ECE232E: LARGE SCALE SOCIAL AND COMPLEX NETWORKS: DESIGN AND ALGORITHMS

Project 3: Reinforcement learning and Inverse Reinforcement learning

May 25, 2020

Wanli Gao, UID: 105431975
Yifan Zhang, UID: 805354474
Tianyi Zhao, UID: 804380974

1 Reinforcement learning (RL)

1.1 Environment

Question 1:
We plot the heat maps of Reward function 1 and Reward function 2:

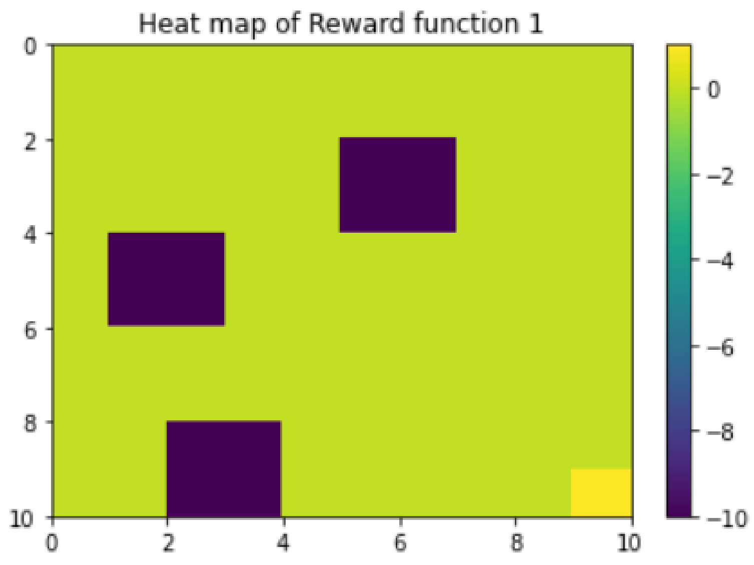


Figure 1.1. Heat map of Reward function 1

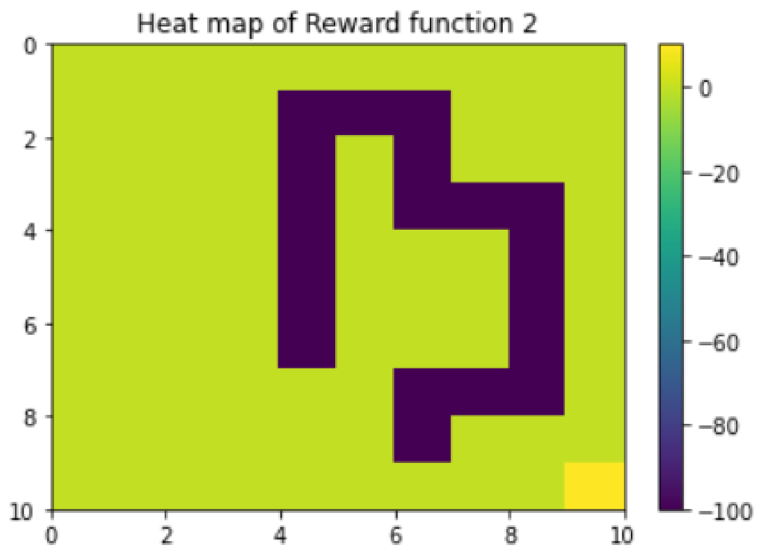
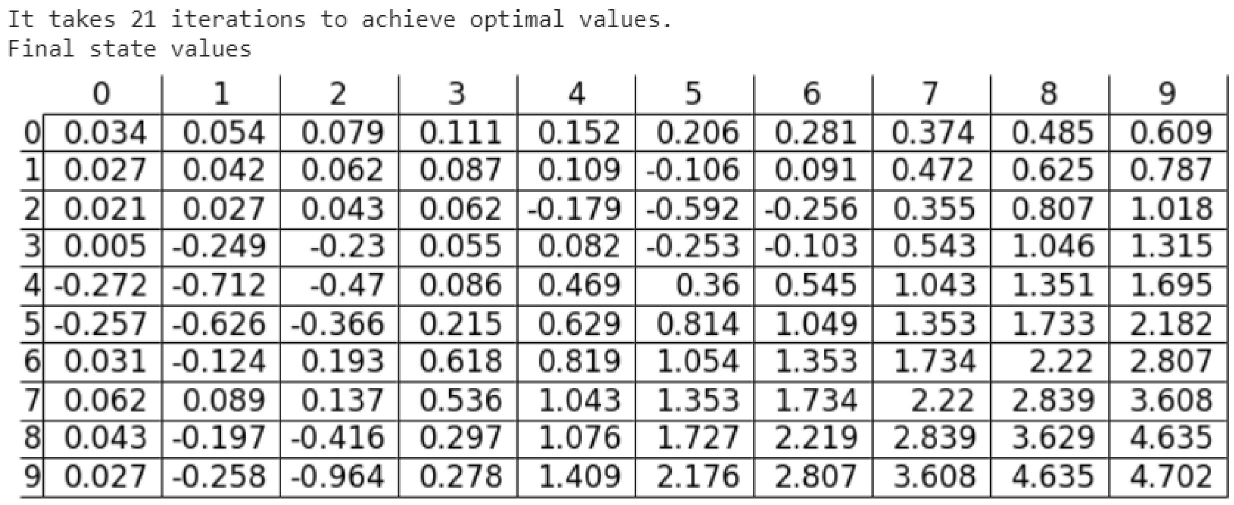


Figure 1.2. Heat map of Reward function 2

2 Optimal policy learning using RL algorithms

Question 2:
We created the environment of the agent using the information provided in section 2, with the parameters given in the guideline. The optimal state values are shown below:

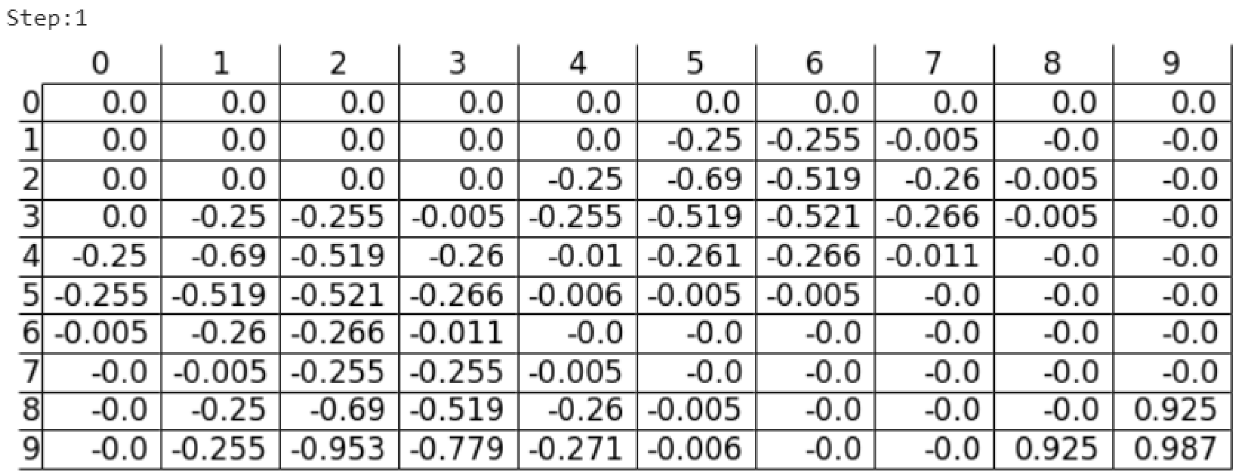


```
It takes 21 iterations to achieve optimal values.
Final state values
```

	0	1	2	3	4	5	6	7	8	9
0	0.034	0.054	0.079	0.111	0.152	0.206	0.281	0.374	0.485	0.609
1	0.027	0.042	0.062	0.087	0.109	-0.106	0.091	0.472	0.625	0.787
2	0.021	0.027	0.043	0.062	-0.179	-0.592	-0.256	0.355	0.807	1.018
3	0.005	-0.249	-0.23	0.055	0.082	-0.253	-0.103	0.543	1.046	1.315
4	-0.272	-0.712	-0.47	0.086	0.469	0.36	0.545	1.043	1.351	1.695
5	-0.257	-0.626	-0.366	0.215	0.629	0.814	1.049	1.353	1.733	2.182
6	0.031	-0.124	0.193	0.618	0.819	1.054	1.353	1.734	2.22	2.807
7	0.062	0.089	0.137	0.536	1.043	1.353	1.734	2.22	2.839	3.608
8	0.043	-0.197	-0.416	0.297	1.076	1.727	2.219	2.839	3.629	4.635
9	0.027	-0.258	-0.964	0.278	1.409	2.176	2.807	3.608	4.635	4.702

