23S-EC ENGR-232E-LEC-1 Project 3: Reinforcement and Inverse Reinforcement Learning

ROBERT OZTURK, YAMAN YUCEL, SARAH WILEN

TOTAL POINTS

398 / 400

QUESTION 1

RL 160 pts

1.1 Q1 **10 / 10**

✓ **- 0 pts** *Correct*

1.2 Q2 **40 / 40**

✓ **- 0 pts** *Correct*

1.3 Q3 **5 / 5**

✓ **- 0 pts** *Correct*

1.4 Q4 **15 / 15**

✓ **- 0 pts** *Correct*

1.5 Q5 **20 / 20**

✓ **- 0 pts** *Correct*

1.6 Q6 **10 / 10**

✓ **- 0 pts** *Correct*

1.7 Q7 **20 / 20**

✓ **- 0 pts** *Correct*

1.8 Q8 **20 / 20**

✓ **- 0 pts** *Correct*

1.9 Q9 **20 / 20**

✓ **- 0 pts** *Correct*

QUESTION 2

IRL 240 pts

2.1 Q10 **8 / 10**

✓ **- 2 pts** *partial correct*

2.2 Q11 **30 / 30**

✓ **- 0 pts** *Correct*

2.3 Q12 **5 / 5**

✓ **- 0 pts** *Correct*

2.4 Q13 **15 / 15**

✓ **- 0 pts** *Correct*

2.5 Q14 **10 / 10**

✓ **- 0 pts** *Correct*

2.6 Q15 **10 / 10**

✓ **- 0 pts** *Correct*

2.7 Q16 **10 / 10**

✓ **- 0 pts** *Correct*

2.8 Q17 **10 / 10**

✓ **- 0 pts** *Correct*

2.9 Q18 **30 / 30**

✓ **- 0 pts** *Correct*

2.10 Q19 **5 / 5**

✓ **- 0 pts** *Correct*

2.11 Q20 **15 / 15**

✓ **- 0 pts** *Correct*

2.12 Q21 **10 / 10**

✓ **- 0 pts** *Correct*

2.13 Q22 **10 / 10**

✓ **- 0 pts** *Correct*

2.14 Q23 **10 / 10**

✓ **- 0 pts** *Correct*

2.15 Q24 **10 / 10**

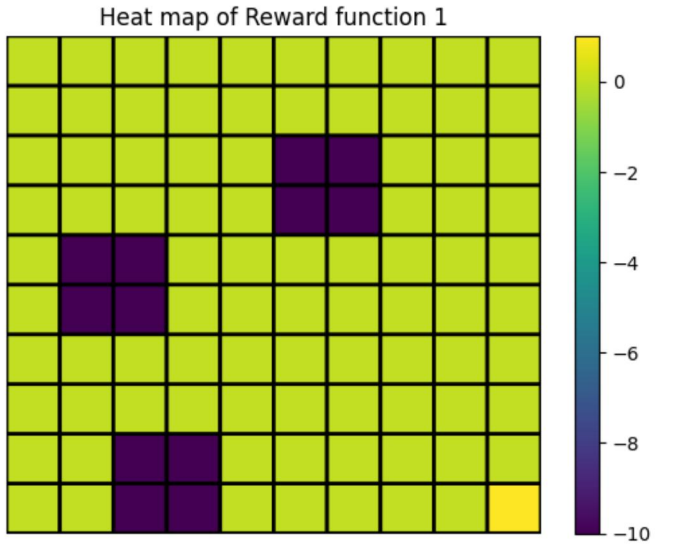
✓ **- 0 pts** *Correct*

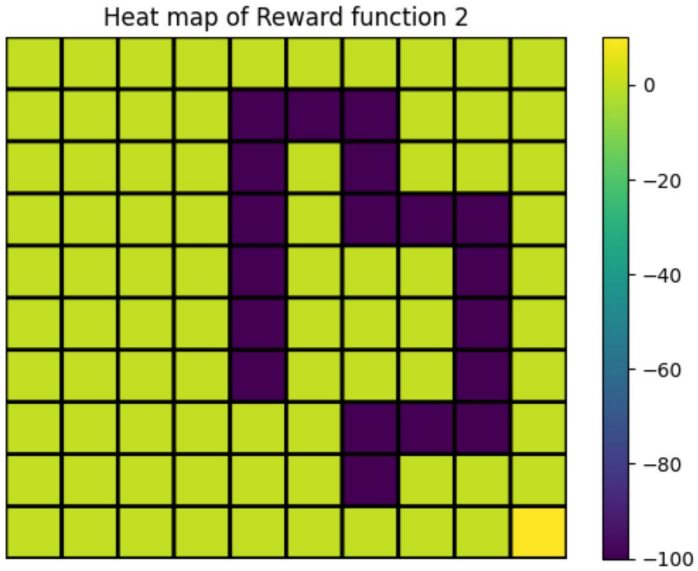
2.16 Q25 **50 / 50**

✓ **- 0 pts** *Correct*

ECE 232E Project 3: Reinforcement Learning and Inverse Reinforcement Learning

Question 1



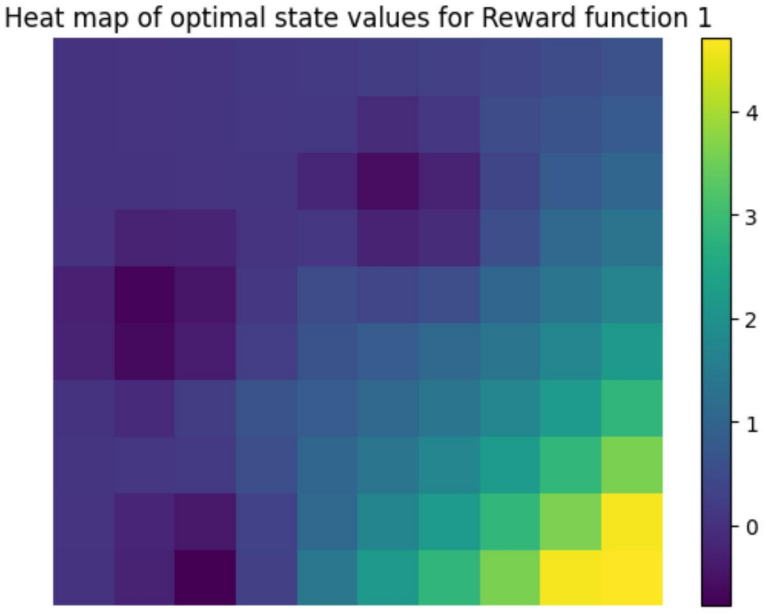


1.1 Q1 **10 / 10**

✓ **- 0 pts** *Correct*

Question 2/3

	0	1	2	3	4	5	6	7	8	9
0	0.0	0.1	0.1	0.1	0.2	0.2	0.3	0.4	0.5	0.6
1	0.0	0.0	0.1	0.1	0.1	-0.1	0.1	0.5	0.6	0.8
2	0.0	0.0	0.0	0.1	-0.2	-0.6	-0.3	0.4	0.8	1.0
3	0.0	-0.2	-0.2	0.1	0.1	-0.3	-0.1	0.5	1.0	1.3
4	-0.3	-0.7	-0.5	0.1	0.5	0.4	0.5	1.0	1.4	1.7
5	-0.3	-0.6	-0.4	0.2	0.6	0.8	1.0	1.4	1.7	2.2
6	0.0	-0.1	0.2	0.6	0.8	1.1	1.4	1.7	2.2	2.8
7	0.1	0.1	0.1	0.5	1.0	1.4	1.7	2.2	2.8	3.6
8	0.0	-0.2	-0.4	0.3	1.1	1.7	2.2	2.8	3.6	4.6
9	0.0	-0.3	-0.8	0.3	1.4	2.2	2.8	3.6	4.6	4.7



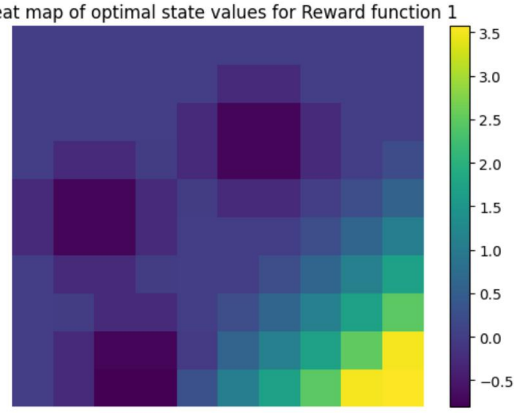
This algorithm took 21 steps to converge.

N=1

	0	1	2	3	4	5	6	7	8	9
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	-0.2	-0.3	-0.0	-0.0	-0.0
2	0.0	0.0	0.0	0.0	-0.2	-0.7	-0.5	-0.3	-0.0	-0.0
3	0.0	-0.2	-0.3	-0.0	-0.3	-0.5	-0.5	-0.3	-0.0	-0.0
4	-0.2	-0.7	-0.5	-0.3	-0.0	-0.3	-0.3	-0.0	-0.0	-0.0
5	-0.3	-0.5	-0.5	-0.3	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0
6	-0.0	-0.3	-0.3	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0
7	-0.0	-0.0	-0.3	-0.3	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0
8	-0.0	-0.3	-0.7	-0.5	-0.3	-0.0	-0.0	-0.0	-0.0	0.9
9	-0.0	-0.3	-0.8	-0.8	-0.3	-0.0	-0.0	-0.0	0.9	1.0

N=6

	0	1	2	3	4	5	6	7	8	9
0	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0
1	-0.0	-0.0	-0.0	-0.0	-0.0	-0.3	-0.3	-0.0	-0.0	-0.0
2	-0.0	-0.0	-0.0	-0.0	-0.3	-0.7	-0.7	-0.3	-0.0	-0.0
3	-0.0	-0.3	-0.3	-0.0	-0.3	-0.7	-0.7	-0.3	-0.0	0.2
4	-0.3	-0.7	-0.7	-0.3	-0.0	-0.3	-0.3	-0.0	0.2	0.6
5	-0.3	-0.7	-0.7	-0.3	-0.0	-0.0	-0.0	0.2	0.6	1.1
6	-0.0	-0.3	-0.3	-0.0	-0.0	-0.0	0.2	0.6	1.1	1.7
7	-0.0	-0.0	-0.3	-0.3	-0.0	0.2	0.6	1.1	1.7	2.5
8	-0.0	-0.3	-0.7	-0.7	-0.0	0.6	1.1	1.7	2.5	3.5
9	-0.0	-0.3	-0.8	-0.8	0.3	1.0	1.7	2.5	3.5	3.6

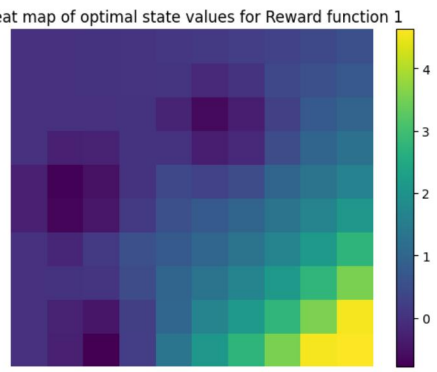


N=11

	0	1	2	3	4	5	6	7	8	9
0	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0	0.1	0.2	0.3
1	-0.0	-0.0	-0.0	-0.0	-0.0	-0.3	-0.2	0.1	0.3	0.5
2	-0.0	-0.0	-0.0	-0.0	-0.3	-0.7	-0.6	0.0	0.5	0.7
3	-0.0	-0.3	-0.3	-0.0	-0.2	-0.6	-0.4	0.2	0.7	1.0
4	-0.3	-0.7	-0.7	-0.2	0.1	0.0	0.2	0.7	1.0	1.4
5	-0.3	-0.7	-0.7	-0.1	0.3	0.5	0.7	1.0	1.4	1.9
6	-0.0	-0.3	-0.1	0.3	0.5	0.7	1.0	1.4	1.9	2.5
7	-0.0	-0.0	-0.2	0.2	0.7	1.0	1.4	1.9	2.5	3.3
8	-0.0	-0.3	-0.7	-0.0	0.7	1.4	1.9	2.5	3.3	4.3
9	-0.0	-0.3	-0.8	-0.0	1.1	1.8	2.5	3.3	4.3	4.4

N=16

	0	1	2	3	4	5	6	7	8	9
0	-0.0	-0.0	0.0	0.0	0.1	0.1	0.2	0.3	0.4	0.5
1	-0.0	-0.0	0.0	0.0	0.0	-0.2	0.0	0.4	0.5	0.7
2	-0.0	-0.0	-0.0	0.0	-0.2	-0.6	-0.3	0.3	0.7	0.9
3	-0.0	-0.3	-0.3	0.0	0.0	-0.3	-0.2	0.5	1.0	1.2
4	-0.3	-0.7	-0.5	0.0	0.4	0.3	0.5	1.0	1.3	1.6
5	-0.3	-0.7	-0.4	0.1	0.6	0.7	1.0	1.3	1.7	2.1
6	-0.0	-0.2	0.1	0.5	0.7	1.0	1.3	1.7	2.1	2.7
7	-0.0	0.0	0.1	0.5	1.0	1.3	1.7	2.1	2.8	3.5
8	-0.0	-0.3	-0.5	0.2	1.0	1.6	2.1	2.8	3.6	4.6
9	-0.0	-0.3	-0.8	0.2	1.3	2.1	2.7	3.5	4.6	4.6



N=21

	0	1	2	3	4	5	6	7	8	9
0	0.0	0.1	0.1	0.1	0.2	0.2	0.3	0.4	0.5	0.6
1	0.0	0.0	0.1	0.1	0.1	-0.1	0.1	0.5	0.6	0.8
2	0.0	0.0	0.0	0.1	-0.2	-0.6	-0.3	0.4	0.8	1.0
3	0.0	-0.2	-0.2	0.1	0.1	-0.3	-0.1	0.5	1.0	1.3
4	-0.3	-0.7	-0.5	0.1	0.5	0.4	0.5	1.0	1.4	1.7
5	-0.3	-0.6	-0.4	0.2	0.6	0.8	1.0	1.4	1.7	2.2
6	0.0	-0.1	0.2	0.6	0.8	1.1	1.4	1.7	2.2	2.8
7	0.1	0.1	0.1	0.5	1.0	1.4	1.7	2.2	2.8	3.6
8	0.0	-0.2	-0.4	0.3	1.1	1.7	2.2	2.8	3.6	4.6
9	0.0	-0.3	-0.8	0.3	1.4	2.2	2.8	3.6	4.6	4.7

From the heat map, we see that the value iteration algorithm almost approximates the reward function over time. You'll notice at the spot where the reward is +1, the state value is the highest. In the states where the reward is -10, the state value is negative.

