Report

Farid Gahramanov

Farid Gahramanov

Introduction

In this project I will investigate how to use reinforcement learning (RL) algorithms for navigation in GridWorld, a virtual environment. The main aim of this project is to show how these algorithms, through their strategies and parameter's, can efficiently identify the best or almost best routes to achieve specific goal nodes in the GridWorld. I evaluated and analyzed these techniques over numerous GridWorld scenarios to increase the analysis's accuracy. In order provide an in-depth analysis, the performance of these algorithms is analyzed using many measures, such as computational time, total score achieved, and convergence speed.

Implementation Details

Environment :

Each cell in the GridWorld environment can represent a particular sort of terrain, such as flat ground, mountains, goals, or threats. GridWorld is a simulated location. The agent may only travel in the following four directions in this grid-formatted environment: up, down, left, and right. The agent's objective is to move from a starting point to a destination location while maximizing accrued rewards and avoiding obstacles. Each cell in the grid has a distinct state.

Algorithm Details :

I will start with Q-learning, it is an off-policy learner, meaning that it gains knowledge about the best course of action apart from the agent's actions. It makes use of the Bellman equation to update its Q-values. It works like this : $Q(s, a) = Q(s, a) + \alpha * (r + \gamma * \max(Q(s', a')) - Q(s, a))$

"s" – is the current state; "a" – is the action taken; "r" – is the reward received; "α"- is the learning rate;

Now about the SARSA, it is an on-policy learner that uses the action taken by the policy generated by Q to update its Q-values. SARSA uses the actual action from the next state, as compared with Q-Learning, which uses the maximum reward of the next state. Both algorithms initialize their Q-values to zero, which represents a state of no prior knowledge. The parameters are:

- "α" – Learning rate, it determines the weight of new information vs. old one.
- "γ" - Discount factor, it indicates the importance of future rewards vs immidiate ones.

The whole parameter listing:

Adjusting these hyper-parameters can have a big impact on how well Q-Learning and SARSA work:

- Learning Rate "α": regulates how quickly the algorithm adds new data to the Q-values, which affects how much is used to explore new territory and how much is previously known.
- Discount Factor "γ": establishes the value of potential rewards in relation to current ones, a lower number gives priority to short-term profits; whereas a higher value emphasizes long-term gains more.
- Epsilon "ϵ": determines the likelihood of selecting a random action over the most popular activity, which controls the exploration approach. More exploration is encouraged by higher epsilon values, which is important in the early phases of learning.
- Epsilon Decay (ϵ decay): controls the rate of epsilon's over time decline. As the agent gains more knowledge about the surroundings, this parameter is essential for progressively changing the approach from exploration to exploitation.

- Minimum Epsilon (ϵ min): establishes the epsilon floor, which keeps the algorithm from being overly greedy and guarantees that there is always some degree of exploration, both of which could prevent the agent from finding new routes or tactics.

For example, a higher learning rate can lead to an unstable learning process, but it also helps the agent swiftly adjust to changes in dynamic situations. On the other hand a lower learning rate guarantees more consistent updates, it may delay the learning process and make it more difficult for the agent to respond appropriately to environmental changes. Another case is when a high discount factor; it might be helpful in difficult settings where long-term gains are important since it makes the agent more forward-thinking and gives future rewards more weight. However, a lower discount factor may be advantageous in simple or highly dynamic contexts where shortterm gains are more indicative of long-term success. The last case is when high initial epsilon values promote exploration, which may improve our awareness of the environment overall but delay down convergence.

Performance Evaluation and Algorithm Comparison

I created three tests, each with a different set of difficulties to evaluate the flexibility and effectiveness of the Q-Learning and SARSA algorithms, to examine how well they performed in various GridWorld settings. This comparison study analyzes the algorithms' general behavior, total scores, and computation time.

Test Environments:

- Test 1: The first GridWorld, sized 10x10, featured 2 goals and 3 pitfalls, providing a relatively straightforward challenge to navigate.
- Test 2: Increasing complexity, the second GridWorld expanded to a 15x15 grid, including 4 goals and 5 pitfalls, creating a denser and more challenging environment.
- Test 3: The third test was the most complex, with a 20x20 grid, 3 goals, and 6 pitfalls, requiring sophisticated navigation strategies due to its larger size and increased number of obstacles.

Time Analysis:

- Q-Learning: Because of its comprehensive exploratory approach, Q-Learning often showed longer processing times. Significant generating load was seen in Test 2, where processing took more than 1282 milliseconds.
- SARSA: In all tests, SARSA demonstrated faster processing times; in Test 3, for example, it finished its run in a mere 125 milliseconds. This effectiveness highlights to us that it is more straightforward approach to problem-solving.

