ECE 232 Project 3 Report

July 24, 2024

Name: Kelly Furuya

1 Introduction

2 Reinforcement Learning (RL)

Q1: For visualization purpose, generate heat maps of Reward function 1 and Reward function 2. For the heat maps, make sure you display the coloring scale. You will have 2 plots for this question

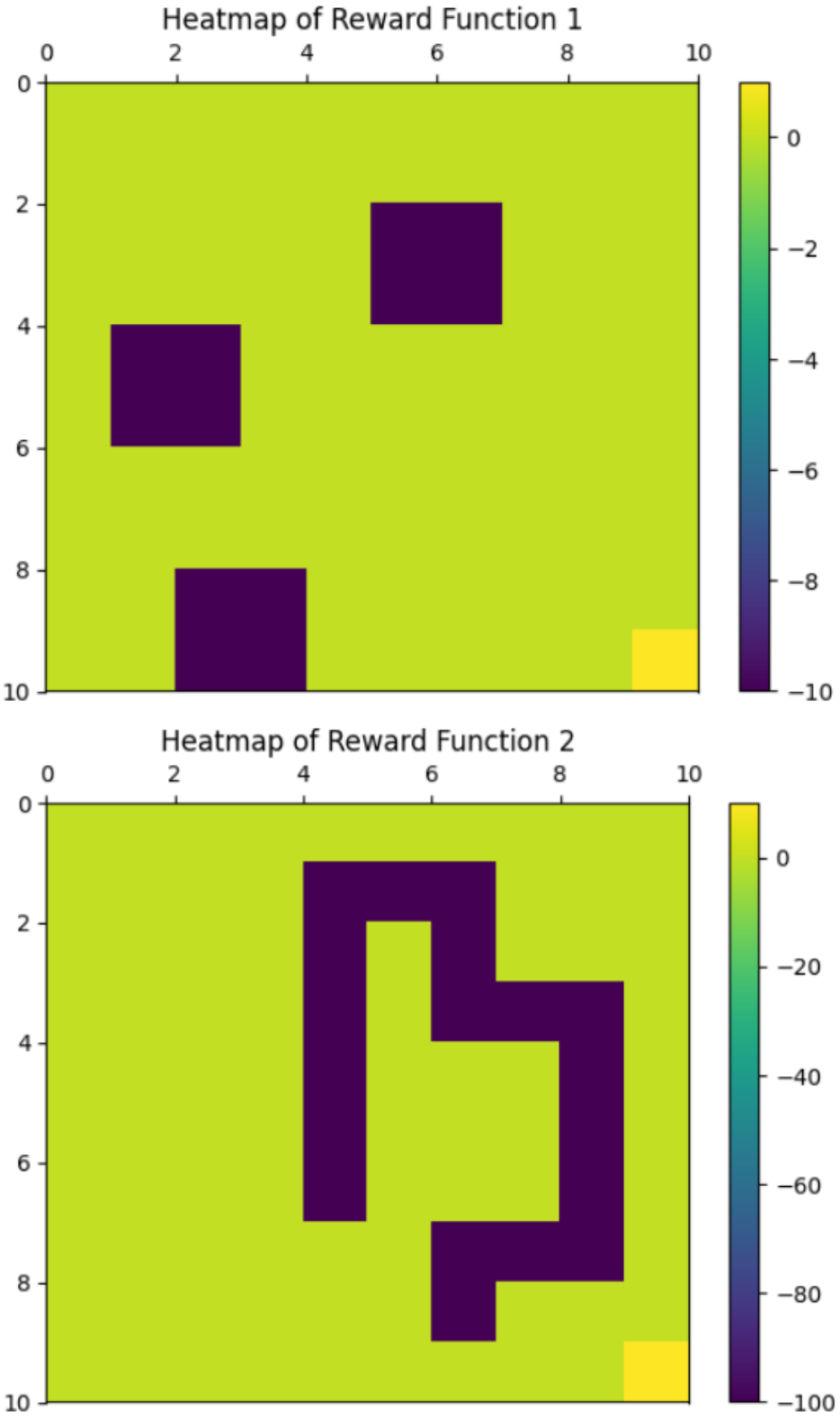

Figure 1: Heatmaps for Reward Functions 1 and 2

3 Optimal policy learning using RL algorithms

Q2: Create the environment of the agent using the information provided in section 2. Then write an optimal state-value function that takes as input the environment of the agent and outputs the optimal value of each state in the grid. For visualization purpose, you should generate a figure similar to that of figure 1 but with the number of state replaced by the optimal value of that state. In this part of question, you should have 1 plot. Let's assume that your value iteration algorithm converges in N steps. Plot snapshots of state values in 5 different steps linearly distributed from 1 to N. Report N and your step numbers. What observations do you have from the plots?

We are given the states, S, already in the form of the grid in Figure 1 of the prompt, the four possible actions, A, as up, down, left, and right, the reward function as Reward function 1 in Figure 6 of the prompt, $w = 0.1$, and $\gamma = 0.8$. We can then create functions in Python to generate the optimal state values for any given step.

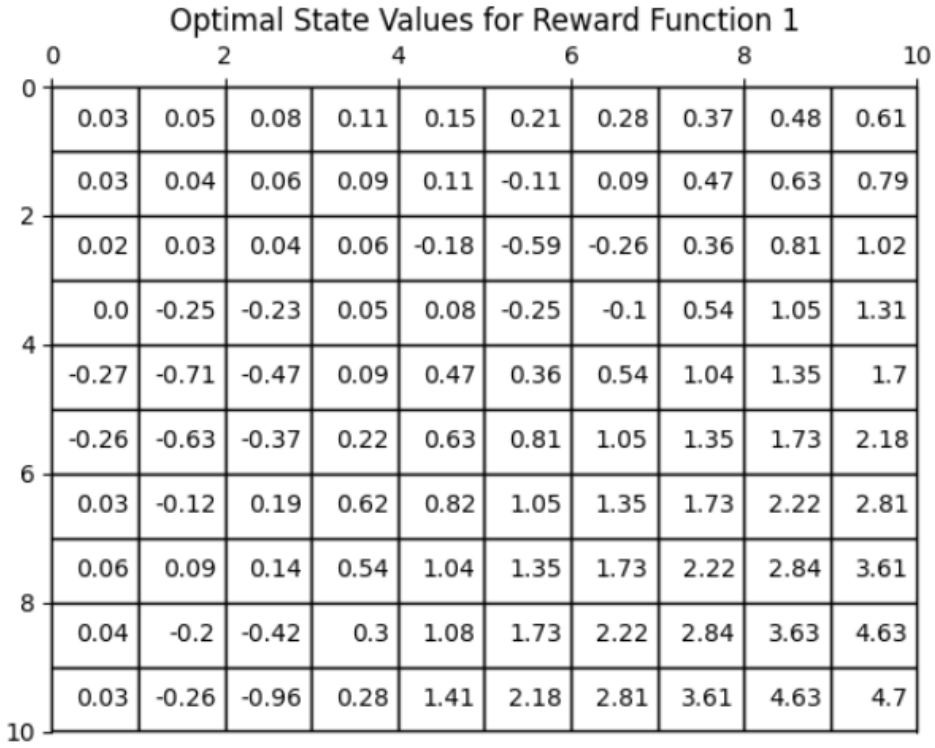

Figure 2: Final Optimal State Values

Figure 2 shows the optimal state values after the algorithm converges and finishes. The algorithm takes 21 steps to converge.

State Values at Step 1

0	2	4	6	8	10
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	-0.25	-0.26	-0.01	-0.0	-0.0
0.0	0.0	0.0	0.0	-0.25	-0.69	-0.52	-0.26	-0.01	-0.0
0.0	-0.25	-0.26	-0.01	-0.26	-0.52	-0.52	-0.27	-0.01	-0.0
-0.25	-0.69	-0.52	-0.26	-0.01	-0.26	-0.27	-0.01	-0.0	-0.0
-0.26	-0.52	-0.52	-0.27	-0.01	-0.01	-0.01	-0.0	-0.0	-0.0
-0.01	-0.26	-0.27	-0.01	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0
-0.0	-0.01	-0.26	-0.26	-0.01	-0.0	-0.0	-0.0	-0.0	-0.0
-0.0	-0.25	-0.69	-0.52	-0.26	-0.01	-0.0	-0.0	-0.0	0.92
-0.0	-0.26	-0.95	-0.78	-0.27	-0.01	-0.0	-0.0	0.92	0.99

State Values at Step 6

0	2	4	6	8	10
-0.0	-0.0	-0.0	-0.0	-0.0	-0.01	-0.01	-0.0	-0.0	-0.0
-0.0	-0.0	-0.0	-0.0	-0.01	-0.27	-0.27	-0.01	-0.0	-0.0
-0.0	-0.01	-0.01	-0.01	-0.28	-0.74	-0.74	-0.27	-0.01	-0.0
-0.01	-0.27	-0.28	-0.02	-0.29	-0.74	-0.74	-0.27	-0.01	0.21
-0.28	-0.74	-0.74	-0.29	-0.02	-0.28	-0.27	-0.01	0.23	0.57
-0.29	-0.74	-0.74	-0.28	-0.01	-0.01	-0.01	0.23	0.6	1.05
-0.01	-0.28	-0.29	-0.02	-0.0	-0.0	0.23	0.6	1.09	1.68
-0.0	-0.02	-0.29	-0.28	-0.01	0.23	0.6	1.09	1.71	2.48
-0.01	-0.28	-0.75	-0.74	-0.04	0.6	1.09	1.71	2.5	3.5
-0.01	-0.29	-1.02	-0.81	0.28	1.05	1.68	2.48	3.5	3.57

Figure 3: State Values at steps 1 and 6

State Values at Step 11

0	2	4	6	8	10
-0.0 -0.0 -0.0 -0.0 -0.0 -0.01 -0.01 0.05 0.16 0.28					
-0.0 -0.0 -0.0 -0.0 -0.01 -0.27 -0.22 0.15 0.3 0.46					
-0.0 -0.01 -0.01 -0.01 -0.28 -0.74 -0.58 0.03 0.48 0.69					
-0.01 -0.27 -0.28 -0.02 -0.23 -0.57 -0.43 0.22 0.72 0.99					
-0.28 -0.74 -0.74 -0.23 0.14 0.03 0.22 0.72 1.02 1.37					
-0.29 -0.74 -0.68 -0.11 0.3 0.49 0.72 1.03 1.41 1.85					
-0.01 -0.28 -0.13 0.29 0.49 0.73 1.03 1.41 1.89 2.48					
-0.0 -0.02 -0.18 0.21 0.72 1.03 1.41 1.89 2.51 3.28					
-0.01 -0.28 -0.66 -0.03 0.75 1.4 1.89 2.51 3.3 4.31					
-0.01 -0.29 -1.0 -0.04 1.08 1.85 2.48 3.28 4.31 4.37					

