensively investigate their characteristics in solving such problems that involve a trade-off between risk and reward.

3.1 Temporal Difference Learning

The theoretical cornerstone of TD learning is the TD(0) algorithm, which lays the groundwork for more complex control algorithms. This algorithm is used to estimate the state-value function $v_\pi(s)$ online under a given policy π. Its update rule is as follows:

$$v_{t+1}(s_t) = v_t(s_t) - \alpha_t(s_t)\left[v_t(s_t) - (r_{t+1} + \gamma v_t(s_{t+1}))\right] \tag{1}$$

In this formula, $v_t(s_t)$ is the current value estimate for state s_t, and $\alpha_t(s_t)$ is the learning rate. The expression $r_{t+1} + \gamma v_t(s_{t+1})$ is known as the "TD target," which is composed of the actual immediate reward r_{t+1} and a discounted estimate of the next state's value (the bootstrapping term). The difference between the TD target and the current estimate $v_t(s_t)$ is the "TD error." However, the basic TD algorithm only estimates state values. For control problems that require selecting optimal actions, such as path planning, we must estimate the state-action value function, $q(s, a)$, which leads to the SARSA algorithm.

3.2 The SARSA Algorithm

The SARSA algorithm is a direct application of the TD method to control problems. Its name is derived from the sequence of data it uses to update Q-values: State, Action, Reward, next State, and

Copyright © 2025 傅祉珏

next Action. SARSA is a classic on-policy algorithm, which means the policy it uses for evaluation and improvement is the same as the one it follows to generate data in the environment (typically an ε-greedy policy).

$$q_{t+1}(s_t, a_t) = q_t(s_t, a_t) - \alpha_t(s_t, a_t)\left[q_t(s_t, a_t) - (r_{t+1} + \gamma q_t(s_{t+1}, a_{t+1}))\right] \tag{2}$$

The key difference from the basic TD algorithm is that the TD target in SARSA, $r_{t+1} + \gamma q_t(s_{t+1}, a_{t+1})$, depends on the actual action a_{t+1} chosen in the next state s_{t+1} according to the current policy. This mechanism causes SARSA to incorporate the risks associated with its exploratory actions (such as random choices under an ε-greedy policy) into its Q-value assessments, thus leading it to learn a relatively ”conservative” or ”safe” policy.

3.3 The Expected SARSA Algorithm

Expected SARSA is a variant of SARSA designed to improve learning stability by reducing the variance in the update process. SARSA's update relies on a single sample of the next action, a_{t+1}, which introduces randomness. In contrast, Expected SARSA replaces this single sample by calculating the expected Q-value over all possible actions in the next state s_{t+1}, according to the current policy π.

$$\text{TD Target} = r_{t+1} + \gamma \mathbb{E}_\pi[q_t(s_{t+1}, \mathcal{A})] = r_{t+1} + \gamma \sum_{a \in \mathcal{A}} \pi(a|s_{t+1}) q_t(s_{t+1}, a) \tag{3}$$

By using an expected value, the update in Expected SARSA is no longer subject to the randomness of a single action choice, making the learning process smoother. In practice, it often exhibits more robust performance than standard SARSA while retaining its on-policy nature.

3.4 The N-step SARSA Algorithm

N-step SARSA is a generalization of SARSA that integrates the ideas of single-step TD learning and Monte Carlo methods. Instead of looking just one step ahead, this algorithm looks ahead n steps to calculate returns, thereby creating a trade-off between bias and variance. Its core idea is the calculation of the n-step return.

The n-step return at time t, denoted $G_{t:t+n}$, is defined as the sum of discounted rewards for the next n steps, plus a discounted estimate of the state-action value at step n (the bootstrapping term).

$$G_{t:t+n} = r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{n-1} r_{t+n} + \gamma^n q_{t+n-1}(s_{t+n}, a_{t+n}) \tag{4}$$

This n-step return is then used to update the Q-value for the state-action pair from n steps prior, (s_t, a_t).

$$q_{t+n+1}(s_t, a_t) = q_{t+n}(s_t, a_t) - \alpha_t(s_t)\left[q_{t+n}(s_t, a_t) - G_{t:t+n}\right] \tag{5}$$

When $n = 1$, this algorithm is equivalent to SARSA; as $n \to \infty$, it approaches the Monte Carlo method. By adjusting the value of n, we can control the bias-variance trade-off during the learning process.

