RL assignment 3

Xiaoyan Jiang
Student ID: 202385007

August 5, 2025

Abstract

This report presents the implementation and comparative analysis of two reinforcement learning algorithms—SARSA and Q-learning—applied to a custom-designed Gridworld environment. The objective is to enable an agent to learn an optimal path from a fixed starting position to one of two terminal goal states, while avoiding trap states that yield heavy penalties. We apply an ϵ-greedy strategy to balance exploration and exploitation and evaluate each method based on learned trajectories and cumulative rewards over episodes. The results show that SARSA tends to learn a safer and more conservative policy, whereas Q-learning explores more aggressively, discovering efficient paths but with higher reward variance. These findings illustrate the behavioral differences between on-policy and off-policy learning in stochastic environments.

Code Repository

The complete implementation can be found at the GitHub repository:
`https://github.com/Xy023max/RLassignment2`

1 Introduction

Reinforcement learning (RL) is a computational framework in which an agent interacts with an environment to learn optimal behavior through trial and error, guided by scalar rewards. Two fundamental model-free RL algorithms, SARSA (State–Action–Reward–State–Action) and Q-learning, represent contrasting approaches to value-based learning: the former is on-policy, learning from the actions actually taken, while the latter is off-policy, learning from the actions that would be taken under a greedy policy.

This report investigates and compares the performance of SARSA and Q-learning in a custom 5x5 Gridworld environment that includes terminal goal states, high-penalty trap states, and movement constraints. The agent starts from a designated initial state and seeks to discover an optimal policy that maximizes expected cumulative rewards while navigating through this structured state space.

We evaluate both algorithms using an ϵ-greedy exploration strategy and analyze their performance based on two key outcomes: the trajectory followed under the learned policy and the total episode rewards accumulated during training. Through empirical results, we highlight the behavioral tendencies of each algorithm and discuss their implications for safety, efficiency, and convergence. The remainder of the report is organized as follows: Section 2 describes the environment and reward design; Section 3 introduces the algorithms; Section 4 presents results and analysis; and Section 5 concludes with discussion and future work.

2 Environment Setup

We define a 5×5 Gridworld environment located in the first quadrant of the Cartesian plane. The environment includes:

- **Start State:** $(1, 5)$, the fixed starting position of the agent.
- **Goal States:** $(5, 5)$ (green) and $(5, 1)$ (yellow), each yielding a terminal reward of $+100$.
- **Trap State:** $(3, 2)$ (red), which yields a heavy penalty of -100 and terminates the episode.
- **Special State:** $(2, 5)$ (blue), a neutral state without reward.

Action Space

The agent can choose from four deterministic actions at each state: `Up`, `Down`, `Left`, and `Right`. Actions that would move the agent off the grid are invalid and result in no movement, with a step penalty still applied.

Reward Structure

- Reaching a goal state (green or yellow): $+100$
- Entering the trap state (red): -100
- All other moves: step penalty of -1

Transition Dynamics

The environment is deterministic: the chosen action leads to the corresponding next state unless it hits the grid boundary. Episodes terminate when the agent reaches either a goal state or the trap state.

3 Methods

3.1 SARSA Algorithm

SARSA (State–Action–Reward–State–Action) is an on-policy temporal-difference learning algorithm. The Q-value update rule is:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \right] \tag{1}$$

This update depends on the action *actually taken* under the current policy. An ϵ-greedy strategy is used for exploration: with probability ϵ, a random action is chosen; otherwise, the action with the highest Q-value is selected.

3.2 Q-learning Algorithm

Q-learning is an off-policy method where the update uses the maximum possible Q-value at the next state, regardless of the policy used to generate the action:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right] \tag{2}$$

Despite using ϵ-greedy for behavior, the learning update targets the greedy policy, making it off-policy.

3.3 Implementation Details

The following hyperparameters are used for both algorithms:

- Learning rate $\alpha = 0.1$
- Discount factor $\gamma = 0.95$
- Exploration rate $\epsilon = 0.1$ (fixed)
- Number of episodes: 500

The ϵ-greedy strategy samples uniformly among all actions with probability ϵ, and otherwise chooses the greedy action with respect to current Q-values.

The Q-values are stored in a 3-dimensional array $Q[s, a]$ where s is the state index (flattened from the 2D grid) and a is one of the four actions. This Q-table is initialized to zero for all (s, a) pairs.

4 Results and Discussion

4.1 Policy Trajectories

Figure 1 and Figure 2 displays the greedy trajectories derived from the learned Q-tables for SARSA and Q-learning, both starting at state $(1, 5)$. In the SARSA case, the agent fails to develop a full trajectory and halts at the starting state,

with only a small arrow (likely indicating a low-confidence action). This suggests either poor convergence or extreme conservatism in the learned policy. By contrast, Q-learning successfully learns a complete trajectory that moves from $(1,5) \to (2,5) \to (2,4) \to (2,3) \to (3,3) \to (4,3) \to (4,2) \to (5,2) \to (5,1)$, terminating at the high-reward green state. The trajectory is visualized using a color gradient from blue to red, reflecting the time order of state visits.

