CSE546: Reinforcement Learning
Assignment 1

Suvigya Vijay

February 22, 2024

Part 1: Defining RL Environments

1. Describe the deterministic and stochastic environments, which were defined (set of actions/states/rewards, main objective, etc).

- **Deterministic Environment:**
  - **Set of Actions:** The agent can take four actions - DOWN, UP, RIGHT, and LEFT.
  - **Set of States:** The environment is represented as a 3x4 grid, and the state is a flattened representation of the grid, indicating the current positions of the agent and the goal.
  - **Rewards:** The agent receives rewards based on the *rewards_map*, which contains rewards at specific grid locations. Additional rewards and penalties are specified at certain grid locations.
  - **Main Objective:** The goal is for the agent to reach the specified goal position within a maximum number of timesteps (10 in this case).
- **Stochastic Environment:**
  - **Extension of Deterministic Environment:** The stochastic environment is an extension of the deterministic environment with the addition of stochasticity.
  - **Stochasticity:** There is a probability ($p_{\text{stochastic}}$) that the agent's chosen action will be replaced by a random action. This introduces randomness into the environment, making it stochastic.

2. Provide visualizations of your environments.

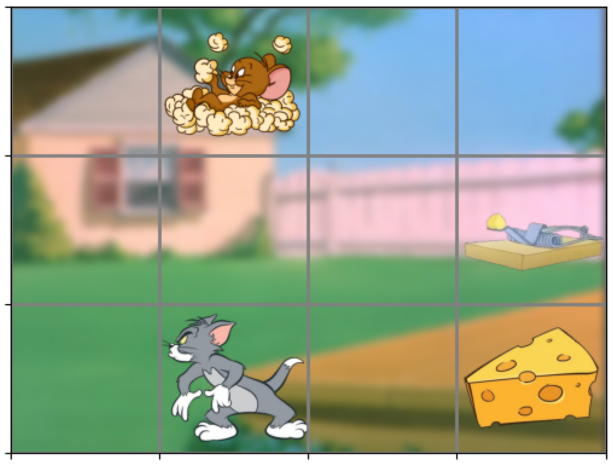


Figure 1: Deterministic Environment Visualization

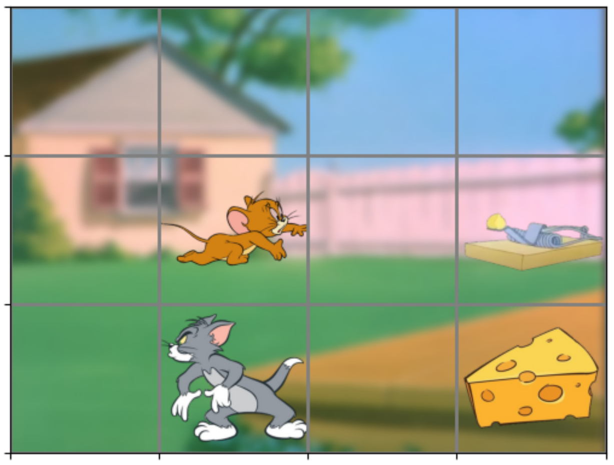


Figure 2: Stochastic Environment Visualization

3. How did you define the stochastic environment?

The stochastic environment is defined by creating a new class `StochasticGridEnv` that inherits from the `DeterministicGridEnv` class. The `step` method in the `StochasticGridEnv` class introduces stochasticity by randomly choosing to replace the agent's chosen action with a random action based on the probability *p_stochastic*.

4. What is the difference between the deterministic and stochastic environments?

In the deterministic environment, the agent's chosen action determines its next state completely. In the stochastic environment, there is a probability that the agent's chosen action will be replaced by a random action, introducing randomness into the agent's decision-making process.

5. Safety in AI: Write a brief review (∼5 sentences) explaining how you ensure the safety of your environments. E.g. how do you ensure that agent choose only actions that are allowed, that agent is navigating within defined state-space, etc.

The safety measures in the code include:

- The `step` method in both environments checks if the agent's position is within the permitted cells, ensuring the agent is navigating within the defined state-space.
- The agent's position is clipped to stay within the grid boundaries, providing a safe and predictable manner for the agent to learn.
- The code includes checks for termination (`terminated`) and truncation (`truncated`) conditions to ensure the safety of the agent and the environment.

Part 2: Applying Tabular Methods

1. Show and discuss the results after:

- **Applying Q-learning to solve the deterministic environment defined in Part 1. Plots should include epsilon decay and total reward per episode.**

  The Q-learning algorithm was successful in optimizing the agent's policy in the deterministic environment, as indicated by the increasing trend in the total rewards per episode. The epsilon decay plot shows how exploration was reduced over time, allowing the agent to exploit the best-known actions.