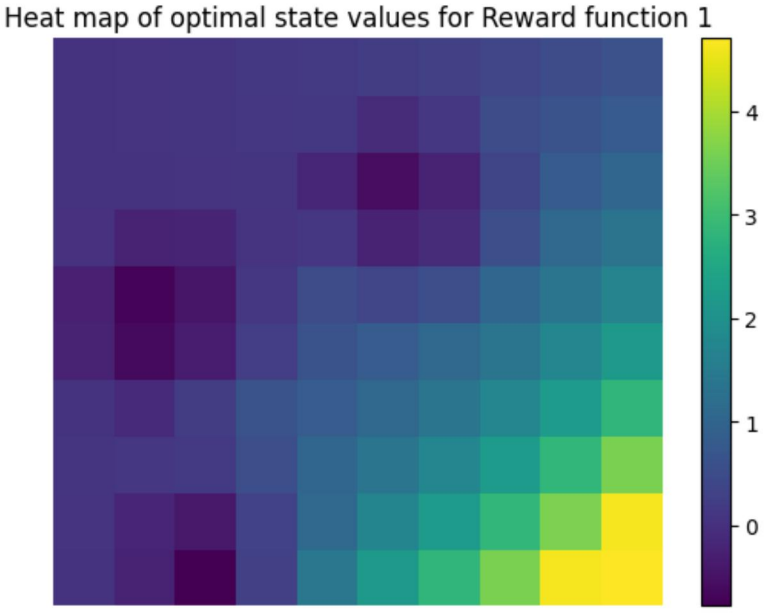
We also observe that the algorithm converges to its final solution very quickly. You'll notice that our optimal state values at just step 6 closely resembles the final solution already.

1.2 Q2 **40 / 40**

✓ **- 0 pts** *Correct*

Question 2/3

	0	1	2	3	4	5	6	7	8	9
0	0.0	0.1	0.1	0.1	0.2	0.2	0.3	0.4	0.5	0.6
1	0.0	0.0	0.1	0.1	0.1	-0.1	0.1	0.5	0.6	0.8
2	0.0	0.0	0.0	0.1	-0.2	-0.6	-0.3	0.4	0.8	1.0
3	0.0	-0.2	-0.2	0.1	0.1	-0.3	-0.1	0.5	1.0	1.3
4	-0.3	-0.7	-0.5	0.1	0.5	0.4	0.5	1.0	1.4	1.7
5	-0.3	-0.6	-0.4	0.2	0.6	0.8	1.0	1.4	1.7	2.2
6	0.0	-0.1	0.2	0.6	0.8	1.1	1.4	1.7	2.2	2.8
7	0.1	0.1	0.1	0.5	1.0	1.4	1.7	2.2	2.8	3.6
8	0.0	-0.2	-0.4	0.3	1.1	1.7	2.2	2.8	3.6	4.6
9	0.0	-0.3	-0.8	0.3	1.4	2.2	2.8	3.6	4.6	4.7



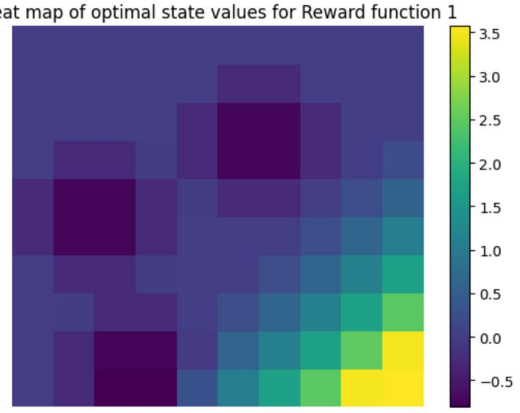
This algorithm took 21 steps to converge.

N=1

	0	1	2	3	4	5	6	7	8	9
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	-0.2	-0.3	-0.0	-0.0	-0.0
2	0.0	0.0	0.0	0.0	-0.2	-0.7	-0.5	-0.3	-0.0	-0.0
3	0.0	-0.2	-0.3	-0.0	-0.3	-0.5	-0.5	-0.3	-0.0	-0.0
4	-0.2	-0.7	-0.5	-0.3	-0.0	-0.3	-0.3	-0.0	-0.0	-0.0
5	-0.3	-0.5	-0.5	-0.3	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0
6	-0.0	-0.3	-0.3	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0
7	-0.0	-0.0	-0.3	-0.3	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0
8	-0.0	-0.3	-0.7	-0.5	-0.3	-0.0	-0.0	-0.0	-0.0	0.9
9	-0.0	-0.3	-0.8	-0.8	-0.3	-0.0	-0.0	-0.0	0.9	1.0

N=6

	0	1	2	3	4	5	6	7	8	9
0	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0
1	-0.0	-0.0	-0.0	-0.0	-0.0	-0.3	-0.3	-0.0	-0.0	-0.0
2	-0.0	-0.0	-0.0	-0.0	-0.3	-0.7	-0.7	-0.3	-0.0	-0.0
3	-0.0	-0.3	-0.3	-0.0	-0.3	-0.7	-0.7	-0.3	-0.0	0.2
4	-0.3	-0.7	-0.7	-0.3	-0.0	-0.3	-0.3	-0.0	0.2	0.6
5	-0.3	-0.7	-0.7	-0.3	-0.0	-0.0	-0.0	0.2	0.6	1.1
6	-0.0	-0.3	-0.3	-0.0	-0.0	-0.0	0.2	0.6	1.1	1.7
7	-0.0	-0.0	-0.3	-0.3	-0.0	0.2	0.6	1.1	1.7	2.5
8	-0.0	-0.3	-0.7	-0.7	-0.0	0.6	1.1	1.7	2.5	3.5
9	-0.0	-0.3	-0.8	-0.8	0.3	1.0	1.7	2.5	3.5	3.6

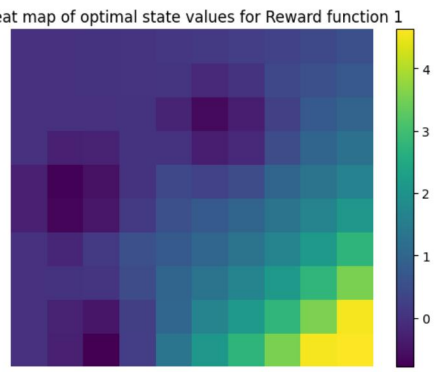


N=11

	0	1	2	3	4	5	6	7	8	9
0	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0	0.1	0.2	0.3
1	-0.0	-0.0	-0.0	-0.0	-0.0	-0.3	-0.2	0.1	0.3	0.5
2	-0.0	-0.0	-0.0	-0.0	-0.3	-0.7	-0.6	0.0	0.5	0.7
3	-0.0	-0.3	-0.3	-0.0	-0.2	-0.6	-0.4	0.2	0.7	1.0
4	-0.3	-0.7	-0.7	-0.2	0.1	0.0	0.2	0.7	1.0	1.4
5	-0.3	-0.7	-0.7	-0.1	0.3	0.5	0.7	1.0	1.4	1.9
6	-0.0	-0.3	-0.1	0.3	0.5	0.7	1.0	1.4	1.9	2.5
7	-0.0	-0.0	-0.2	0.2	0.7	1.0	1.4	1.9	2.5	3.3
8	-0.0	-0.3	-0.7	-0.0	0.7	1.4	1.9	2.5	3.3	4.3
9	-0.0	-0.3	-0.8	-0.0	1.1	1.8	2.5	3.3	4.3	4.4

N=16

	0	1	2	3	4	5	6	7	8	9
0	-0.0	-0.0	0.0	0.0	0.1	0.1	0.2	0.3	0.4	0.5
1	-0.0	-0.0	0.0	0.0	0.0	-0.2	0.0	0.4	0.5	0.7
2	-0.0	-0.0	-0.0	0.0	-0.2	-0.6	-0.3	0.3	0.7	0.9
3	-0.0	-0.3	-0.3	0.0	0.0	-0.3	-0.2	0.5	1.0	1.2
4	-0.3	-0.7	-0.5	0.0	0.4	0.3	0.5	1.0	1.3	1.6
5	-0.3	-0.7	-0.4	0.1	0.6	0.7	1.0	1.3	1.7	2.1
6	-0.0	-0.2	0.1	0.5	0.7	1.0	1.3	1.7	2.1	2.7
7	-0.0	0.0	0.1	0.5	1.0	1.3	1.7	2.1	2.8	3.5
8	-0.0	-0.3	-0.5	0.2	1.0	1.6	2.1	2.8	3.6	4.6
9	-0.0	-0.3	-0.8	0.2	1.3	2.1	2.7	3.5	4.6	4.6



N=21

	0	1	2	3	4	5	6	7	8	9
0	0.0	0.1	0.1	0.1	0.2	0.2	0.3	0.4	0.5	0.6
1	0.0	0.0	0.1	0.1	0.1	-0.1	0.1	0.5	0.6	0.8
2	0.0	0.0	0.0	0.1	-0.2	-0.6	-0.3	0.4	0.8	1.0
3	0.0	-0.2	-0.2	0.1	0.1	-0.3	-0.1	0.5	1.0	1.3
4	-0.3	-0.7	-0.5	0.1	0.5	0.4	0.5	1.0	1.4	1.7
5	-0.3	-0.6	-0.4	0.2	0.6	0.8	1.0	1.4	1.7	2.2
6	0.0	-0.1	0.2	0.6	0.8	1.1	1.4	1.7	2.2	2.8
7	0.1	0.1	0.1	0.5	1.0	1.4	1.7	2.2	2.8	3.6
8	0.0	-0.2	-0.4	0.3	1.1	1.7	2.2	2.8	3.6	4.6
9	0.0	-0.3	-0.8	0.3	1.4	2.2	2.8	3.6	4.6	4.7

From the heat map, we see that the value iteration algorithm almost approximates the reward function over time. You'll notice at the spot where the reward is +1, the state value is the highest. In the states where the reward is -10, the state value is negative.

We also observe that the algorithm converges to its final solution very quickly. You'll notice that our optimal state values at just step 6 closely resembles the final solution already.

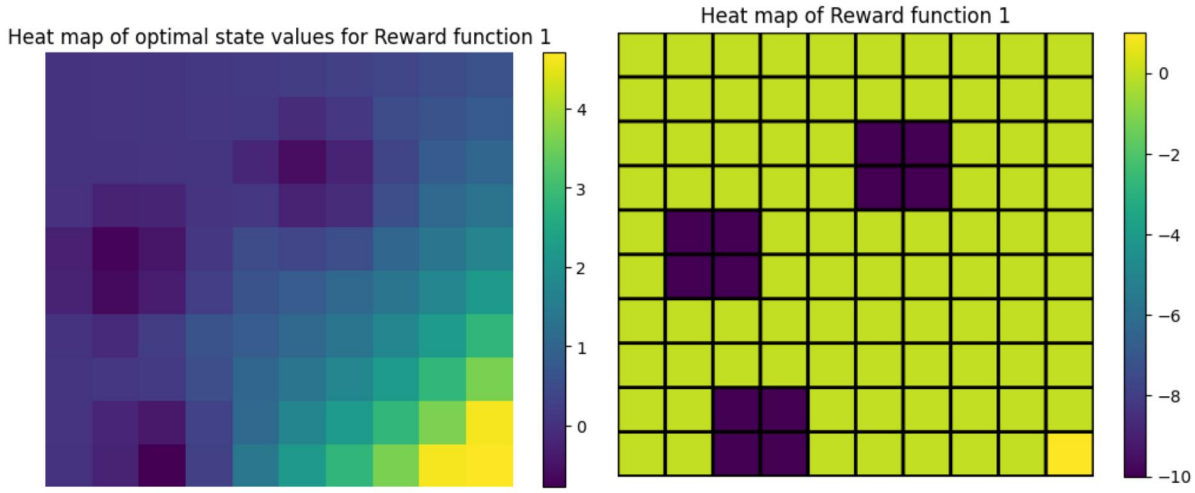
1.3 Q3 **5 / 5**

✓ **- 0 pts** *Correct*

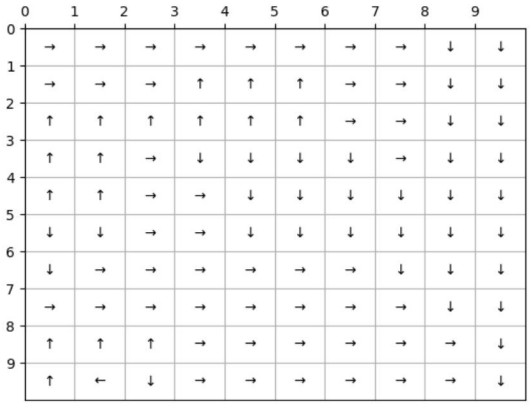
Question 4

We notice that the distribution of the optimal state values attempts to “approximate” the reward function. As you’ll notice, the areas where the reward is highest, the optimal state values are also the highest. The areas where the reward is the lowest (the most negative), the optimal state values are also the lowest (the most negative). This makes sense because iteratively, the agent has interpreted the reward to result in optimal state values that approximate the reward function.

We also notice that while the reward function has a very definite space where the value is -10, 0, or 1, but the optimal state value has values that are “more continuous”. For example, around the largest reward value on the bottom right corner, we get a gradient. This is due to the discount factor, which introduces a gradient decrease between states and surrounding neighbors that have a very high or very low value.



Question 5



The optimal policy does match our intuition. The highest reward is on the bottom corner and all arrows point towards that corner while avoiding the three states that have the very negative reward.