Copyright © 2025 傅祉珏

3.5 The Q-learning Algorithm

Q-learning is another landmark algorithm in TD learning. Unlike SARSA, Q-learning is an off-policy algorithm. This means that the policy it learns about (the target policy, typically the greedy policy) can be different from the policy it uses to generate experiences (the behavior policy, typically an ε-greedy policy). This property allows Q-learning to learn the optimal policy directly, regardless of the exploratory actions taken during training.

$$q_{t+1}(s_t, a_t) = q_t(s_t, a_t) - \alpha_t(s_t)\left[q_t(s_t, a_t) - \left(r_{t+1} + \gamma \max_{a \in \mathcal{A}(s_{t+1})} q_t(s_{t+1}, a)\right)\right] \tag{6}$$

The crucial difference lies in its TD target, $r_{t+1} + \gamma \max_a q_t(s_{t+1}, a)$, which incorporates a max operator. During the update, it always selects the action a that yields the maximum Q-value in the next state s_{t+1} to compute the target value, completely ignoring which action was actually taken. This ”greedy” update mechanism allows it to break free from the constraints of the behavior policy and converge directly toward the optimal action-value function, q^*.E

4 Experiments

To conduct an in-depth investigation into the dynamic characteristics and performance differences of SARSA, Expected SARSA, N-step SARSA, and Q-learning during the actual solution process, we designed and executed a series of comparative experiments. All experiments were based on a unified reinforcement learning framework where an agent explores a discrete grid world. This environment was modeled as a Markov Decision Process, in which the agent receives a penalty of -1 for each move and a large penalty of -100 for stepping into a cliff, which also returns it to the start position. The episode terminates when the agent reaches the goal, which has a reward of 0. This reward structure is designed to compel the algorithms to find the shortest path while strictly avoiding risk. In all tests, we uniformly set the learning rate $\alpha = 0.5$, the discount factor $\gamma = 1.0$, and the exploration rate $\varepsilon = 0.1$, and trained for 500 episodes to ensure sufficient opportunity for convergence.

4.1 Analysis in the Classic Cliff Walk Environment

Our core investigation began with the standard 4×12 Cliff Walk environment. The reward convergence curves and the final state-value (V-value) heatmaps from the experimental results clearly reveal the classic performance divergence between on-policy and off-policy algorithms.

First, observing the reward convergence curves, the SARSA and Expected SARSA algorithms (blue and orange curves) rapidly converge to a relatively high reward range (around -25) and exhibit stable performance after an initial phase of exploration. This aligns perfectly with theoretical expectations. As on-policy algorithms, they learn the value of the ε-greedy policy they are actually executing. Since this policy inherently includes the risk of random exploration (i.e., the probability of falling when near the cliff), the algorithms incorporate this risk into their Q-value evaluation. Consequently, the learned policy actively avoids the cliff by choosing a longer, ”safe path.” By taking the safe path, the agent rarely falls off the cliff in the later stages of training, thus avoiding the large -100 penalty and achieving a stable and higher sum of rewards per episode.