State Values at Step 16

0	2	4	6	8	10
-0.0 -0.0 0.01 0.03 0.07 0.13 0.2 0.3 0.41 0.53					
-0.0 -0.0 0.01 0.03 0.05 -0.17 0.01 0.39 0.55 0.71					
-0.0 -0.01 -0.0 0.01 -0.23 -0.64 -0.33 0.28 0.73 0.94					
-0.01 -0.27 -0.27 0.0 0.01 -0.33 -0.18 0.47 0.97 1.24					
-0.28 -0.73 -0.54 0.01 0.39 0.28 0.47 0.97 1.27 1.62					
-0.29 -0.7 -0.44 0.14 0.55 0.74 0.97 1.28 1.66 2.1					
-0.01 -0.2 0.12 0.54 0.74 0.98 1.28 1.66 2.14 2.73					
-0.0 0.01 0.06 0.46 0.97 1.28 1.66 2.14 2.76 3.53					
-0.01 -0.26 -0.48 0.22 1.0 1.65 2.14 2.76 3.55 4.56					
-0.01 -0.29 -0.99 0.2 1.33 2.1 2.73 3.53 4.56 4.62					

Figure 4: State Values at steps 11 and 16

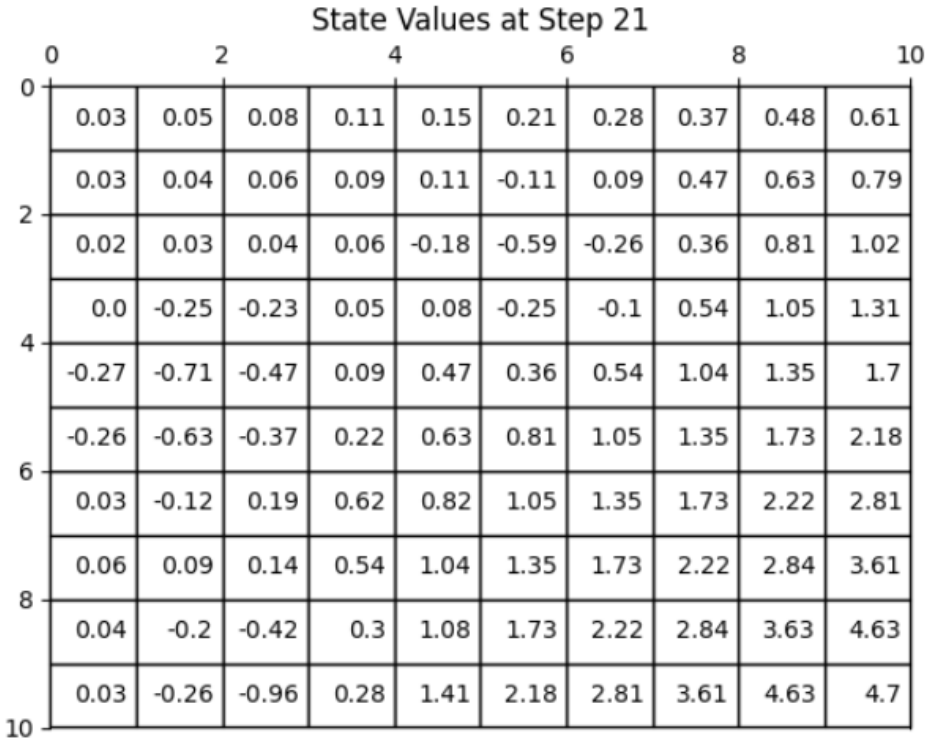

Figure 5: State Values at steps 21

Figures 3-5 shows the intermediary state values at steps 1, 6, 11, 16, and 21. These plots show that the algorithm at Step 1, already gives the states with negative reward values, a very low score while also immediately giving the states around the only positive reward value a comparatively high score. As the steps progress, we can see that the rest of the states begin to slowly fill in with some negative scores before primarily consisting of positive values, gradually increasing the gap between the highest and lowest score. These intermediate steps help clearly show that the algorithm can quickly identify the negative reward areas and assign them negative values to discourage travelling there. The top left corner, the corner farthest away from the positive reward value, remained relatively unchanged throughout the steps, really only receiving a proper score at the very end. This is likely the result of the algorithm preferring to move towards the positive reward value and only travelling to this far corner when the introduced randomness forces it there.

Q3: Generate a heat map of the optimal state values across the 2- D grid.

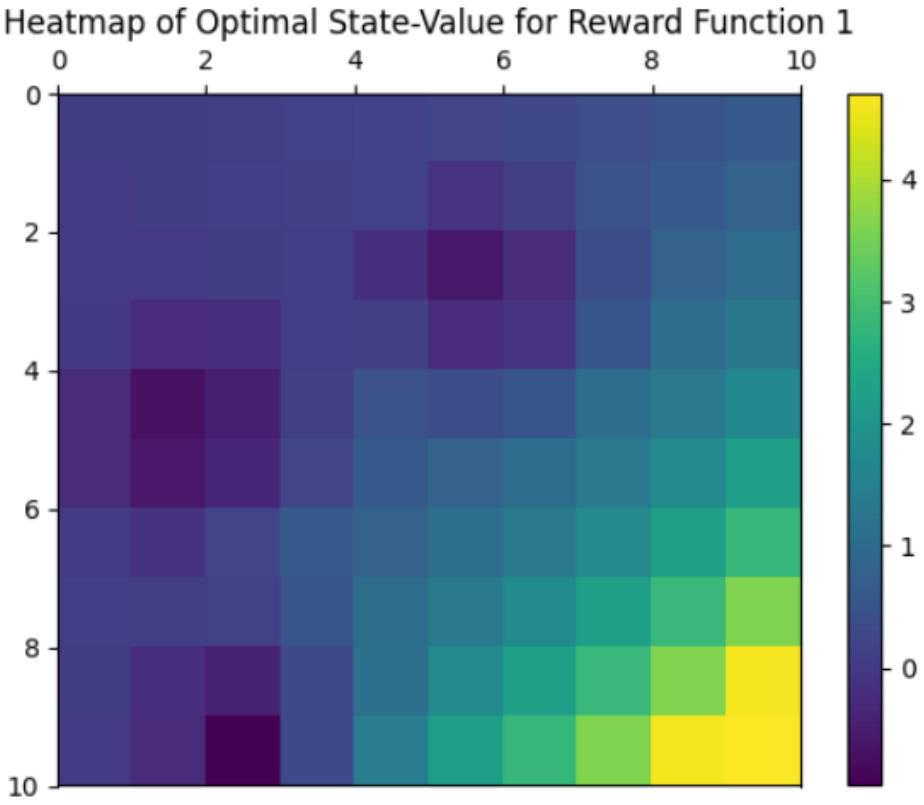

Figure 6: Heatmap of the optimal state values for Reward function 1

Figure 6 shows the heat map for the optimal state values for Reward function 1.

Q4: Explain the distribution of the optimal state values across the 2-D grid.

The heat map more visually shows the algorithm's prefence to avoid the areas with negative reward values and attraction to areas with positive reward values. The dark blue areas have a lower score while the green and yellow areas have much higher scores. The reason the algorithm doesn't just stay on the positive reward value is because of γ which helps introduce a degree of randomness to the algorithm's movements although it would prefer to remain in the positive reward state.

Q5: Implement the computation step of the value iteration algorithm (lines 14-17) to compute the optimal policy of the agent navigating the 2-D state-space. Generate a figure similar to that of figure 1 but with the number state replaced by the optimal action at that state using arrows. Does the optimal policy of the agent match your intuition? Please provide a brief explanation. Is it possible for the agent to compute the optimal action to take at each state by observing the optimal values of it's neighboring states?

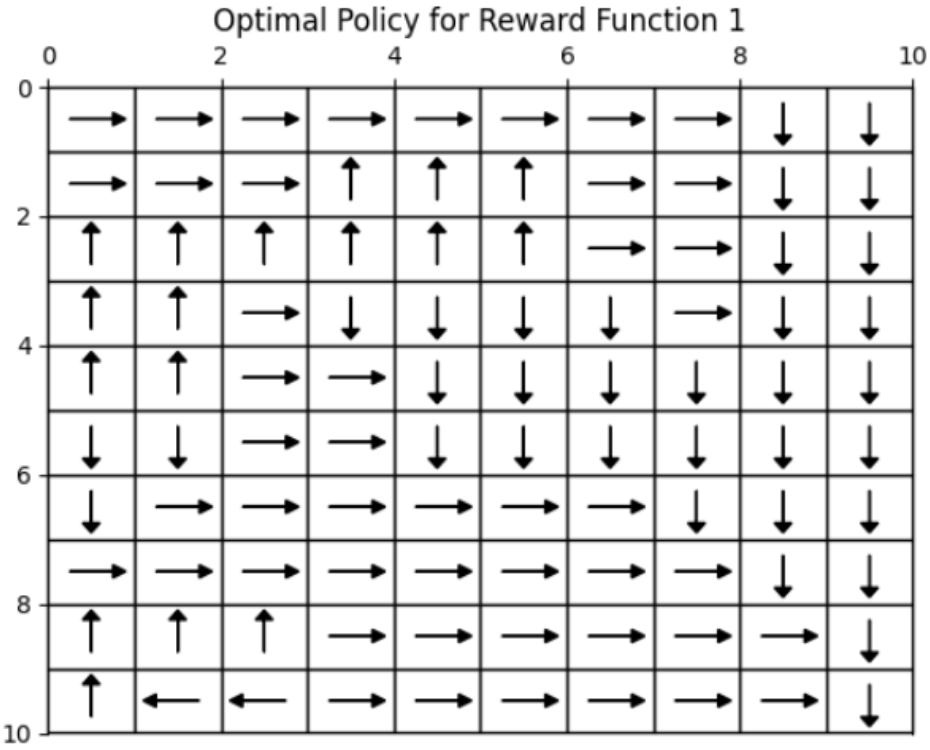

Figure 7: Optimal policy action for reward function 1

Figure 7 shows the moves the algorithm wants to take for each state. These movements more or less follow our intuition that it will want to move away from the negative value states and move to the neighboring state with the highest value. We can see that the algorithm clearly wants to head towards the positive reward value since the values of each state near it get higher as the algorithm moves closer to the bottom right corner.