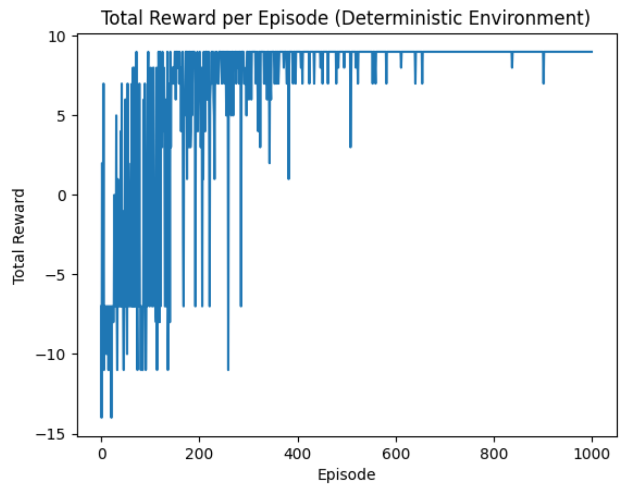


Figure 3: Total Reward per Episode

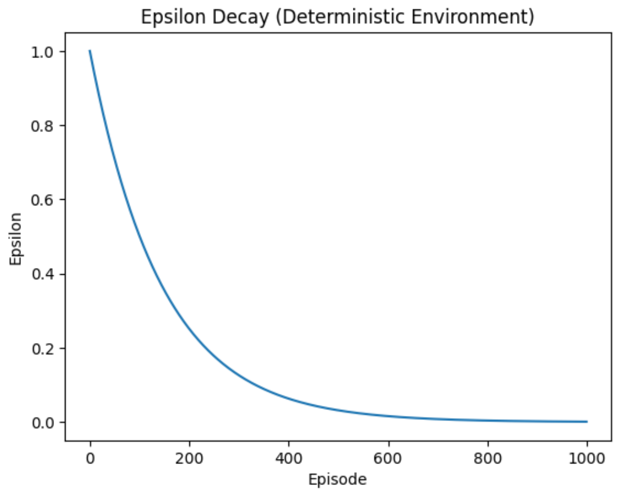


Figure 4: Epsilon Decay

- **Applying Q-learning to solve the stochastic environment defined in Part 1. Plots should include epsilon decay and total reward per episode.**

  In the stochastic environment, the agent faced more challenges due to the introduction of randomness in action outcomes. Despite this, the agent learned a good policy, as reflected by the gradual improvement in the total rewards per episode, even though it is more erratic compared to the deterministic case.

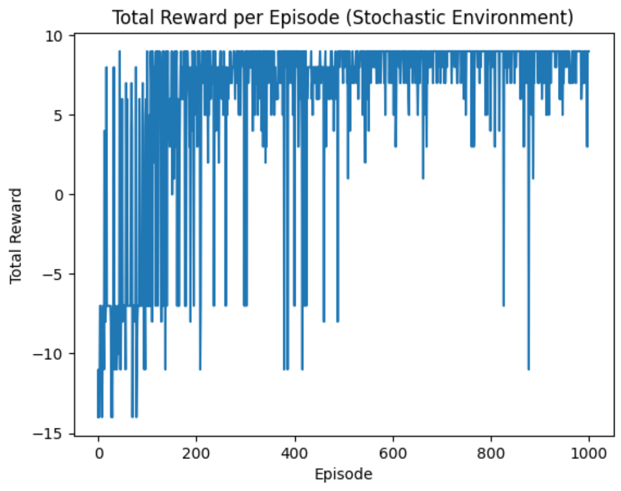


Figure 5: Total Reward per Episode

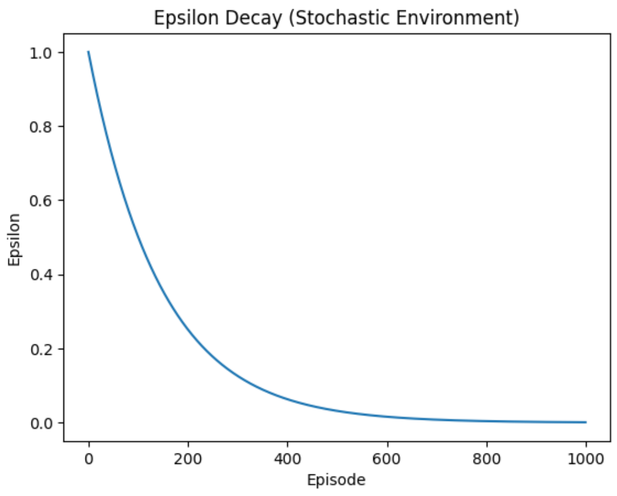


Figure 6: Epsilon Decay

- **Applying any other algorithm of your choice to solve the deterministic environment defined in Part 1. Plots should include total reward per episode.**

  The SARSA algorithm exhibits trends similar to those observed with Q-learning when applied to the deterministic environment. The epsilon decay plot indicates a steady transition from exploration to exploitation. The total rewards per episode show a trend towards stability and improved performance, akin to the Q-learning results, suggesting effective policy learning.