Copyright © 2025 傅祉珏

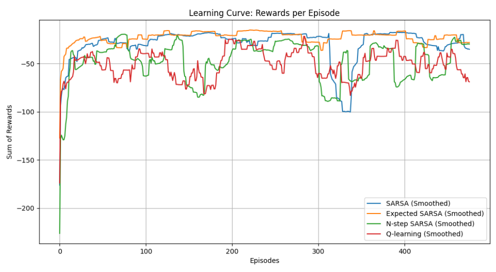


Fig 1: Classic Environment Reward Curve

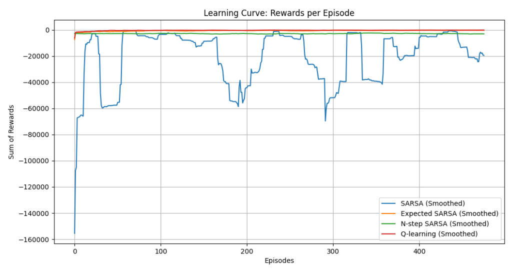


Fig 2: Complex Environment Reward Curve

In stark contrast is the Q-learning algorithm (red curve), whose reward curve remains at a low level (around -60) and exhibits high volatility throughout the entire training process. The fundamental reason lies in its off-policy nature. Its update rule, which relies on the max operator, is designed to directly learn the value of the optimal policy, q^*. This optimal policy, however, happens to be the "risky path" that runs along the edge of the cliff. During training, the agent still follows the ε-greedy policy in practice. Therefore, as it learns to walk along the cliff edge, any random downward exploration will cause it to fall. This creates a core paradox: Q-learning learns the optimal policy, but its actual performance during training is poor due to its persistent, high-risk exploration.

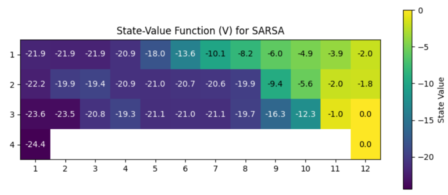
However, when we shift our analysis to the final V-value heatmaps, the conclusion is reversed. Q-learning's V-values (which are closer to 0, i.e., less negative) are significantly better than those of SARSA and Expected SARSA. This is precisely what demonstrates the success of Q-learning. The V-value heatmap reflects the expected return of the final learned optimal policy. Q-learning found the shortest path (approx. 13 steps), and its starting state's V-value converges to around -12. In contrast, SARSA learned a longer, safe path (approx. 25+ steps), and its V-value converges to around -24. This perfectly explains the aforementioned phenomenon: Q-learning's low training reward is a consequence of its courageous exploration of the optimal path, while its higher final V-values are proof that it did, in fact, find that theoretically most efficient route.

4.2 Performance Analysis in a Large and Complex Environment

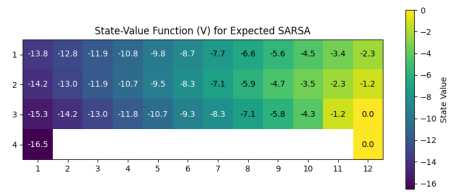
To validate the robustness of the algorithms in more complex scenarios, we extended our experiments to a large 21×21 maze containing numerous cliff traps. In this environment, the performance differences between algorithms were drastically amplified, exposing their deeper characteristics.

The experimental results show that the SARSA algorithm struggles to converge effectively in this complex environment. Its reward curve remains at an extremely low level and is filled with noise, performing far worse than Expected SARSA and Q-learning. The root cause is the high-variance nature of SARSA's update mechanism. In a large state space, SARSA's reliance on a single sample of the next action to update Q-values causes the disturbances from this randomness to be severely magnified. A single unfortunate exploratory choice can propagate a large negative Q-value backward, "polluting" the value table and causing the agent to "lose its way" in the vast maze, making it difficult to learn a coherent policy. In comparison, Expected SARSA smooths out this randomness by taking the expectation over all future actions, resulting in a more robust update and demonstrating far superior stability and convergence in this complex environment.