Total Score:

In Test 1, Q-Learning fared better than SARSA, achieving an 87 vs a -106 score, demonstrating its capacity to optimize well in less complex configurations. Tests 2 and 3 demonstrated the difficulties both algorithms encountered in increasingly complicated situations. Test 2 showed SARSA struggle significantly, highlighting weaknesses in its cautious method under highdensity challenge settings, whereas Q-Learning was able to retain a positive score of 85.

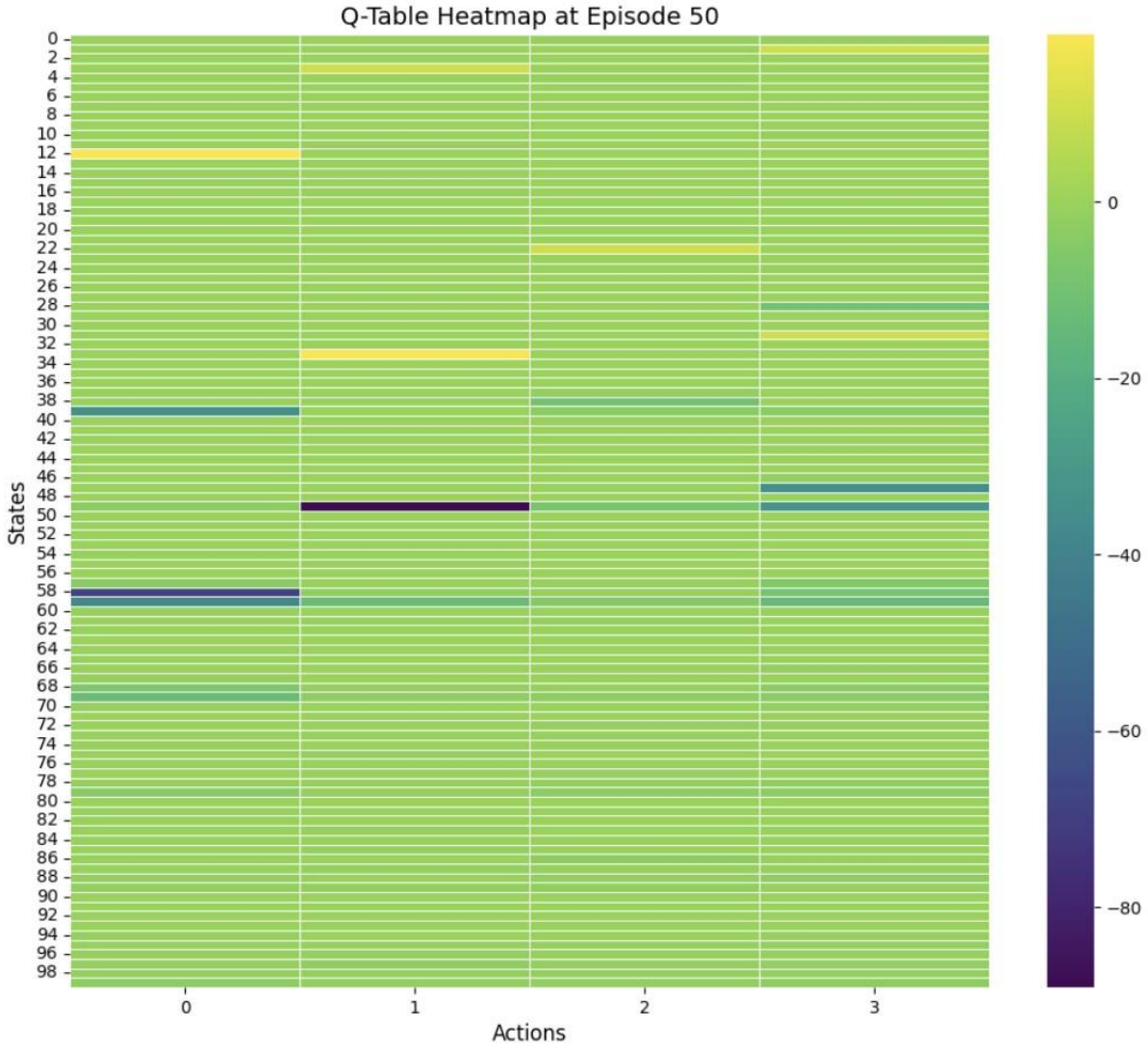
In these experiments I understood possible drawbacks of each algorithm in various scenarios.