The agent relies on being able to "see" the optimal values of its neighboring states in order to make a decision on which state to move to, so yes, it can compute the optimal action to take by observing the optimal values of it's neighboring states.

Q6: Modify the environment of the agent by replacing Reward function 1 with Reward function 2. Compute the optimal value of each state in the grid.

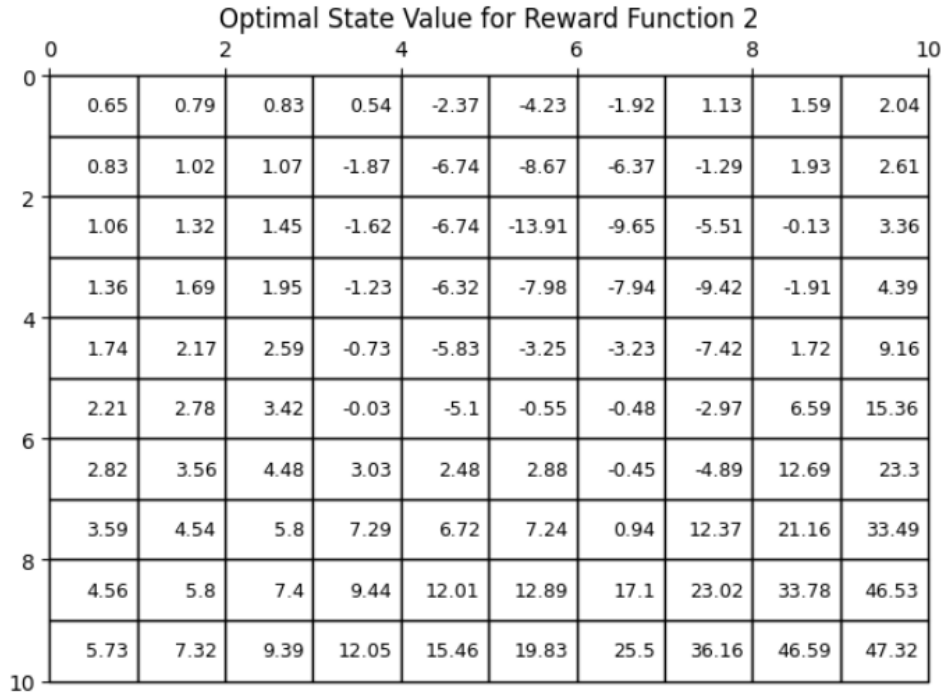

Figure 8: Optimal State Values for Reward Function 2

Figure 8 shows the optimal state values for Reward Function 2. We can see immediately that one of the most noticeable difference between the two reward functions is the greater range. Reward Function 1 had a state value range from -0.96 to 4.7 while Reward Function 2 has a state value range of -13.91 to 47.32. This is due to the much larger positive and negative reward values in Reward Function 2.

Q7: Generate a heat map of the optimal state values across the 2-D grid. Explain the distribution of the optimal state values across the 2-D grid.

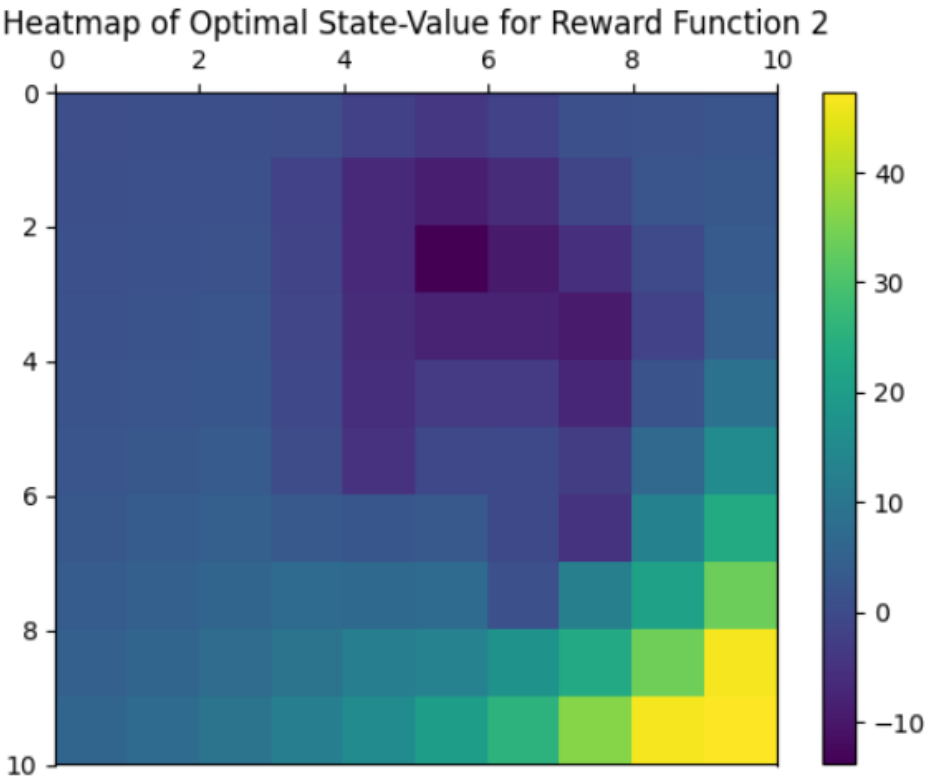

Figure 9: Heat map of the optimal state values for Reward Function 2

Figure 9 shows the resulting heat map for reward function 2. We can see that, similar to Reward Function 1, the agent clearly prefers to avoid the areas with negative reward values, travelling there much less frequently than the area with a positive reward value. The areas immediately surrounding the negative values are also less frequently visited, as indicated by their dark color, but are a lighter color than the negative state values. The yellow and green states are visited much more often as they have higher state values and are more attracitve to the agent. Noticeable, the area in the middle of the negative reward values, although having a reward value of 0, is visited even less frequently than the surrounding negative values. This is likely the result of the agent almost never travelling to that square as it wiill always want to move away from the negative values and towards states that will give a higher value. Although that spot has a reward value of 0, the neighboring negative reward values make the state value much lower, making it undesireable to the agent.

Q8: Compute the optimal policy of the agent navigating the 2-D state-space. Generate a figure simlar to that of figure 1 but with the number of state replaced by the optimal action at that state. Does the optimal policy of the agent match your inuition?

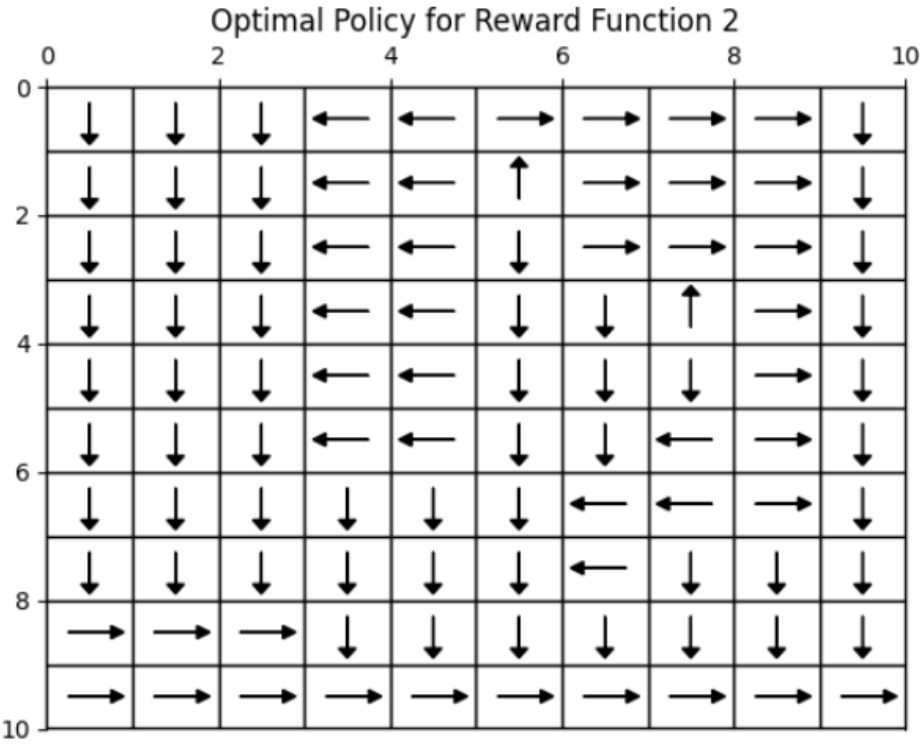

Figure 10: Optimal policy action for reward function 2

Once again, the optimal action roughly follows our intuition that the agent will try to move away from the negative state values and move towards the highest neighbor. We can clearly see that the agent moves away from the areas that have negative reward values and tries to move towards the one state with a positive reward value. Just like the observations in Q7, we can see that the agent will also try to avoid going to the states in the middle of the open loop created by the negative reward values as these states have very low state values.

Q9: Change the hyper parameter w to 0.6 and find the optimal policy map similar to the previous question for both reward functions. Explain the differences you observe. What do you think about the value of new w compared to the previous value? Choose the w that you think will give rise to better optimal policy anduse that w for the next stages of the project.