Figure 2.1. The optimal values of states using reward function 1

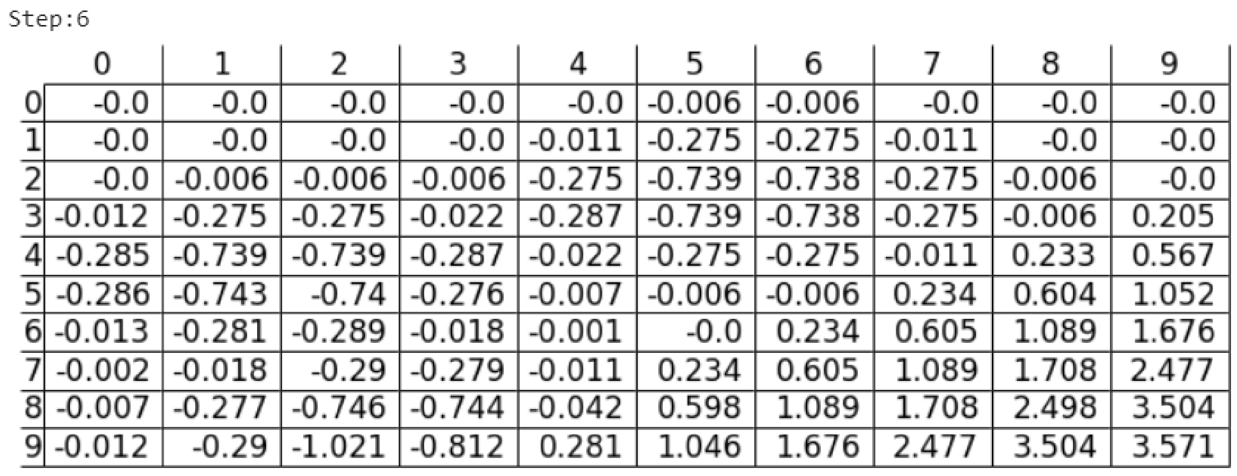
It takes 21 iterations to converge. We plot the snapshots of step 1, 6, 11, 16, 21:



```
Step: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	-0.25	-0.255	-0.005	-0.0	-0.0
2	0.0	0.0	0.0	0.0	-0.25	-0.69	-0.519	-0.26	-0.005	-0.0
3	0.0	-0.25	-0.255	-0.005	-0.255	-0.519	-0.521	-0.266	-0.005	-0.0
4	-0.25	-0.69	-0.519	-0.26	-0.01	-0.261	-0.266	-0.011	-0.0	-0.0
5	-0.255	-0.519	-0.521	-0.266	-0.006	-0.005	-0.005	-0.0	-0.0	-0.0
6	-0.005	-0.26	-0.266	-0.011	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0
7	-0.0	-0.005	-0.255	-0.255	-0.005	-0.0	-0.0	-0.0	-0.0	-0.0
8	-0.0	-0.25	-0.69	-0.519	-0.26	-0.005	-0.0	-0.0	-0.0	0.925
9	-0.0	-0.255	-0.953	-0.779	-0.271	-0.006	-0.0	-0.0	0.925	0.987

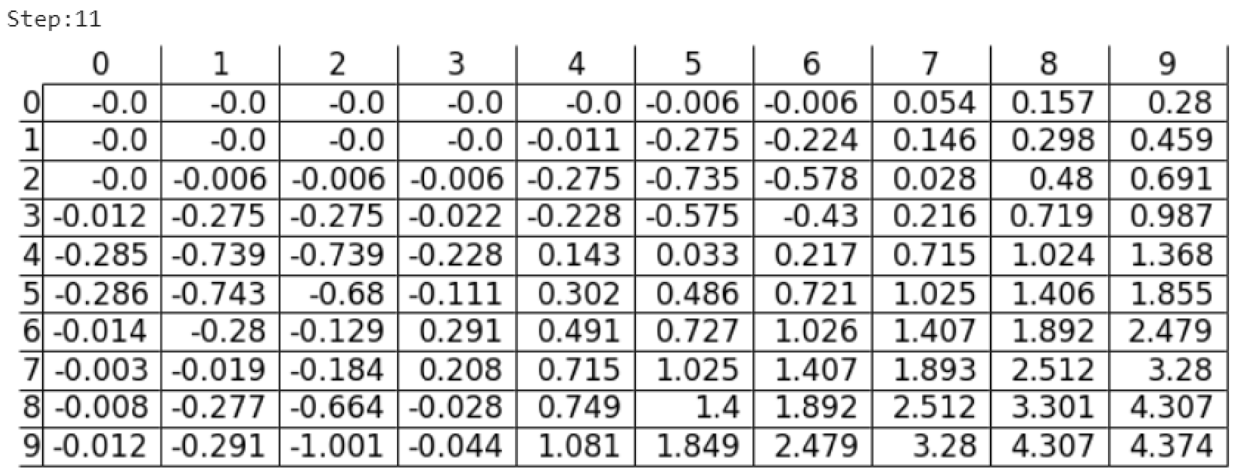
Figure 2.2. Snapshot of step 1



Step:6

	0	1	2	3	4	5	6	7	8	9
0	-0.0	-0.0	-0.0	-0.0	-0.0	-0.006	-0.006	-0.0	-0.0	-0.0
1	-0.0	-0.0	-0.0	-0.0	-0.011	-0.275	-0.275	-0.011	-0.0	-0.0
2	-0.0	-0.006	-0.006	-0.006	-0.275	-0.739	-0.738	-0.275	-0.006	-0.0
3	-0.012	-0.275	-0.275	-0.022	-0.287	-0.739	-0.738	-0.275	-0.006	0.205
4	-0.285	-0.739	-0.739	-0.287	-0.022	-0.275	-0.275	-0.011	0.233	0.567
5	-0.286	-0.743	-0.74	-0.276	-0.007	-0.006	-0.006	0.234	0.604	1.052
6	-0.013	-0.281	-0.289	-0.018	-0.001	-0.0	0.234	0.605	1.089	1.676
7	-0.002	-0.018	-0.29	-0.279	-0.011	0.234	0.605	1.089	1.708	2.477
8	-0.007	-0.277	-0.746	-0.744	-0.042	0.598	1.089	1.708	2.498	3.504
9	-0.012	-0.29	-1.021	-0.812	0.281	1.046	1.676	2.477	3.504	3.571

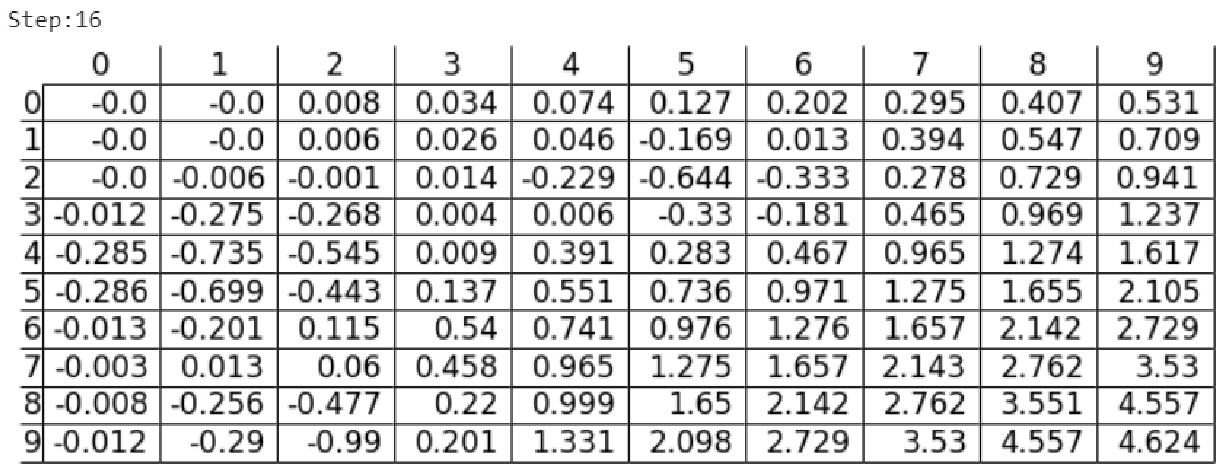
Figure 2.3. Snapshot of step 6



Step:11

	0	1	2	3	4	5	6	7	8	9
0	-0.0	-0.0	-0.0	-0.0	-0.0	-0.006	-0.006	0.054	0.157	0.28
1	-0.0	-0.0	-0.0	-0.0	-0.011	-0.275	-0.224	0.146	0.298	0.459
2	-0.0	-0.006	-0.006	-0.006	-0.275	-0.735	-0.578	0.028	0.48	0.691
3	-0.012	-0.275	-0.275	-0.022	-0.228	-0.575	-0.43	0.216	0.719	0.987
4	-0.285	-0.739	-0.739	-0.228	0.143	0.033	0.217	0.715	1.024	1.368
5	-0.286	-0.743	-0.68	-0.111	0.302	0.486	0.721	1.025	1.406	1.855
6	-0.014	-0.28	-0.129	0.291	0.491	0.727	1.026	1.407	1.892	2.479
7	-0.003	-0.019	-0.184	0.208	0.715	1.025	1.407	1.893	2.512	3.28
8	-0.008	-0.277	-0.664	-0.028	0.749	1.4	1.892	2.512	3.301	4.307
9	-0.012	-0.291	-1.001	-0.044	1.081	1.849	2.479	3.28	4.307	4.374

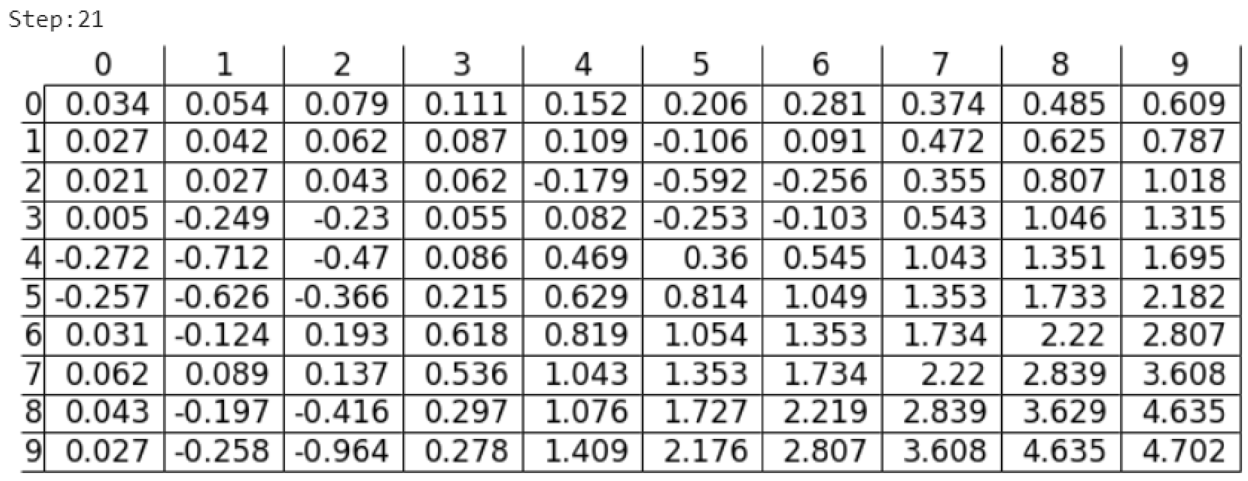
Figure 2.4. Snapshot of step 11



Step:16

	0	1	2	3	4	5	6	7	8	9
0	-0.0	-0.0	0.008	0.034	0.074	0.127	0.202	0.295	0.407	0.531
1	-0.0	-0.0	0.006	0.026	0.046	-0.169	0.013	0.394	0.547	0.709
2	-0.0	-0.006	-0.001	0.014	-0.229	-0.644	-0.333	0.278	0.729	0.941
3	-0.012	-0.275	-0.268	0.004	0.006	-0.33	-0.181	0.465	0.969	1.237
4	-0.285	-0.735	-0.545	0.009	0.391	0.283	0.467	0.965	1.274	1.617
5	-0.286	-0.699	-0.443	0.137	0.551	0.736	0.971	1.275	1.655	2.105
6	-0.013	-0.201	0.115	0.54	0.741	0.976	1.276	1.657	2.142	2.729
7	-0.003	0.013	0.06	0.458	0.965	1.275	1.657	2.143	2.762	3.53
8	-0.008	-0.256	-0.477	0.22	0.999	1.65	2.142	2.762	3.551	4.557
9	-0.012	-0.29	-0.99	0.201	1.331	2.098	2.729	3.53	4.557	4.624

Figure 2.5. Snapshot of step 16