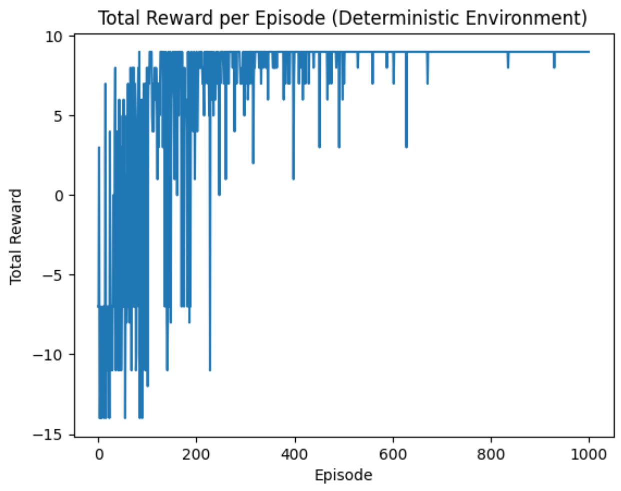


Figure 7: Total Reward per Episode

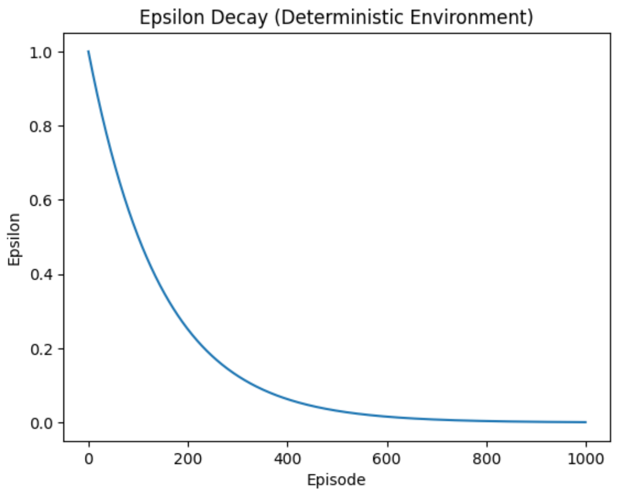


Figure 8: Epsilon Decay

- **Applying any other algorithm of your choice to solve the stochastic environment defined in Part 1. Plots should include total reward per episode.**

In the stochastic environment, SARSA's performance trends again align closely with those from Q-learning. The epsilon decay demonstrates a similar decrease, and the total rewards per episode plot, while more erratic due to environmental stochasticity, shows an overall trend towards learning a stable and effective policy, as seen with Q-learning.

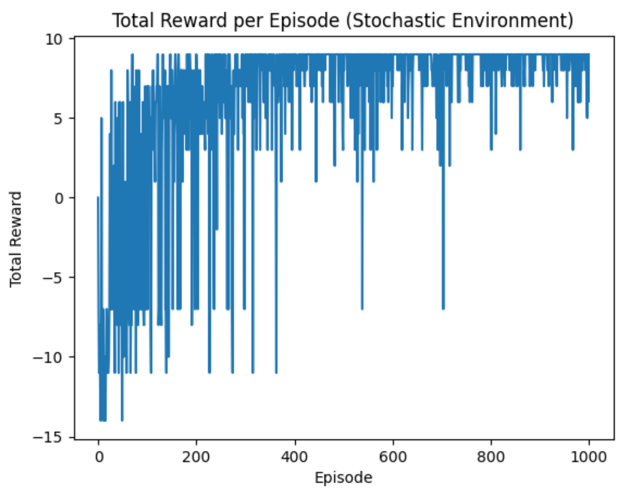


Figure 9: Total Reward per Episode

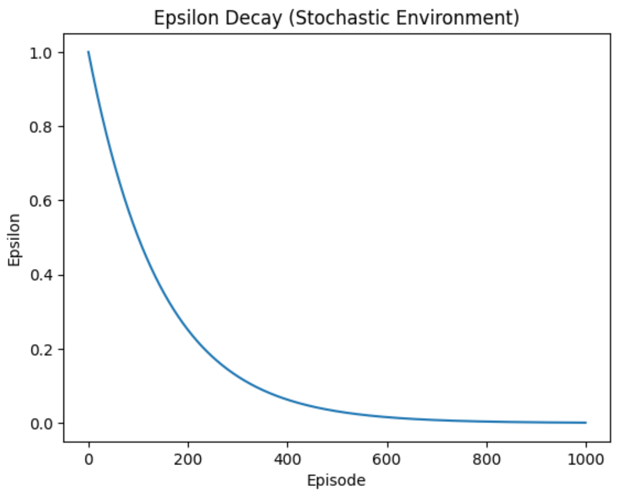
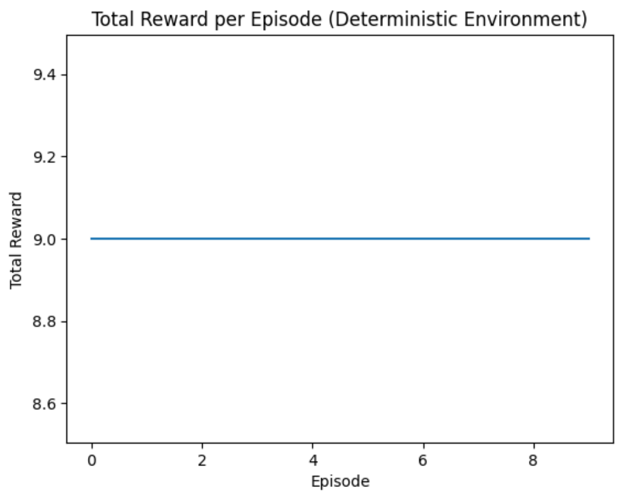
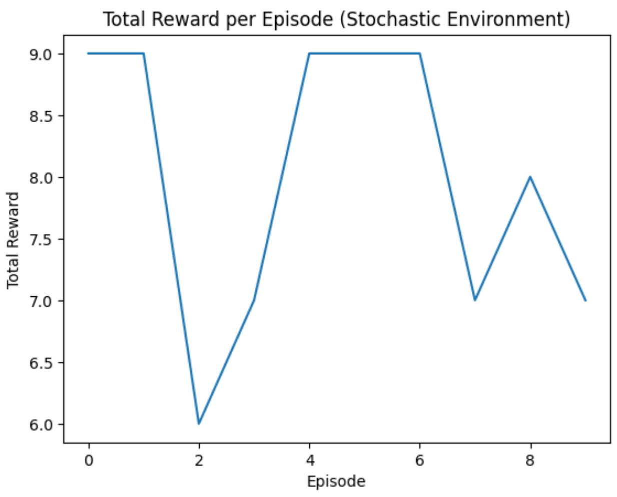