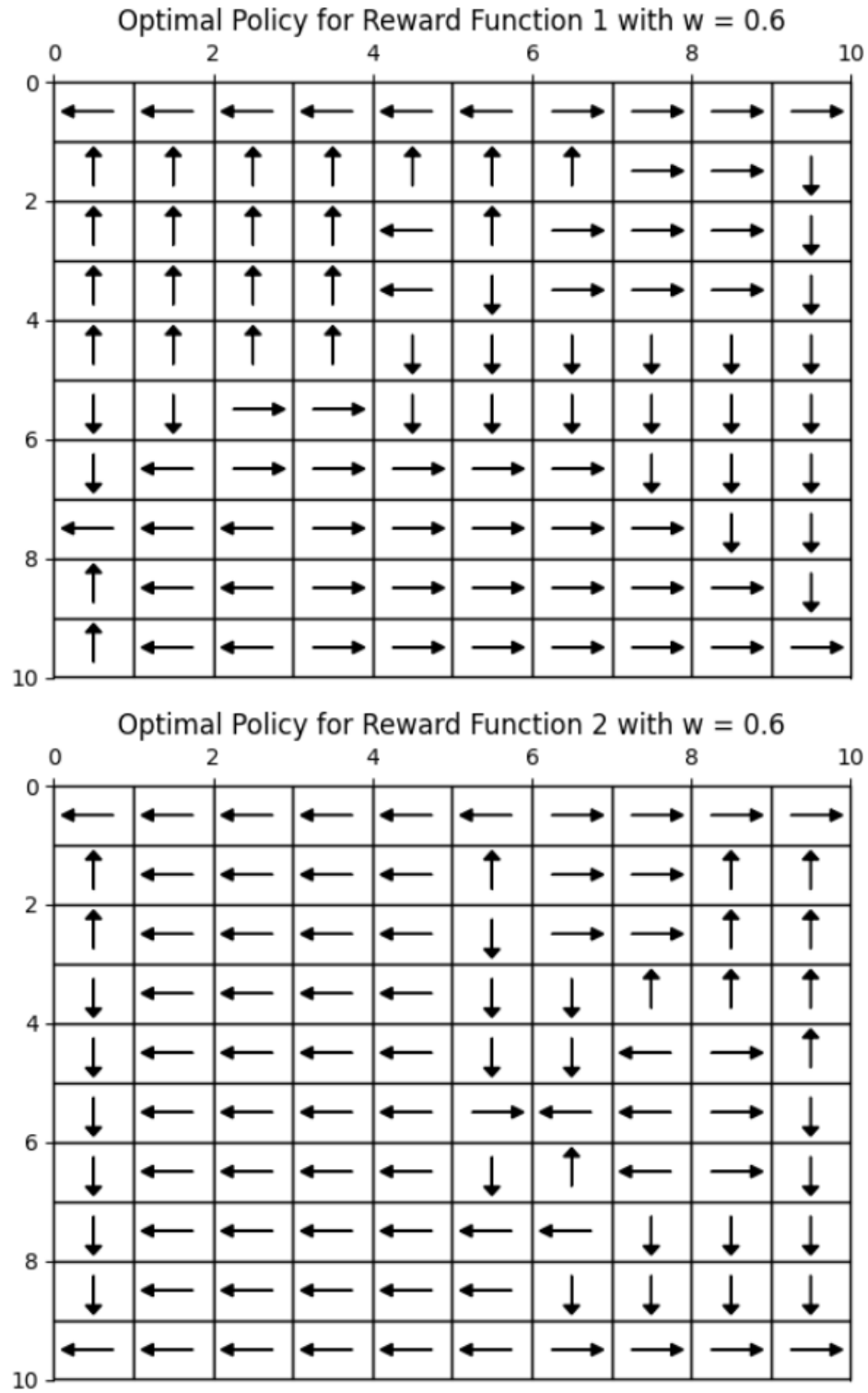

Figure 11: Optimal policy actions for Reward Function 1 and 2 with $w = 0.6$

The two mappings in Figure 11 show that changing w results in fairly different optimal policies for both reward functions. Although the algorithm still tries to move to a higher state value, the way the values are calculated is changed enough that the effects of the negative reward values

spreads further. This means that the states near a state with a negative reward will have a lower state value than when $w = 0.1$ and this effect will spread to neighboring states with a greater impact. If we look at the state values for the two reward functions, we can see this effect more clearly.

Optimal State Value for Reward Function 1 with w = 0.6

	0	2	4	6	8	10
0	-0.02	-0.03	-0.04	-0.09	-0.24	-0.61	-0.6	-0.23	-0.07	-0.03
	-0.06	-0.09	-0.13	-0.25	-0.86	-2.96	-2.94	-0.8	-0.17	-0.04
2	-0.23	-0.49	-0.57	-0.73	-3.08	-6.15	-6.07	-2.84	-0.41	-0.02
	-1.01	-2.95	-3.04	-1.52	-3.47	-6.2	-6.04	-2.79	-0.34	0.1
4	-3.58	-6.27	-6.28	-3.49	-1.55	-2.99	-2.79	-0.61	0.09	0.3
	-3.82	-6.56	-6.41	-3.21	-0.79	-0.5	-0.3	0.12	0.39	0.56
6	-1.47	-3.6	-3.79	-1.39	-0.36	-0.01	0.19	0.42	0.68	0.94
	-0.92	-1.73	-3.94	-3.25	-0.74	0.08	0.41	0.7	1.09	1.54
8	-1.12	-3.52	-6.93	-6.54	-2.81	-0.07	0.61	1.08	1.68	2.54
	-1.29	-4.1	-9.33	-8.95	-3.34	-0.05	0.83	1.53	2.54	2.97
10										

Optimal State Value for Reward Function 2 with w = 0.6

	0	2	4	6	8	10
0	-0.24	-0.58	-2.46	-10.74	-38.96	-53.72	-41.27	-13.5	-4.83	-2.64
	-0.3	-0.95	-5.14	-30.82	-67.59	-84.81	-74.44	-40.01	-12.31	-4.85
2	-0.36	-1.14	-6.02	-34.71	-76.43	-117.77	-97.52	-75.39	-39.99	-13.58
	-0.39	-1.18	-6.15	-34.89	-74.32	-94.82	-94.03	-94.08	-74.14	-41.85
4	-0.35	-1.14	-6.02	-34.18	-70.12	-69.28	-67.41	-89.37	-85.31	-57.96
	-0.29	-1.04	-5.65	-32.72	-65.45	-60.0	-50.2	-68.16	-84.9	-61.26
6	-0.2	-0.8	-4.51	-27.31	-45.31	-56.2	-62.78	-84.64	-79.42	-49.45
	-0.1	-0.37	-1.73	-8.02	-30.76	-47.79	-73.23	-77.54	-60.97	-26.43
8	-0.05	-0.14	-0.58	-2.38	-8.81	-31.08	-44.68	-48.36	-20.35	11.35
	-0.03	-0.07	-0.23	-0.83	-2.87	-9.19	-25.82	-2.84	14.57	22.97
10										

Figure 12: Optimal state values for Reward Function 1 and 2 with $w = 0.6$

Figure 12 shows that the optimal state values for every state is lower when $w = 0.6$ than when

$w = 0.1$. As a result, there are far more negative state values than before with only the states nearest the positive reward function remaining positive.

The overall path that the agent might take from any given state in order to reach the positive reward state is noticeably longer when $w = 0.6$. If the goal of the agent is to reach and stay at the state that will give it the highest reward, then the w value that allows it to reach that state faster will likely give a better optimal policy. So in this case, we will use $w = 0.1$ for the remainder of this project.

4 Inverse Reinforcement Learning (IRL)

Q10: Express, c, x, D, b in terms of R, P_a, P_{a1}, t_i, u, λ, and R_{max}.

In order to express the standard LP formulation in terms of R, P_a, P_{a1}, t_i, u, λ, and R_{max}, we can use block matrices to reorganize and define the variables and constraints.

$$c = \begin{bmatrix} 1 \\ -\lambda \\ 0 \end{bmatrix}$$

$$x = \begin{bmatrix} t \\ u \\ R \end{bmatrix}$$

We can quickly check these two variables by multiplying the transpose of c by x where we would get $\Sigma_{i=1}^{|S|}(t_i - \lambda u_i)$. Next we can work on the constraints.

$$D = \begin{bmatrix} I_{|S|\times|S|} & 0 & -(P_{a1} - P_{a2})(I - \gamma P_{a1})^{-1} \\ I_{|S|\times|S|} & 0 & -(P_{a1} - P_{a3})(I - \gamma P_{a1})^{-1} \\ I_{|S|\times|S|} & 0 & -(P_{a1} - P_{a4})(I - \gamma P_{a1})^{-1} \\ 0 & 0 & -(P_{a1} - P_{a2})(I - \gamma P_{a1})^{-1} \\ 0 & 0 & -(P_{a1} - P_{a3})(I - \gamma P_{a1})^{-1} \\ 0 & 0 & -(P_{a1} - P_{a4})(I - \gamma P_{a1})^{-1} \\ 0 & -I_{|S|\times|S|} & I_{|S|\times|S|} \\ 0 & -I_{|S|\times|S|} & -I_{|S|\times|S|} \\ 0 & 0 & I_{|S|\times|S|} \\ 0 & 0 & -I_{|S|\times|S|} \end{bmatrix}$$

$$b = \begin{bmatrix} 0_{|S|\times 1} \\ 0_{|S|\times 1} \\ 0_{|S|\times 1} \\ 0_{|S|\times 1} \\ 0_{|S|\times 1} \\ 0_{|S|\times 1} \\ 0_{|S|\times 1} \\ 0_{|S|\times 1} \\ (R_{max})_{|S|\times 1} \\ (R_{max})_{|S|\times 1} \end{bmatrix}$$

where P_{a2}, P_{a3}, and P_{a4} represent the transtion probabilities for the non-optimal move. Again, we can double check this by performing matrix multiplication for Dx and see if it is less than b.

$$Dx = \begin{bmatrix} t - R(P_{a1} - P_{a2})(I - \gamma P_{a1})^{-1} \\ t - R(P_{a1} - P_{a3})(I - \gamma P_{a1})^{-1} \\ t - R(P_{a1} - P_{a4})(I - \gamma P_{a1})^{-1} \\ -R(P_{a1} - P_{a2})(I - \gamma P_{a1})^{-1} \\ -R(P_{a1} - P_{a3})(I - \gamma P_{a1})^{-1} \\ -R(P_{a1} - P_{a4})(I - \gamma P_{a1})^{-1} \\ -u + R \\ -u - R \\ R \\ R \end{bmatrix}$$

We were already given the constraint that $[(P_{a1}(i) - P_a(i))(I - \gamma P_{a1})^{-1}R] \geq t_i$ meaning that the first three elements in the matrix Dx will be negative for all values of i and thus less than 0. Next, we were given the constraint that $[(P_{a1}(i) - P_a(i))(I - \gamma P_{a1})^{-1}R \succeq 0$. If we multiply both sides by -1 we will see that the fourth, fifth, and sixth elements in Dx will also be less than 0. The constraint $-u \preceq R \preceq u$ proves that, once again, the seventh element will be less than 0 as $R \preceq u$ so $R - u$ will be negative. The same constraint proves that $-u - R$ will also result in a negative value. Lastly, the constraint $|R_i| \leq R_{max}$ proves that the last two elements of Dx will be less than the last two elements of b

Q11: Sweep λ from 0 to 5 to get 500 evenly spaced values for λ. For each value of λ compute $O_A(s)$ by following the process described. Then compute the accuracy for all 500 values and plot the data.

