Homework 04

Graded

21 Hours, 6 Minutes Late

Student

Kartikay Gupta

Total Points

65 / 65 pts

Question 1

Question 1 **14** / 14 pts

1.1 **Question 1a** **7** / 7 pts

✔ **– 0 pts** Correct

– 7 pts Missing

– 4 pts Not close to the expected value function

– 2 pts Missing max-norm value

1.2 **Question 1b** **7** / 7 pts

✔ **– 0 pts** Correct

– 7 pts Missing

– 3 pts Very high average number of steps, and value function far from optimal

– 3 pts Very low average number of steps, and value function far from optimal

– 2 pts Missing standard deviation

Question 2

Question 2 **18** / 18 pts

2.1 **Question 2a** **7** / 7 pts

✔ **– 0 pts** Correct

– 7 pts Missing

– 1 pt No discussion about the hyperparameter choices

– 3 pts The learning curve does not indicate learning, likely because of the hyperparameters selected

– 5 pts The learning curve does not indicate learning, even with reasonable hyperparameters selected

2.2 **Question 2b** **8** / 8 pts

✔ **– 0 pts** Correct

– 8 pts Missing

– 3 pts The learning curve does not indicate learning, likely because of the hyperparameters selected

– 5 pts The learning curve does not indicate learning, even with reasonable hyperparameters selected

2.3 **Question 2c** **3** / 3 pts

✔ **– 0 pts** Correct

– 3 pts Missing

– 3 pts Policy is not close to the expected.

– 1.5 pts Policy not close to expected in only some states

– 0.5 pts Policy for the water state is missing or incorrect

– 1 pt Policy for one or more states near the goal state are incorrect

Question 3

Question 3 **18** / 18 pts

3.1 **Question 3a** **7** / 7 pts

✔ **– 0 pts** Correct

– 1 pt No discussion about the hyperparameter choices

– 3 pts The learning curve does not indicate learning, likely because of the hyperparameters selected

– 5 pts The learning curve does not indicate learning, even with reasonable hyperparameters selected

– 7 pts Missing

3.2 **Question 3b** **8** / 8 pts

✔ **– 0 pts** Correct

– 8 pts Missing

– 3 pts The learning curve does not indicate learning, likely because of the hyperparameters selected

– 6 pts The learning curve does not indicate learning, even with reasonable hyperparameters selected

3.3 **Question 3c** **3** / 3 pts

✔ **– 0 pts** Correct

– 3 pts Missing

– 3 pts Policy is not close to the expected.

– 1.5 pts Policy not close to expected in only some states

– 0.5 pts Policy for the water state is missing or incorrect

– 1 pt Policy for one or more states near the goal state are incorrect

Question 4

Question 4 **15** / 15 pts

4.1 **Question 4a** **6** / 6 pts

✔ **– 0 pts** Correct

– 2 pts First learning curve missing

– 2 pts First learning curve does not show fast convergence to ~80% of J*

– 2 pts Second learning curve missing

– 2 pts Second learning curve does not show slow convergence to ~80% of J*

– 2 pts J* not shown or mentioned

– 1 pt Only one J* given for both starting states

– 1 pt Used arbitrary n rather than $n =$ number of value iteration steps

– 3 pts No MDP details (no screenshots, description, variable settings, etc.)

– 6 pts Missing

4.2 **Question 4b** **6** / 6 pts

✔ **– 0 pts** Correct

– 6 pts Missing

– 2 pts First learning curve missing

– 2 pts First (or second) learning curve does not show fast convergence to ~80% of J*

– 2 pts Second learning curve missing

– 2 pts Second (or first) learning curve does not show slow convergence to ~80% of J*

– 2 pts J* not shown or mentioned

– 1 pt Used arbitrary n rather than $n =$ number of value iteration steps

– 3 pts No MDP details (no screenshots, description, variable settings, not using the same MDP as before, etc.)

4.3 **Question 4c** **3** / 3 pts

✔ **– 0 pts** Completed

– 3 pts Not completed

No questions assigned to the following page.

CMPSCI 687 - Extra Credit Assignment (Optional) - Fall 2023

Due **December 05, 2023**, 11:55pm Eastern Time

1 Instructions

This assignment consists only of a programming portion. While you may discuss problems with your peers (e.g., to discuss high-level approaches), you must answer the questions on your own. In your submission, do explicitly list all students with whom you discussed this assignment. Submissions must be typed (handwritten and scanned submissions will not be accepted). You must use LaTeX. The assignment should be submitted on Gradescope as a PDF with marked answers via the Gradescope interface. The source code should be submitted via the Gradescope programming assignment as a .zip file. Include with your source code instructions for how to run your code. You **must** use Python 3 for your homework code. You may not use any reinforcement learning or machine learning-specific libraries in your code, e.g., TensorFlow, PyTorch, or scikit-learn. You *may* use libraries like numpy and matplotlib, though. The automated system will not accept assignments after 11:55pm on December 05. The tex file for this homework can be found here.

Before starting this homework, please review this course's policies on plagiarism by reading Section 10 of the syllabus.

Programming (65 Points Total)

In this assignment, you will implement three algorithms based on Temporal Difference (TD): TD-Learning for policy evaluation; SARSA; and Q-Learning. You will deploy all three algorithms on the 687-Gridworld. **Notice that you may not use existing RL code for this problem—you must implement the learning algorithms and the environments entirely on your own and from scratch.** The complete definition of the 687-GridWorld domain can be found in Homework 3.

General instructions:

- You are free to initialize the q-function in any way that you want. More details about this later.
- Whenever displaying values (e.g., the value of a state), use 4 decimal places; e.g, $v(s) = 9.4312$.
- Whenever showing the value function and policy learned by your algorithm, present them in a format resembling that depicted in Figure 1.