Figure 10: Epsilon Decay

- **Provide the evaluation results. Run your environment for at least 10 episodes, where the agent chooses only greedy actions from the learnt policy. Plot should include the total reward per episode.**

  The evaluation of both Q-learning and SARSA algorithms in deterministic and stochastic environments, when run for at least 10 episodes using greedy actions from the learned policy, is depicted in the plots below. In the deterministic environment, the total rewards per episode are consistently high and stable, indicating that the agent has effectively learned an optimal or near-optimal policy. On the other hand, in the stochastic environment, the total rewards per episode exhibit more variability, reflecting the inherent uncertainty and complexity of the environment. However, the rewards are still relatively high, which suggests that the agent has learned a robust policy that performs well on average despite the stochastic nature of the environment.





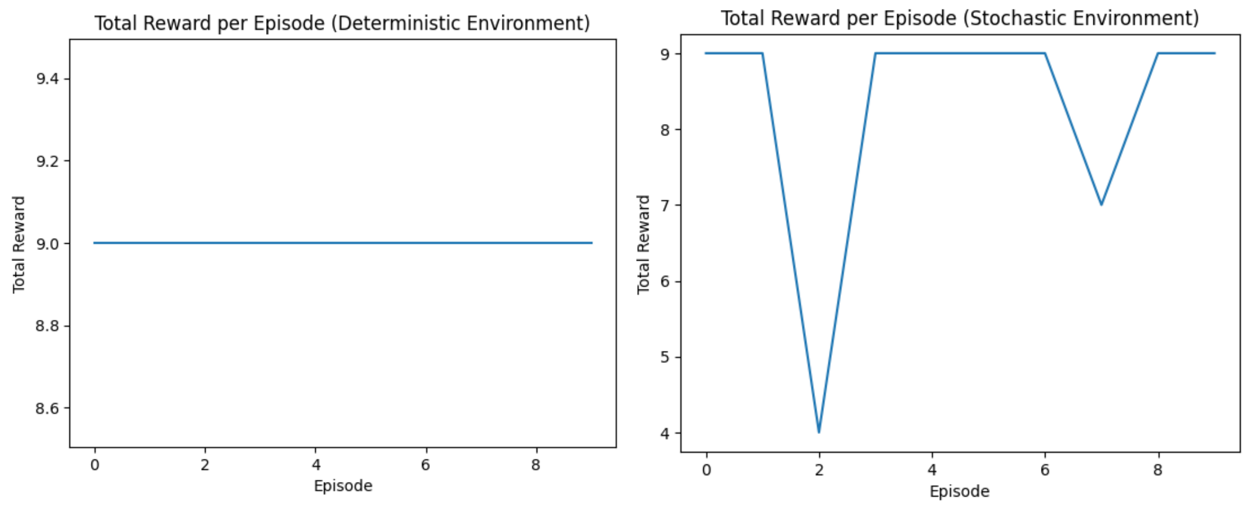


Figure 11: Evaluations of Q-Learning and SARSA

2. Compare the performance of both algorithms on the same deterministic environment (e.g., show one graph with two reward dynamics) and give your interpretation of the results.

The performance comparison of Q-learning and SARSA in the deterministic environment, as shown in the graph, indicates that both algorithms have similar reward dynamics. However, there are episodes where the rewards obtained by one algorithm exceed those of the other. This could be due to the difference in learning strategies; Q-learning is off-policy, learning the optimal policy irrespective of the agent's actions, whereas SARSA is on-policy, learning the policy based on the actions actually taken. In a deterministic environment, where outcomes are predictable, Q-learning might slightly edge out SARSA due to its nature of finding the optimal policy.