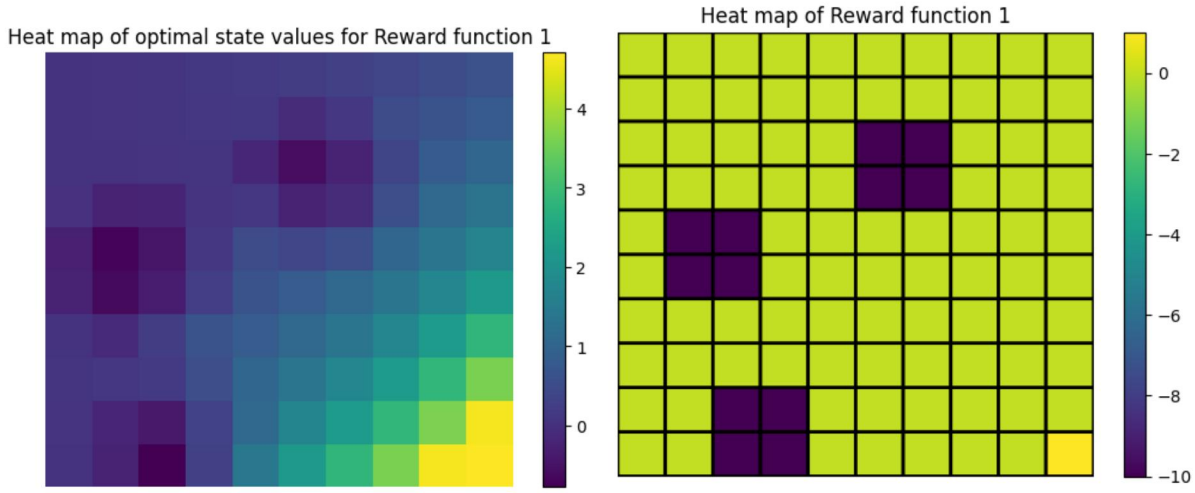
1.4 Q4 **15 / 15**

✓ **- 0 pts** *Correct*

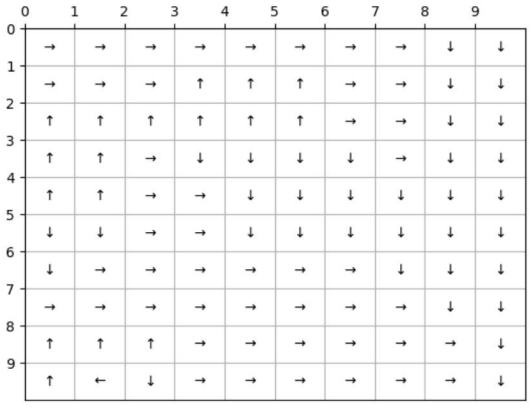
Question 4

We notice that the distribution of the optimal state values attempts to “approximate” the reward function. As you’ll notice, the areas where the reward is highest, the optimal state values are also the highest. The areas where the reward is the lowest (the most negative), the optimal state values are also the lowest (the most negative). This makes sense because iteratively, the agent has interpreted the reward to result in optimal state values that approximate the reward function.

We also notice that while the reward function has a very definite space where the value is -10, 0, or 1, but the optimal state value has values that are “more continuous”. For example, around the largest reward value on the bottom right corner, we get a gradient. This is due to the discount factor, which introduces a gradient decrease between states and surrounding neighbors that have a very high or very low value.



Question 5



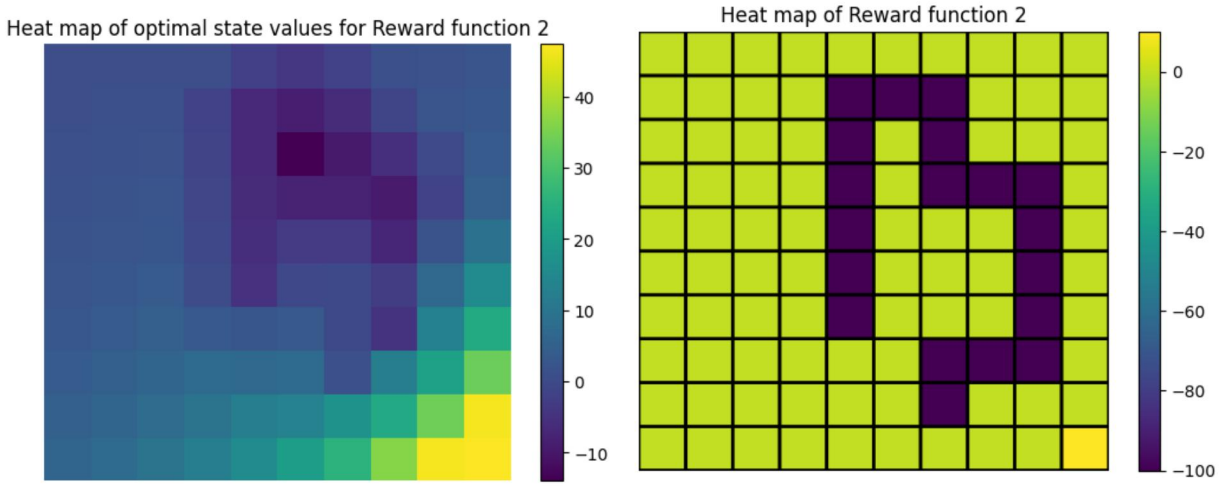
The optimal policy does match our intuition. The highest reward is on the bottom corner and all arrows point towards that corner while avoiding the three states that have the very negative reward.

Yes, the agent can compute the optimal action to take at each state by observing the optimal values of neighboring states. For example, starting at the top left corner, the next highest reward is to the right, so the action is to the right. After that, the next highest reward is again to the right. You'll notice that it takes the path to maximize the reward. The best action is towards the neighbor of the highest optimal value.

Question 6/7

Converges in 31 steps.

	0	1	2	3	4	5	6	7	8	9
0	0.6	0.8	0.8	0.5	-2.4	-4.2	-1.9	1.1	1.6	2.0
1	0.8	1.0	1.1	-1.9	-6.7	-8.7	-6.4	-1.3	1.9	2.6
2	1.1	1.3	1.5	-1.6	-6.7	-13.9	-9.6	-5.5	-0.1	3.4
3	1.4	1.7	1.9	-1.2	-6.3	-8.0	-7.9	-9.4	-1.9	4.4
4	1.7	2.2	2.6	-0.7	-5.8	-3.3	-3.2	-7.4	1.7	9.2
5	2.2	2.8	3.4	-0.0	-5.1	-0.5	-0.5	-3.0	6.6	15.4
6	2.8	3.6	4.5	3.0	2.5	2.9	-0.5	-4.9	12.7	23.3
7	3.6	4.5	5.8	7.3	6.7	7.2	0.9	12.4	21.2	33.5
8	4.6	5.8	7.4	9.4	12.0	12.9	17.1	23.0	33.8	46.5
9	5.7	7.3	9.4	12.0	15.5	19.8	25.5	36.2	46.6	47.3



As we noticed with the previous reward function, the optimal state value map attempts to approximate the reward function. This is what we intuitively expect. The area with the highest reward is on the bottom right corner in both the reward function and the optimal state value. The areas where the reward is lowest (purple in the reward function), the optimal state values are also the lowest (purple in the optimal state value). This makes sense because iteratively, the agent has interpreted the reward to result in optimal state values that approximate the reward function.

1.5 Q5 **20 / 20**

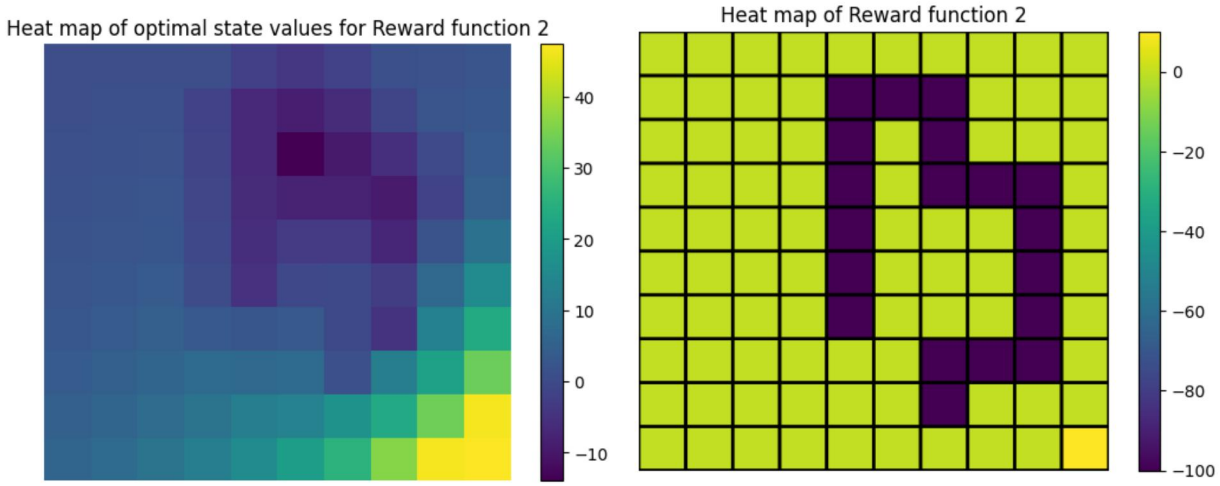
✓ **- 0 pts** *Correct*

Yes, the agent can compute the optimal action to take at each state by observing the optimal values of neighboring states. For example, starting at the top left corner, the next highest reward is to the right, so the action is to the right. After that, the next highest reward is again to the right. You'll notice that it takes the path to maximize the reward. The best action is towards the neighbor of the highest optimal value.

Question 6/7

Converges in 31 steps.

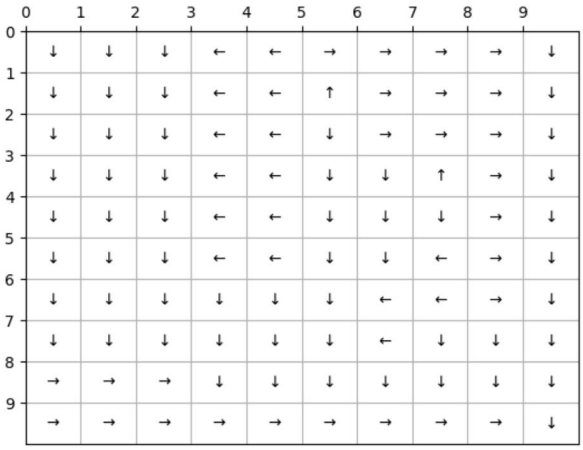
	0	1	2	3	4	5	6	7	8	9
0	0.6	0.8	0.8	0.5	-2.4	-4.2	-1.9	1.1	1.6	2.0
1	0.8	1.0	1.1	-1.9	-6.7	-8.7	-6.4	-1.3	1.9	2.6
2	1.1	1.3	1.5	-1.6	-6.7	-13.9	-9.6	-5.5	-0.1	3.4
3	1.4	1.7	1.9	-1.2	-6.3	-8.0	-7.9	-9.4	-1.9	4.4
4	1.7	2.2	2.6	-0.7	-5.8	-3.3	-3.2	-7.4	1.7	9.2
5	2.2	2.8	3.4	-0.0	-5.1	-0.5	-0.5	-3.0	6.6	15.4
6	2.8	3.6	4.5	3.0	2.5	2.9	-0.5	-4.9	12.7	23.3
7	3.6	4.5	5.8	7.3	6.7	7.2	0.9	12.4	21.2	33.5
8	4.6	5.8	7.4	9.4	12.0	12.9	17.1	23.0	33.8	46.5
9	5.7	7.3	9.4	12.0	15.5	19.8	25.5	36.2	46.6	47.3



As we noticed with the previous reward function, the optimal state value map attempts to approximate the reward function. This is what we intuitively expect. The area with the highest reward is on the bottom right corner in both the reward function and the optimal state value. The areas where the reward is lowest (purple in the reward function), the optimal state values are also the lowest (purple in the optimal state value). This makes sense because iteratively, the agent has interpreted the reward to result in optimal state values that approximate the reward function.

We also notice that while the reward function has very definite space where the value is -100, 0, or 10, but the optimal state values has values that are “more continuous”. For example, around the largest reward value on the bottom right corner, we get a gradient. This is due to the discount factor, which introduces a gradient decrease between states and surrounding neighbors that have a very high or very low value.

Question 8



This is what we expect. Similar to the previous reward, the actions point towards the bottom right corner, which has the highest reward and points actions away from the low reward area (the area around columns 4-8).

Question 9

Reward function 1

16 steps to converge

1.6 Q6 **10 / 10**

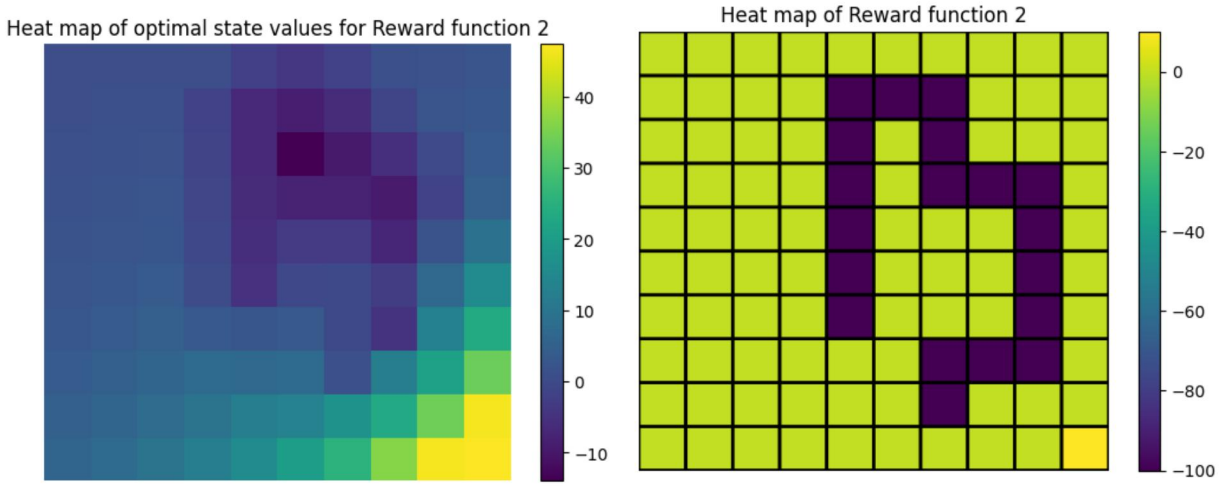
✓ **- 0 pts** *Correct*

Yes, the agent can compute the optimal action to take at each state by observing the optimal values of neighboring states. For example, starting at the top left corner, the next highest reward is to the right, so the action is to the right. After that, the next highest reward is again to the right. You'll notice that it takes the path to maximize the reward. The best action is towards the neighbor of the highest optimal value.

Question 6/7

Converges in 31 steps.