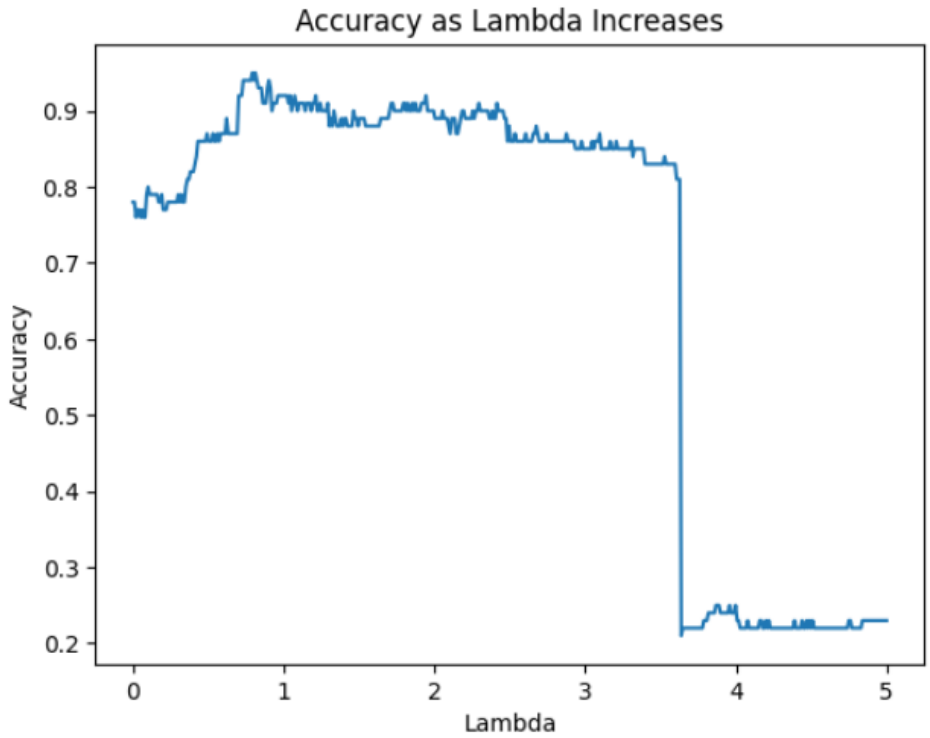

Figure 13: Change in accuracy of the IRL policy as λ increases.

We can see from the graph that as λ increases, so does the accuracy before plateauing and then sharply droping off. It makes sense that at the lower and higher ends of λ the acucracy would be

lower as a low λ encourages too many reward values to be close to 1 while a high λ encourages too many reward values to be close to 0. As a result, the best accuracy is at a lambda somewhere between.

Q12: Use the plot in question 11 to comput the value of λ for which accuracy is maximum. Denoe this as $\lambda_{max}^{(1)}$.

In this case, $\lambda_{max}^{(1)}$ is 0.792 with an accuracy of 95%.

Q13: For $\lambda_{max}^{(1)}$, generate heat maps of the ground truth reward and the extracted reward.

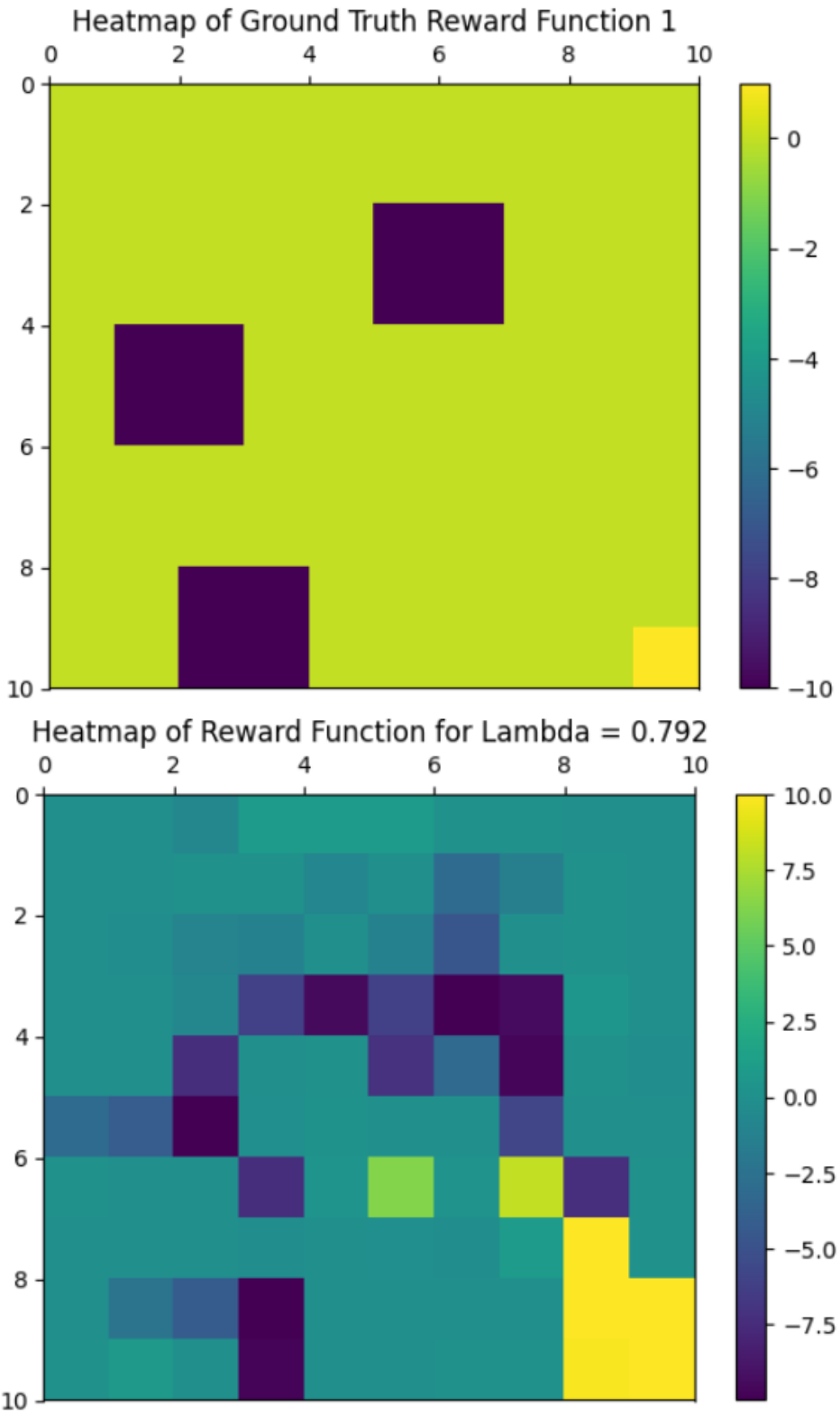

Figure 14: Heatmaps for gound truth and for $\lambda = 0.792$

The two heatmpas show that the IRL function was able to accurately predict the general area

with the positive reward value. It could also roughtly predict where the negative reward values were but with a bit more noise.

Q14: Use the extracted reward function computed in question 13 to compute the optimal values of the states in the 2-D grid. Generate a heat map of the optimal state values across the 2-D grid.

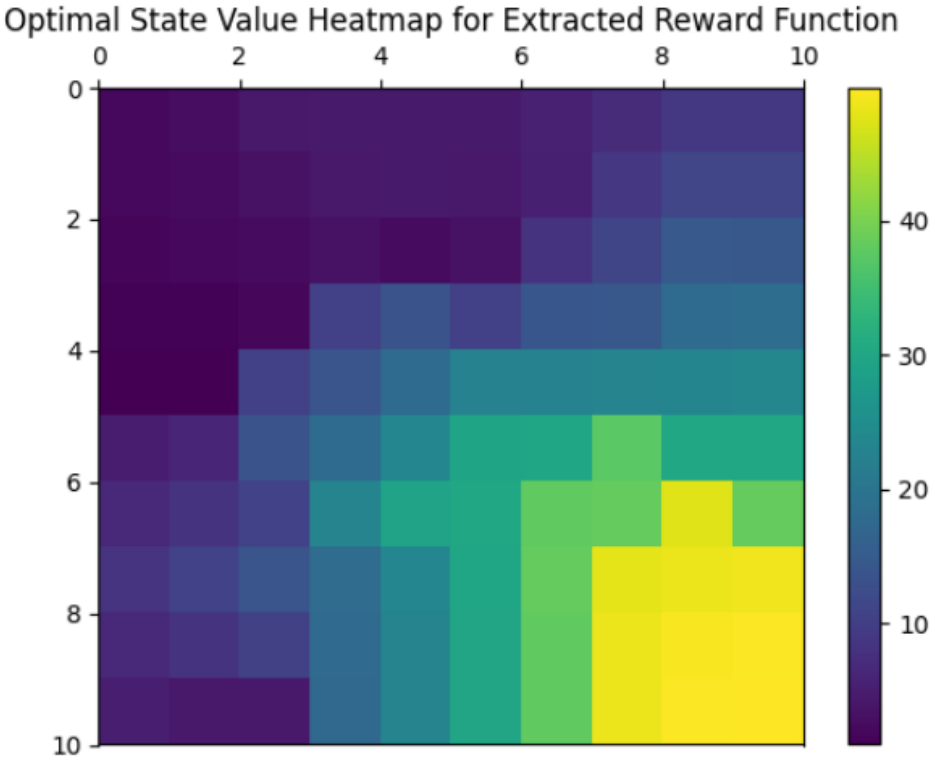

Figure 15: Optimal state values for the extracted reward function with $\lambda = 0.792$.

Q15: Compare the heat maps of Question 3 and Question 14 and provide a brief explanation on their similarities and differences.

The two heatmaps both indicate that the bottom right corner has states with significantly higher values than the rest of the map that slowly transition to a lower value the farther away from the corner the agent travels. They also show that a large portion of the map is either 0 or a value close to 0.