Because it may take more risks and return greater rewards in less complicated contexts, QLearning's exploration approach typically performs better there. However, although successful, SARSA's conservative approach may not be as effective in situations when a more risky tactic may perform better.

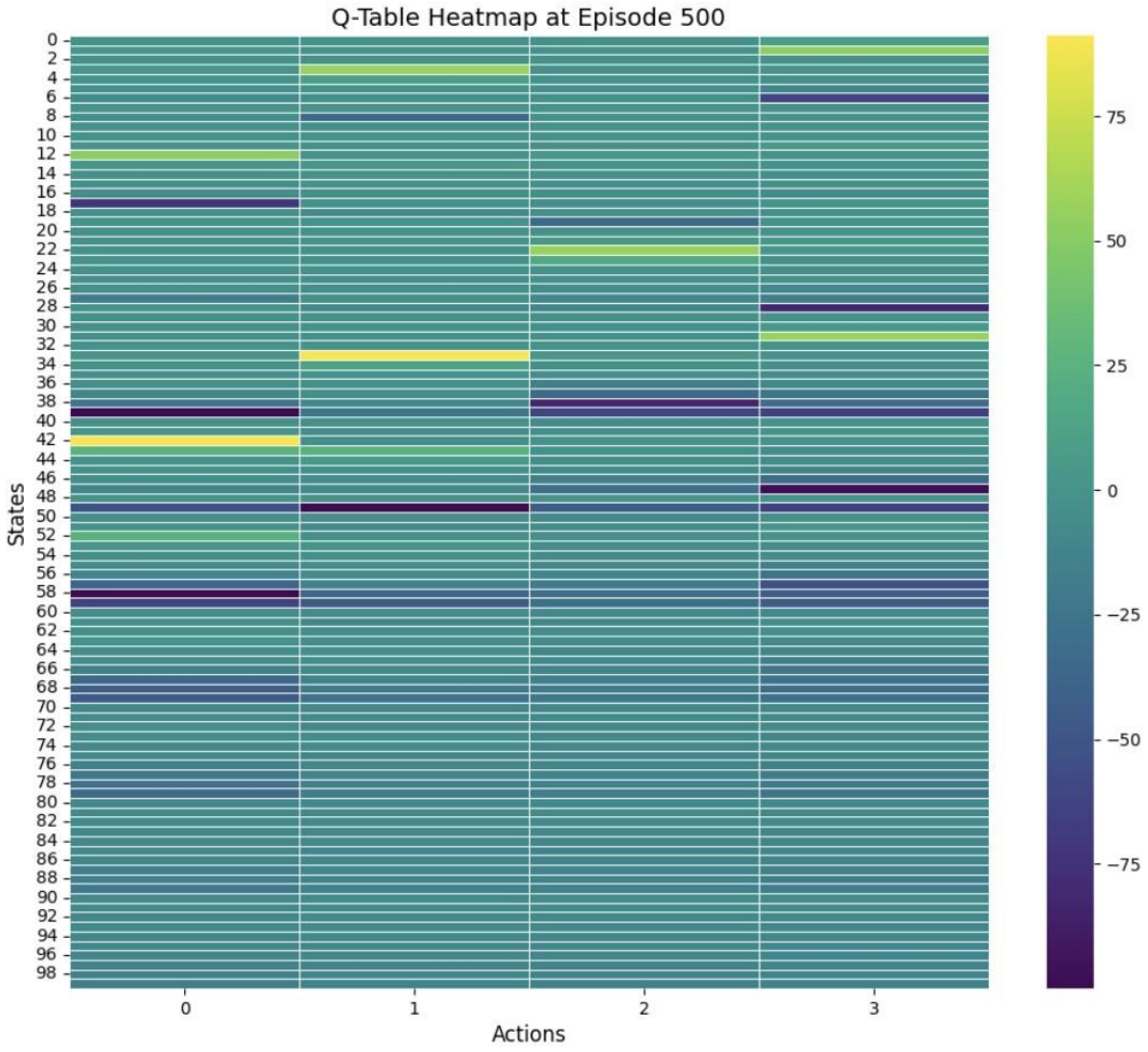
Analyzes and visualization of trained Q-Tables

I used heatmaps to visually illustrate the growth of the Q-Tables over several training episodes. Each heatmap corresponds to a Q-Table from a single episode and shows how the Q-values change. This is done by storing the Q-Table at regular times (for example, every 50 episodes) throughout the training period. These stored Q-Tables are then loaded and heatmaps created with tools like as numpy, matplotlib, and seaborn. In the early phases, the Q-Table values are often closer to zero, suggesting that the agent is still studying the environment and has yet to develop useful methods. As training develops, specific patterns appear in the heatmap. By episode 50, we can observe several places where the Q-values are much higher or lower, indicating that the agent is beginning to learn whether behaviors in specific conditions are more or less promising. By episode 500, the Q-Table values show more substantial distinction, indicating the agent's improved method of picking behaviors that result in bigger rewards. The heatmap's color variety suggests that the agent has successfully discriminated between good and harmful activities.