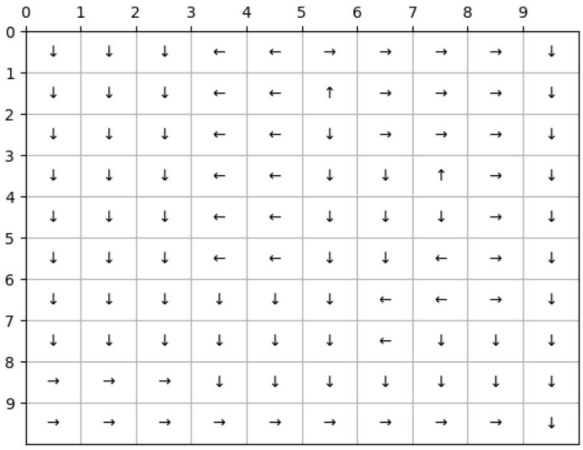
	0	1	2	3	4	5	6	7	8	9
0	0.6	0.8	0.8	0.5	-2.4	-4.2	-1.9	1.1	1.6	2.0
1	0.8	1.0	1.1	-1.9	-6.7	-8.7	-6.4	-1.3	1.9	2.6
2	1.1	1.3	1.5	-1.6	-6.7	-13.9	-9.6	-5.5	-0.1	3.4
3	1.4	1.7	1.9	-1.2	-6.3	-8.0	-7.9	-9.4	-1.9	4.4
4	1.7	2.2	2.6	-0.7	-5.8	-3.3	-3.2	-7.4	1.7	9.2
5	2.2	2.8	3.4	-0.0	-5.1	-0.5	-0.5	-3.0	6.6	15.4
6	2.8	3.6	4.5	3.0	2.5	2.9	-0.5	-4.9	12.7	23.3
7	3.6	4.5	5.8	7.3	6.7	7.2	0.9	12.4	21.2	33.5
8	4.6	5.8	7.4	9.4	12.0	12.9	17.1	23.0	33.8	46.5
9	5.7	7.3	9.4	12.0	15.5	19.8	25.5	36.2	46.6	47.3



As we noticed with the previous reward function, the optimal state value map attempts to approximate the reward function. This is what we intuitively expect. The area with the highest reward is on the bottom right corner in both the reward function and the optimal state value. The areas where the reward is lowest (purple in the reward function), the optimal state values are also the lowest (purple in the optimal state value). This makes sense because iteratively, the agent has interpreted the reward to result in optimal state values that approximate the reward function.

We also notice that while the reward function has very definite space where the value is -100, 0, or 10, but the optimal state values has values that are “more continuous”. For example, around the largest reward value on the bottom right corner, we get a gradient. This is due to the discount factor, which introduces a gradient decrease between states and surrounding neighbors that have a very high or very low value.

Question 8



This is what we expect. Similar to the previous reward, the actions point towards the bottom right corner, which has the highest reward and points actions away from the low reward area (the area around columns 4-8).

Question 9

Reward function 1

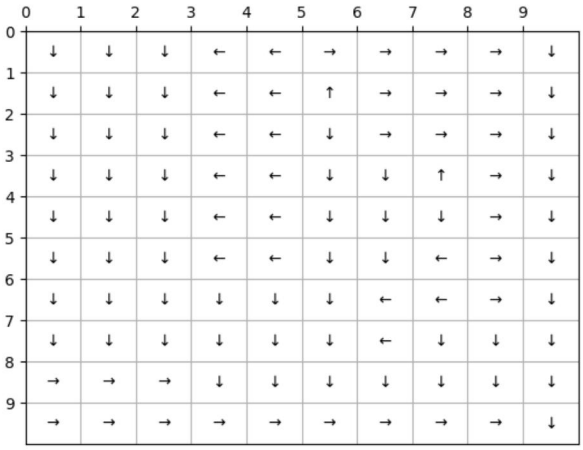
16 steps to converge

1.7 Q7 **20 / 20**

✓ **- 0 pts** *Correct*

We also notice that while the reward function has very definite space where the value is -100, 0, or 10, but the optimal state values has values that are “more continuous”. For example, around the largest reward value on the bottom right corner, we get a gradient. This is due to the discount factor, which introduces a gradient decrease between states and surrounding neighbors that have a very high or very low value.

Question 8



This is what we expect. Similar to the previous reward, the actions point towards the bottom right corner, which has the highest reward and points actions away from the low reward area (the area around columns 4-8).

Question 9

Reward function 1

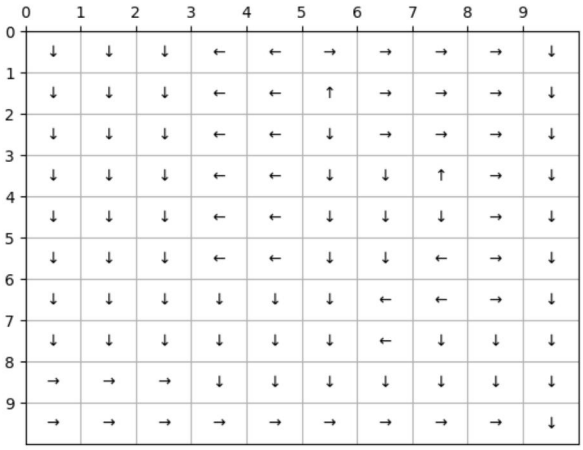
16 steps to converge

1.8 Q8 **20 / 20**

✓ **- 0 pts** *Correct*

We also notice that while the reward function has very definite space where the value is -100, 0, or 10, but the optimal state values has values that are “more continuous”. For example, around the largest reward value on the bottom right corner, we get a gradient. This is due to the discount factor, which introduces a gradient decrease between states and surrounding neighbors that have a very high or very low value.

Question 8



This is what we expect. Similar to the previous reward, the actions point towards the bottom right corner, which has the highest reward and points actions away from the low reward area (the area around columns 4-8).

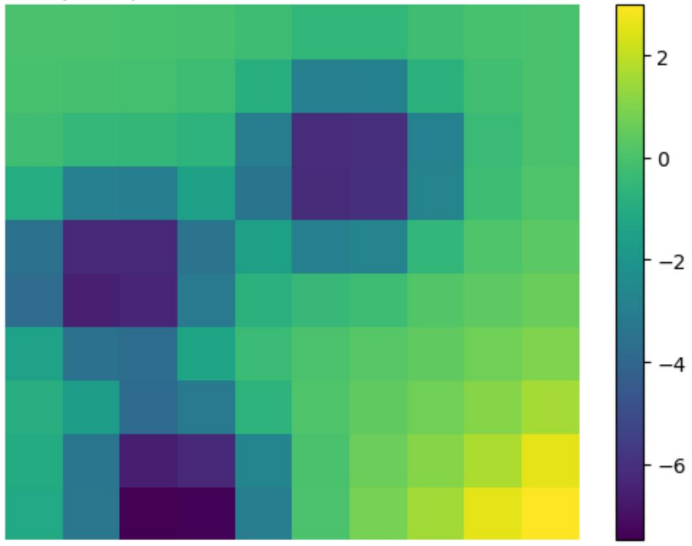
Question 9

Reward function 1

16 steps to converge

	0	1	2	3	4	5	6	7	8	9
0	-0.0	-0.0	-0.0	-0.1	-0.2	-0.6	-0.6	-0.2	-0.1	-0.0
1	-0.1	-0.1	-0.1	-0.2	-0.9	-3.0	-2.9	-0.8	-0.2	-0.0
2	-0.2	-0.5	-0.6	-0.7	-3.1	-6.1	-6.1	-2.8	-0.4	-0.0
3	-1.0	-3.0	-3.0	-1.5	-3.5	-6.2	-6.0	-2.8	-0.3	0.1
4	-3.6	-6.3	-6.3	-3.5	-1.5	-3.0	-2.8	-0.6	0.1	0.3
5	-3.8	-6.6	-6.4	-3.2	-0.8	-0.5	-0.3	0.1	0.4	0.6
6	-1.5	-3.6	-3.8	-1.4	-0.4	-0.0	0.2	0.4	0.7	0.9
7	-0.9	-1.7	-3.9	-3.2	-0.7	0.1	0.4	0.7	1.1	1.5
8	-1.1	-3.4	-6.6	-6.3	-2.7	-0.1	0.6	1.1	1.7	2.5
9	-1.1	-3.3	-7.5	-7.4	-3.1	-0.0	0.8	1.5	2.5	3.0

Heat map of optimal state values for Reward function 1

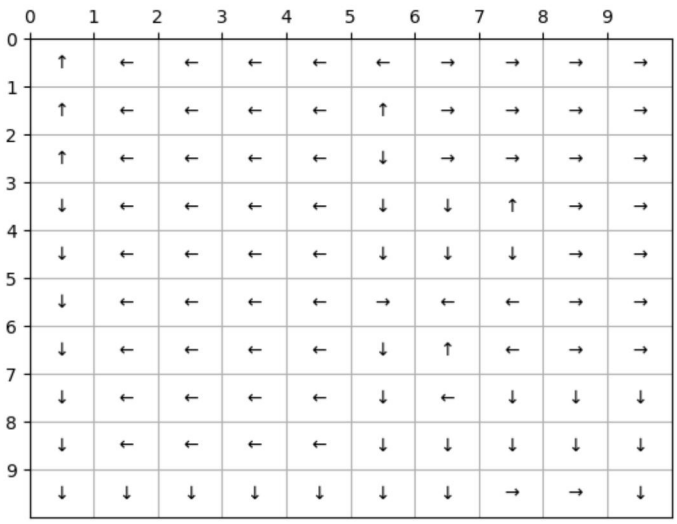


	0	1	2	3	4	5	6	7	8	9
0	↑	←	←	←	←	←	→	→	→	→
1	↑	↑	↑	↑	↑	↑	↑	→	→	→
2	↑	↑	↑	↑	←	↑	→	→	→	↓
3	↑	↑	↑	↑	←	↓	→	→	→	↓
4	↑	↑	↑	↑	↓	↓	↓	↓	↓	↓
5	↓	↓	→	→	↓	↓	↓	↓	↓	↓
6	↓	←	→	→	→	→	→	↓	↓	↓
7	←	←	←	→	→	→	→	→	↓	↓
8	↑	←	←	→	→	→	→	→	→	↓
9	↑	↓	↓	↓	↓	→	→	→	→	↓

Reward function 2

22 steps to converge

	0	1	2	3	4	5	6	7	8	9
0	-0.2	-0.6	-2.5	-10.7	-38.9	-53.7	-41.0	-13.1	-4.3	-2.0
1	-0.3	-1.0	-5.1	-30.8	-67.6	-84.7	-74.0	-39.3	-11.2	-3.2
2	-0.4	-1.1	-6.0	-34.7	-76.4	-117.6	-96.8	-74.1	-37.9	-10.1
3	-0.4	-1.2	-6.2	-34.9	-74.3	-94.6	-93.5	-92.5	-68.9	-32.8
4	-0.4	-1.1	-6.0	-34.2	-70.1	-69.0	-66.8	-86.9	-73.3	-36.5
5	-0.3	-1.0	-5.6	-32.7	-65.4	-59.7	-49.6	-65.8	-71.3	-36.5
6	-0.2	-0.8	-4.5	-27.3	-45.3	-56.0	-62.3	-83.1	-71.1	-35.0
7	-0.1	-0.4	-1.7	-8.0	-30.7	-47.5	-72.9	-77.0	-59.4	-24.0
8	-0.0	-0.1	-0.6	-2.3	-8.6	-30.4	-43.9	-48.0	-20.0	11.8
9	-0.0	-0.0	-0.2	-0.7	-2.3	-7.8	-24.4	-2.5	14.7	23.1

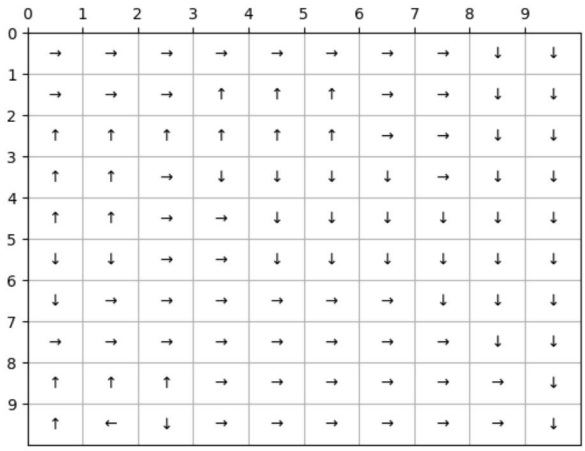


The wind parameter gives rise to exploration. It allows for the agent ot gather new info and try new actions that may give better outcomes. Without the wind (random) factor, we are limited to specific strategies. The wind parameter may help avoid local optimum by allowing us a probability to move in a random direction.

The wind parameter increasing to 0.6 means there is a higher chance for the agent to move in a direction other than the intended action. We see this resulting in quicker convergence for both reward function 1 and reward function 2. However, we also see that the overall optimal values are lower than when w=0.2.

Another interesting point is the higher rewards are a lot more contrasted and distinct than when w=0.2. This could suggest the higher wind function allows for a better approximation of the reward function. With this said, we notice that the optimal policy results in some actions that are stuck. For example, at the bottom-left, we can move up twice, but then will be stuck. This must be where the exploration, w, term comes in to allow for random movement in a direction that is not intentional.

After examining wind parameters between 0.1 and 0.6, the parameter that allows for our agent to reach the reward without getting stuck is shown to be 0.1. We will continue with this value. Refer to the following optimal policy map for reward function 1. We see that there are no places our agent will get stuck.


0 1 2 3 4 5 6 7 8 9
0 → → → → → → → → ↓ ↓
1 → → → ↑ ↑ ↑ → → ↓ ↓
2 ↑ ↑ ↑ ↑ ↑ ↑ → → ↓ ↓
3 ↑ ↑ → ↓ ↓ ↓ ↓ → ↓ ↓
4 ↑ ↑ → → ↓ ↓ ↓ ↓ ↓ ↓
5 ↓ ↓ → → ↓ ↓ ↓ ↓ ↓ ↓
6 ↓ → → → → → → ↓ ↓ ↓
7 → → → → → → → → ↓ ↓
8 ↑ ↑ ↑ → → → → → → ↓
9 ↑ ← ↓ → → → → → → ↓

1.9 Q9 **20 / 20**

✓ **- 0 pts** *Correct*

Question 10:

(We have modified plotting functions for next sections)

We have written c, x, b and D using LaTeX since it is a burden in Google Doc.