Value Function

4.0187	4.5548	5.1575	5.8336	6.4553
4.3716	5.0324	5.8013	6.6473	7.3907
3.8672	4.3900	0.0000	7.5769	8.4637
3.4182	3.8319	0.0000	8.5738	9.6946
2.9977	2.9309	6.0733	9.6946	0.0000

Policy

→	→	→	↓	↓
→	→	→	↓	↓
↑	↑		↓	↓
↑	↑		↓	↓
↑	↑	→	→	G

Figure 1: Optimal value function and optimal policy for the 687-GridWorld domain.

Question assigned to the following page: 1.1

1. TD-Learning on the 687-Gridworld [14 Points, total]

First, implement the TD-Learning algorithm for policy evaluation and use it to construct an estimate, $\hat{v}$, of the value function of the optimal policy, π^*, for the 687-Gridworld. **Remember that both the optimal value function and optimal policy for the 687-Gridworld domain were identified in a previous homework by using the Value Iteration algorithm. For reference, both of them are shown in Figure 1.** When sampling trajectories, set d_0 to be a uniform distribution over $\mathcal{S}$ to ensure that you are collecting returns from all states. Do not let the agent be initialized in an Obstacle. Notice that using this alternative definition of d_0 (instead of the original one) is important because we want to generate trajectories from all states of the MDP, not just states that are visited under the optimal deterministic policy shown in Figure 1.

You should run your algorithm until the value function estimate does not change significantly between successive iterations; i.e., whenever the Max-Norm between $\hat{v}_{t-1}$ and $\hat{v}_t$ is at most δ, for some value of δ that you should set empirically. For each run of your algorithm, count how many episodes were necessary for it to converge to an estimate of v^{π^*}. **You should run your algorithm (as described above) 50 times and report the value of α that was used. You should pick a value of α that causes the algorithm to converge to a solution that is close to v^{π^*}, while not requiring an excessively large number of episodes.** Then:

(Question 1a. 7 Points) Report the average value function estimated by the algorithm—i.e., the average of all 50 value functions learned by the algorithm at the end of its execution. Report, also, the Max-Norm between this average value function and the optimal value function for the 687-Gridworld domain.

Answer 1a

For $\alpha = 0.1$ and $\delta = 0.001$, which managed to reduce the maximum norm between the averaged value function (across 50 iterations) and the optimal value function to 0.1222 in approximately $20,000$ episodes on average.

Elevated values of α such as 0.2 and 0.3 converged in fewer episodes (several thousand), yet the maximum norm was notably higher, approximately 0.6 to 0.7.

The anticipated averaged value function over 50 iterations is needed.

```
avg value function:
3.9105   4.4655   5.0778   5.7845   6.2908

4.2642   4.9682   5.7608   6.6529   7.2470

3.7807   4.2792   0.0000   7.6173   8.4182

3.3364   3.7375   0.0000   8.6694   9.6889

2.9199   2.9188   6.0440   9.7616   0.0000
```

Figure 2: Max norm between Average value function and optimal value function

Questions assigned to the following page: 2.1 and 1.2

(Question 1b. 7 Points) Report the average and the standard deviation of the number of episodes needed for the algorithm to converge.

Answer 1b.
avg no of episodes = 26087.62
std for no of episodes = 24455.886625424155

```
max norm from true values = 0.16446348051164517
avg no of episodes = 26087.62
std for no of episodes = 24455.886625424155
```

Figure 3: Average and std deviation of this algorithm

2. SARSA on the 687-Gridworld [18 Points, total]

Implement the SARSA algorithm and use it to construct $\hat{q}$, an estimate of the optimal **action-value function**, q^{π^*}, of the 687-Gridworld domain; and to construct an estimate, $\hat{\pi}$, of its optimal policy, π^*. You will have to make four main design decisions: *(i)* which value of α to use; *(ii)* how to initialize the q-function; you may, e.g., initialize it with zeros, with random values, or optimistically. Remember that $q(s_\infty, \cdot) = 0$ by definition; *(iii)* how to explore; you may use different exploration strategies, such as ϵ-greedy exploration or softmax action selection; and *(iv)* how to control the exploration rate over time; you might choose to keep the exploration parameter (e.g., ϵ) fixed over time, or have it decay as a function of the number of episodes. **Report the design decisions *(i, ii, iii, and iv)* you made and discuss what was the reasoning behind your choices.** Then:

(Question 2a. 7 Points) Construct a learning curve where you show, on the x axis, the total number of actions taken by the agent, from the beginning of the training process (i.e., the total number of steps taken across all episodes up to that moment in time). On the y axis, you should show the number of episodes completed up to that point. If learning is successful, this graph should have an increasing slope, indicating that as the agent takes more and more actions (and thus executes more learning updates), it requires fewer actions/timesteps to complete each episode. Your graph should, ideally, look like the figure shown in Example 6.5 of the RL book (2nd edition). To construct this graph, run your algorithm 20 times and show the average curve across those runs.

Answer 2a.

(a) $\alpha = 0.1$: I observed that setting a very high learning rate causes drastic updates in learning, yielding poor results. Lower values of α seem to work well, providing a smoother decrease in the Mean Squared Error (MSE) versus Episodes curve.

(b) Initialized the q-function with zeros, and it yielded good results.

(c) Used ε-greedy exploration: Despite trying softmax action selection, which did not show significant improvement, I opted for ε-decay by setting the initial $\varepsilon = 0.8$ and subtracting 0.05 after each episode. This choice is based on the idea that high exploration initially, gradually transitioning to more greedy strategies, tends to work well for convergence.