```
Step:21
```

	0	1	2	3	4	5	6	7	8	9
0	0.034	0.054	0.079	0.111	0.152	0.206	0.281	0.374	0.485	0.609
1	0.027	0.042	0.062	0.087	0.109	-0.106	0.091	0.472	0.625	0.787
2	0.021	0.027	0.043	0.062	-0.179	-0.592	-0.256	0.355	0.807	1.018
3	0.005	-0.249	-0.23	0.055	0.082	-0.253	-0.103	0.543	1.046	1.315
4	-0.272	-0.712	-0.47	0.086	0.469	0.36	0.545	1.043	1.351	1.695
5	-0.257	-0.626	-0.366	0.215	0.629	0.814	1.049	1.353	1.733	2.182
6	0.031	-0.124	0.193	0.618	0.819	1.054	1.353	1.734	2.22	2.807
7	0.062	0.089	0.137	0.536	1.043	1.353	1.734	2.22	2.839	3.608
8	0.043	-0.197	-0.416	0.297	1.076	1.727	2.219	2.839	3.629	4.635
9	0.027	-0.258	-0.964	0.278	1.409	2.176	2.807	3.608	4.635	4.702

Figure 2.6. Snapshot of step 21

We can observe that the neighbors of the non-zero reward states first get non-zero state values. Then after several iterations the neighbors of these states with non-zero values also update their values to non-zero values. In other words, states with non-zero reward is extending outward to the states with zero reward. Therefore, the adjacent state values are updated. We can see from step 1 to step 21 that states with zero values become less and less as the step increases.

Question 3:
We plot the heat map of the optimal state values across the 2-D grid:

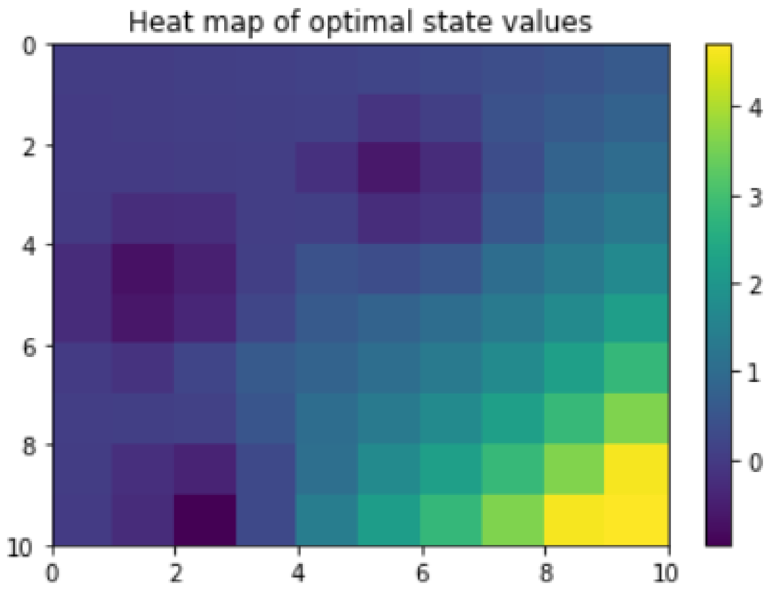
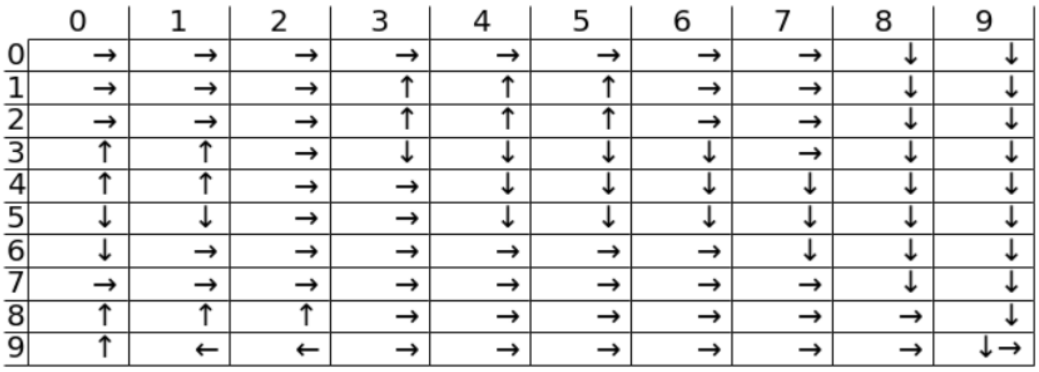


Figure 2.7. Heat map of the optimal state values (Reward function 1)

Question 4:
From the distribution of the optimal state values, we can see that state 99 has highest magnitude in the heat map, as it has the only positive reward. State away from state 99 will drop with respect to the distance between itself and state 99. This is because the further the distance is, it will take more steps to

reach the positive reward. With the discount factor 0.8, there is decay for state transits to further states, so the overall trend of magnitude is lower at the top-left of the heat map and higher at bottom-right. What's more, we can see that the magnitude around the states with negative reward is lower. This makes sense because the value is the expected total discounted reward.

Question 5:
The optimal policy is shown below:

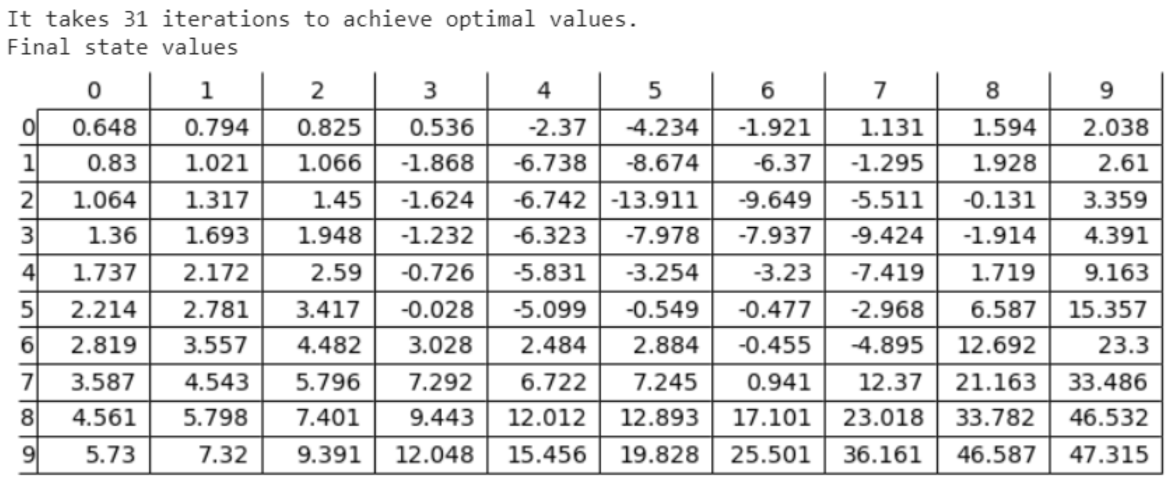


	0	1	2	3	4	5	6	7	8	9
0	→	→	→	→	→	→	→	→	↓	↓
1	→	→	→	↑	↑	↑	→	→	↓	↓
2	→	→	→	↑	↑	↑	→	→	↓	↓
3	↑	↑	→	↓	↓	↓	↓	→	↓	↓
4	↑	↑	→	→	↓	↓	↓	↓	↓	↓
5	↓	↓	→	→	↓	↓	↓	↓	↓	↓
6	↓	→	→	→	→	→	→	↓	↓	↓
7	→	→	→	→	→	→	→	→	↓	↓
8	↑	↑	↑	→	→	→	→	→	→	↓
9	↑	←	←	→	→	→	→	→	→	↓→

Figure 2.8. The optimal policy of the agent using reward function 1

The optimal policy of the agent match our intuition.
What we think before this figure comes out is that our agent will go to the state 99 as quick as possible. In our figure shown below, the agent will go to state 99 either by moving right or moving down, it is what will happen in our intuition. Therefore, it is possible for the agent to compute the optimal action to take at each state by observing the optimal values of it's neighboring states.

Question 6:
The state values using reward function 2 is shown below:



```
It takes 31 iterations to achieve optimal values.
Final state values
```

	0	1	2	3	4	5	6	7	8	9
0	0.648	0.794	0.825	0.536	-2.37	-4.234	-1.921	1.131	1.594	2.038
1	0.83	1.021	1.066	-1.868	-6.738	-8.674	-6.37	-1.295	1.928	2.61
2	1.064	1.317	1.45	-1.624	-6.742	-13.911	-9.649	-5.511	-0.131	3.359
3	1.36	1.693	1.948	-1.232	-6.323	-7.978	-7.937	-9.424	-1.914	4.391
4	1.737	2.172	2.59	-0.726	-5.831	-3.254	-3.23	-7.419	1.719	9.163
5	2.214	2.781	3.417	-0.028	-5.099	-0.549	-0.477	-2.968	6.587	15.357
6	2.819	3.557	4.482	3.028	2.484	2.884	-0.455	-4.895	12.692	23.3
7	3.587	4.543	5.796	7.292	6.722	7.245	0.941	12.37	21.163	33.486
8	4.561	5.798	7.401	9.443	12.012	12.893	17.101	23.018	33.782	46.532
9	5.73	7.32	9.391	12.048	15.456	19.828	25.501	36.161	46.587	47.315

Figure 2.9. The optimal values of states using reward function 2

Question 7:

We plot the heat map of the optimal state values across the 2-D grid of Reward function 2:

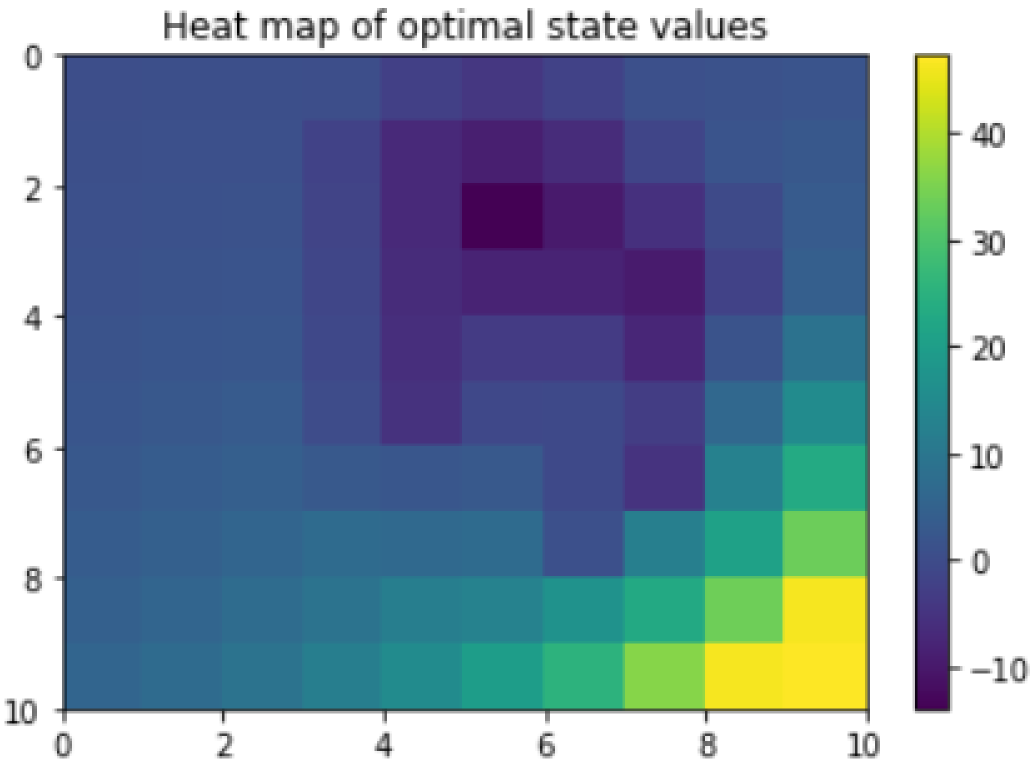
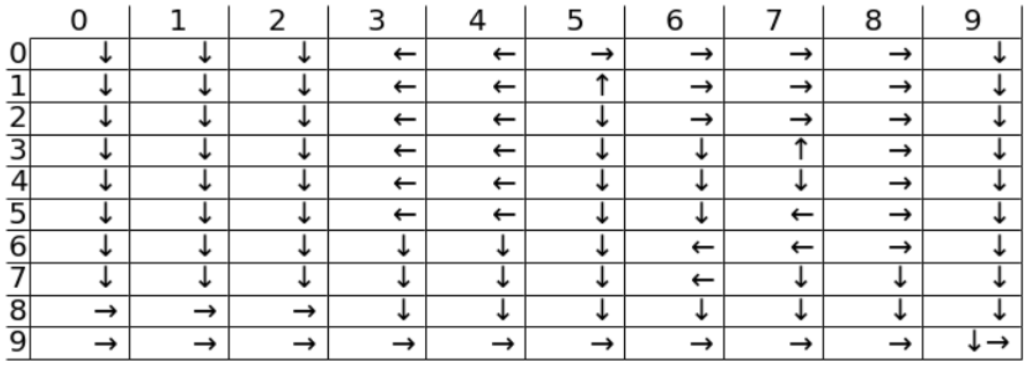


Figure 2.10. Heat map of the optimal state values (Reward function 2)

Similar to reward function 1, we can see that state 99 has highest magnitude in the heat map, as it has the only positive reward. Compared with reward function 1, this state value is larger, about ten times of that in reward function 1, as the reward in state 99 is 10 instead of 1. Also, the overall trend of magnitude is lower at the top-left of the heat map and higher at bottom-right.
From the distribution above, we can also see that there is a circle-like region with lowest magnitude in right-half of the heat map. This is caused by states with negative reward -100 and their neighbors.

Question 8:

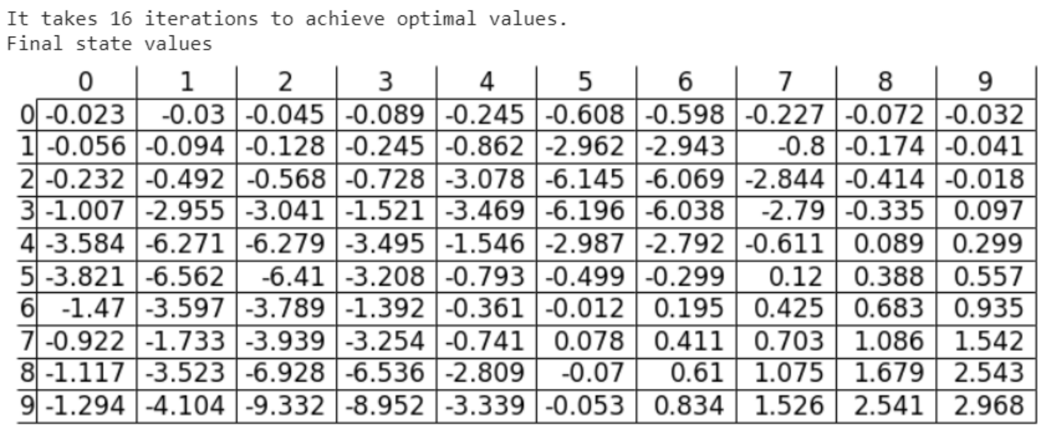


	0	1	2	3	4	5	6	7	8	9
0	↓	↓	↓	←	←	→	→	→	→	↓
1	↓	↓	↓	←	←	↑	→	→	→	↓
2	↓	↓	↓	←	←	↓	→	→	→	↓
3	↓	↓	↓	←	←	↓	↓	↑	→	↓
4	↓	↓	↓	←	←	↓	↓	↓	→	↓
5	↓	↓	↓	←	←	↓	↓	←	→	↓
6	↓	↓	↓	↓	↓	↓	←	←	→	↓
7	↓	↓	↓	↓	↓	↓	←	↓	↓	↓
8	→	→	→	↓	↓	↓	↓	↓	↓	↓
9	→	→	→	→	→	→	→	→	→	↓→

Figure 2.11. Heat map of the optimal state values (Reward function 2)

The optimal policy of the agent matches our intuition. We all learned that the final state has the largest optimal value, the overall trend is the upper state tends to go down and the left states tends to go right. However, we can clearly see that there are some negative optimal values in the figure shown above, the trends are interrupted by these points. The neighborhoods for these low state values tend to flow to states with higher values. Therefore, this optimal policy of the agent still matches our intuition.

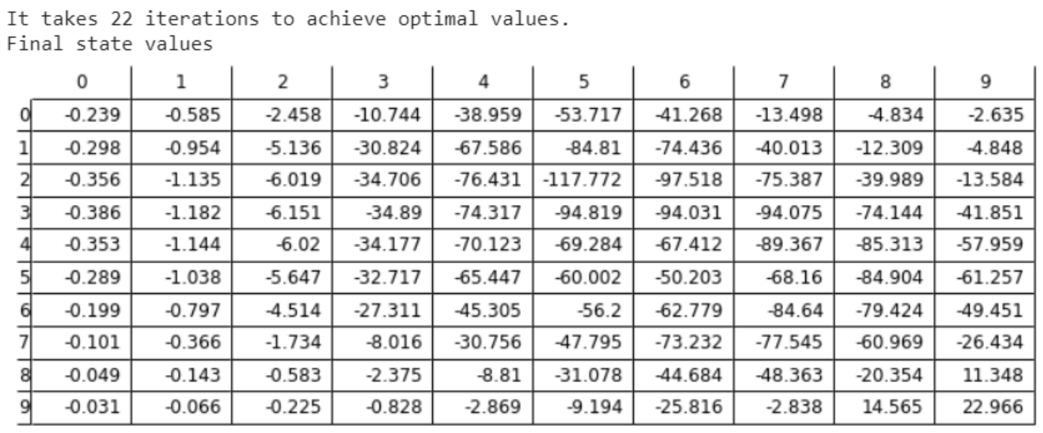
Question 9:
In this question, we changed w to w = 0.6 and find the optimal policy map similar to previous question for reward function 1 and reward function 2. The results are shown below:



```
It takes 16 iterations to achieve optimal values.
Final state values
```

	0	1	2	3	4	5	6	7	8	9
0	-0.023	-0.03	-0.045	-0.089	-0.245	-0.608	-0.598	-0.227	-0.072	-0.032
1	-0.056	-0.094	-0.128	-0.245	-0.862	-2.962	-2.943	-0.8	-0.174	-0.041
2	-0.232	-0.492	-0.568	-0.728	-3.078	-6.145	-6.069	-2.844	-0.414	-0.018
3	-1.007	-2.955	-3.041	-1.521	-3.469	-6.196	-6.038	-2.79	-0.335	0.097
4	-3.584	-6.271	-6.279	-3.495	-1.546	-2.987	-2.792	-0.611	0.089	0.299
5	-3.821	-6.562	-6.41	-3.208	-0.793	-0.499	-0.299	0.12	0.388	0.557
6	-1.47	-3.597	-3.789	-1.392	-0.361	-0.012	0.195	0.425	0.683	0.935
7	-0.922	-1.733	-3.939	-3.254	-0.741	0.078	0.411	0.703	1.086	1.542
8	-1.117	-3.523	-6.928	-6.536	-2.809	-0.07	0.61	1.075	1.679	2.543
9	-1.294	-4.104	-9.332	-8.952	-3.339	-0.053	0.834	1.526	2.541	2.968

Figure 2.12. The optimal values of states with w changed using reward function 1