It also more or less indicates where the negative rewards are as there are 2-3 distinct areas that are darker than the rest of the heatmpa. The location of these dark areas roughly aligns with the negative reward values of Reward Function 1. Just like with the reward value heat maps, the algorithm is clearly able to realize there are areas that the optimal policy tries to avoid and asisgns lower state values to those areas while giving higher values to the areas the optimal policy tries to go towards.

Q16: Use the extracted reward function found in question 13 to compute the optimal policy of the agent. Generate a figure similar to that of figure 1 but with the number of state replaced by the optimal action at that state.

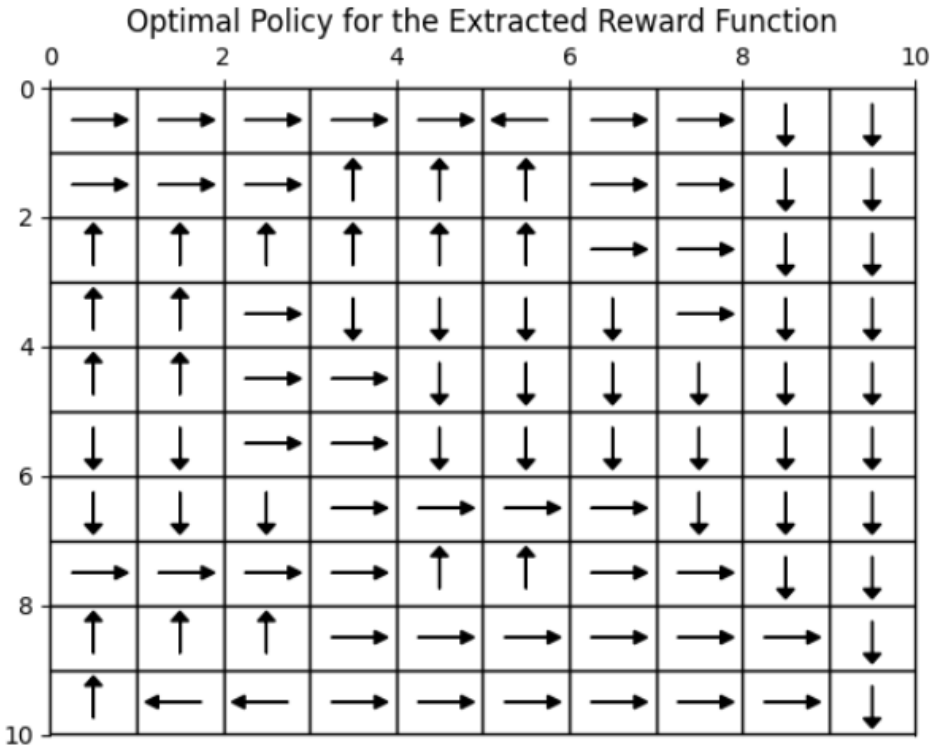

Figure 16: Optimal policy for the extracted reward function with $\lambda = 0.792$.

Q17: Compare the figures of Question 5 and Question 16 and provide a brief explanation on their similarities and differences.

Predictably, the two policies are similar. They both attempt to move to the bottom right corner of the grid with most of the moves and clearly try to move away from similar areas. There are only a few states where the two policies differ. $(0, 5)$, $(6, 1)$, $(6, 2)$, $(7, 4)$, and $(7, 5)$. These are all somewhat near the negative reward states and likely affected by them in some way. Figure 17 below shows these differing moves highlighted in red.

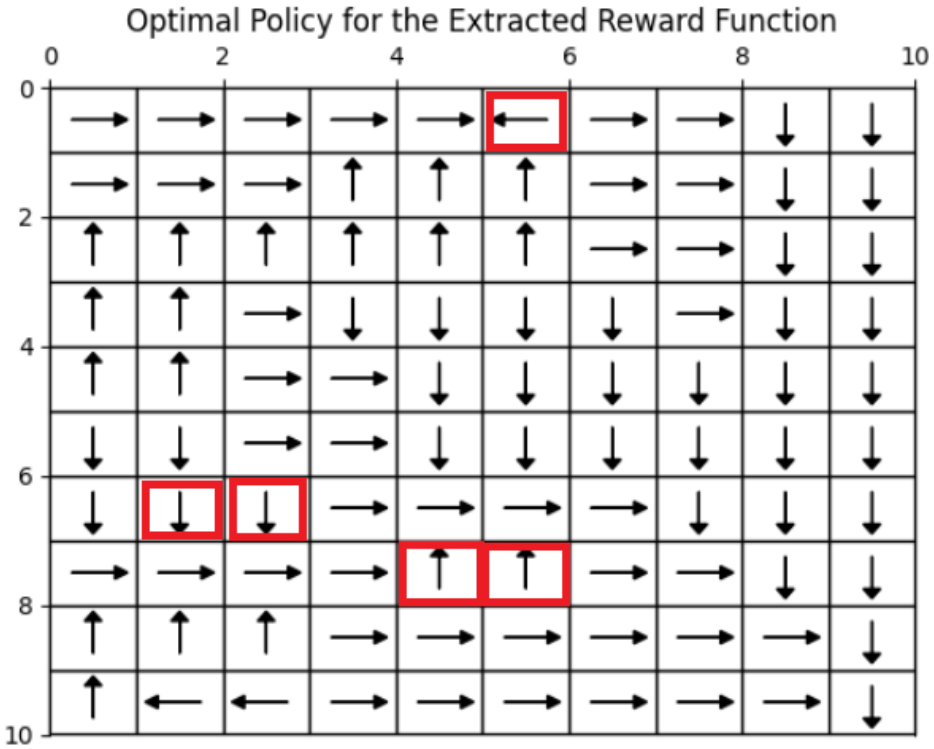

Figure 17: Optimal policy differenes.

Q18: Repeat Question 11 for Reward Function 2

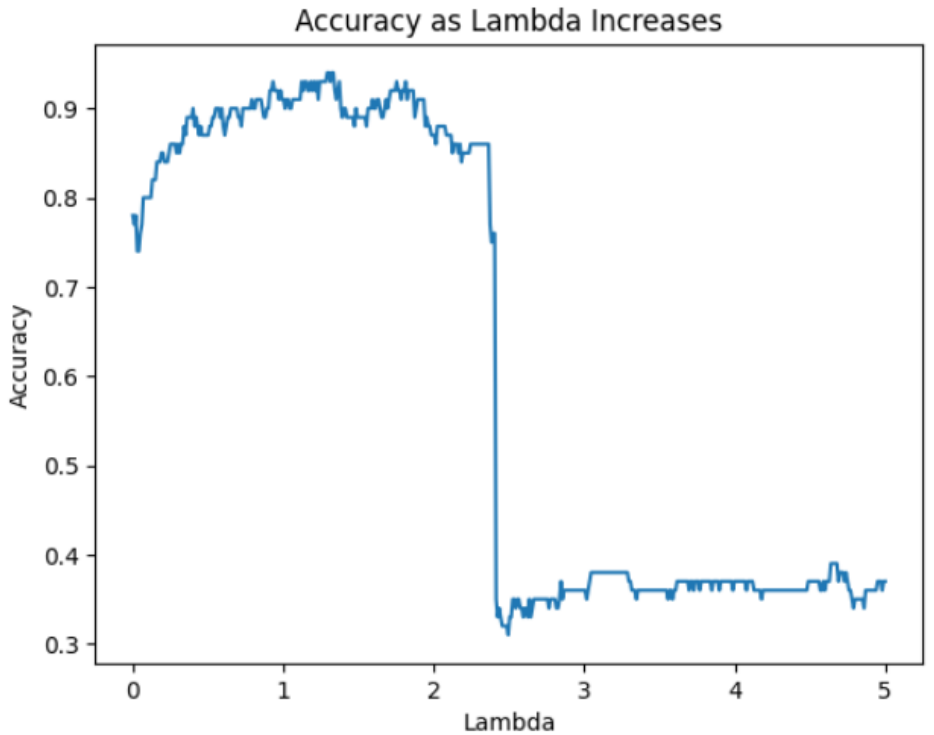

Figure 18: Change in accuracy of the IRL policy as λ increase for Reward Function 2.

Q19: Use the plot in question 18 to compute the value of λ for which accuracy is maximum. Denote this value as $\lambda_{max}^{(2)}$.

For this second version, the accuracy was highest, 94%, when $\lambda = 1.293$. Noticeably, the dropoff for the accuracy for reward function 2 occured at a lower λ than for reward function 1.

Q20: For $\lambda_{max}^{(2)}$, generate heat mpas of the ground truth reward and the extracted reward.