Questions assigned to the following page: 2.1 and 2.2

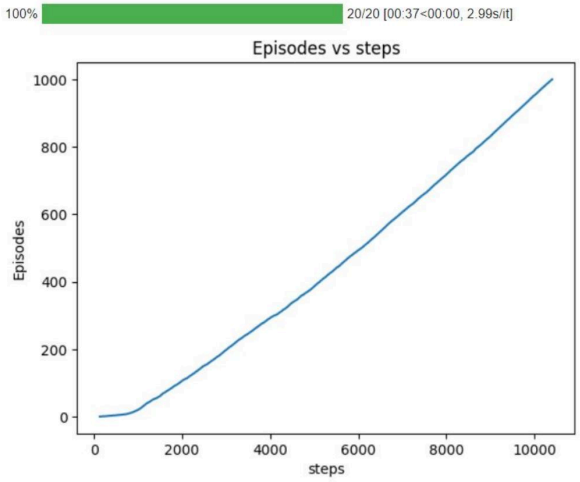

Figure 4: the total number of steps and the number of episodes completed up to that point

(Question 2b. 8 Points) Construct another learning curve: one where you show the number of episodes on the x axis, and, on the y axis, the mean squared error between your current estimate of the value function, $\hat{v}$, and the true optimal value function, v^{π^*}. The mean squared error between two value functions, v_1 and v_2, can be computed as $\frac{1}{|\mathcal{S}|}\sum_s (v_1(s) - v_2(s))^2$. Remember that SARSA estimates a q-function ($\hat{q}$), however, not a value function ($\hat{v}$). To compute $\hat{v}$, remember that $\hat{v}(s) = \sum_a \pi(s,a)\hat{q}(s,a)$. Remember, also, that the SARSA policy π, at each given moment, depends on your exploration strategy. Let us say, for example, that you are using ϵ-greedy exploration. Then, in the general case when more than one action may be optimal with respect to $\hat{q}(s,a)$, the ϵ-greedy policy would be as follows:

$$\pi(s,a) = \begin{cases} \frac{1-\epsilon}{|\mathcal{A}^*|} + \frac{\epsilon}{|\mathcal{A}|} & \text{if } a \in \mathcal{A}^* \\ \frac{\epsilon}{|\mathcal{A}|} & \text{otherwise,} \end{cases} \tag{1}$$

where $\mathcal{A}^* = \arg\max_{a \in \mathcal{A}} \hat{q}(s,a)$. To construct the learning curve, run your algorithm 20 times and show the average curve across those runs; i.e., the average mean squared error, computed over 20 runs, as a function of the number of episodes.

Answer 2b

This learning curve I got when, the algorithm was run 20 times. The average Mean Squared Error (MSE) was computed across these 20 runs, plotted as a function of the number of episodes. This approach allowed us to visualize the average performance and convergence of the algorithm over multiple runs, offering insights into its learning behavior and improvement trends across episodes.

Questions assigned to the following page: 3.1, 2.3, and 2.2

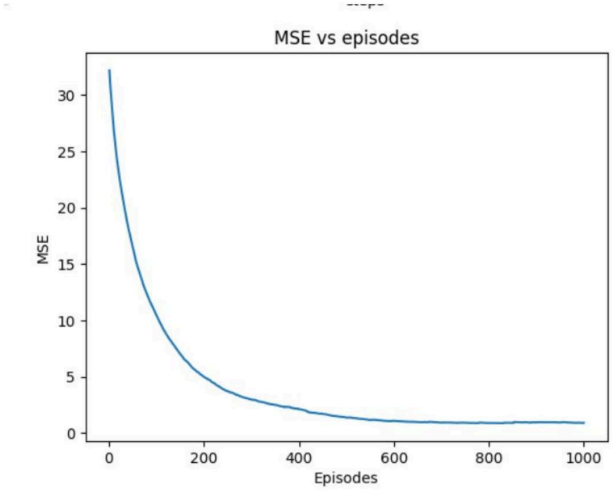

Figure 5: MSE and the number of episodes completed up to that point

(Question 2c. 3 Points) Show the *greedy policy* with respect to the q-values learned by SARSA.

Answer 2c.

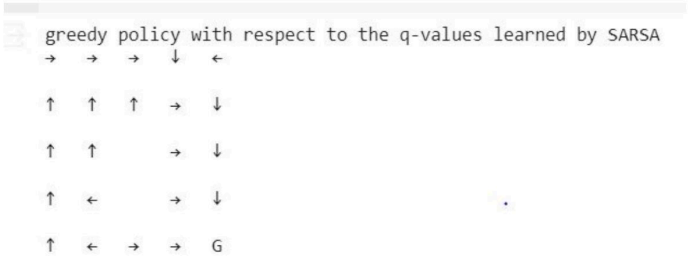

Figure 6: *greedy policy* with respect to the q-values learned by SARSA

3. Q-Learning on the 687-Gridworld [18 Points, total]

Answer the same questions as above (*2a, 2b, and 2c*), but now using the Q-Learning algorithm; i.e., you will be using Q-Learning to solve the 687-Gridworld domain. When computing $\hat{v}$, for question *3c*, you should compute the value function of the greedy policy: $\hat{v}(s) = \max_a \hat{q}(s, a)$. The amount of points associated with questions *3a, 3b, and 3c*, will be the same as those associated with the corresponding items of Question 2.

Answer 3a.

- Used $\alpha = 0.1$: A high learning rate caused drastic updates during learning, leading to unsatisfactory results. Lower values of α showed better performance, displaying a smoother decrease in the Mean Squared Error (MSE) versus episodes curve.
- Initialized the q-function with zeros, resulting in positive outcomes.
- Employed ε-greedy exploration. Although attempting softmax action selection didn't bring significant improvements.

Questions assigned to the following page: 3.1, 3.2, and 3.3

- Implemented ε-decay by starting with an initial ε value of 0.8 and subtracting 0.05 after each episode. This strategy emphasized high exploration initially, gradually transitioning towards more greedy strategies, which appeared effective for convergence.