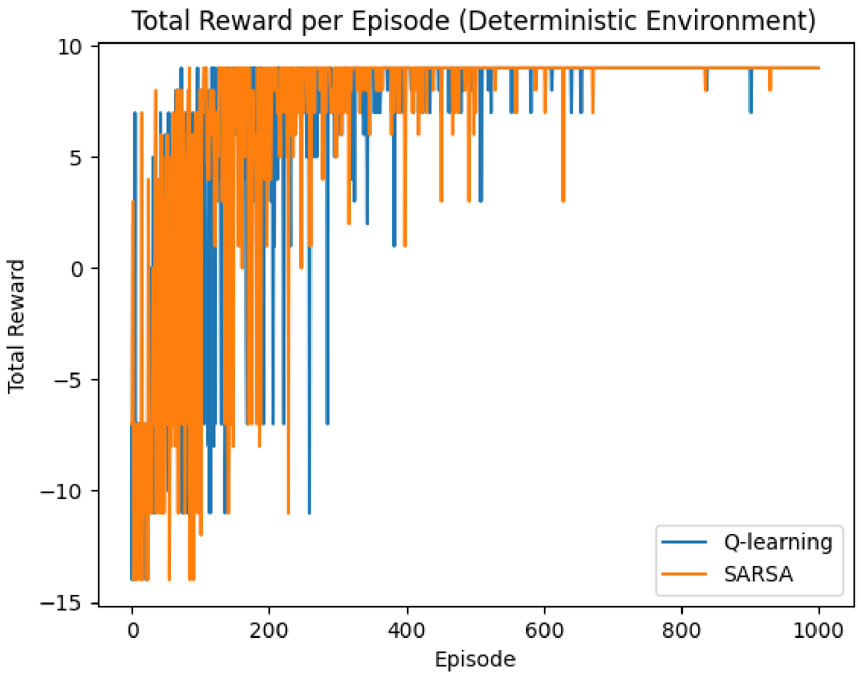


Figure 12: Q-Learning and SARSA on Deterministic Environment

3. Compare how both algorithms perform in the same stochastic environment (e.g., show one graph with two reward dynamics) and give your interpretation of the results.

In the stochastic environment, the comparison graph reveals that both Q-learning and SARSA experience significant variability in the total reward per episode. This reflects the inherent uncertainties and variations in the environment's dynamics. It is also evident that neither algorithm consistently outperforms the other, which could imply that the stochastic environment challenges both learning approaches in a way that makes their performance more dependent on the specific episode dynamics.

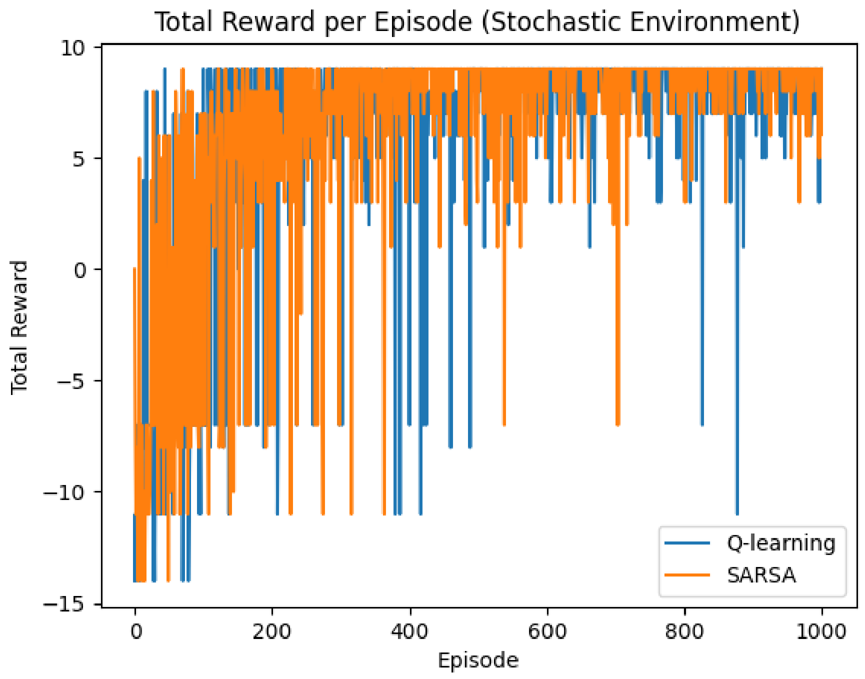


Figure 13: Q-Learning and SARSA on Stochastic Environment

4. Briefly explain the tabular methods, including Q-learning, that were used to solve the problems. Provide their update functions and key features.

Tabular methods in reinforcement learning involve maintaining a table of values (such as Q-values) which are updated iteratively to learn the optimal policy. Q-learning and SARSA are two such tabular methods.

Q-learning is an off-policy method, which means it learns the value of the optimal policy independently of the agent's actions. Its update function is given by:

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

where α is the learning rate, γ is the discount factor, r_{t+1} is the reward received after taking action a_t in state s_t, and $\max_a Q(s_{t+1}, a)$ is the maximum Q-value for the next state s_{t+1}.

The key feature of Q-learning is that it updates its Q-values using the maximum expected future rewards, regardless of the policy being followed.

SARSA is an on-policy method, which updates its Q-values using the action actually taken by the policy. Its update function is:

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

where a_{t+1} is the actual next action taken, as opposed to the optimal action in Q-learning.

The key feature of SARSA is that it updates its Q-values based on the action chosen by the current policy, which may not be the greedy action.

Both methods converge to the optimal policy as long as all state-action pairs are visited an infinite number of times and the learning rate meets certain conditions.

5. Briefly explain the criteria for a good reward function. If you tried multiple reward functions, give your interpretation of the results.