```
It takes 22 iterations to achieve optimal values.
Final state values
```

	0	1	2	3	4	5	6	7	8	9
0	-0.239	-0.585	-2.458	-10.744	-38.959	-53.717	-41.268	-13.498	-4.834	-2.635
1	-0.298	-0.954	-5.136	-30.824	-67.586	-84.81	-74.436	-40.013	-12.309	-4.848
2	-0.356	-1.135	-6.019	-34.706	-76.431	-117.772	-97.518	-75.387	-39.989	-13.584
3	-0.386	-1.182	-6.151	-34.89	-74.317	-94.819	-94.031	-94.075	-74.144	-41.851
4	-0.353	-1.144	-6.02	-34.177	-70.123	-69.284	-67.412	-89.367	-85.313	-57.959
5	-0.289	-1.038	-5.647	-32.717	-65.447	-60.002	-50.203	-68.16	-84.904	-61.257
6	-0.199	-0.797	-4.514	-27.311	-45.305	-56.2	-62.779	-84.64	-79.424	-49.451
7	-0.101	-0.366	-1.734	-8.016	-30.756	-47.795	-73.232	-77.545	-60.969	-26.434
8	-0.049	-0.143	-0.583	-2.375	-8.81	-31.078	-44.684	-48.363	-20.354	11.348
9	-0.031	-0.066	-0.225	-0.828	-2.869	-9.194	-25.816	-2.838	14.565	22.966

Figure 2.13. The optimal values of states with w changed using reward function 2

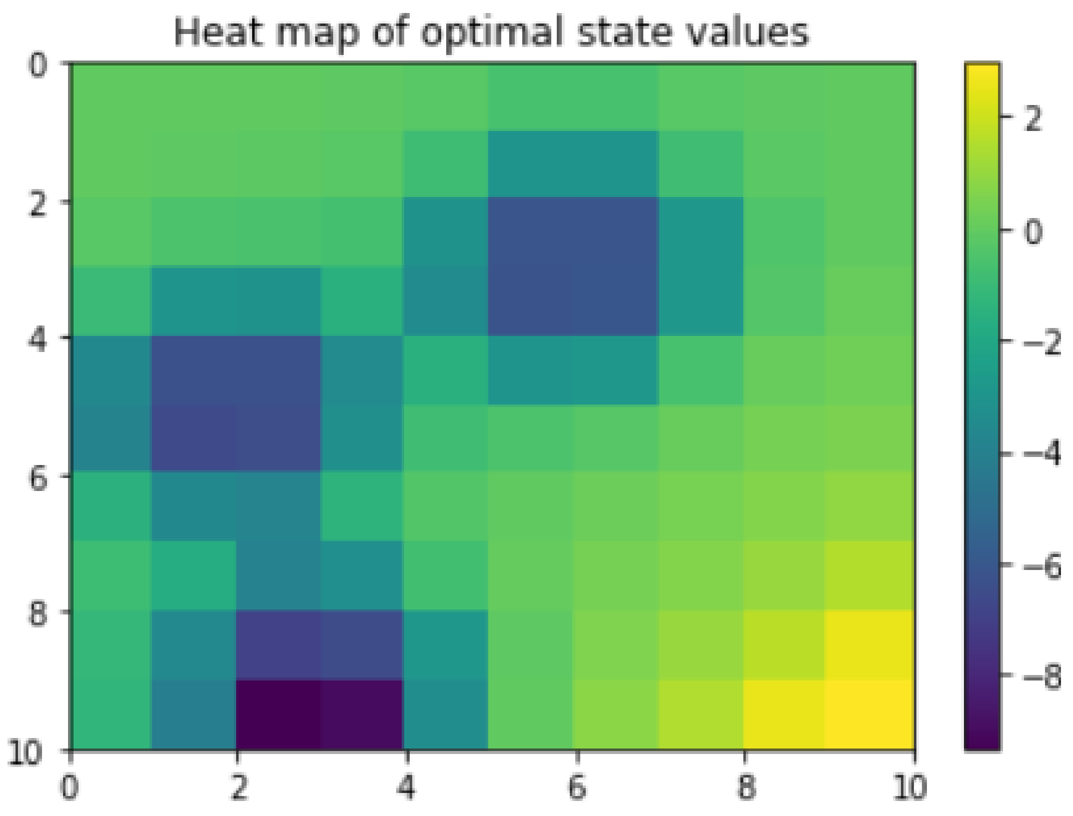


Figure 2.14. Heat map of the optimal state values with w changed using reward function 1

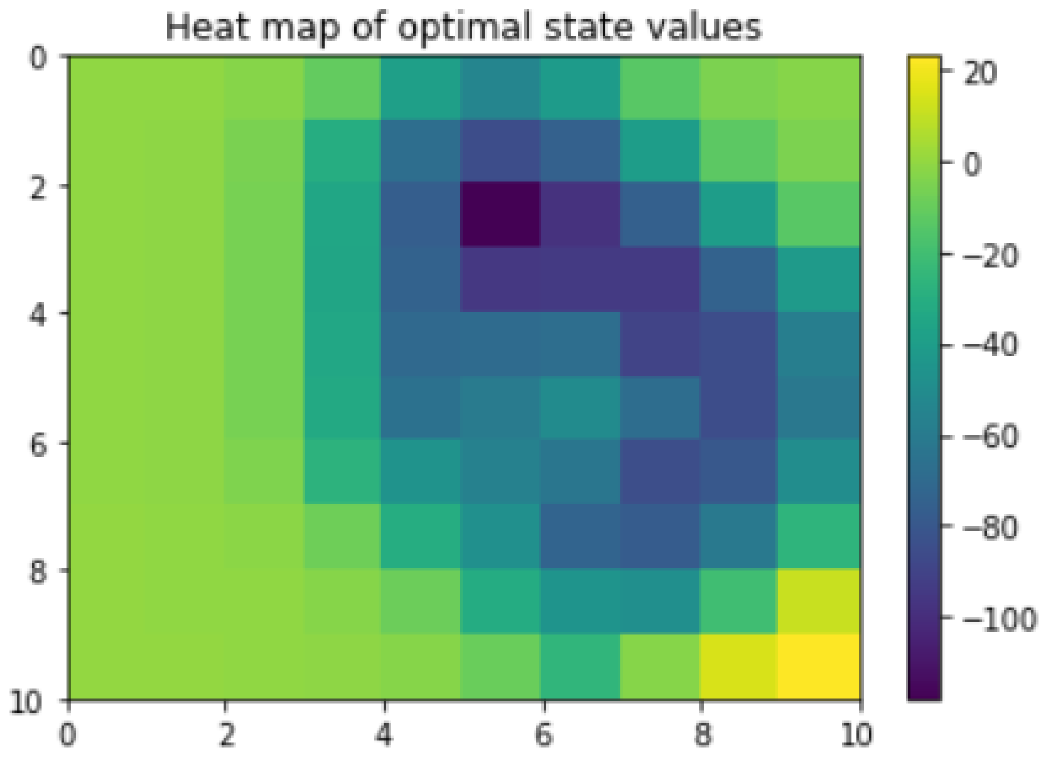
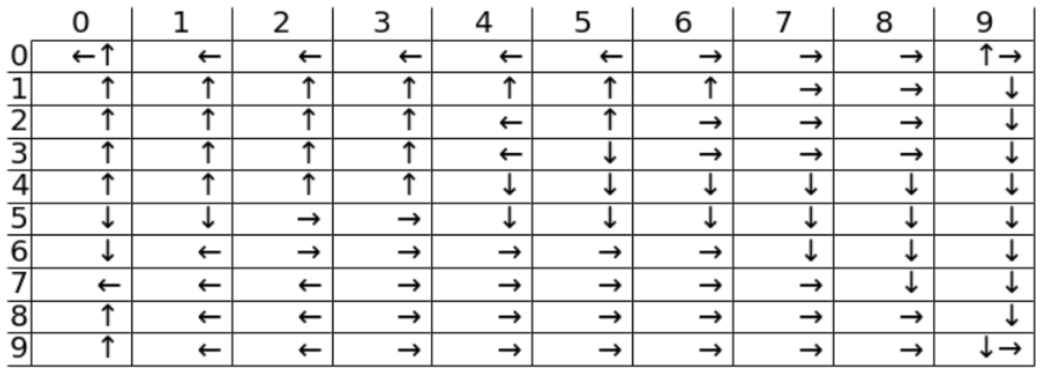


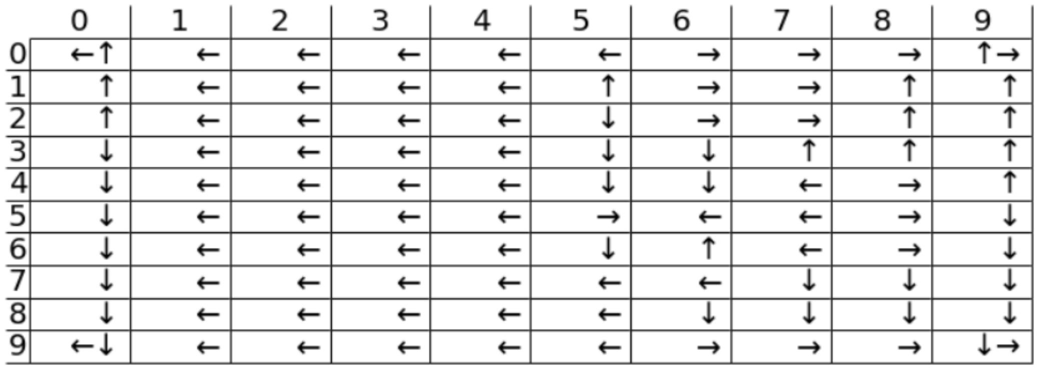
Figure 2.15. Heat map of the optimal state values with w changed using reward function 2

From the state values and heat map, we can see that there are more negative state values for both reward functions with w = 0.6, compared with w = 0.1. Especially, the state values at states with negative reward and their neighbors are much lower than that with w = 0.1. This make sense as the probability of a state transits to states with negative rewards is much larger, as larger w will lead to more random transitions of a state.



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

Figure 2.16. The optimal policy of the agent with w changed using reward function 1



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

Figure 2.17. The optimal policy of the agent with w changed using reward function 2

We can see that for both reward functions with w = 0.6, there are much more states where optimal actions point to the top or left of the map, compared with w = 0.1. Intuitively, this should not be an optimal action as the goal is to reach state 99, which is at the right-bottom of the table. This happens as higher w will lead to higher probability of random transitions, which may not be a good action that will make the agent transits to the final state (state 99).