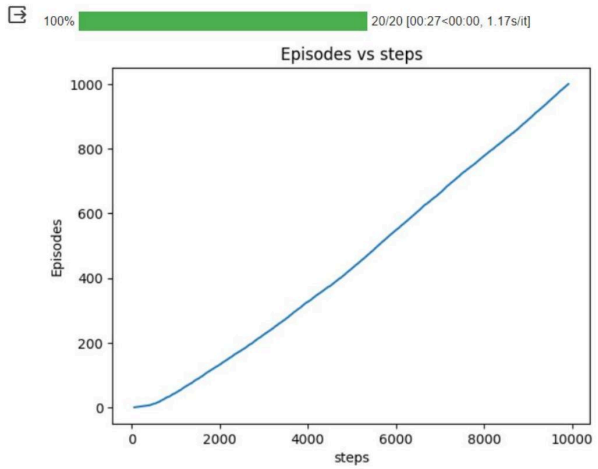

Figure 7: Qlearning of 687 the total number of steps and the number of episodes completed up to that point

Answer 3b.
The learning curve is as follows:

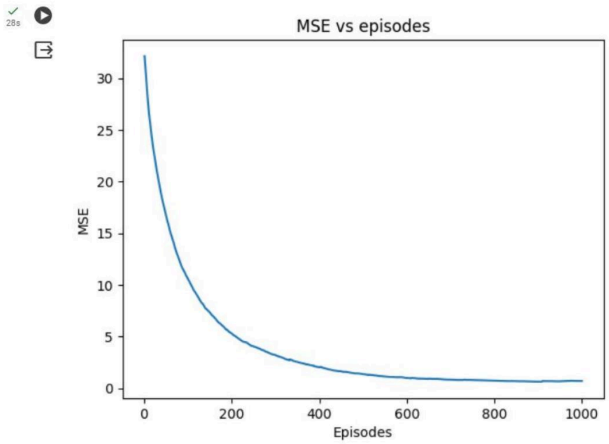

Figure 8: Qlearning on 687 grid world MSE and the number of episodes completed up to that point

Answer 3c.
The learning curve is as follows:

Question assigned to the following page: 3.3

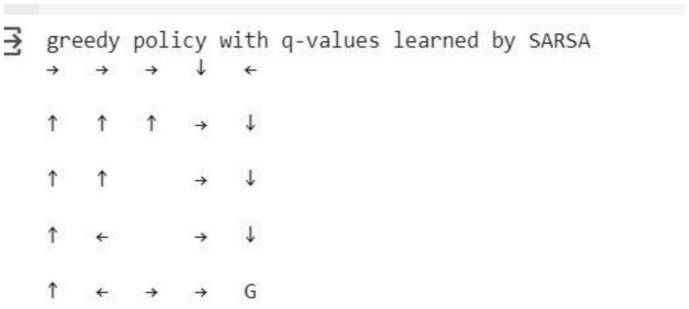

Figure 9: Qlearning on 687 grid world greedy policy with q-values learned by SARSA

---

4. Creating Your Own Gridworld [15 Points, total]

In this part of the assignment, you will investigate different properties of RL algorithms using a pilot program developed by one of the TAs. This program allows you to easily and interactively create you own gridworld MDPs. First, clone this repository and follow the installation instructions. All instructions for using the program are in the repository's README.

(Question 4a. 6 Points) You will first investigate how different definitions of the initial state distribution, d_0, may affect the difficulty of solving reinforcement learning problems. Start by launching the program and use it to define the components of your MDP: $\mathcal{S}$, p, d_0, $R(s')$, and γ. Notice that you can solve this MDP (i.e., find an optimal policy) via the Value Iteration algorithm by clicking the "Solve!" option. Once you have done so, the program will display the number of iterations, n, required for Value Iteration to converge. Next, click "Show Value Function" to display the optimal value function, v^*, identified by Value Iteration, as well as the expected return of the corresponding optimal policy; that is, J^*.

You will now create your own MDP and investigate two initial state distributions, d_0 and d_0', which you are free to specify. Let n be the number of iterations required for Value Iteration to converge. Your goal is to identify/specify d_0 and d_0' such that:

- When using d_0, it takes SARSA *fewer* than n episodes to achieve an average return that corresponds to 80% of the maximum return possible, that is $0.8\, J^*_{d_0}$, where $J^*_{d_0}$ is the expected return of the optimal policy when using the initial state distribution d_0.
- When using d_0', it takes SARSA *more* than $10n$ episodes to achieve an average return that corresponds to 80% of the maximum return possible, that is $0.8\, J^*_{d_0'}$, where $J^*_{d_0'}$ is the expected return of the optimal policy when using the initial state distribution d_0'.

When solving this question, please use the program's default hyperparameters for SARSA. Furthermore, notice that the *average return* mentioned above can be easily determined by visually inspecting the exponential moving average (EMA) line plotted along with the learning curves presented by the program.

Given the MDP you created and the two initial state distributions you specified:

- Present a screenshot of your MDP;
- Present the learning curves of SARSA for each of the initial state distributions, d_0 and d_0';
- Describe in English what the distributions you designed, d_0 and d_0', are modeling and why you chose them;

Question assigned to the following page: 4.1

- Report the value of n. Also report $J^*_{d_0}$ (the expected return of the optimal policy for the MDP with initial state distribution d_0) and $J^*_{d'_0}$ (the expected return of the optimal policy for the MDP with initial state distribution d'_0).

The chosen Markov Decision Process (MDP) features a single terminal state highlighted in green. Transitioning to this terminal state yields a reward of 100.0, while all other state transitions offer a reward of zero. The transition probabilities mirror those of the 687-GridWorld.

In the initial distribution of this MDP, the agent consistently starts in the blue-colored state with a probability of 1.0. This initial state is positioned in close proximity to the terminal state.