A good reward function should clearly align with the objectives of the task, providing positive reinforcement for desirable actions and negative reinforcement for undesirable ones. It should encourage exploration without leading to suboptimal policy adherence, and maintain a balance that neither overpowers nor underrepresents the significance of actions.

In the implementation, the reward function provides a negative reward for each step, which incentivizes the agent to find the shortest path to the goal. A reward of zero was also tried at each step, and was ineffective, indicating that it did not provide enough incentive for the agent to reach the goal. Additional rewards or penalties are added to the $rewards\_map$ to influence the agent's path, such as negative rewards for dangerous paths or positive rewards for desirable intermediate states.

Part 3: Stock Trading Environment

1. Show and discuss the results after applying the Q-learning algorithm to solve the stock trading problem. Plots should include epsilon decay and total reward per episode.

The application of Q-learning to the stock trading problem resulted in a discernible pattern of epsilon decay and volatile reward accumulation over episodes. The epsilon decay plot demonstrates a standard exponential decay, indicating the agent's shift from exploration to exploitation as the learning progresses.

The total rewards per episode plot, however, shows high variability, with peaks suggesting instances of high profitability mixed with periods of lower performance. This could be due to the complex and unpredictable nature of the stock trading environment, where optimal strategies can vary greatly with market conditions and may not be consistently repeatable.

The agent eventually stabilizes to a good-enough policy that is best for earning consistent rewards in the high risk stock trading environment, which can be seen after episode 900 with low exploration.

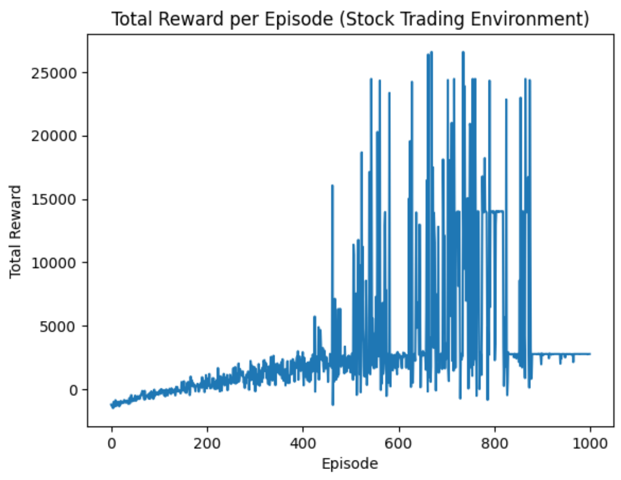


Figure 14: Total Reward per Episode

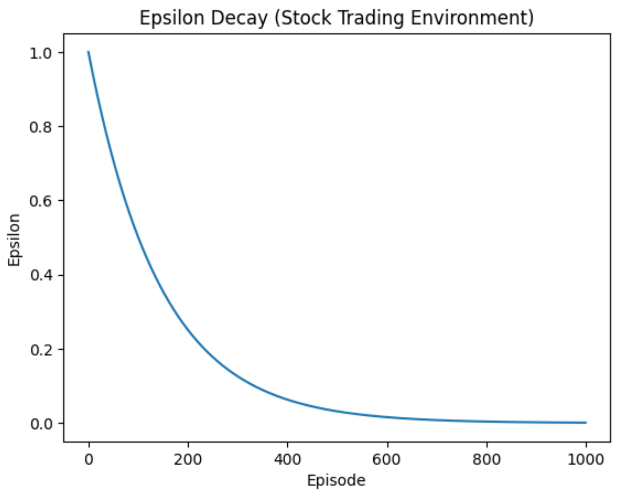


Figure 15: Epsilon Decay

2. Provide the evaluation results. Evaluate your trained agent's performance (you will have to set the train parameter set to False), by only choosing greedy actions from the learnt policy. Plot should include the agent's account value over time.

The evaluation of the trained agent's performance by choosing greedy actions from the learned policy yielded a profit of 52030 as depicted in the plot.