From the comparison above, we can conclude that the new w = 0.6 performs worse than w = 0.1, as it will cause too much random transitions, instead of a greedy one. Therefore, we still use w = 0.1, which give rise to better optimal policy, and use this w for the next stages of the project.

3 Inverse Reinforcement learning (IRL)

Question 10:

$$x = \begin{bmatrix} R_{|S|\times 1} \\ t_{|S|\times 1} \\ u_{|S|\times 1} \end{bmatrix}$$

$$c = \begin{bmatrix} 0_{|S|\times 1} \\ 1_{|S|\times 1} \\ -\lambda * 1_{|S|\times 1} \end{bmatrix}$$

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

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

Question 11:
We repeat the process for all 500 values of λ to get 500 data points. The λ against Accuracy plot is as follows:

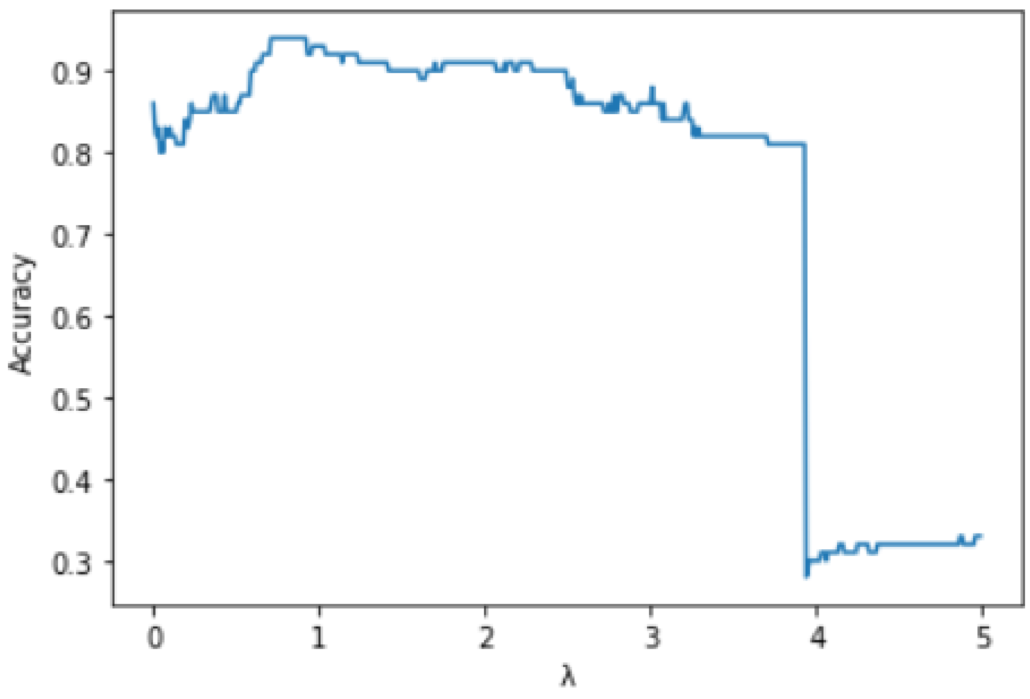


Figure 3.1. λ against Accuracy (Reward function 1)

Question 12:
From the plot in question 11, we can see that the when λ is in [0.71, 0.92], the accuracy reaches maximum 0.94. So we set $\lambda_{max}^{(1)}$ to be 0.82, which is in the middle of the range.

Question 13:
We generate heat maps of the ground truth reward and the extracted reward by using $\lambda_{max}^{(1)} = 0.82$:

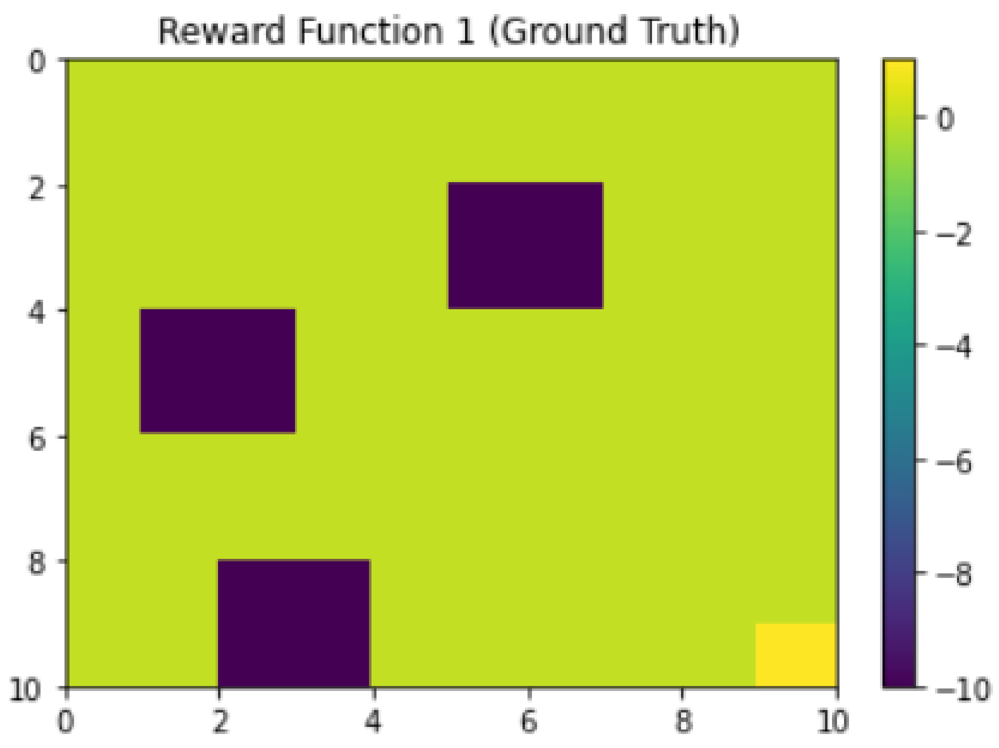


Figure 3.2. Heat map of the ground truth reward

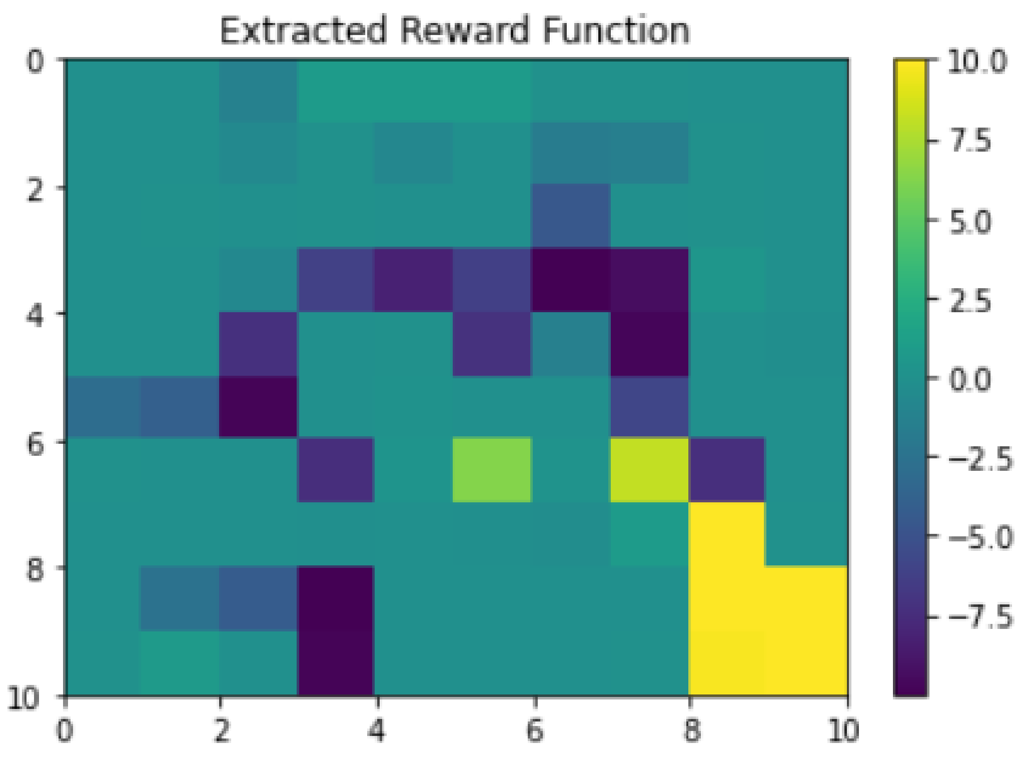


Figure 3.3. Heat map of the extracted reward

Question 14:
We use the extracted reward function computed in question 13 to compute the optimal values of the states

in the 2-D grid, then we generate a heat map of the optimal state values across the 2-D grid as follows:

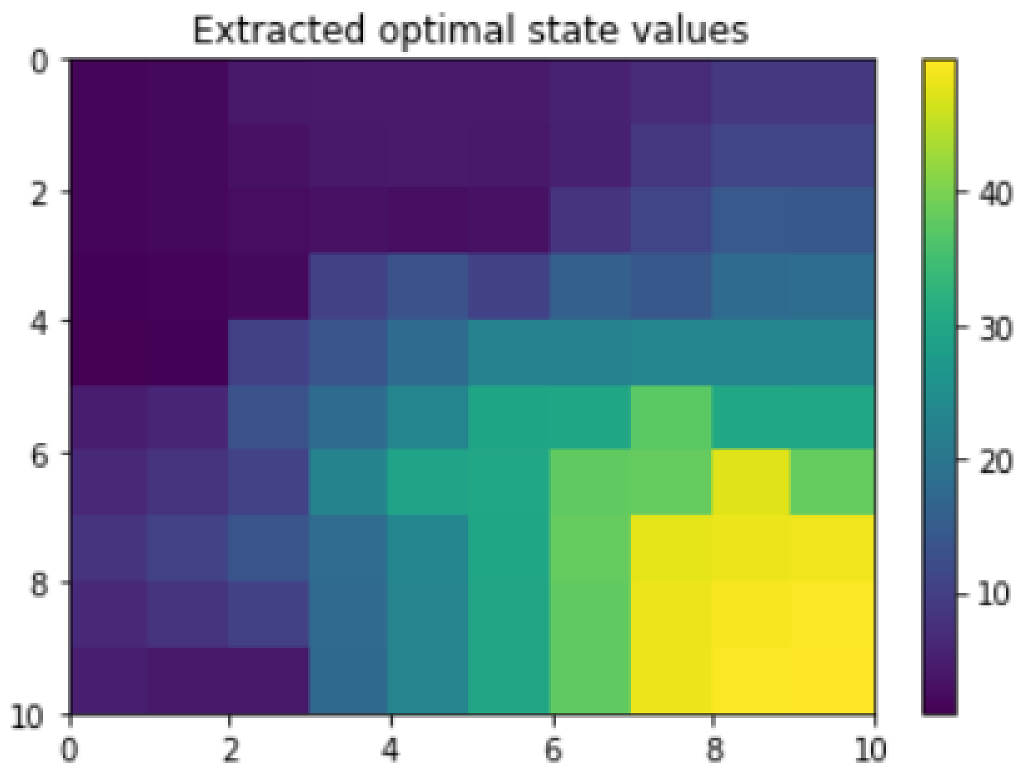
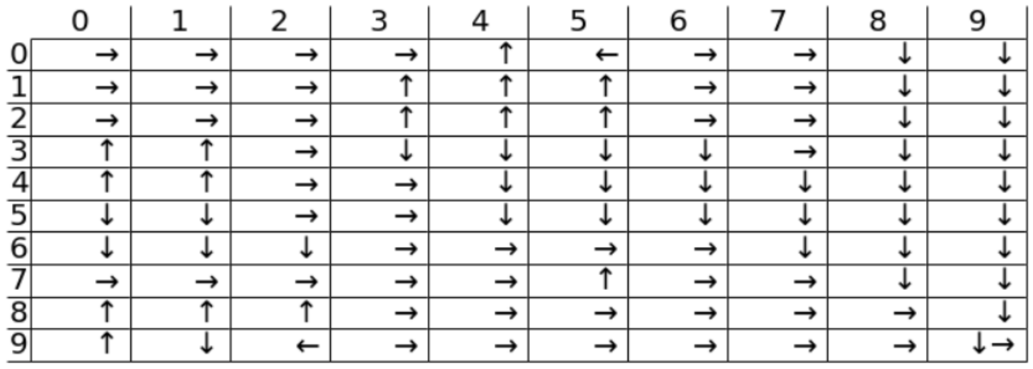


Figure 3.4. Heat map of the optimal state values across the 2-D grid (Reward function 1)

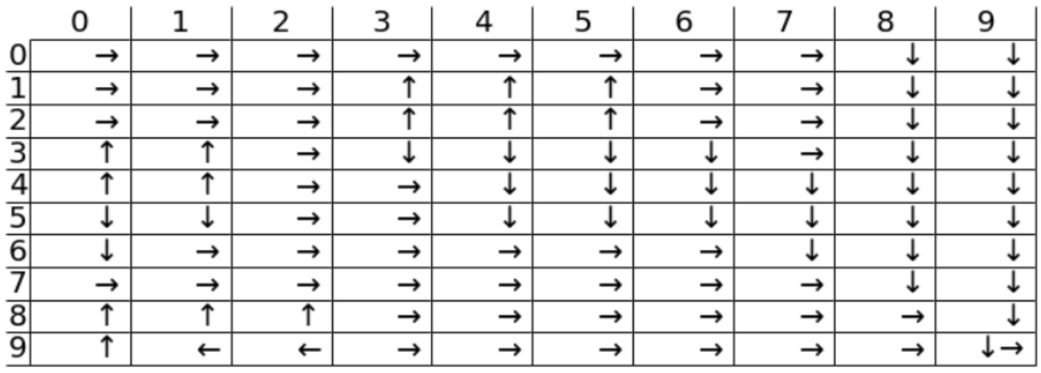
Question 15:
Similarities:
The heat map for question 3 and question 14 both follow Gaussian shape. Both heat maps hold similar trend: the up left corner represents the smallest one and the bottom right corner represents the largest one. Generally, the larger the distance to the bottom-right corner is, the smaller the state value is in the heat map. This make sense as the extracted reward function are close to the ground truth reward function, where most states hold zero reward. After value iteration, the state values of both cases should be close.