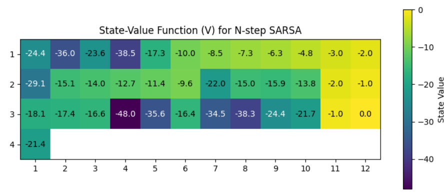
Copyright © 2025 傅祉珏



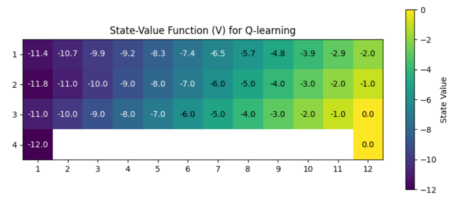
(a) Value Heatmap of SARSA



(b) Value Heatmap of Expected SARSA

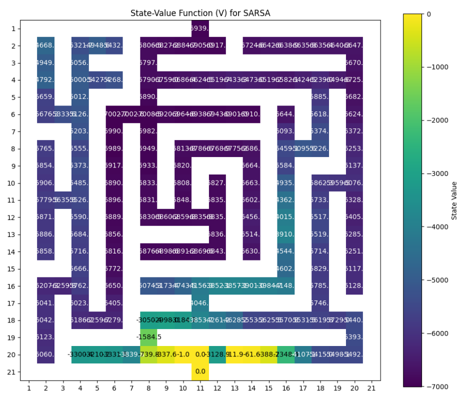


(c) Value Heatmap of N-step SARSA

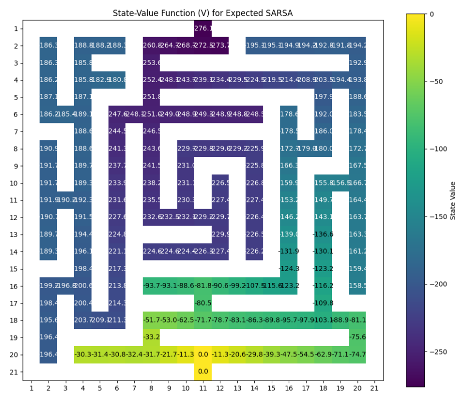


(d) Value Heatmap of Q-learning

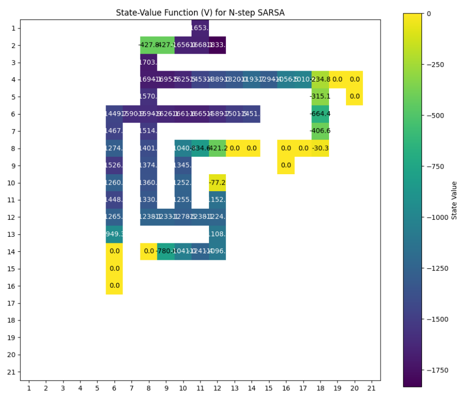
Fig 3: Classic Environment State-Value Heatmap



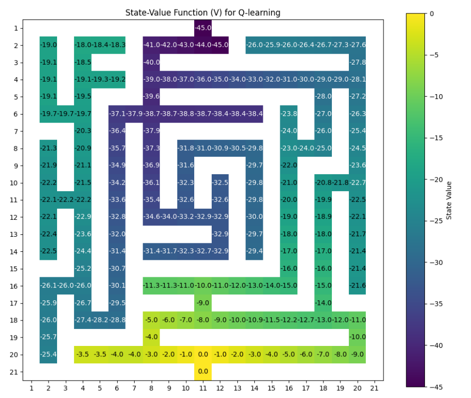
(a) Value Heatmap of SARSA



(b) Value Heatmap of Expected SARSA



(c) Value Heatmap of N-step SARSA



(d) Value Heatmap of Q-learning

Fig 4: Complex Environment State-Value Heatmap

Even more extreme is the case of the N-step SARSA algorithm. As seen from its V-value heatmap, it failed to even complete the exploration of the entire maze, with a large number of states left unvisited. This profoundly reveals the decisive impact of the hyperparameter n (the number of steps) on the algorithm's performance. The value of n controls the bias-variance trade-off: a smaller n leads to high bias and low variance (closer to SARSA), while a larger n results in low bias and high variance (closer to Monte Carlo). In an environment that is both large and full of traps, a fixed value of n (in this case, 5) is likely inappropriate. The long n-step return chain means that the large negative reward from a single fall off a cliff can be retrospectively attributed to a series of state-action pairs from n steps prior, a challenge known as the Credit Assignment Problem. This erroneous penalty assignment severely pollutes the Q-table, leading to a chaotic and disordered policy that ultimately prevents the agent from exploring effectively. This demonstrates that while N-step SARSA can theoretically accelerate learning, its high sensitivity to the hyperparameter n makes it difficult to tune and apply in unknown and complex environments.