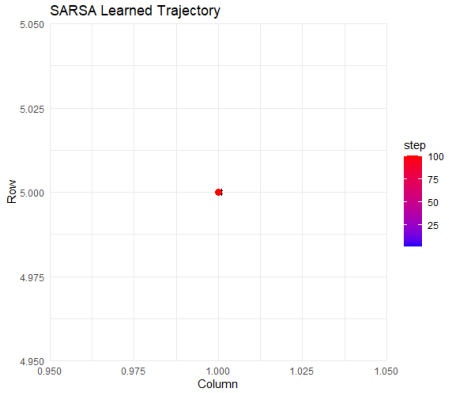

Figure 1: Trajectory learned by SARSA

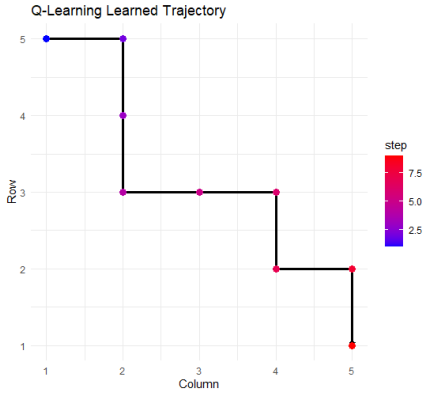

Figure 2: Trajectory learned by Q-learning

4.2 Cumulative Reward Comparison

Figure 3 compares the learning performance of both algorithms using cumulative episode rewards over training. The red curve represents SARSA, and the green curve represents Q-learning. Both series are shown with raw data and smoothed trend lines.

SARSA's reward curve exhibits relatively low variance and lies predominantly between -125 and -100, with most episodes converging near -100. A few episodes rise slightly above this threshold, but the overall reward level is lower. The smoothed trend line for SARSA indicates early improvement and eventual stabilization around episode 375.

Q-learning, on the other hand, displays higher reward variability. Its episodes span a broader range—from approximately -250 up to 0—with frequent oscillations. The average performance gradually improves, and the trend line shows a slow but steady rise from around -135 to -128, suggesting continued learning but less stability than SARSA.

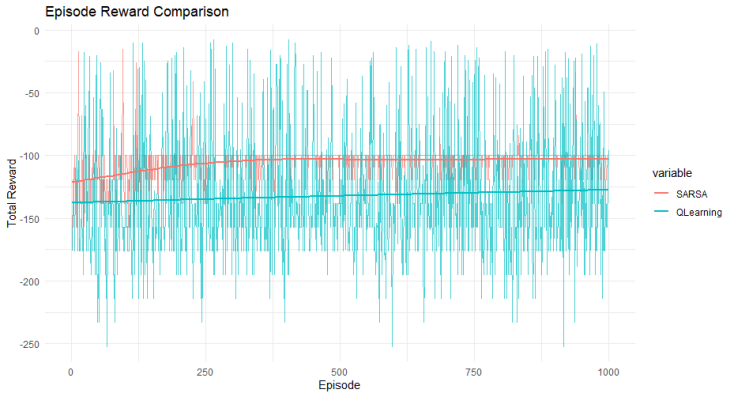

Figure 3: Cumulative episode rewards: red (SARSA), green (Q-learning). Solid lines indicate moving averages.

4.3 Discussion

These results reveal a fundamental trade-off between learning stability and long-term optimality. Q-learning, being an off-policy algorithm, learns from the greedy policy regardless of the behavior policy, enabling it to converge to the optimal path—as evidenced by its successful trajectory from $(1, 5)$ to the goal. However, its training process exhibits higher variance and frequently encounters negative rewards, especially in early episodes. This reflects Q-learning's tendency to explore aggressively, which can lead to high penalty situations before converging (Watkins and Dayan, 1992).

By contrast, SARSA, due to its on-policy nature, updates values based on the current behavior, which includes exploratory actions. As a result, it tends to learn safer but sometimes suboptimal policies, particularly avoiding risky states such as the trap at $(3, 2)$. The trajectory plot indicates its failure to reach the terminal state consistently, suggesting that while SARSA's reward curve is more stable, its policy may be trapped in local optima (Sutton et al., 1998).

In summary, although SARSA yields smoother and safer learning dynamics, Q-learning outperforms it in terms of asymptotic optimality. This echoes

findings from prior empirical studies (Van Seijen et al., 2009) which show that Q-learning, despite higher early variance, often leads to more optimal solutions when sufficient exploration is allowed. Tuning exploration parameters (e.g., ε decay schedules) and hybrid methods like Expected SARSA may help balance this trade-off.

References

Sutton, R. S., Barto, A. G., et al. (1998). *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge.

Van Seijen, H., Fatemi, M., Szepesvári, C., and Sutton, R. S. (2009). A theoretical and empirical analysis of expected sarsa. In *Advances in neural information processing systems*, pages 815–823.

Watkins, C. J. and Dayan, P. (1992). Q-learning. *Machine learning*, 8(3-4):279–292.