The rationale behind this design choice lies in positioning the agent near the goal state upon initialization. This proximity increases the likelihood of the agent reaching the goal state within a reduced number of steps, aligning with the SARSA algorithm's optimization objectives. In the initial setup of the MDP

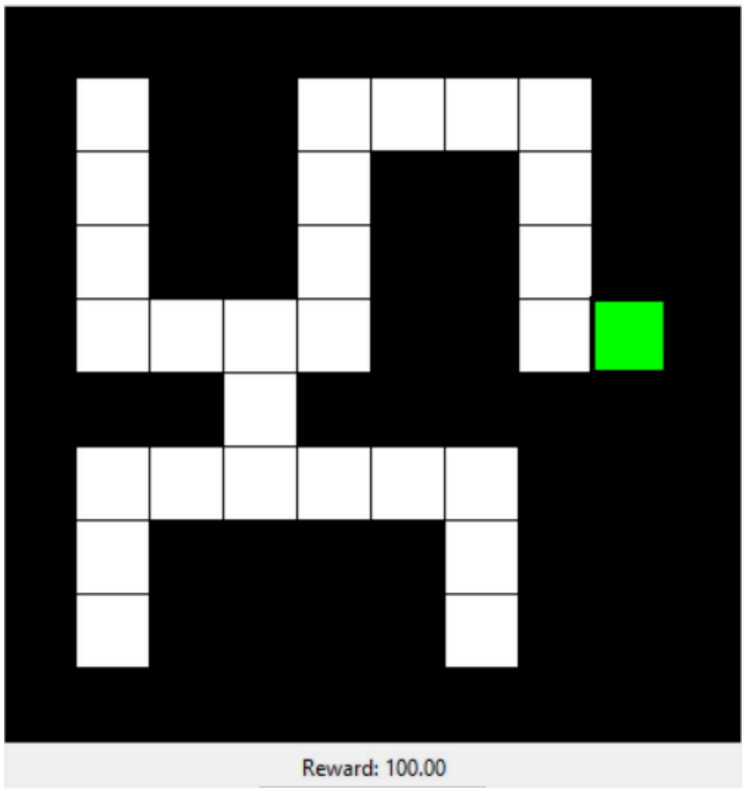

Figure 10: MDP with first initial distribution d_0

shown, the agent consistently begins in the blue state with a probability of 1.0.

The initial state is positioned near the terminal state. The rationale behind this design choice is that initializing the agent in close proximity to the goal state increases the likelihood of reaching it in fewer steps, particularly when utilizing the SARSA algorithm.

The value of 'n' is set to 133. The expected return of the optimal policy ($J^*_{d_0}$) for the MDP with an initial state distribution of d_0 is 438.22.

The SARSA learning curve for the initial distribution d_0 is shown in Figure . In this scenario, SARSA required less than 133 iterations to achieve 80 **MDP with second initial distribution d'_0:**

The MDP with the second initial distribution d'_0 is described as follows:

Question assigned to the following page: 4.1

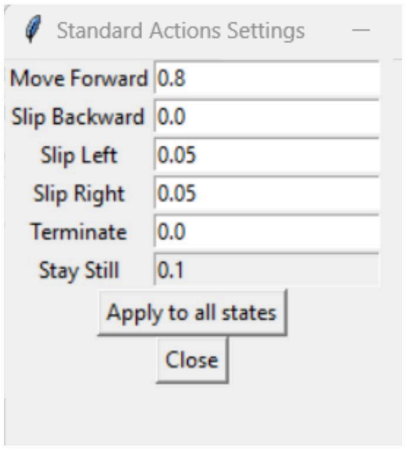

Figure 11: Transition probabilities for MDP

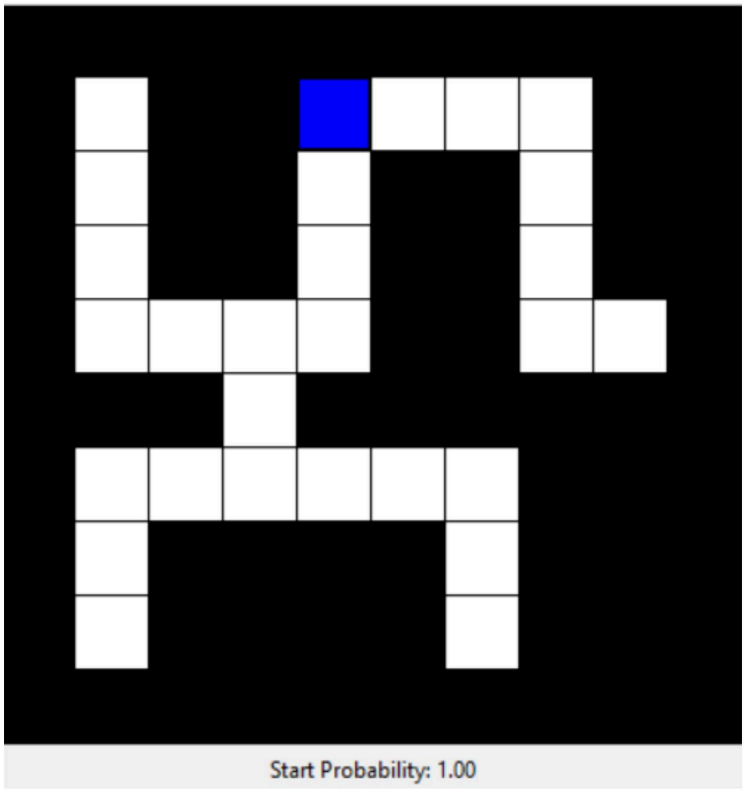

Figure 12: Initial distribution d_0

The selected Markov Decision Process, illustrated in Figure 10, comprises a single terminal state highlighted in green. Transitioning to this terminal state yields a reward of 100.0, while all other state transitions offer a reward of zero. The transition probabilities are consistent with those of the 687-GridWorld, as depicted in Figure 11. **MDP with second initial distribution d_0':**

In the initial distribution of the MDP portrayed in Figure 14, the agent is consistently initialized in the