5 Conclusion

Through the solution and analysis of the Cliff Walk problem, our research profoundly reveals the inherent philosophical trade-off between optimality and safety that exists between on-policy and off-policy reinforcement learning algorithms. The experimental results do not merely present a linear ranking of algorithm performance; rather, they clearly demonstrate that off-policy algorithms, represented by Q-learning, aim to approximate a theoretical optimum without regard for the costs of exploration. Consequently, they can find the most efficient path but exhibit high-risk behavior during the learning process. In contrast, on-policy algorithms, represented by SARSA and its variants, learn faithfully from their own experience, incorporating the uncertainty and risks of exploration into their value assessments, thereby learning a policy that is robust and safe, albeit potentially suboptimal. The core insight from this finding is that the choice of algorithm is not a purely technical issue but a strategic decision dependent on the specific application context. For tasks where near-infinite trial and error is permissible in a simulated environment to pursue peak performance, Q-learning is the undisputed choice. However, for real-world applications where any single failure could have severe consequences (such as robotics navigation), the robustness of SARSA becomes critically important.

This conclusion offers new perspectives and future directions for algorithmic optimization. Since each policy paradigm possesses its own irreplaceable advantages, future research can focus on how to dynamically and adaptively fuse their strengths, rather than rigidly adhering to one.

First, one could explore more intelligent safe exploration strategies. Instead of allowing Q-learning to blindly conduct ε-greedy exploration while learning the optimal policy, a risk-aware exploration mechanism could be designed. For instance, the algorithm could simultaneously estimate an action's Q-value and its uncertainty (or variance), prioritizing for exploration those actions that are "potentially high-reward and have low uncertainty." This would significantly reduce the probability of catastrophic failures during the learning process while still guaranteeing convergence to the optimal solution.

Second, hybrid or curriculum-based learning algorithms could be designed. An agent could adopt a multi-stage learning strategy: in the initial training phase, it could use a conservative

Copyright © 2025 傅祉珏

on-policy approach like SARSA to quickly learn a safe, error-free baseline policy and build a fundamental understanding of the environment. In the later stages, once the agent is capable of avoiding core risks, its update target could gradually transition to the max operator of Q-learning to optimize the existing safe policy and approach the theoretical optimum. This paradigm shift from ”survival” to ”perfection” holds the promise of achieving both safety and optimality.

Finally, future research could incorporate risk as part of the optimization objective, rather than solely maximizing expected return. The framework of Risk-Sensitive Reinforcement Learning could be introduced, incorporating risk measures such as the variance of returns or Conditional Value at Risk (CVaR) into the objective function. This would allow the agent to autonomously trade-off between varying degrees of ”aggressive” and ”conservative” policies based on a predefined risk preference. The design of these adaptive and hybrid strategies means we no longer view each algorithm in isolation but as components in a toolbox to be intelligently combined based on real-time feedback from the problem. This approach promises to achieve a superior combination of computational efficiency and convergence stability across a broader range of complex decision-making problems.

References

[1] 董豪, 丁子涵, 仉尚航等. 深度强化学习: 基础、研究与应用 [M]. 第 1 版. 北京: 电子工业出版社, 2021.

[2] 张伟楠, 沈键, 俞勇. 动手学强化学习 [M]. 第 1 版. 北京: 人民邮电出版社, 2022.

[3] 赵世钰. 强化学习的数学原理 [M]. 第 1 版. 北京: 清华大学出版社, 2025.

[4] 赵世钰. 强化学习的数学原理：英文 [M]. 第 1 版. 北京: 清华大学出版社, 2024.

[5] De Asis K, Hernandez-Garcia J, Holland G, et al. Multi-step reinforcement learning: A unifying algorithm[C]//Proceedings of the AAAI conference on artificial intelligence. 2018, 32(1).