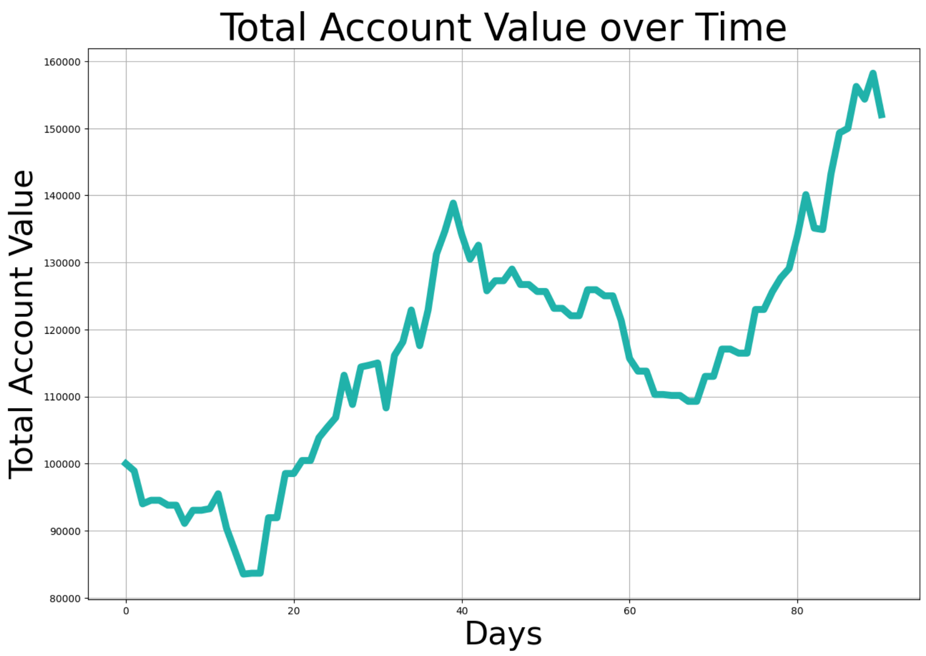
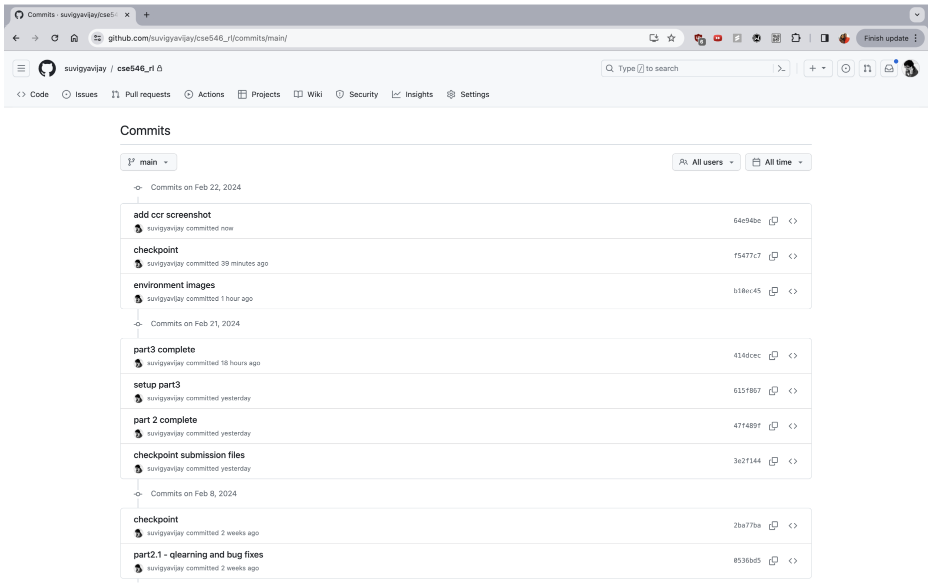


Figure 16: Agent's Account Value Over Time

Extra Points

1. Git Expert

Link to Private Repo: https://github.com/suvigyavijay/cse546_rl



Commits

Commits on Feb 22, 2024

add ccr screenshot
suvigyavijay committed now — 64e94be

checkpoint
suvigyavijay committed 39 minutes ago — f5477c7

environment images
suvigyavijay committed 1 hour ago — b10ec45

Commits on Feb 21, 2024

part3 complete
suvigyavijay committed 18 hours ago — 414dcec

setup part3
suvigyavijay committed yesterday — 615f867

part 2 complete
suvigyavijay committed yesterday — 47f489f

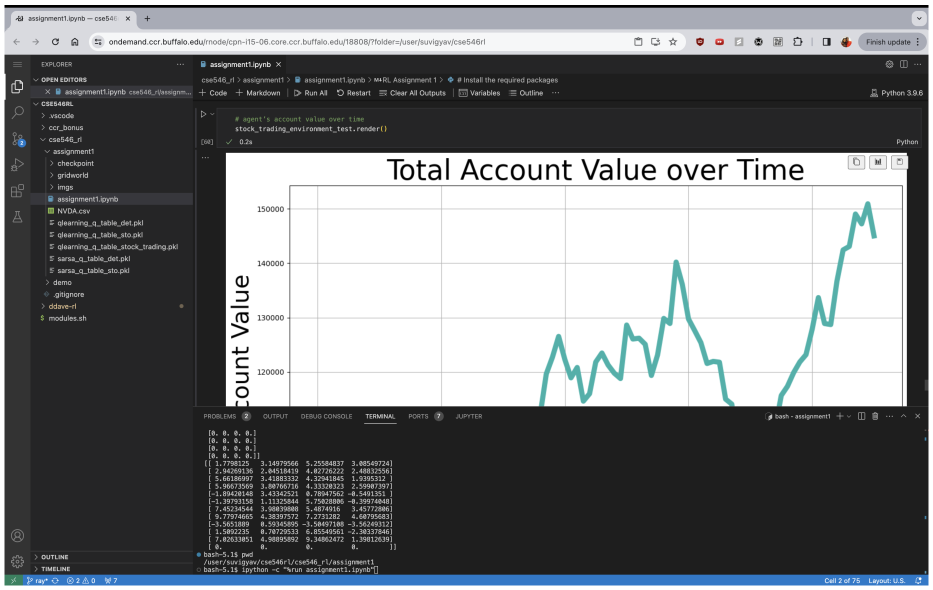
checkpoint submission files
suvigyavijay committed yesterday — 3e2f144