LaTeX script:

```
c = $\begin{bmatrix}
        1_{|S| \times 1} \\
        -\lambda_{|S| \times 1} \\
        0_{|S| \times 1}
        \end{bmatrix}$,
x = $\begin{bmatrix}
        t \\
        u \\
        R
        \end{bmatrix}$,
b = $\begin{bmatrix}
        0_{|2(|\mathcal{A} \backslash a_1|-1)|S|| \times 1} \\
        0_{|S| \times 1} \\
        0_{|S| \times 1} \\
        (R_{max})_{|S| \times 1}
        (R_{max})_{|S| \times 1}
        \end{bmatrix}$
\\
D = $\begin{bmatrix}
        I_{|S| \times|S|} & 0 &(P_a-P_{a_1})(I-\gamma P_{a_1})^{-1}\\
        0 & 0 & (P_a-P_{a_1})(I-\gamma P_{a_1})^{-1}\\
        0 & -I_{|S| \times|S|} & I_{|S| \times|S|} \\
        0 & -I_{|S| \times|S|} & -I_{|S| \times|S|} \\
        0 & 0 & I_{|S| \times|S|} \\
        0 & 0 & -I_{|S| \times|S|}
        \end{bmatrix}$
```

Output:

$$c = \begin{bmatrix} 1_{|S|\times 1} \\ -\lambda_{|S|\times 1} \\ 0_{|S|\times 1} \end{bmatrix}, x = \begin{bmatrix} t \\ u \\ R \end{bmatrix}, b = \begin{bmatrix} 0_{|2(|\mathcal{A}\setminus a_1|-1)|S||\times 1} \\ 0_{|S|\times 1} \\ 0_{|S|\times 1} \\ (R_{max})_{|S|\times 1}(R_{max})_{|S|\times 1} \end{bmatrix}$$

$$D = \begin{bmatrix} I_{|S|\times|S|} & 0 & (P_a - P_{a_1})(I-\gamma P_{a_1})^{-1} \\ 0 & 0 & (P_a - P_{a_1})(I-\gamma P_{a_1})^{-1} \\ 0 & -I_{|S|\times|S|} & I_{|S|\times|S|} \\ 0 & -I_{|S|\times|S|} & -I_{|S|\times|S|} \\ 0 & 0 & I_{|S|\times|S|} \\ 0 & 0 & -I_{|S|\times|S|} \end{bmatrix}$$

Question 11:

LP formulation found in Q10 can be easily solved using a LP solver, only catch is that those solvers try to minimize, so c should be multiplied with -1 to solve our maximization problem.

Equally spaced 500 **λ** values between 0 and 5 are created to optimize the LP solution. Following plot is obtained:

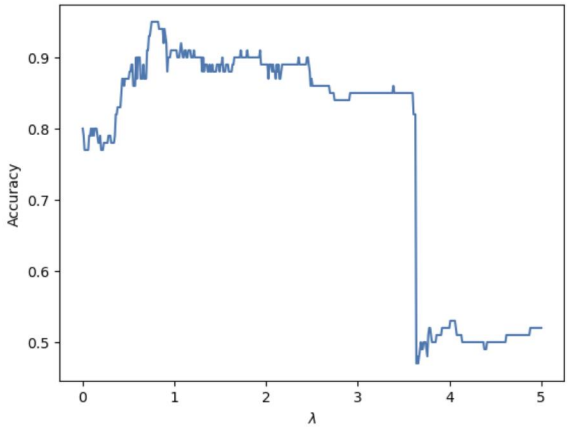


Figure Q11: **λ** vs Accuracy using Reward Function 1

Question 12:

From the figure Q11, we can see that optimal **λ** is 0.7515 with accuracy 0.95.

2.1 Q10 **8 / 10**

✓ **- 2 pts** *partial correct*

Output:

$$c = \begin{bmatrix} 1_{|S|\times 1} \\ -\lambda_{|S|\times 1} \\ 0_{|S|\times 1} \end{bmatrix}, x = \begin{bmatrix} t \\ u \\ R \end{bmatrix}, b = \begin{bmatrix} 0_{|2(|\mathcal{A}\setminus a_1|-1)|S||\times 1} \\ 0_{|S|\times 1} \\ 0_{|S|\times 1} \\ (R_{max})_{|S|\times 1}(R_{max})_{|S|\times 1} \end{bmatrix}$$

$$D = \begin{bmatrix} I_{|S|\times|S|} & 0 & (P_a - P_{a_1})(I-\gamma P_{a_1})^{-1} \\ 0 & 0 & (P_a - P_{a_1})(I-\gamma P_{a_1})^{-1} \\ 0 & -I_{|S|\times|S|} & I_{|S|\times|S|} \\ 0 & -I_{|S|\times|S|} & -I_{|S|\times|S|} \\ 0 & 0 & I_{|S|\times|S|} \\ 0 & 0 & -I_{|S|\times|S|} \end{bmatrix}$$

Question 11:

LP formulation found in Q10 can be easily solved using a LP solver, only catch is that those solvers try to minimize, so c should be multiplied with -1 to solve our maximization problem.

Equally spaced 500 **λ** values between 0 and 5 are created to optimize the LP solution. Following plot is obtained:

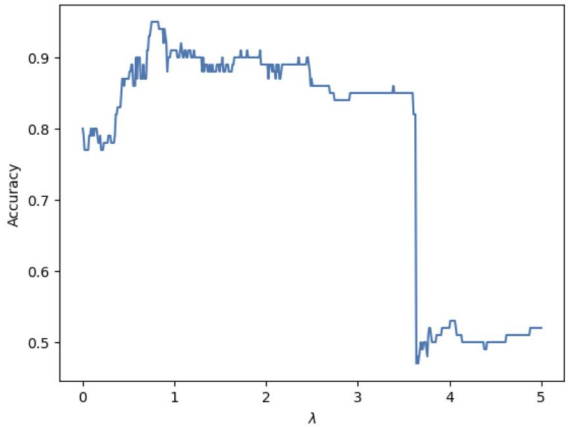


Figure Q11: **λ** vs Accuracy using Reward Function 1

Question 12:

From the figure Q11, we can see that optimal **λ** is 0.7515 with accuracy 0.95.

2.2 Q11 **30 / 30**

✓ **- 0 pts** *Correct*

Output:

$$c = \begin{bmatrix} 1_{|S|\times 1} \\ -\lambda_{|S|\times 1} \\ 0_{|S|\times 1} \end{bmatrix}, x = \begin{bmatrix} t \\ u \\ R \end{bmatrix}, b = \begin{bmatrix} 0_{|2(|\mathcal{A}\setminus a_1|-1)|S||\times 1} \\ 0_{|S|\times 1} \\ 0_{|S|\times 1} \\ (R_{max})_{|S|\times 1}(R_{max})_{|S|\times 1} \end{bmatrix}$$

$$D = \begin{bmatrix} I_{|S|\times|S|} & 0 & (P_a - P_{a_1})(I-\gamma P_{a_1})^{-1} \\ 0 & 0 & (P_a - P_{a_1})(I-\gamma P_{a_1})^{-1} \\ 0 & -I_{|S|\times|S|} & I_{|S|\times|S|} \\ 0 & -I_{|S|\times|S|} & -I_{|S|\times|S|} \\ 0 & 0 & I_{|S|\times|S|} \\ 0 & 0 & -I_{|S|\times|S|} \end{bmatrix}$$

Question 11:

LP formulation found in Q10 can be easily solved using a LP solver, only catch is that those solvers try to minimize, so c should be multiplied with -1 to solve our maximization problem.

Equally spaced 500 **λ** values between 0 and 5 are created to optimize the LP solution. Following plot is obtained:

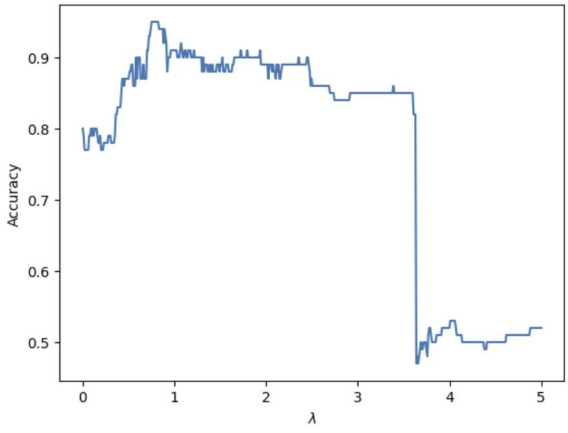


Figure Q11: **λ** vs Accuracy using Reward Function 1

Question 12:

From the figure Q11, we can see that optimal **λ** is 0.7515 with accuracy 0.95.

2.3 Q12 **5 / 5**

✓ **- 0 pts** *Correct*

Question 13

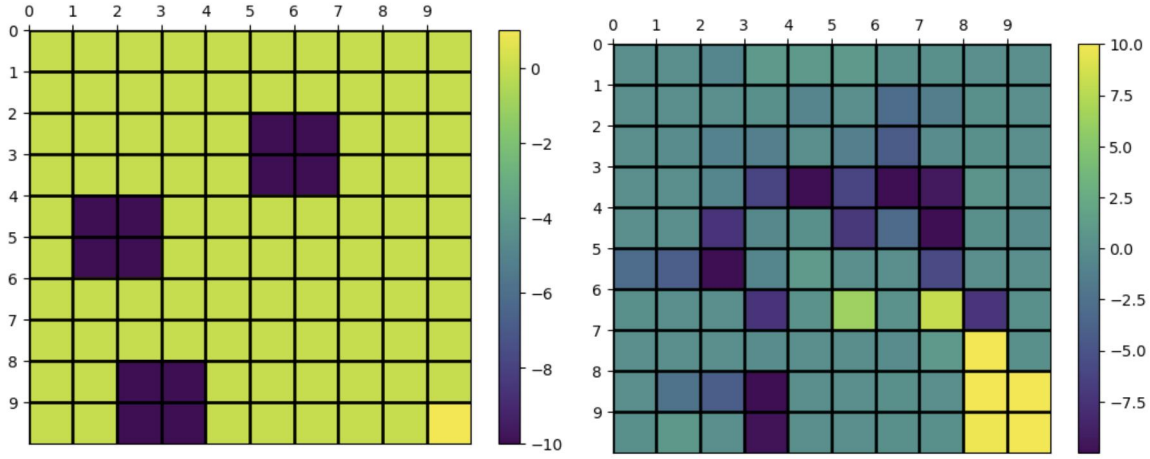


Figure Q13: (Left) Ground Truth Reward, (Right) Extracted Reward

Since we have used Rmax = max(|R_1|), Extracted Reward function have a range of -10 to 10. However, we can see that two heatmaps are similar to each other. Sharp penalty regions can not be seen at the extracted reward function. Penalty and reward regions are smoothed across the neighboring states. However, it is easy to distinguish the original penalty and reward regions at the Extracted Reward function.

Question 14

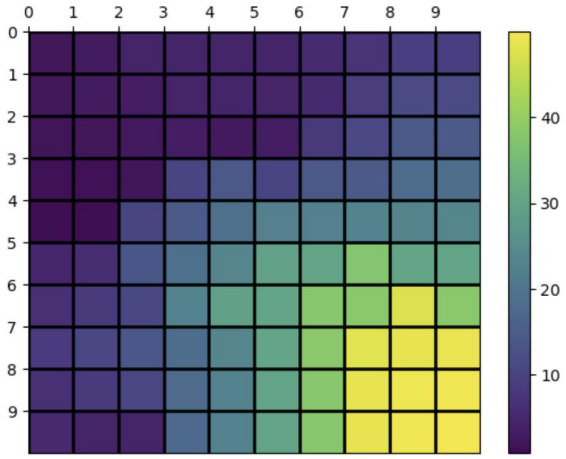


Figure Q14: Optimal values of the states

Same method applied to Q3 is used here. Using the reward matrix and transition probabilities, optimal values of the states are obtained.

2.4 Q13 **15 / 15**

✓ **- 0 pts** *Correct*

Question 13

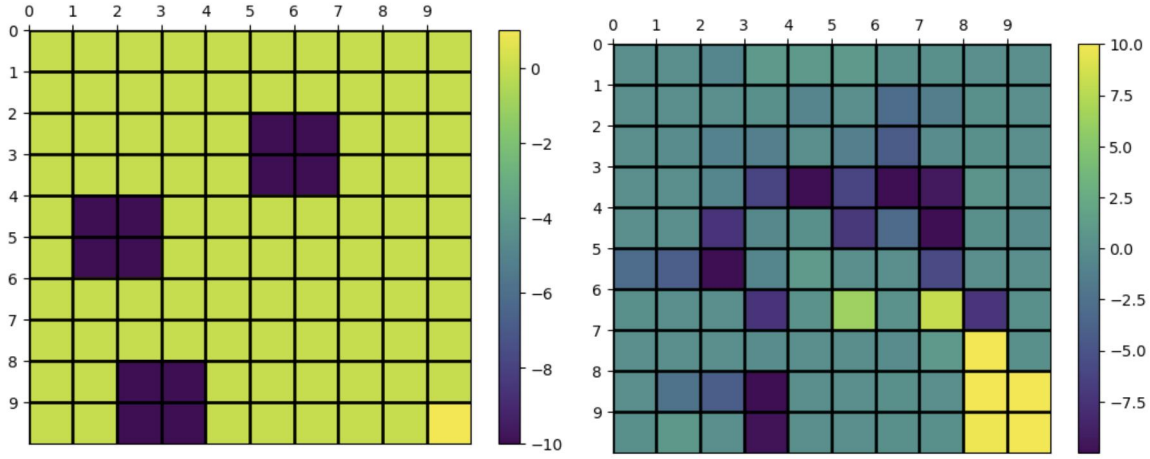


Figure Q13: (Left) Ground Truth Reward, (Right) Extracted Reward

Since we have used Rmax = max(|R_1|), Extracted Reward function have a range of -10 to 10. However, we can see that two heatmaps are similar to each other. Sharp penalty regions can not be seen at the extracted reward function. Penalty and reward regions are smoothed across the neighboring states. However, it is easy to distinguish the original penalty and reward regions at the Extracted Reward function.

Question 14

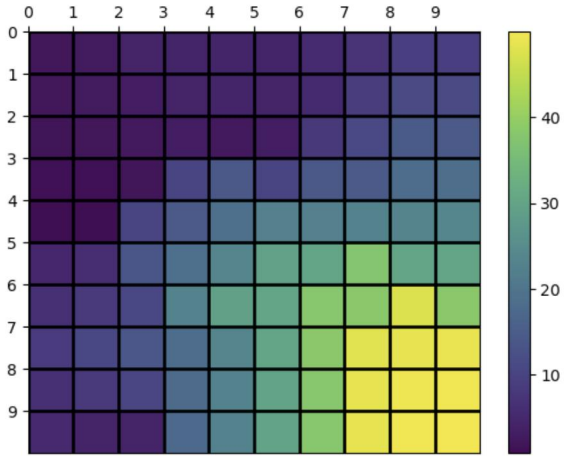


Figure Q14: Optimal values of the states

Same method applied to Q3 is used here. Using the reward matrix and transition probabilities, optimal values of the states are obtained.