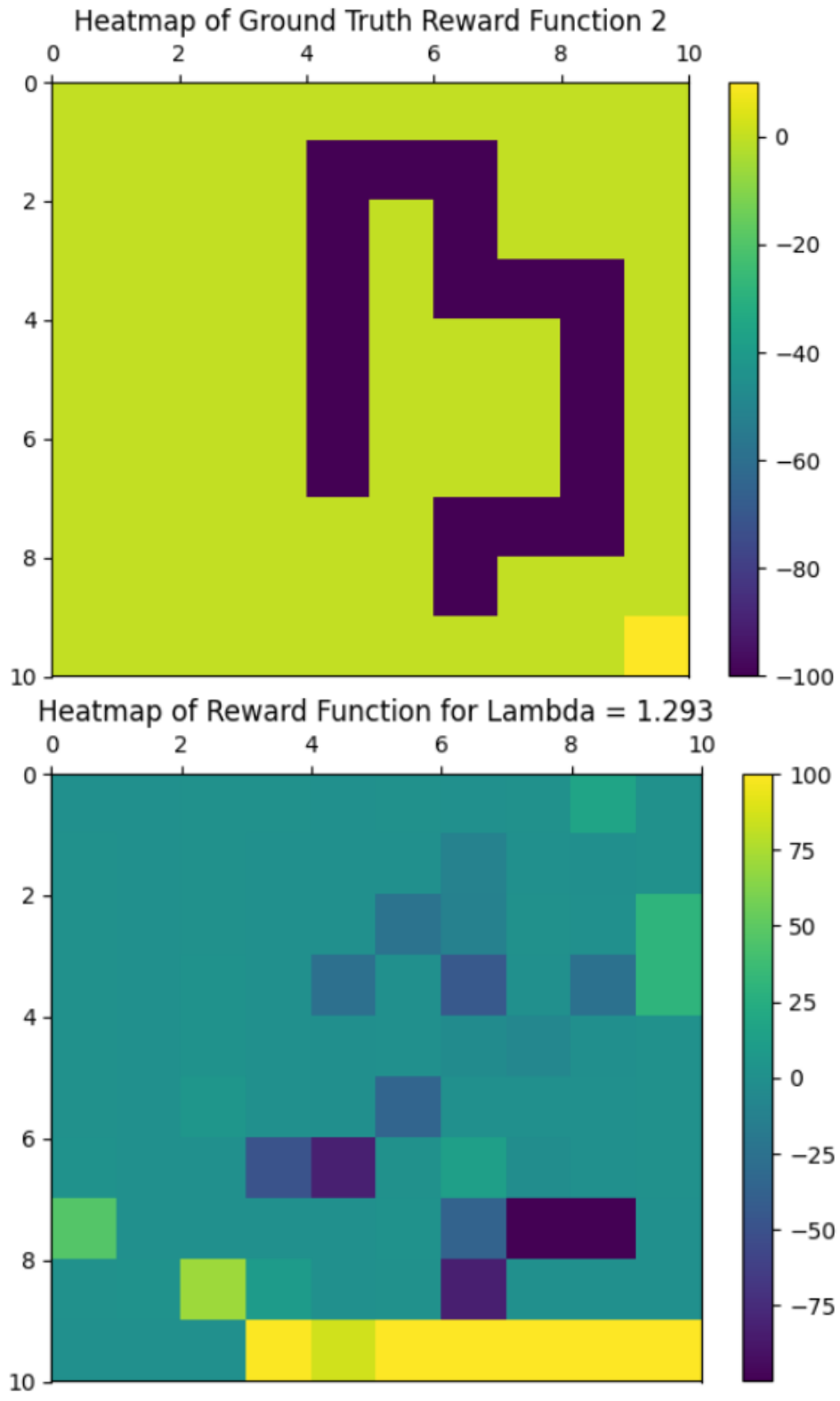

Figure 19: Heatmaps for gound truth and for $\lambda = 1.293$

The ground truth heatmap for the reward function clearly shows the partial loop of negative values and the singular positive reward in the bottom right corner. The extrated reward heatmap,

however, struggles to accurately depict the same. It still understands that there is a positive reward at the bottom of the grid, but it inaccurately gives most of the bottom row high, positive reward values. It does recognize that there is an area that the optimal policy attempts to navigate away from on the right side of the grid, but cannot depict the area clearly.

Q21: Use the extracted reward function from question 20 to compute the optimal values of the states in the 2-D grid. Generate a heat map of the optimal state values.

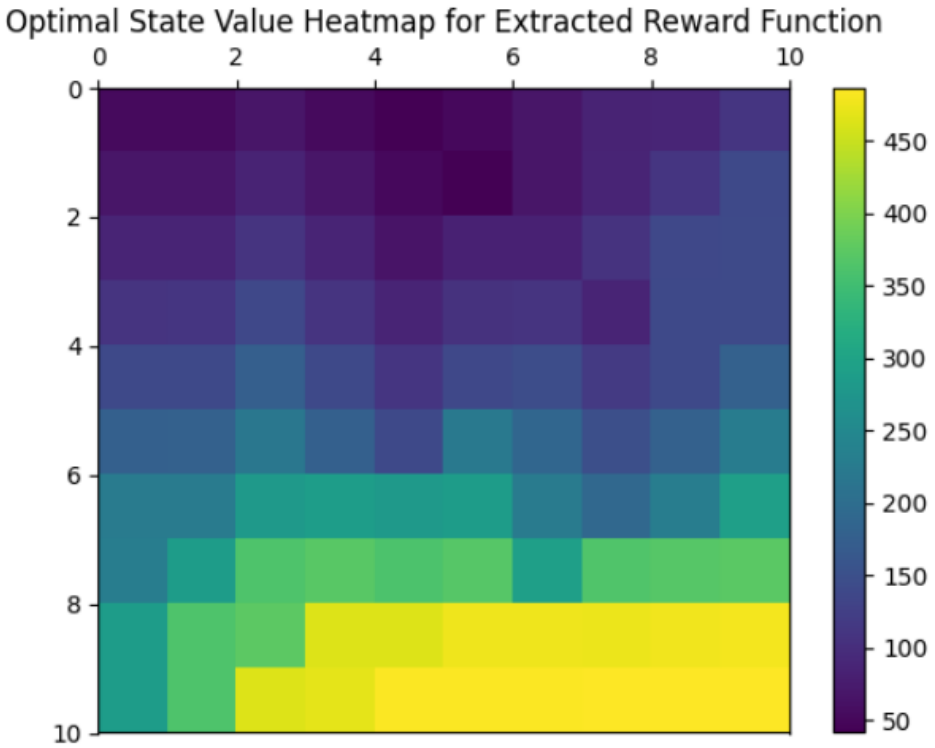

Figure 20: Optimal state values for the extracted reward function with $\lambda = 1.293$.

Q22: Compare the heat maps of Question 7 and Question 21 and provide a brief explanation on their similarities and differences.

Both heatmaps show that there is a strongly positive state value in the bottom right corner and a patch of much lower state values above that around the 2nd and 3rd rows and a distinctly darker state surrounded by a series of low value states in the shape of an "R" or "P".

The most noticeable difference between the two heat maps is the much larger area of highly positive state values on the bottom portion of the grid of the optimal state values for the extracted reward function. This mirrors the heat map of the extracted reward function and was expected based off of that. The distribution of state values for the extracted reward function also struggles to accurately indicate the areas with negative reward values. A roughly similar shape can be seen although there is a lot of noise making it difficult to see.

Q23: Use the extracted reward function from question 20 to compute the optimal policy of the agent. Plot the actions using arrows.

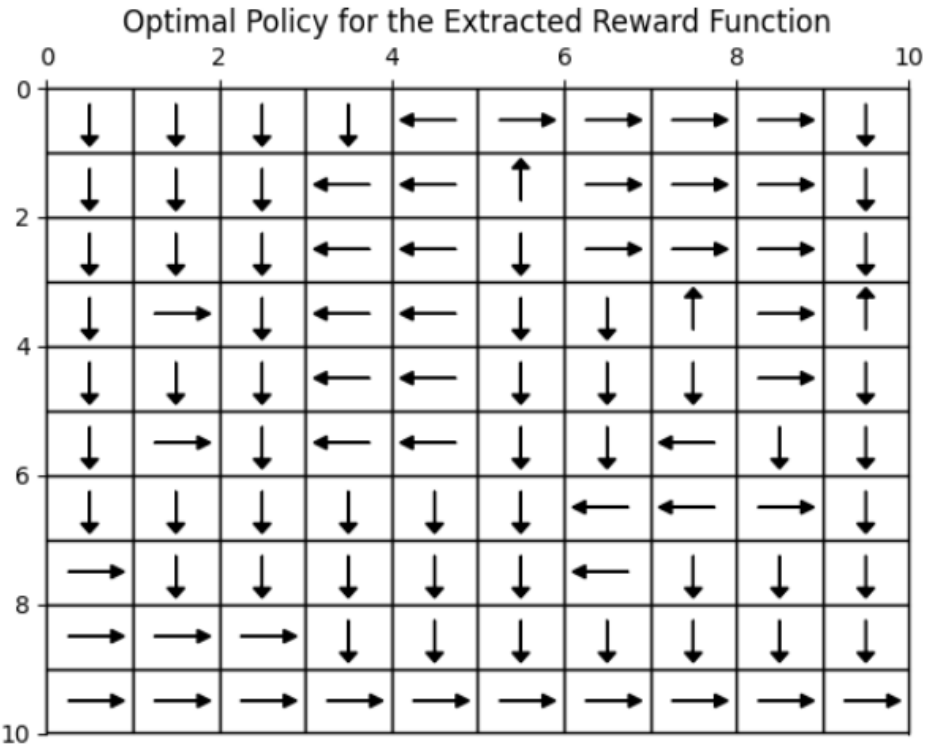

Figure 21: Optimal policy for the extracted reward function with $\lambda = 1.293$.

Q24: Compare the figures of Question 9 and Question 23 and provide a brief explanation on their similarities and differences.

The two optimal policies are fairly similar. They both attempt to move the agent towards the bottom right corner and have a very similar pattern for the area corresponding to the negative reward states of the ground truth.

However, there are multiple states where the two differ and tell the agent to move in different directions. $(0, 3)$, $(3, 1)$, $(3, 9)$, $(5, 1)$, $(5, 8)$, and $(7, 0)$. Unlike the policies for reward function 1, these differences do not all line up with the negative reward values of the ground truth. Instead, they seem to happen at the outer rings of influence of the extracted state values. Figure 22 below shows these different moves highlighted in red.

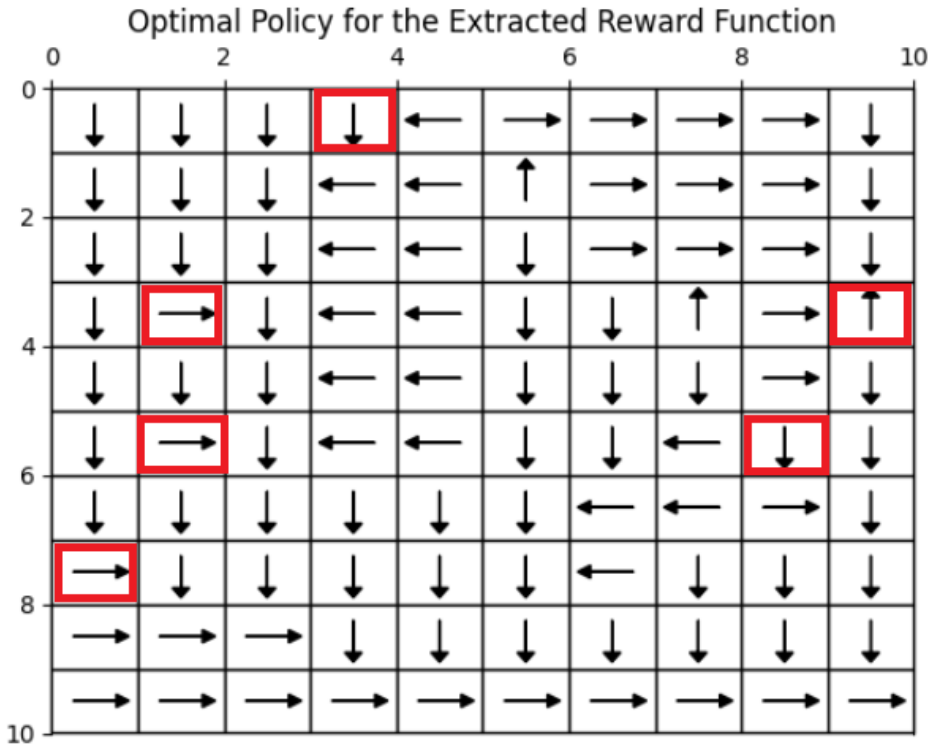

Figure 22: Optimal policy differences.

