CS 329 – Foundations of AI : Multiagent Systems

Reinforcement Learning in a Custom GridWorld Maze:

Learning Behaviors of Q-Learning and SARSA

Date: November 23, 2025

Team Members:

- **Banoth Arvind Nayak (23110058)**
- **Chitteti Vagdevi Reddy (23110080)**
- **Tejavath Vinod Kumar (23110338)**
- **Kunal Singh Rathore (23110181)**

Abstract

In this project, we designed a small yet fully functional Reinforcement Learning (RL) environment from scratch and trained two classic tabular RL agents—Q-Learning and SARSA—to learn how to navigate a 5×5 GridWorld maze. The maze contains a start position, a goal, and a vertical wall of obstacles that forces the agent to take a detour. We compare learning curves, convergence behavior, and final policies of the two algorithms. Our analysis highlights the differences between off-policy and on-policy learning, especially in terms of exploration risk and stability.

Introduction

Problem Statement

Reinforcement Learning allows an agent to learn optimal behavior by interacting with an environment, receiving rewards, and gradually improving its policy. Even in small discrete

environments, the choice of algorithm influences how the agent explores, how quickly it converges, and how safely it behaves.

The problem we focused on was:
How do Q-Learning and SARSA behave when learning to navigate a GridWorld maze with obstacles?
We wanted to see which method converges faster, which is more stable, and how their final strategies differ.

Objectives

Our main goals were:

- To create our own GridWorld Maze environment in Python without using external RL libraries.
- To implement and train **Q-Learning** (off-policy) and **SARSA** (on-policy) agents.
- To collect and compare:
    - rewards per episode
    - episode lengths
    - final learned policies
- To understand how the update rules of the algorithms influence their learning behavior in practice.

Methodology

Algorithms

Q-Learning

Q-Learning updates its Q-values using the best possible future action:

$$Q(S, A) \leftarrow Q(S, A) + \alpha(R + \gamma \max_{a'} Q(S', a') - Q(S, A))$$

It is **off-policy**, meaning it learns the greedy optimal policy even while following an ε-greedy exploration strategy. In practice, this makes it more aggressive and usually faster to converge.

SARSA

SARSA updates based on the action the agent *actually* takes under the current policy:

$$Q(S, A) \leftarrow Q(S, A) + \alpha(R + \gamma Q(S', A') - Q(S, A))$$

SARSA is **on-policy**, making it more cautious because it accounts for the risk of exploratory actions. It may converge slower but behaves more safely.

Environment

We implemented a simple 5×5 GridWorld maze.

Environment Details:

- Grid size: **5 × 5**
- Start position: **(0, 0)**
- Goal position: **(4, 4)**
- Obstacles: **(1,1), (2,1), (3,1)** (a vertical wall)
- Actions: Up, Down, Left, Right
- Rewards:
    - **+10** for reaching goal
    - **–1** per step
- Episode ends if the agent reaches the goal or exceeds **50 steps**

This environment creates an intentional bottleneck, forcing the agent to take a detour and making the learning behavior easy to observe.

Training Setup

To get more stable and readable learning curves, we trained each algorithm **5 times** and averaged the results.
Other details:

- Episodes per run: **500**

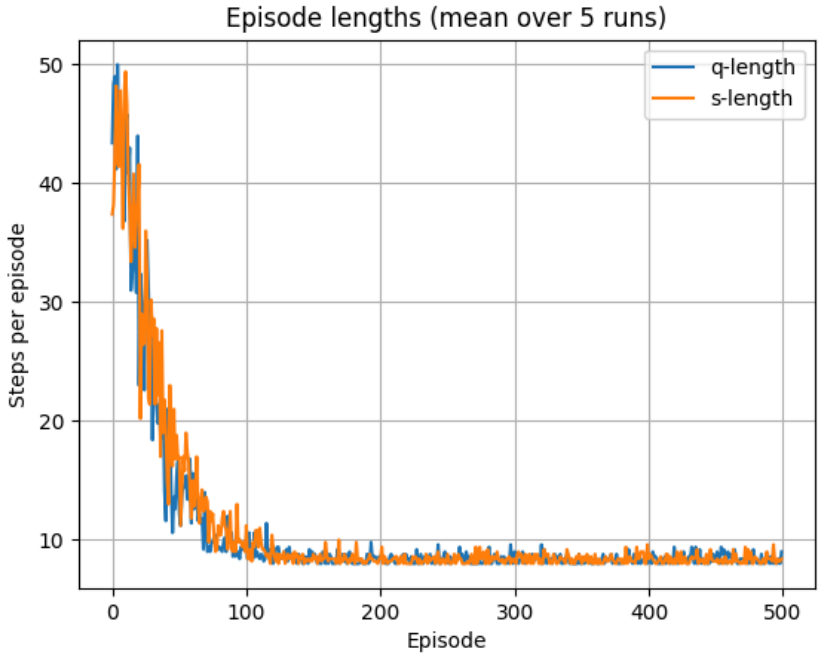
- Learning rate $\alpha = 0.1$
- Discount $\gamma = 0.99$
- ε-greedy exploration with ε decaying from $1.0 \rightarrow 0.05$
- Decay rate: **0.98** (slightly faster to reduce noise)
- We applied a **moving average of 50 episodes** for smoothing.

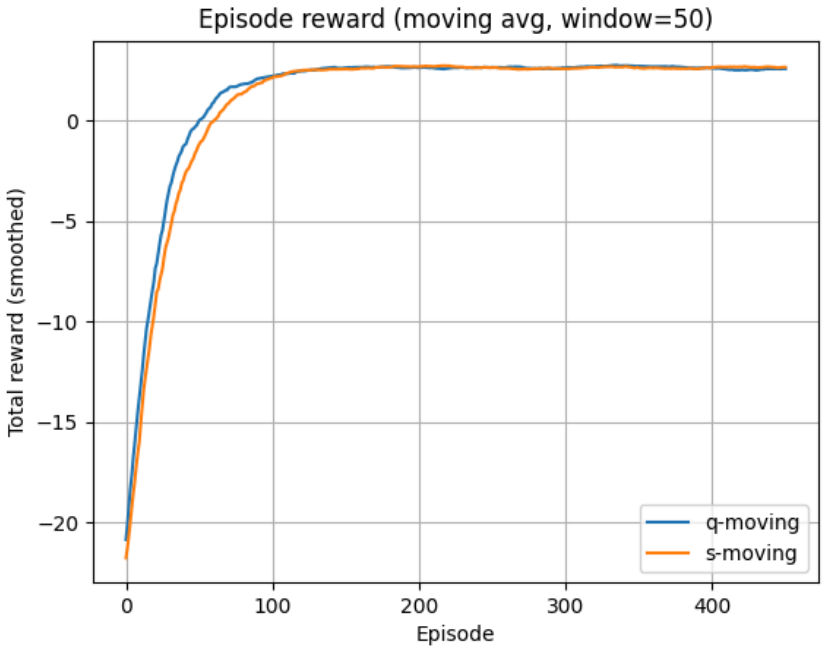
We recorded:

- total reward per episode
- steps taken per episode
- final greedy policy

Experimental Results

Episode Rewards



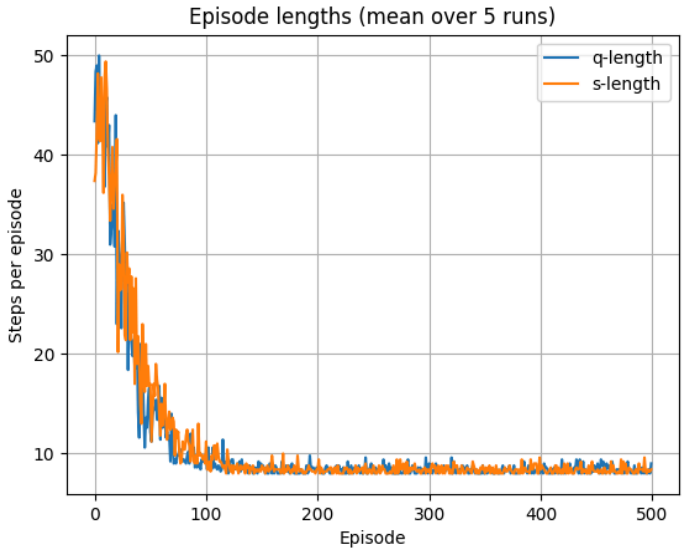


Our raw reward plots were naturally noisy, but after smoothing, the trends became clear:

- **Q-Learning** rises faster and stabilizes earlier.
- **SARSA** has a slower start because it considers exploratory moves in its update.
- After around 150–200 episodes, both agents reach similar reward levels.

In general, Q-Learning behaved more aggressively, while SARSA trained more cautiously.

Episode Lengths



Episode lengths also showed a noticeable difference:

- Q-Learning quickly discovers a short, efficient path around the obstacle.
- SARSA's episode length decreases more slowly because it avoids risky shortcuts early on.
- Eventually both agents converge to a path of around 8–10 steps, which is near-optimal for our maze.

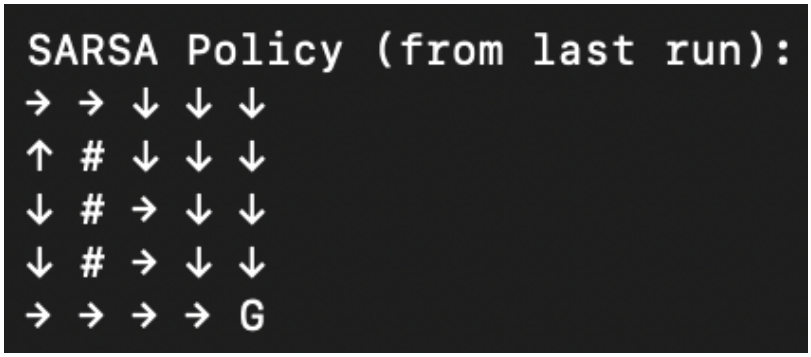
Learned Policies

Both learned policies eventually make sense for the maze layout.

Q-Learning Final Policy

```
Q-Learning Policy (from last run):
→ → → ↓ ↓
↑ # ↓ ↓ ↓
↓ # → ↓ ↓
↓ # ↓ ↓ ↓
→ → → → G
```

SARSA Final Policy



Interpretation

- Q-Learning generally prefers the most direct and efficient path.
- SARSA tends to be more cautious around the obstacle region, which is expected given its on-policy nature.
- Both policies successfully reach the goal by avoiding the obstacle wall.

Contributions

Banoth Arvind Nayak (23110058)

- Developed the complete GridWorld environment
- Integrated Q-Learning and SARSA into a unified training workflow
- Ran multi-run averaging experiments and generated all plots
- Helped interpret algorithm behavior and finalize the analysis

Chitteti Vagdevi (23110080)

- Verified environment correctness, transitions, and reward mechanisms
- Assisted in tuning hyperparameters, especially ε-decay and smoothing
- Helped debug agent movement and convergence issues

Tejavath Vinod Kumar (23110338)

- Analyzed learning curves and contributed to writing the experimental results
- Compared the final policies and studied differences between algorithms
- Helped organize experiment outputs and document findings

Kunal Singh Rathore (23110181)

- Structured the project repository and coordinated code formatting
- Prepared the written report and integrated all figures and explanations
- Reviewed code, ensured readability, and maintained version consistency

---

Conclusion

Through our experiments on the custom GridWorld maze, we observed clear differences between Q-Learning and SARSA. Q-Learning converges faster and learns a more direct path to the goal, while SARSA behaves more safely during exploration and takes longer to converge. Despite the differences during training, both ultimately discover near-optimal policies.

Designing our own environment and experimenting with two fundamental RL algorithms helped us understand how exploration, update rules, and policy types influence learning performance.

References

1. Sutton, Richard S. & Barto, Andrew G. *Reinforcement Learning: An Introduction*, MIT Press, 2018.