Question assigned to the following page: 4.1

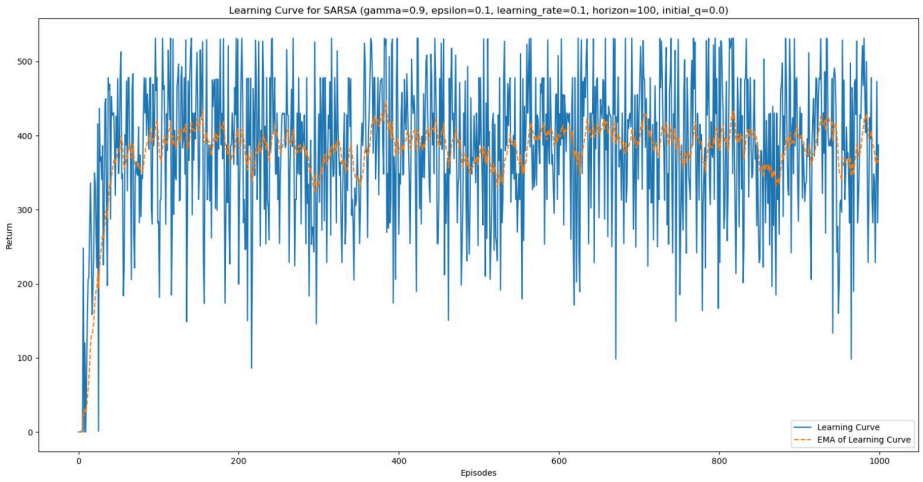

Figure 13: SARSA learning curve for initial distribution d_0

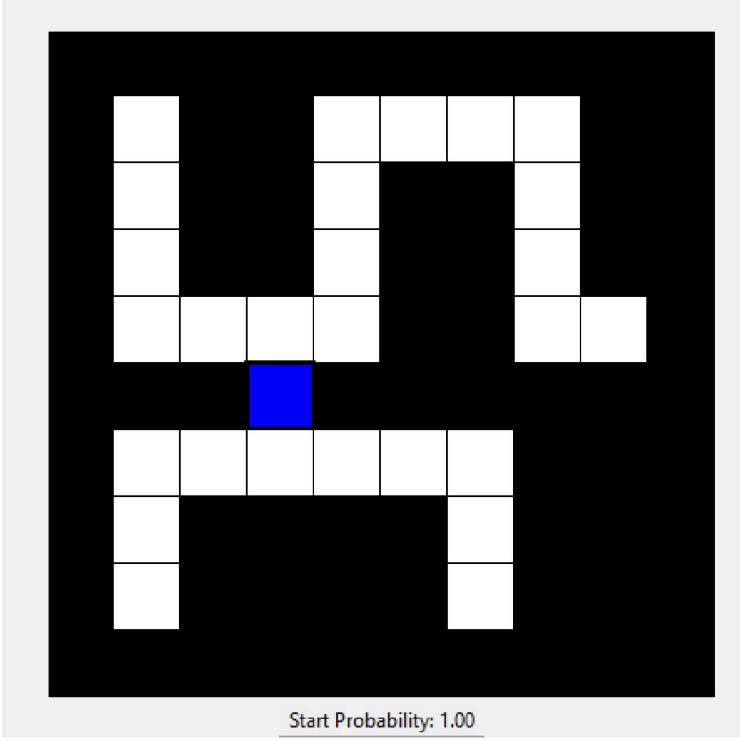

Figure 14: MDP with second initial distribution d_0'

blue state with a probability of 1.0. The initial state is notably distant from the terminal state. The rationale behind this design decision is that by initializing the agent farther from the goal state,

Questions assigned to the following page: 4.2 and 4.1

it's more likely to take numerous steps to reach the goal state when using the SARSA algorithm.

The value of 'n' is set to 133. The expected return of the optimal policy ($J^*_{d'_0}$) for the MDP with an initial state distribution of d'_0 is 225.6.

The SARSA learning curve for the initial distribution d'_0 is illustrated in Figure 18. For this scenario, SARSA required fewer than 133 iterations to achieve 80

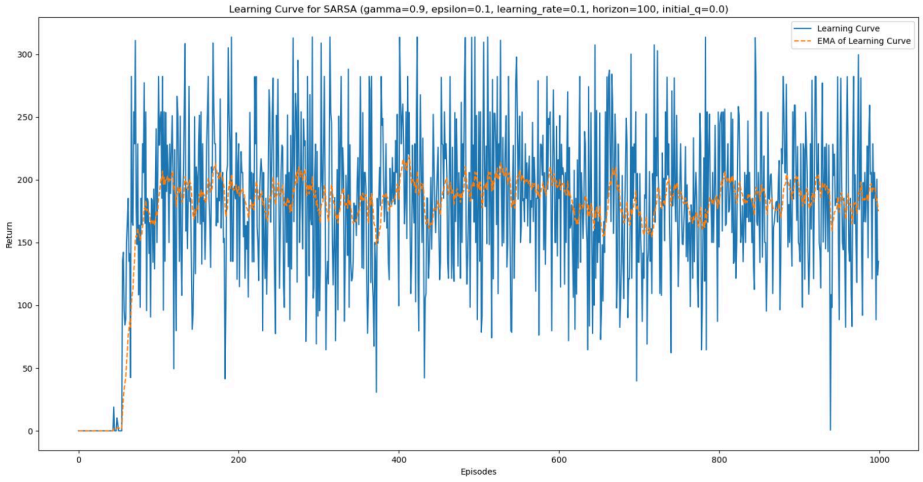

Figure 15: SARSA learning curve for initial distribution d'_0

(Question 4b. 6 Points) You will now investigate how the values used to initialize the q-function being learned by Q-Learning impact its learning speed. To do so, either use the same MDP as in the previous question (and select only one of the initial state distributions you designed) or create a new MDP. Your goal is to identify two ways of initializing the q-function being learned by Q-Learning: q_0 and q'_0. You should identify these q-function initializations such that:

- When using q_0 as its initial q-function, it takes Q-learning *fewer* than $3n$ episodes to achieve an average return that corresponds to 80% of the maximum return possible, that is $0.8\,J^*$;
- When using q'_0 as its initial q-function, it takes Q-learning *more* than $10n$ episodes to achieve an average return that corresponds to 80% of the maximum return possible, that is $0.8\,J^*$.

When solving this question, you can use any Q-Learning hyperparameters you like. As before, ***(i)*** present a screenshot of your MDP (unless it is the same as you defined in the previous question); ***(ii)*** present the learning curves of Q-Learning when using q_0 as its initial q-function, and when using q'_0 as its initial q-function. Clearly identify which learning curve is associated with which initial q-function; and ***(iii)*** report the values of n, J^*, d_0, and p.