2.5 Q14 **10 / 10**

✓ **- 0 pts** *Correct*

Question 15

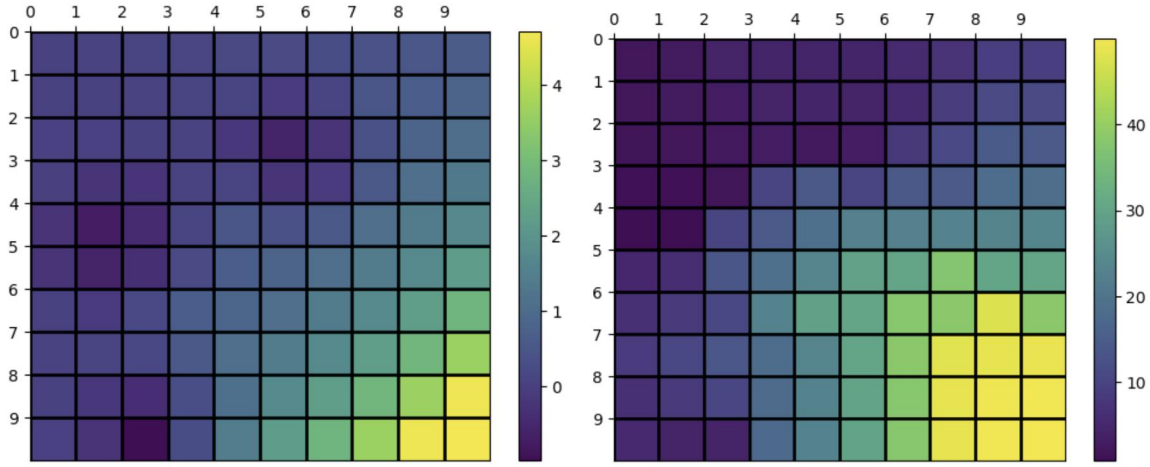


Figure Q15: (Left) Q3 Optimal Values of the States, (Right) Q14 Optimal Values of the States

At the first look, the heatmap of two solutions are nearly identical, since for both reward functions (ground truth and extracted), the reward is getting bigger as the state is closer to the right bottom corner.

However, the range of values are different and widely spread due to the range of the reward function. Also, since there is smoothness in the reward function of extracted reward, highly negative and highly positive optimal values are more dispersed. The smoothness of the reward function is also reflected to the optimal values of states. Therefore, we observe more states having the highest values and having the lowest values. Also, note that reward rectangle and penalty rectangle can not be easily distinguished from the results of Q14.

Question 16

Optimal agent policy for the extracted reward function is as follows:

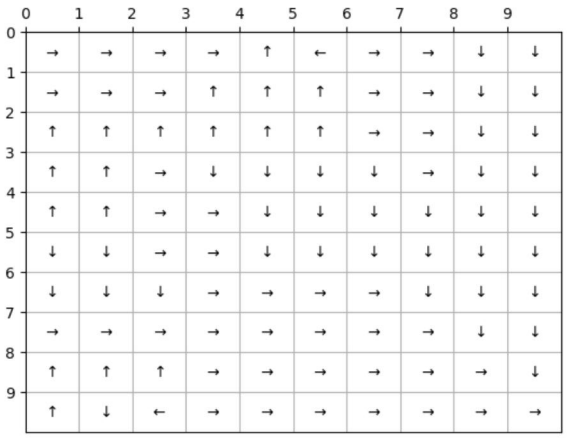


Figure Q16: Optimal policy for the extracted reward function

2.6 Q15 **10 / 10**

✓ **- 0 pts** *Correct*

Question 15

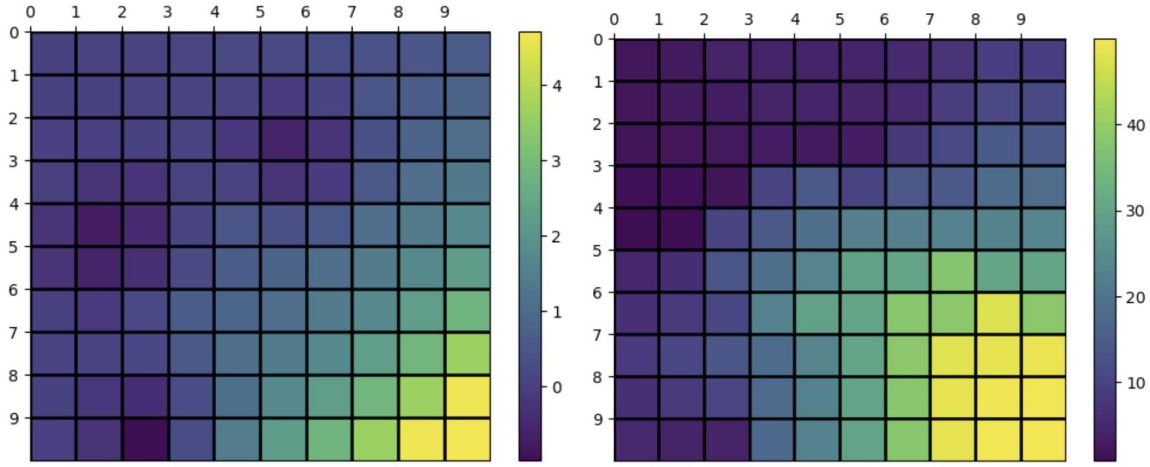


Figure Q15: (Left) Q3 Optimal Values of the States, (Right) Q14 Optimal Values of the States

At the first look, the heatmap of two solutions are nearly identical, since for both reward functions (ground truth and extracted), the reward is getting bigger as the state is closer to the right bottom corner.

However, the range of values are different and widely spread due to the range of the reward function. Also, since there is smoothness in the reward function of extracted reward, highly negative and highly positive optimal values are more dispersed. The smoothness of the reward function is also reflected to the optimal values of states. Therefore, we observe more states having the highest values and having the lowest values. Also, note that reward rectangle and penalty rectangle can not be easily distinguished from the results of Q14.

Question 16

Optimal agent policy for the extracted reward function is as follows:

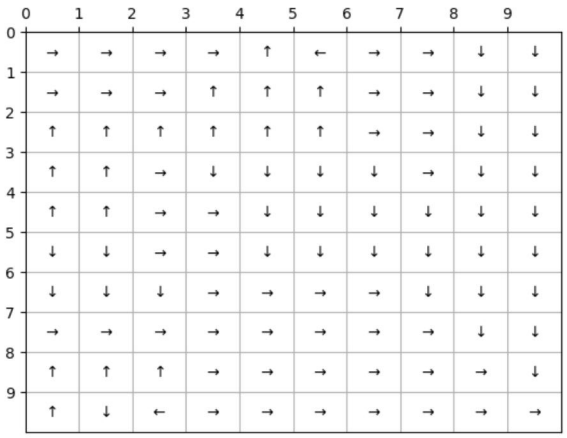


Figure Q16: Optimal policy for the extracted reward function

2.7 Q16 **10 / 10**

✓ **- 0 pts** *Correct*

Question 17

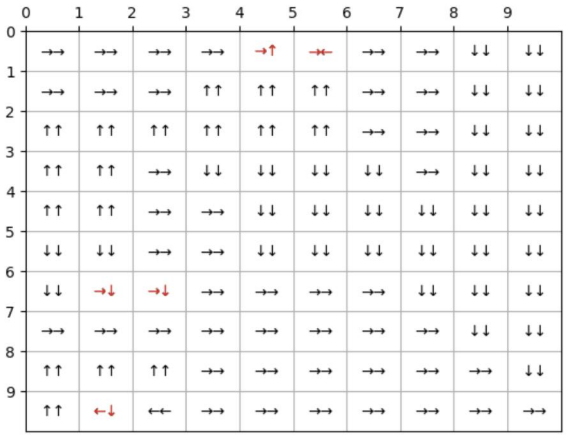
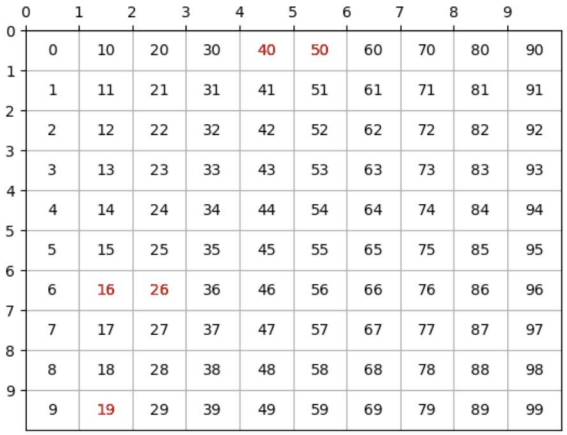


Figure Q17.1: Comparison between optimal policies found at Q5 and Q16. (Arrows at the left belong to Q5, Arrows at the right belong to Q16. Difference in policy marked with red)

Policy for following states are different: 16, 19, 26, 40, 50



	0	1	2	3	4	5	6	7	8	9
0	0	10	20	30	40	50	60	70	80	90
1	1	11	21	31	41	51	61	71	81	91
2	2	12	22	32	42	52	62	72	82	92
3	3	13	23	33	43	53	63	73	83	93
4	4	14	24	34	44	54	64	74	84	94
5	5	15	25	35	45	55	65	75	85	95
6	6	16	26	36	46	56	66	76	86	96
7	7	17	27	37	47	57	67	77	87	97
8	8	18	28	38	48	58	68	78	88	98
9	9	19	29	39	49	59	69	79	89	99

Figure Q17.2: States that have different policies are marked with red

From Figure Q17.1 and Q17.2 we can see that 95 states have the same policy. Both policies show that the agent tries to go the nearest diagonal and then to the bottom left corner while avoiding penalty regions. The similarity of policies show that although there are some differences in the reward matrix and optimal state values, the agent performs very similarly.

However, for states 19, 40 and 50, reconstructed optimal policy is meaningless and consumes time. This is due to the spreadness of penalty regions, the agent tries to avoid those regions by jumping to edges. Also, for states 16 and 26 since reward function does not have penalization at 17 and 27 as much as the ground truth reward function, optimal policy is found as moving to those regions.

2.8 Q17 **10 / 10**

✓ **- 0 pts** *Correct*

Question 18

Here we repeat the task of question 11, this time with an expert based on the second reward function. The figure below displays the plot of accuracy versus λ.

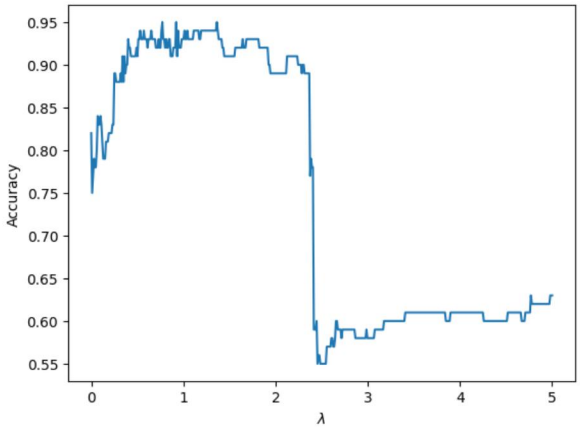


Figure Q18: Accuracy of IRL Algorithm vs. λ for Reward Function 2

Looking at the graph, we can see the accuracy increase as λ approaches. It then begins to drop off, with a steep fall around λ = 2.5. λ acts as a regularizer; at high values we see it work to prevent overfitting and improve the generalization of the algorithm. However, at high values λ prevents proper fitting, limiting the expressiveness of the reward vector. Since $\sum_{i=1}^{|S|} (t_i - \lambda u_i)$, with the constraint that $-u \leq R \leq u$, we see that high values of λ force low u and R subsequently. Thus there is an optimal value for λ we determine.

Question 19

We want to report the value of λ for which the accuracy is maximum $\lambda_2^{(max)}$. In this case, rounded to the third decimal, $\lambda_2^{(max)} = .772$, giving an accuracy of 95.0%.

Question 20

Now, for $\lambda_2^{(max)}$, we generate heat maps of the ground truth reward and extracted reward. The ground truth reward is the provided reward function2 and we obtain the extracted reward as we did in question question 13 via the IRL algorithm. We see the two heat maps below.

2.9 Q18 **30 / 30**

✓ **- 0 pts** *Correct*

Question 18

Here we repeat the task of question 11, this time with an expert based on the second reward function. The figure below displays the plot of accuracy versus λ.

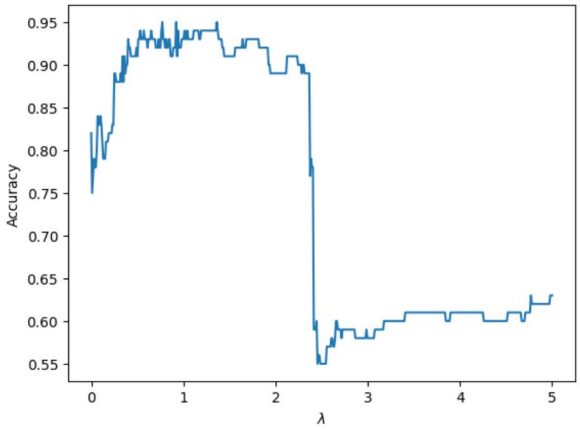


Figure Q18: Accuracy of IRL Algorithm vs. λ for Reward Function 2

Looking at the graph, we can see the accuracy increase as λ approaches. It then begins to drop off, with a steep fall around λ = 2.5. λ acts as a regularizer; at high values we see it work to prevent overfitting and improve the generalization of the algorithm. However, at high values λ prevents proper fitting, limiting the expressiveness of the reward vector. Since $\sum_{i=1}^{|S|} (t_i - \lambda u_i)$, with the constraint that $-u \leq R \leq u$, we see that high values of λ force low u and R subsequently. Thus there is an optimal value for λ we determine.

Question 19

We want to report the value of λ for which the accuracy is maximum $\lambda_2^{(max)}$. In this case, rounded to the third decimal, $\lambda_2^{(max)} = .772$, giving an accuracy of 95.0%.

Question 20

Now, for $\lambda_2^{(max)}$, we generate heat maps of the ground truth reward and extracted reward. The ground truth reward is the provided reward function2 and we obtain the extracted reward as we did in question question 13 via the IRL algorithm. We see the two heat maps below.

2.10 Q19 **5 / 5**

✓ **- 0 pts** *Correct*

Question 18

Here we repeat the task of question 11, this time with an expert based on the second reward function. The figure below displays the plot of accuracy versus λ.