Commits on Feb 8, 2024

checkpoint
suvigyavijay committed 2 weeks ago — 2ba77ba

part2.1 - qlearning and bug fixes
suvigyavijay committed 2 weeks ago — 0536bd5

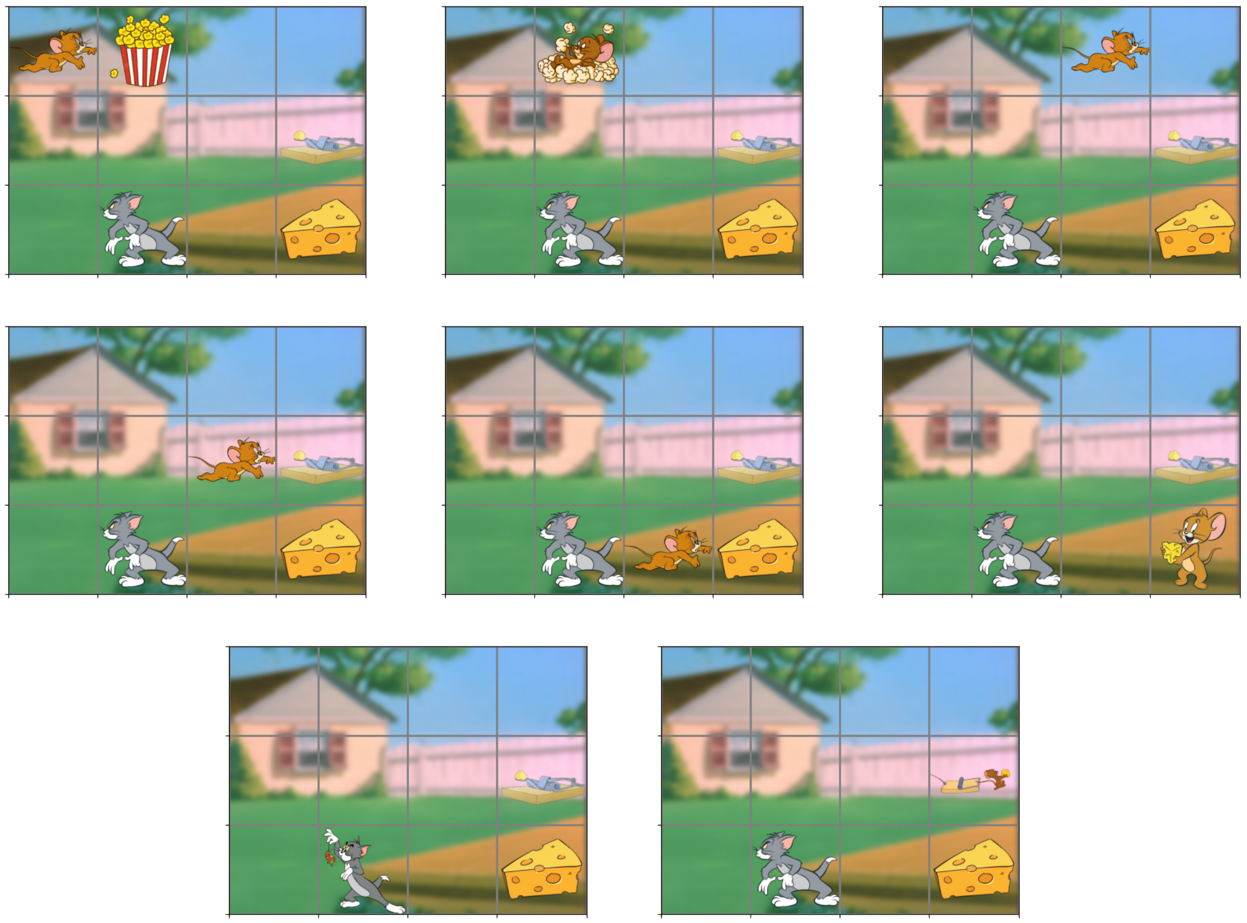
2. CCR Submission



```
# agent's account value over time
stock_trading_environment_test.render()
```

```
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]
[[ 1.7798125   3.14979566  5.25584837  3.08549724]
 [ 2.94269136  2.04518419  4.02726222  2.48832556]
 [ 5.66186997  3.41883332  4.32941845  1.9395312 ]
 [ 5.96673569  3.80766716  4.33320323  2.59907397]
 [-1.89420148  3.43342521  0.78947562 -0.5491351 ]
 [-1.39793158  1.11325844  5.75028806 -0.39974048]
 [ 7.45234544  3.98039808  5.4874916   3.45772806]
 [ 9.77974665  4.38397572  7.2731282   4.60795683]
 [-3.5651889   0.59345895 -3.50497108 -3.56249312]
 [ 1.5092235   0.70729533  6.85549561 -2.30337846]
 [ 7.02633051  4.98895892  9.34862472  1.39812639]
 [ 0.          0.          0.          0.        ]]
bash-5.1$ pwd
/user/suvigyav/cse546rl/cse546_rl/assignment1
bash-5.1$ ipython -c "%run assignment1.ipynb"
```

3. Grid-World Scenario Visualization



References:

[1] A1 Video Overview

[2] RL Env Creation and Random Agent

[3] RL Environment visualization

[4] Gymnasium documentation

[5] Q-Learning Tutorial

[6] SARSA Tutorial