Value iteration converged in $n = 19$ steps, whereas q_0 and q'_0 required $n = 54$ and $n = 423$ iterations, respectively. The optimal value function J^*_{d0} is 4.0186. Lower q-values prompt the algorithm to seek terminal states for increased rewards. Conversely, higher q-values prolong convergence, necessitating more iterations to reach the original value.

The initial distribution $d0$ assigns a value of 1 to coordinates $[0, 0]$ and 0 elsewhere.

Questions assigned to the following page: 4.2 and 4.3

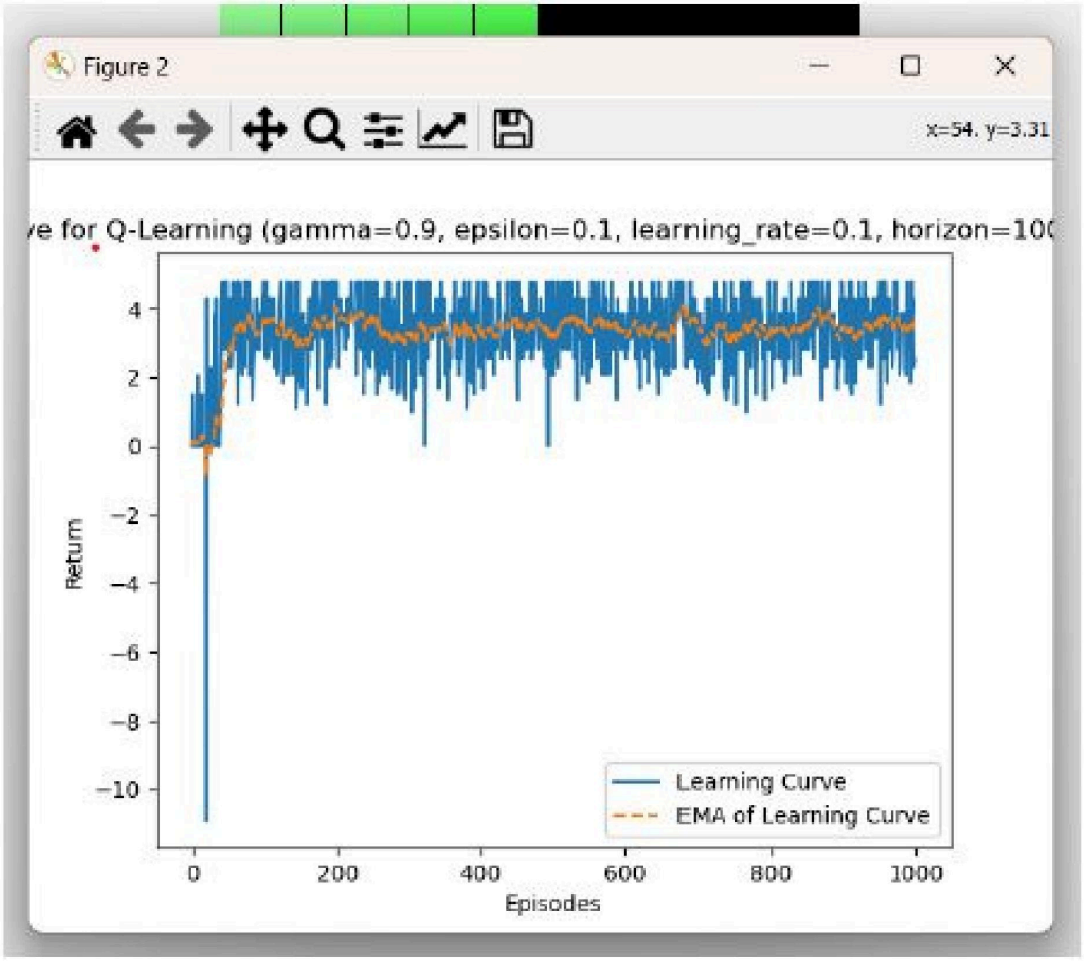

Figure 16: SARSA learning curve for initial distribution d'_0, q0 = 3

(Question 4c. 3 Points) Finally, please fill out our (short!) survey. The responses you submit are not anonymous since they will be used to grade this assignment. Importantly, however, notice that we will not associate any feedback you provide with your assignment: your feedback will only be used to improve the program designed by the TA. It will *not* impact your grade. Thank you!