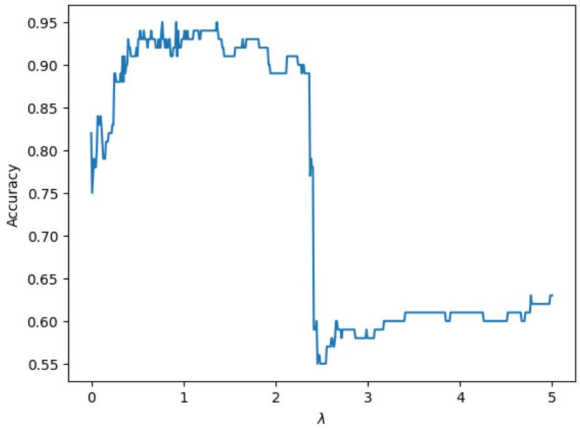


Figure Q18: Accuracy of IRL Algorithm vs. λ for Reward Function 2

Looking at the graph, we can see the accuracy increase as λ approaches. It then begins to drop off, with a steep fall around λ = 2.5. λ acts as a regularizer; at high values we see it work to prevent overfitting and improve the generalization of the algorithm. However, at high values λ prevents proper fitting, limiting the expressiveness of the reward vector. Since $\sum_{i=1}^{|S|} (t_i - \lambda u_i)$, with the constraint that $-u \leq R \leq u$, we see that high values of λ force low u and R subsequently. Thus there is an optimal value for λ we determine.

Question 19

We want to report the value of λ for which the accuracy is maximum $\lambda_2^{(max)}$. In this case, rounded to the third decimal, $\lambda_2^{(max)} = .772$, giving an accuracy of 95.0%.

Question 20

Now, for $\lambda_2^{(max)}$, we generate heat maps of the ground truth reward and extracted reward. The ground truth reward is the provided reward function2 and we obtain the extracted reward as we did in question question 13 via the IRL algorithm. We see the two heat maps below.

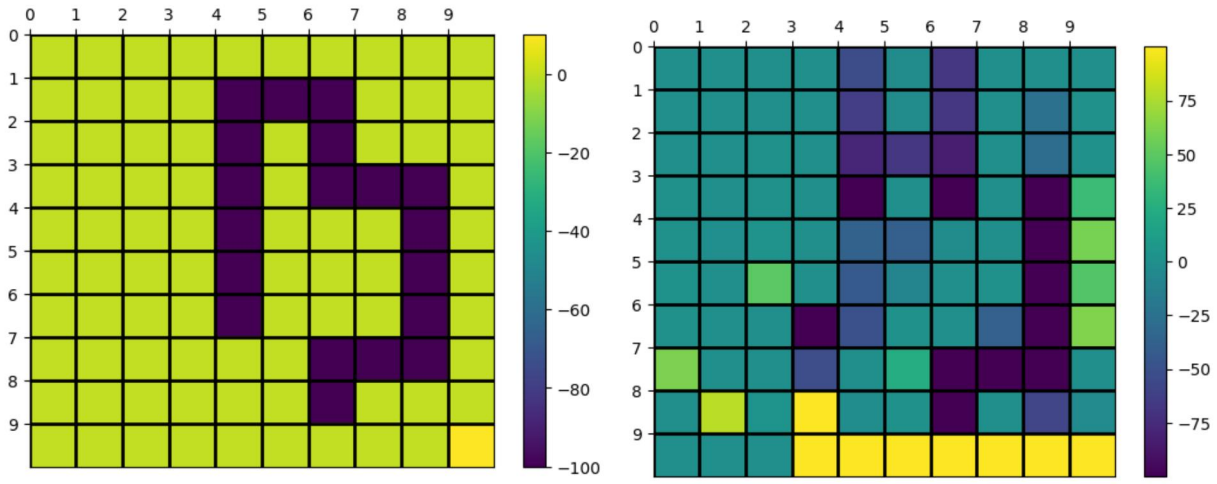


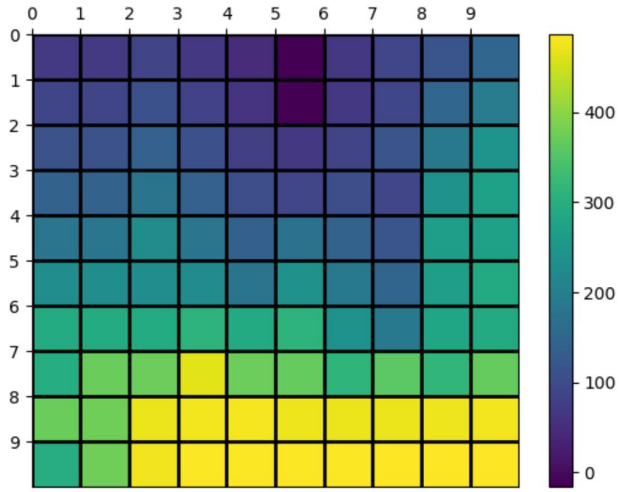
Figure Q20: (Left) Heatmap for Ground Truth Reward 2, (Right) Extracted Heatmap for $\lambda_2^{(max)}$

Looking at the two heatmaps, we see that both present a similar picture with the extracted map being less discrete and perhaps "fuzzier" in presentation. In the extracted heatmap, areas around state 99 have high reward values, while states on or in the vicinity of the snaking lower reward area of the ground truth are similarly low in reward. On the other hand, we see pockets of higher reward on the left side as well as a line of high reward states in the final row. There is also a difference in scale.

We very much expect discrepancies and a loss of sharpness in this process, as it is ultimately not a lossless endeavor. The expert we learn from cannot capture all the intricacies of the reward function. Additionally, it cannot provide an exact scale.

Question 21

We use the extracted reward function from the previous question to compute the optimal values of all the states using value iteration. The heatmap is seen below.



2.11 Q20 **15 / 15**

✓ **- 0 pts** *Correct*

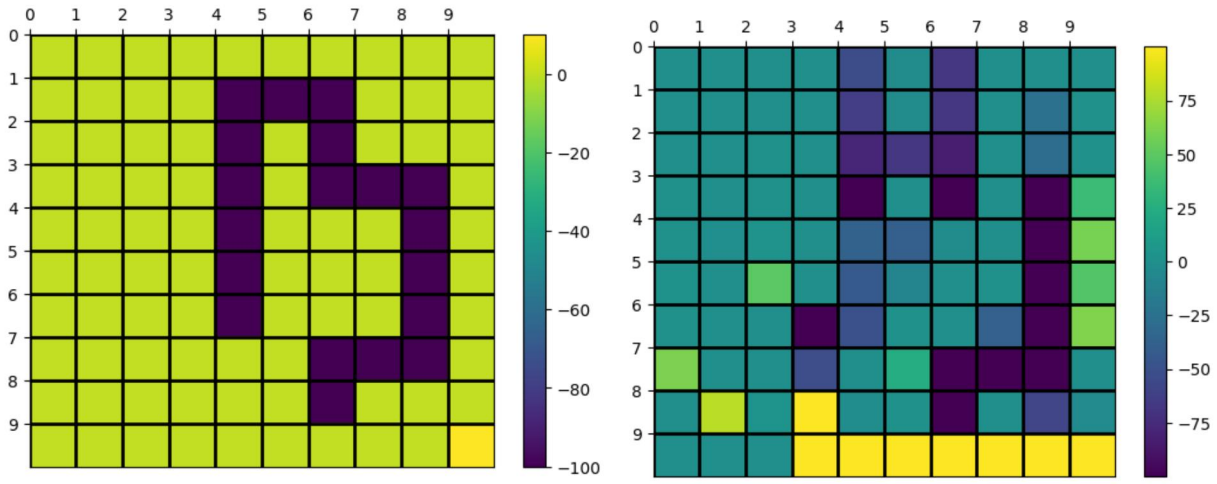


Figure Q20: (Left) Heatmap for Ground Truth Reward 2, (Right) Extracted Heatmap for $\lambda_2^{(max)}$

Looking at the two heatmaps, we see that both present a similar picture with the extracted map being less discrete and perhaps "fuzzier" in presentation. In the extracted heatmap, areas around state 99 have high reward values, while states on or in the vicinity of the snaking lower reward area of the ground truth are similarly low in reward. On the other hand, we see pockets of higher reward on the left side as well as a line of high reward states in the final row. There is also a difference in scale.

We very much expect discrepancies and a loss of sharpness in this process, as it is ultimately not a lossless endeavor. The expert we learn from cannot capture all the intricacies of the reward function. Additionally, it cannot provide an exact scale.

Question 21

We use the extracted reward function from the previous question to compute the optimal values of all the states using value iteration. The heatmap is seen below.

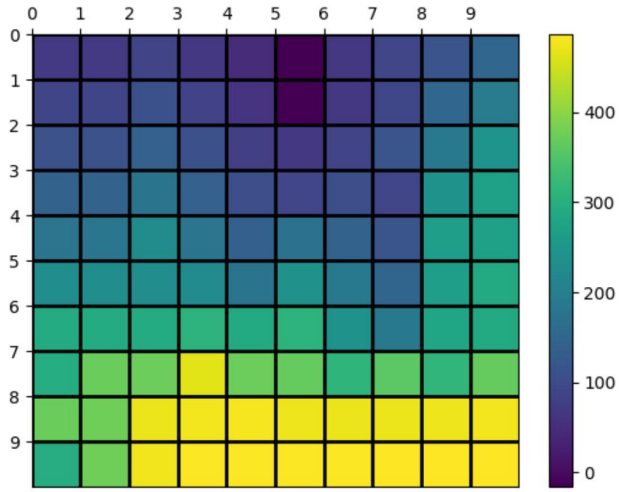


Figure Q21: Heatmap for Optimal State Values Using $\lambda_2^{(max)}$

Question 22

We now compare the heatmaps generated in Question 7 and from the previous Question 21, seen above. The heatmap from question 7 is displayed below.

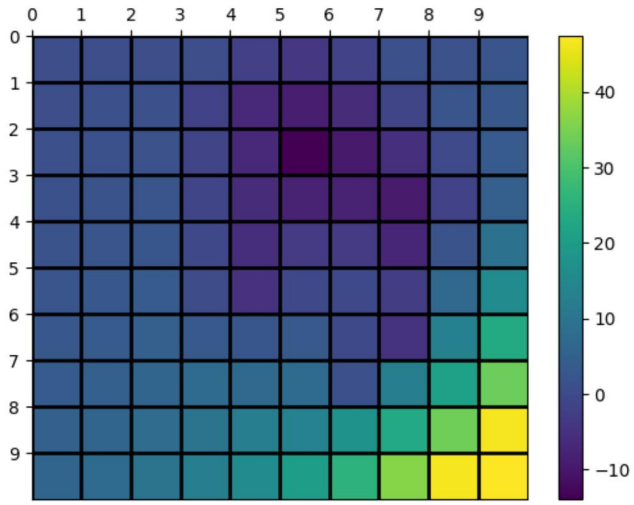


Figure Q22: Heatmap for Optimal States Values for Reward Function 2

In regards to the similarities, we see that areas of high and low values are similar, with the lower right corner containing high values and the middle to upper areas being low in value. The gradation between the two areas is also similar with intermediate values between the two extremes.

A marked difference we see is the abundance of high state values in the extracted value heatmap. Although the lower right corner is high in value we see this value spread to left past the midpoint of the columns and up an additional row. We also see the area of lowest value having shifted slightly higher in the extracted heatmap. The scaling is also different between the two maps. The values from the ground truth are also slightly more discrete and less fuzzy than the extracted heatmap values.

Question 23

Here we have used the extracted reward function to compute the policy for the agent. It is shown below.

2.12 Q21 **10 / 10**

✓ **- 0 pts** *Correct*

Figure Q21: Heatmap for Optimal State Values Using $\lambda_2^{(max)}$

Question 22

We now compare the heatmaps generated in Question 7 and from the previous Question 21, seen above. The heatmap from question 7 is displayed below.

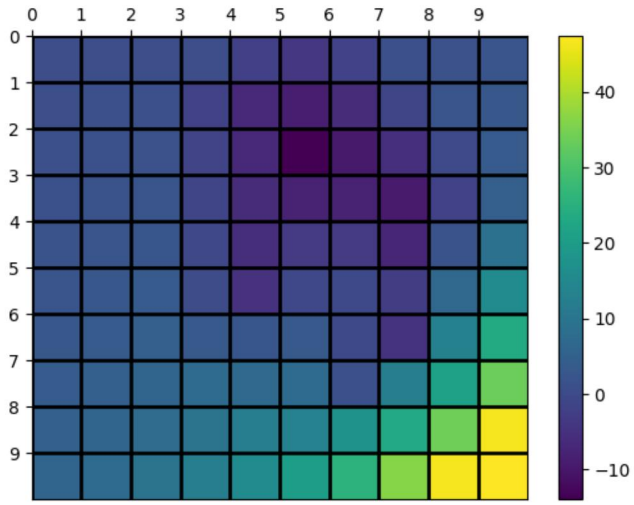


Figure Q22: Heatmap for Optimal States Values for Reward Function 2

In regards to the similarities, we see that areas of high and low values are similar, with the lower right corner containing high values and the middle to upper areas being low in value. The gradation between the two areas is also similar with intermediate values between the two extremes.

A marked difference we see is the abundance of high state values in the extracted value heatmap. Although the lower right corner is high in value we see this value spread to left past the midpoint of the columns and up an additional row. We also see the area of lowest value having shifted slightly higher in the extracted heatmap. The scaling is also different between the two maps. The values from the ground truth are also slightly more discrete and less fuzzy than the extracted heatmap values.

Question 23

Here we have used the extracted reward function to compute the policy for the agent. It is shown below.

2.13 Q22 **10 / 10**

✓ **- 0 pts** *Correct*

Figure Q21: Heatmap for Optimal State Values Using $\lambda_2^{(max)}$

Question 22

We now compare the heatmaps generated in Question 7 and from the previous Question 21, seen above. The heatmap from question 7 is displayed below.

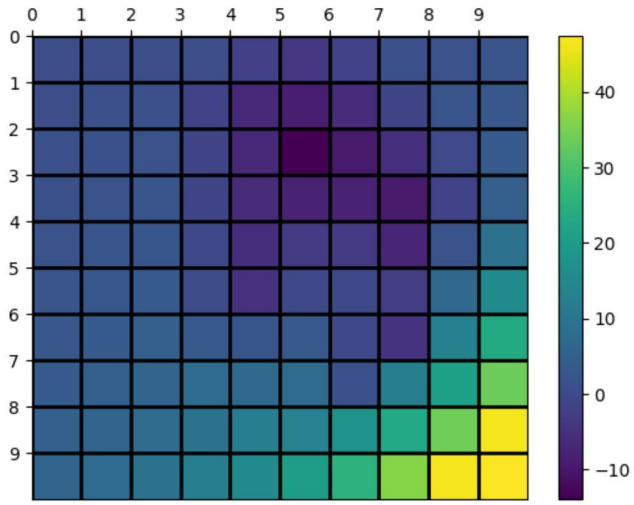


Figure Q22: Heatmap for Optimal States Values for Reward Function 2

In regards to the similarities, we see that areas of high and low values are similar, with the lower right corner containing high values and the middle to upper areas being low in value. The gradation between the two areas is also similar with intermediate values between the two extremes.