Farid Gahramanov



---- This is the heatmap for episode 50 and that was gotten from GridTestOne.pkl result.



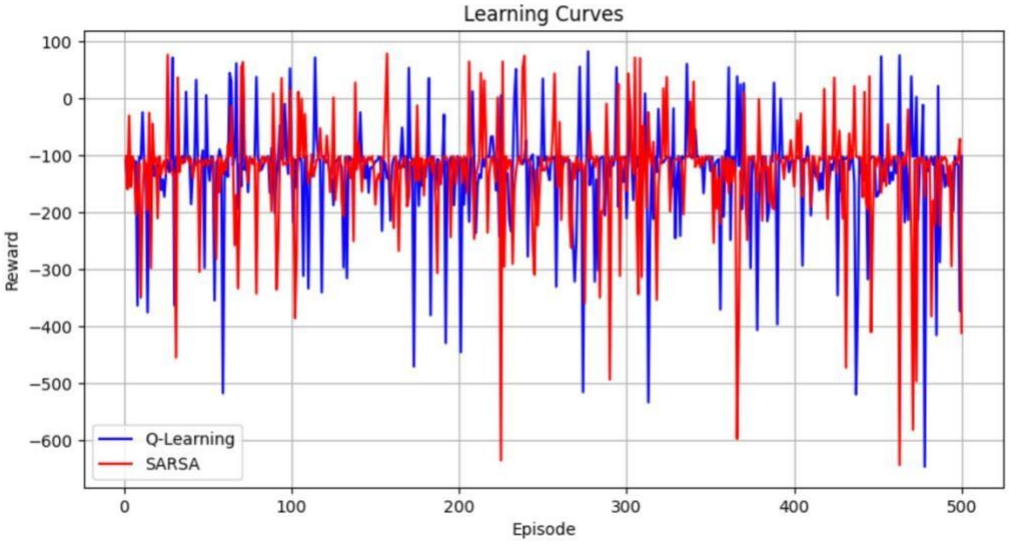
--- This is the heatmap for episode 500 and that was gotten from GridTestOne.pkl result.

These heatmaps help to demonstrate that the Q-Learning algorithm optimizes its policy over time. By viewing these representations, we can see that the learning progression and evolving strategy contained in the Q-Tables. This graphic demonstrates the success of the learning process.

Farid Gahramanov

Graphs & Demonstration

In this part of the report, I demonstrate the learning curves for two reinforcement learning algorithms: Q-Learning and SARSA. Learning curves are essential tools for illustrating how an agent learns over time. These curves indicate the rewards that were collected each episode over 500 episodes of training for each algorithm. To generate these curves, I constructed each algorithm's agent with particular characteristics such as the environment and discount rates, learning rates, and epsilon values. The environment serves as a standard for evaluating the agent's behaviors and distributing rewards accordingly. Each episode is a complete journey through the environment that ends when the agent achieves a goal, fails, or takes the maximum number of steps allowed. The rewards for each episode were recorded in two separate numpy arrays for Q-Learning and SARSA. These arrays were imported and plotted using Matplotlib. The plot's x-axis displays the episode number, while the y-axis shows the reward that was collected in that episode. The blue line shows the Q-Learning algorithm, whereas the red line represents the SARSA algorithm.



Farid Gahramanov

Conclusion

As we learned from this project, the learning curves demonstrate that both Q-Learning and SARSA algorithms learn from their interactions with the environment, although they behave differently. Q-Learning tends to explore more aggressively since it learns from the potential optimal action (off-policy), which may result in bigger rewards but also a greater danger of significant penalties. SARSA, on the other hand, learns acts based on current policy (on-policy), making it more cautious and frequently resulting in smoother learning curves with fewer high cuts in rewards. Different GridWorld parameters can have a significant impact on the algorithms' performance. Scenarios with more obstacles or penalties might require more cautious solutions, perhaps making SARSA more successful. In contrast, environments with large rewards and fewer risks may profit from Q-Learning's brave strategy of obtaining bigger returns at increasing risk.