Question assigned to the following page: 4.2

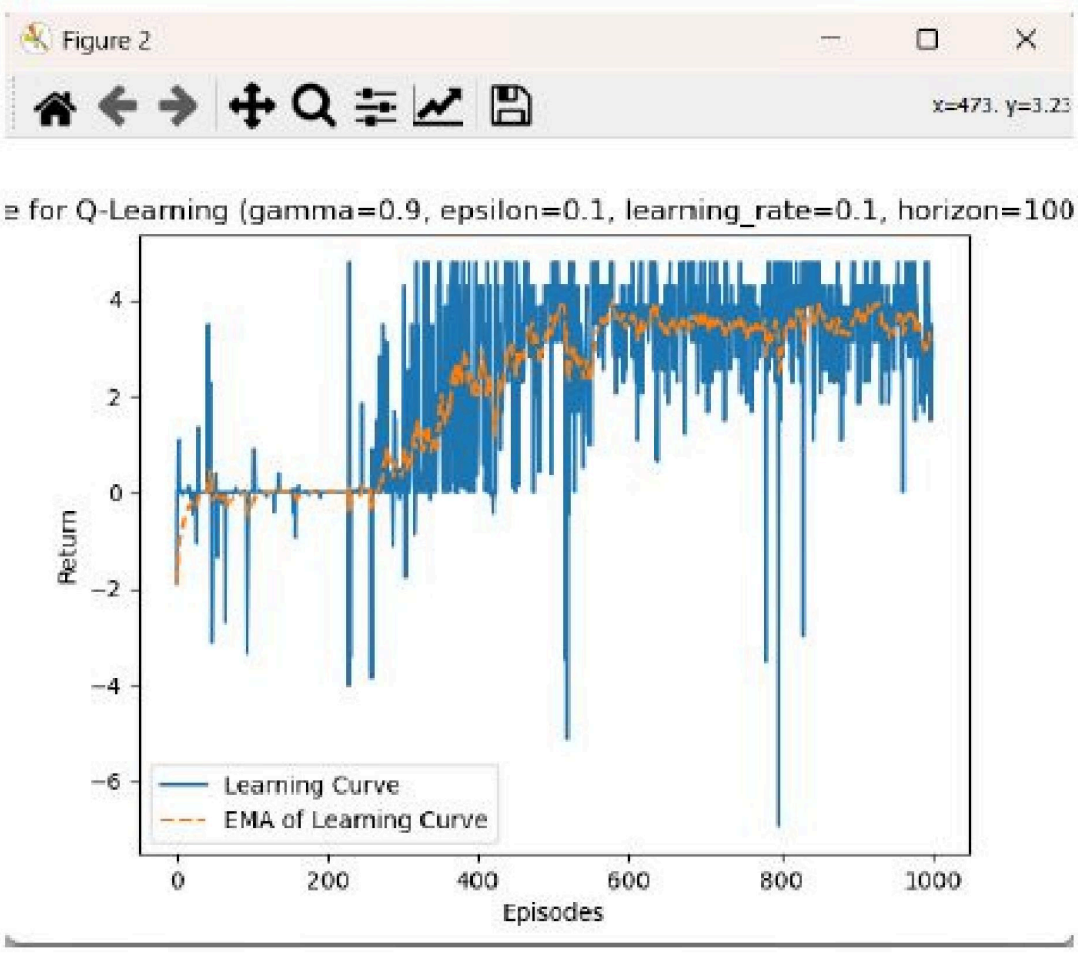

Figure 17: SARSA learning curve for initial distribution d_0', q0 = 3

Question assigned to the following page: 4.2

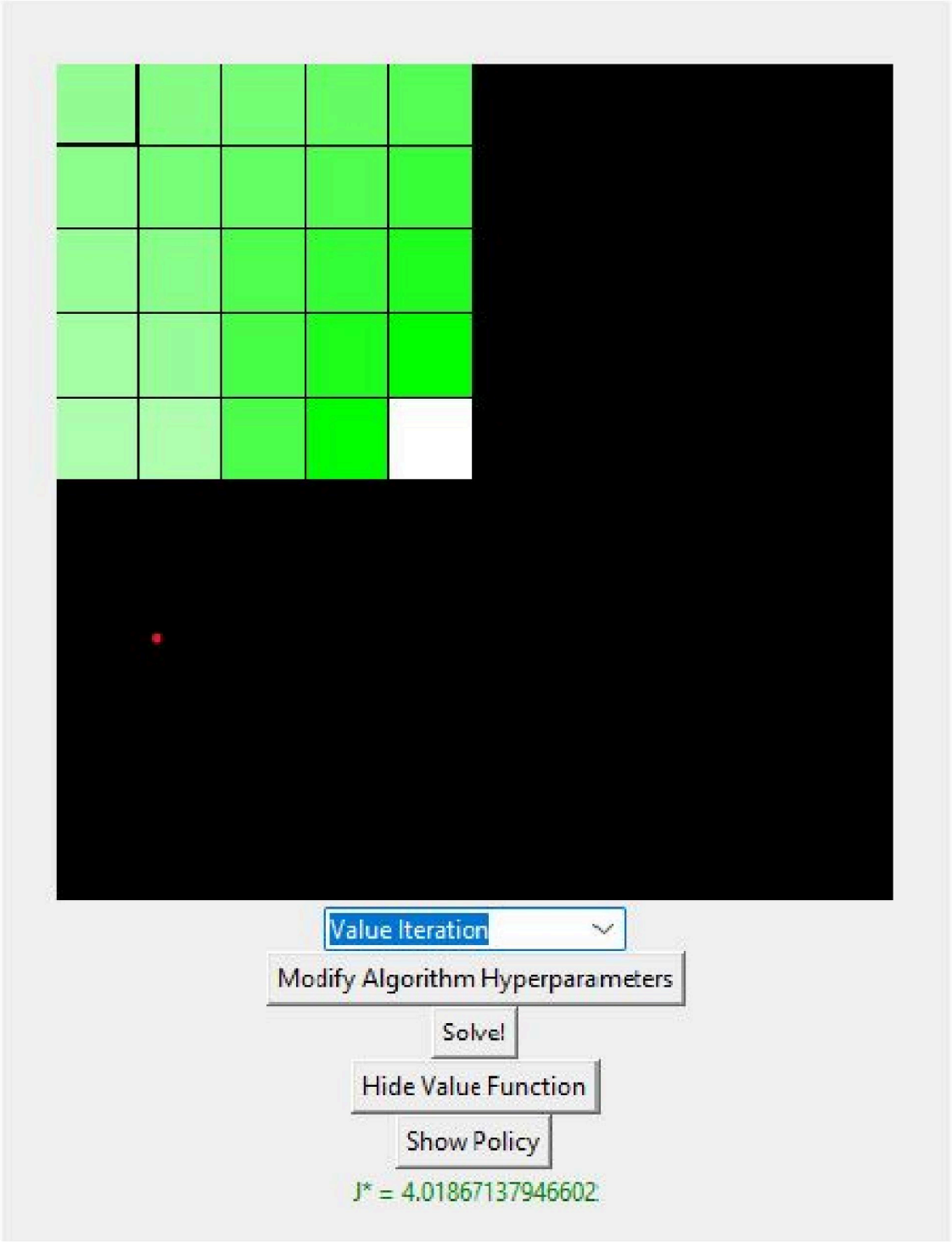

Figure 18: J*d0