A marked difference we see is the abundance of high state values in the extracted value heatmap. Although the lower right corner is high in value we see this value spread to left past the midpoint of the columns and up an additional row. We also see the area of lowest value having shifted slightly higher in the extracted heatmap. The scaling is also different between the two maps. The values from the ground truth are also slightly more discrete and less fuzzy than the extracted heatmap values.

Question 23

Here we have used the extracted reward function to compute the policy for the agent. It is shown below.

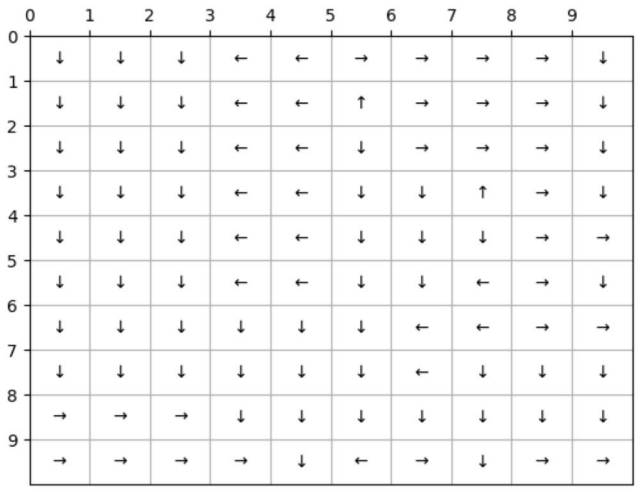


Figure Q23: Policy for Agent Using Extracted Reward Function Via $\lambda_2^{(max)}$

Question 24

Below we replicate the policy for the agent via the ground truth reward function 2 for comparison.

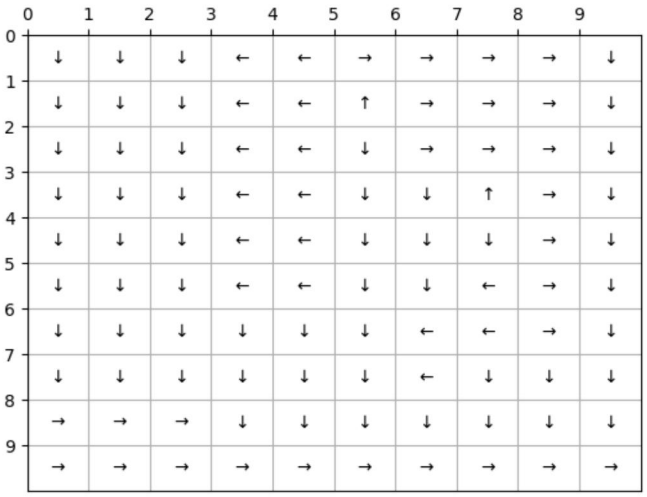


Figure Q24.1: Policy for Agent Using Ground Truth Reward Function 2

We see that both policies follow a similar guideline, directing the agent away from the region of low reward eventually downwards and right to the area of highest reward. All directions are included and the pathing is similar.

Divegently, we see multiple areas in the extracted policy where the agent tries to move off the grid as well as an instance of the optimal action being away from the area of highest reward. These are highlighted in the figure below.

2.14 Q23 **10 / 10**

✓ **- 0 pts** *Correct*

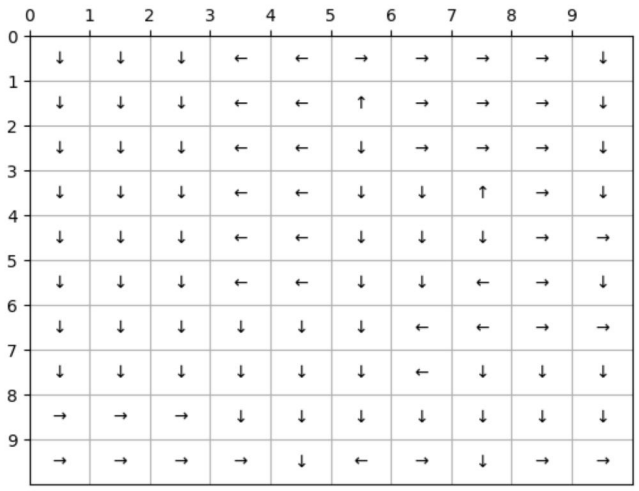


Figure Q23: Policy for Agent Using Extracted Reward Function Via $\lambda_2^{(max)}$

Question 24

Below we replicate the policy for the agent via the ground truth reward function 2 for comparison.

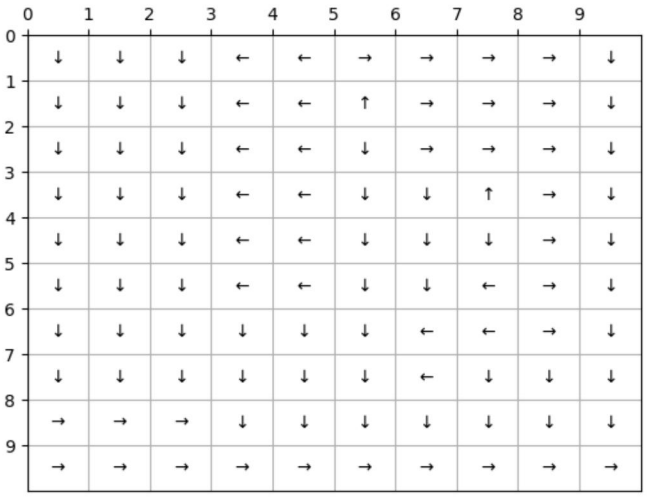


Figure Q24.1: Policy for Agent Using Ground Truth Reward Function 2

We see that both policies follow a similar guideline, directing the agent away from the region of low reward eventually downwards and right to the area of highest reward. All directions are included and the pathing is similar.

Divegently, we see multiple areas in the extracted policy where the agent tries to move off the grid as well as an instance of the optimal action being away from the area of highest reward. These are highlighted in the figure below.

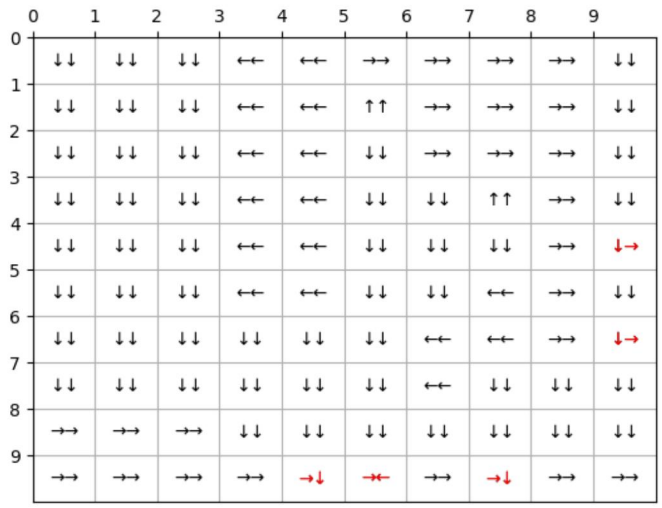


Figure Q24.2: Policy Discrepencies of Ground Truth (left arrow) and Extracted (right arrow)

We see the attempts to the exit the grid in states 49 and 79 on on the bottom row and 94 and 96 in the final column. We see the left pointing arrow directly adjacent to an exit state in state 59 on the bottom row.

Question 25

As was discussed in question 24 there are two major discrepancies in the optimal policy for the agent. We see four instances where the agent attempts to exit the grid, states 49 and 79 on the bottom row and states 94 and 96. The local rewards may be outweighing that of the long run, perhaps due to not enough emphasis on exploration and/or too loose of a convergence criterion in $\in$. This may be caused due to the inability of the algorithm to fully recreate the more discrete nature of the ground truth reward function. We could hardcode the off-grid to having values of negative infinity to remove this issue, but that is not possible with a simple change to only the value iteration algorithm.

Another discrepancy is a falsely pointing arrow in state 59 that points left rather than right towards the area of highest reward in the bottom right. As shown in the figure below, the values of the states surrounding state 59 are markedly close in value, suggesting that a slight tightening of the convergence criterion may amend this error.

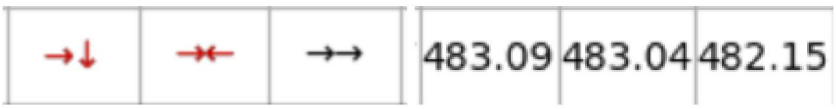


Figure Q25.1: (Left) Policy Differences as Presented in Figure Q24.2 for states 49, 59, 69, (Right) Value for States 49, 59, 69 Via Extracted Reward Function

Rerunning the value iteration with $\in$ reduced a factor of ten yielded the following discrepancy chart.

2.15 Q24 **10 / 10**

✓ **- 0 pts** *Correct*

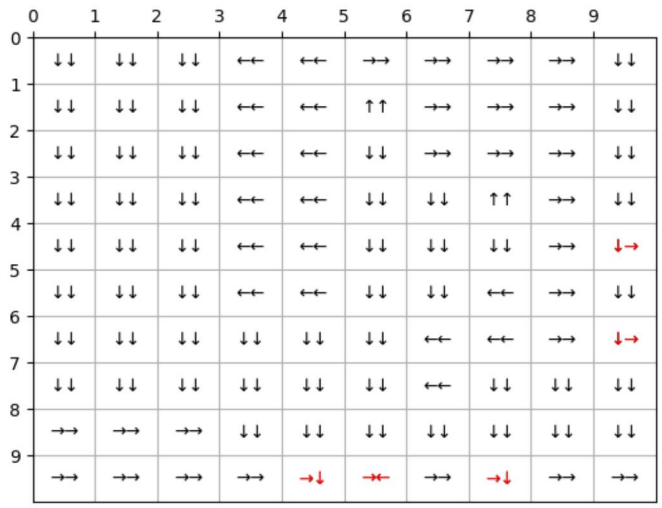


Figure Q24.2: Policy Discrepencies of Ground Truth (left arrow) and Extracted (right arrow)

We see the attempts to the exit the grid in states 49 and 79 on on the bottom row and 94 and 96 in the final column. We see the left pointing arrow directly adjacent to an exit state in state 59 on the bottom row.

Question 25

As was discussed in question 24 there are two major discrepancies in the optimal policy for the agent. We see four instances where the agent attempts to exit the grid, states 49 and 79 on the bottom row and states 94 and 96. The local rewards may be outweighing that of the long run, perhaps due to not enough emphasis on exploration and/or too loose of a convergence criterion in $\in$. This may be caused due to the inability of the algorithm to fully recreate the more discrete nature of the ground truth reward function. We could hardcode the off-grid to having values of negative infinity to remove this issue, but that is not possible with a simple change to only the value iteration algorithm.

Another discrepancy is a falsely pointing arrow in state 59 that points left rather than right towards the area of highest reward in the bottom right. As shown in the figure below, the values of the states surrounding state 59 are markedly close in value, suggesting that a slight tightening of the convergence criterion may amend this error.

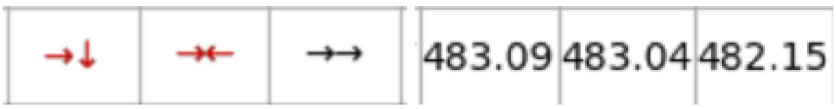


Figure Q25.1: (Left) Policy Differences as Presented in Figure Q24.2 for states 49, 59, 69, (Right) Value for States 49, 59, 69 Via Extracted Reward Function

Rerunning the value iteration with $\in$ reduced a factor of ten yielded the following discrepancy chart.

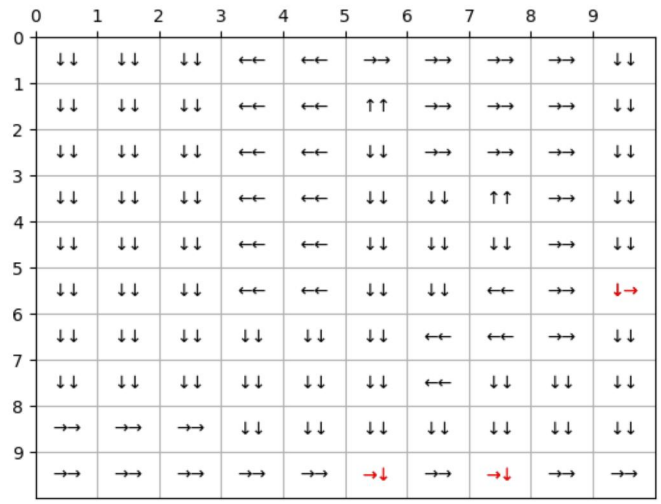


Figure Q25.2 Policy Discrepencies of Ground Truth (left arrow) and Extracted (right arrow) with $\in = .001$

We no longer see any arrows pointing opposite the direction of the highest reward states and the new best accuracy is 97%, occurring at $\lambda = .922$. The graph is shown below.

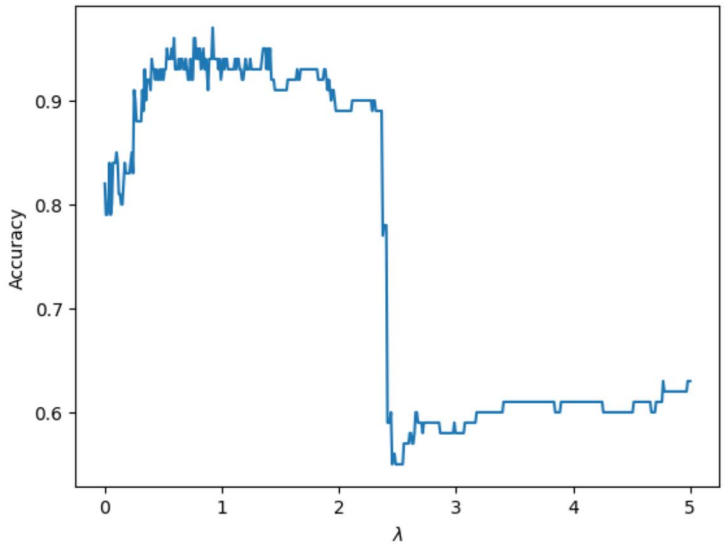


Figure Q25.3: Accuracy of IRL Algorithm vs. λ for Reward Function 2 with $\in = .001$

2.16 Q25 **50 / 50**

✓ **- 0 pts** *Correct*