Q25: From the figure in question 23, you should observe that the optimal policy of the agent has two major discrepencies. Please identify and provde the causes for these two discrepencies. One of the discrepancies can be fixed easily by a slight modification to the value iteration algorithm. Perform this modification and re-run the modified value iteration algorithm to comput the optimal policy of the agent. Also, recompute the maximum accuracy after this modification. Is there a change in maximum accuracy?

First, we will take a look at the state values from the extracted reward function to see why the agent chose each movement at each state.

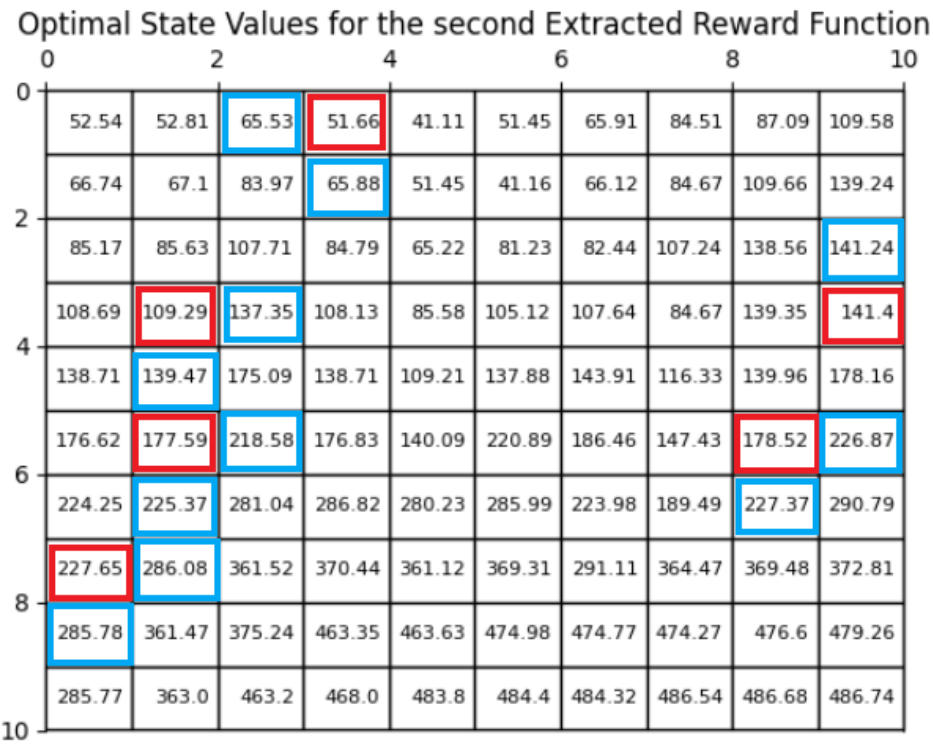

Figure 23: Optimal state values with discrepencies in red and similar state values in blue.

Figure 23 shows the optimal state values that result from using the value iteration algorithm on the second extracted reward function. The states that has the policy discrepencies are still highlighted in red while the blue highlighted states point out the two states that the two policies tried to move to. We can see that for all of these blue states, their values are fairly similar to each other, indicating where the problem comes from. There is an interesting case, however, for the state $(3, 9)$ as the states that the two policies disagree on moving to, are flipped. The optimal policy from the ground truth wants to move down while the optimal policy from the extracted reward function wants to move up. We can see that both of these states share very similar state values.

We can attempt to remedy the optimal policy's indecision by having the algorithm run for more steps. We can see from the figures in Question 2 that as the algorithm runs for more steps, the state values continue to change. In order to have the algorithm run for more steps, we can reduce the size of ϵ until the discrepencies disappear.

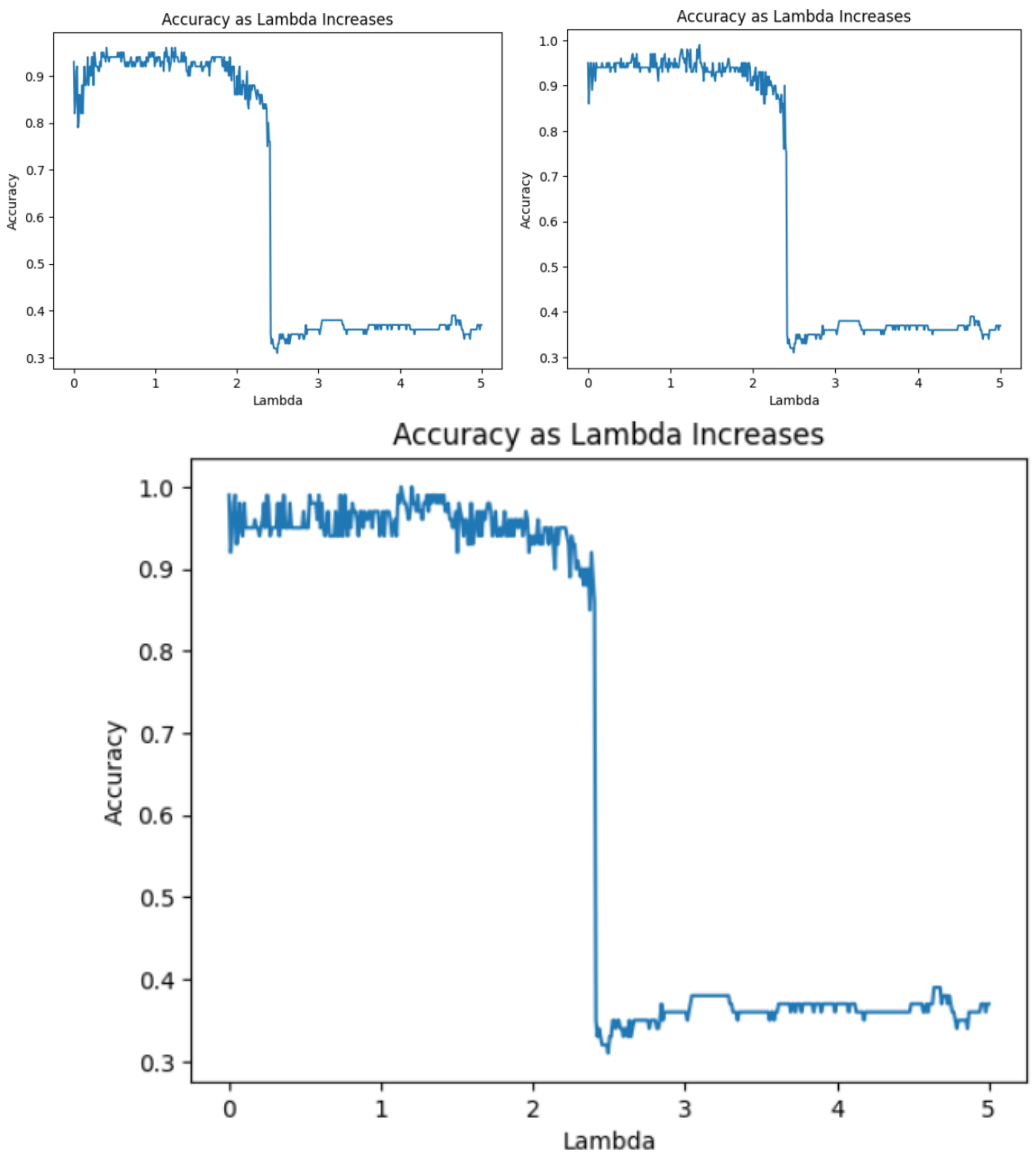

Figure 24: Accuracy plots for $\epsilon = 0.0001, 0.00001, 0.000001$, respectively

The results in Figure 24 show that as ϵ decreases, the accuracy slowly increase.

Table 1: λ and accuracy values for each ϵ value.

ϵ	λ	**Maximum Accuracy**
0.0001	0.401	96%
0.00001	1.353	99%
0.000001	1.132	100%

We can see from Table 1 that the IRL algorithm is able to obtain 100% accuracy when $\epsilon = 0.000001$ meaning that the two optimal policies line up now as shown in Figure 25.

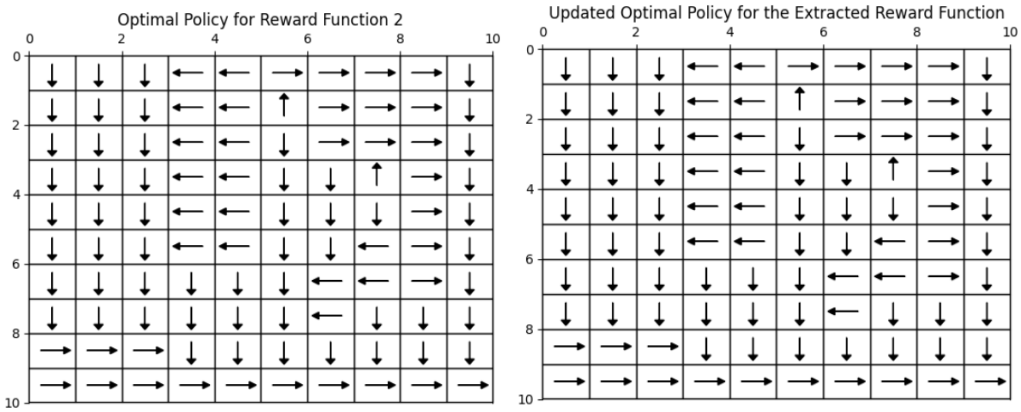

Figure 25: Ground truth optimal policy and updated optimal policy after setting ϵ to 0.000001