[6] Kumar A, Zhou A, Tucker G, et al. Conservative q-learning for offline reinforcement learning[J]. Advances in neural information processing systems, 2020, 33: 1179-1191.

[7] Kurth-Nelson Z, Redish A D. Temporal-difference reinforcement learning with distributed representations[J]. PLoS One, 2009, 4(10): e7362.

[8] Tesauro G. Temporal difference learning and TD-Gammon[J]. Communications of the ACM, 1995, 38(3): 58-68.

[9] Van Seijen H, Van Hasselt H, Whiteson S, et al. A theoretical and empirical analysis of expected sarsa[C]//2009 ieee symposium on adaptive dynamic programming and reinfo rcement learning. IEEE, 2009: 177-184.

[10] Watkins C J C H, Dayan P. Q-learning[J]. Machine learning, 1992, 8(3): 279-292.

[11] Zhao D, Wang H, Shao K, et al. Deep reinforcement learning with experience replay based on SARSA[C]//2016 IEEE symposium series on computational intelligence (SSCI). IEEE, 2016: 1-6.

Copyright © 2025 傅祉珏

Appendix

A Pseudocode for Different Algorithms

Algorithm 1 SARSA

```
Require: Initialize learning rate $\alpha_t(s,a) = \alpha > 0$ and $q_0(s,a)$ for all $(s,a)$. Initialize $\varepsilon$-greedy policy $\pi_0$ derived from $q_0$ and $\varepsilon \in (0,1)$.
Ensure: Find the optimal policy that can guide the agent from an initial state $s_0$ to a terminal state.
 1: for each episode do
 2:     Generate action $a_0$ at state $s_0$ according to $\pi_0(s_0)$.
 3:     if $s_t$ (for $t = 0, 1, 2, ...$) is not a terminal state then
 4:         Collect an experience tuple $(r_{t+1}, s_{t+1}, a_{t+1})$ resulting from $(s_t, a_t)$.
 5:         $q_{t+1}(s_t, a_t) \leftarrow q_t(s_t, a_t) - \alpha[q_t(s_t, a_t) - (r_{t+1} + \gamma q_t(s_{t+1}, a_{t+1}))]$.
 6:         For $a = \arg\max_a q_{t+1}(s_t, a)$, set $\pi_{t+1}(a|s_t) = 1 - \varepsilon + \frac{\varepsilon}{|\mathcal{A}(s_t)|}$.
 7:         Otherwise, set $\pi_{t+1}(a|s_t) = \frac{\varepsilon}{|\mathcal{A}(s_t)|}$.
 8:         $s_t \leftarrow s_{t+1}, a_t \leftarrow a_{t+1}$.
 9:     end if
10: end for
```

For Expected Sarsa and n-step Sarsa algorithms, only the TD Target in the update rule needs to be replaced with the corresponding TD Target.

Algorithm 2 Q-learning (Off-Policy Algorithm)

```
Require: Initialize learning rate $\alpha_t(s,a) = \alpha > 0$, $q_0(s,a)$, and a behavior policy $\pi_b(a|s)$ for all $(s,a)$.
Ensure: Learn the optimal target policy $\pi_T$ from experience samples generated by $\pi_b$.
1: for each episode $(s_0, a_0, r_1, s_1, a_1, r_2, ...)$ generated by $\pi_b$ do
2:     for each step of the episode, $t = 0, 1, 2, ...$ do
3:         $q_{t+1}(s_t, a_t) \leftarrow q_t(s_t, a_t) - \alpha(s_t, a_t)[q_t(s_t, a_t) - (r_{t+1} + \gamma \max_a q_t(s_{t+1}, a))]$.
4:         For $a = \arg\max_a q_{t+1}(s_t, a)$, set $\pi_{T,t+1}(a|s_t) = 1$.
5:         Otherwise, set $\pi_{T,t+1}(a|s_t) = 0$.
6:     end for
7: end for
```

Copyright © 2025 傅祉珏