Difference:
We can notice that scale of the state values calculated from the extracted reward function is larger than that from the ground truth reward function. This is caused by the higher upper bound of the extracted reward function in linear programming.
Question 16:
We use the extracted reward function found in question 13 to compute the optimal policy of the agent:



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

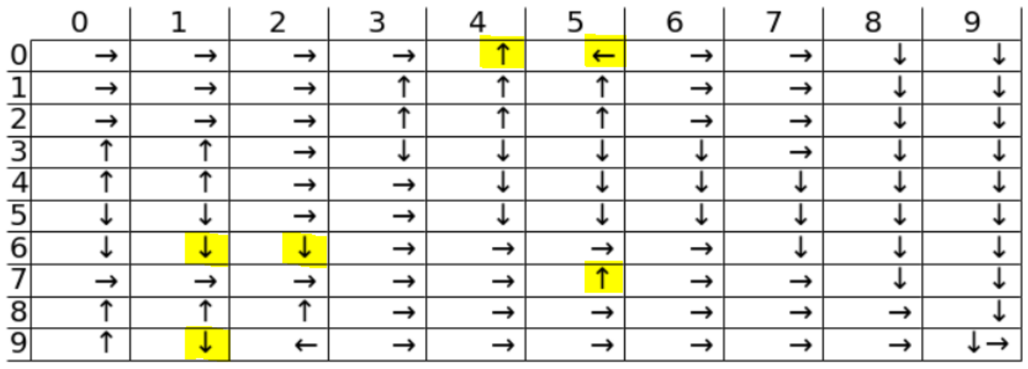
Figure 3.5. Optimal policy of the agent (Reward function 1)

Question 17:



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

Figure 3.6. Optimal policy of the agent with reward function 1



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

Figure 3.7. Optimal policy of the agent with extracted reward function 1

From the two plots above, we can see that the optimal policy from reward function 1 and the extracted reward function 1 are almost the same, which makes sense as the distribution of state values with the extracted reward function is very similar to that with the ground truth reward function, which has been stated in Question 15.
We can also see that there are 6 states where optimal policy is different, which matches the accuracy 0.94. The differences are marked in Figure 3.7. It can be explained by the difference of the extracted reward

function and the ground truth reward. From the heat map of the extracted reward function in Question 13 we can see that the states where the optimal policy is different have non-zero neighbors, which can explain why the optimal policy in these states are different from the original reward function.
Question 18:
We repeat the process for all 500 values of λ to get 500 data points. The relationship between λ and Accuracy is as follows:

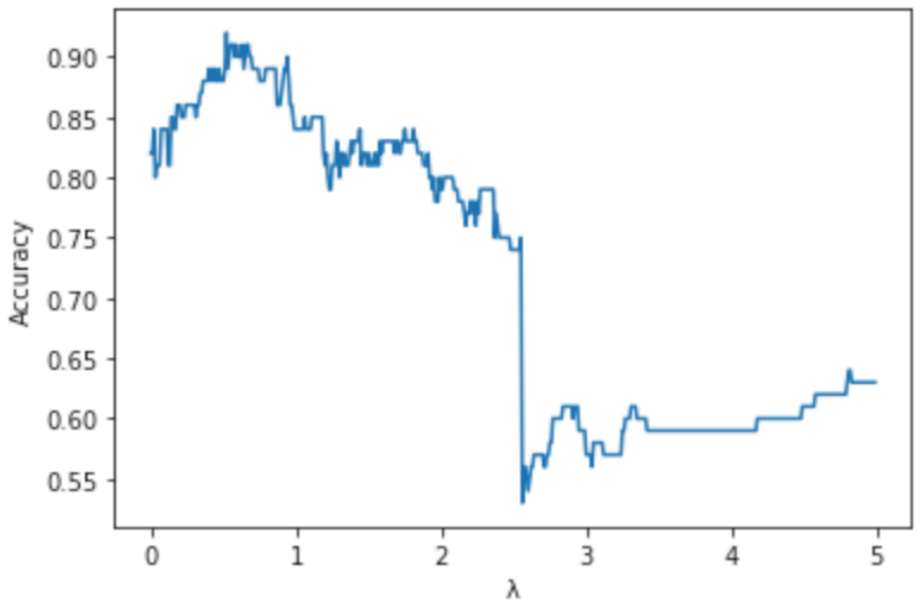


Figure 3.8. λ against Accuracy (Reward function 2)

Question 19:
We use the plot in question 18 to compute the value of λ for which accuracy is maximum:

$$\lambda_{max}^{(1)} \approx 0.52$$

The corresponding accuracy is 0.92.
Question 20:
We generate heat maps of the ground truth reward and the extracted reward by using $\lambda_{max}^{(1)} = 0.52$:

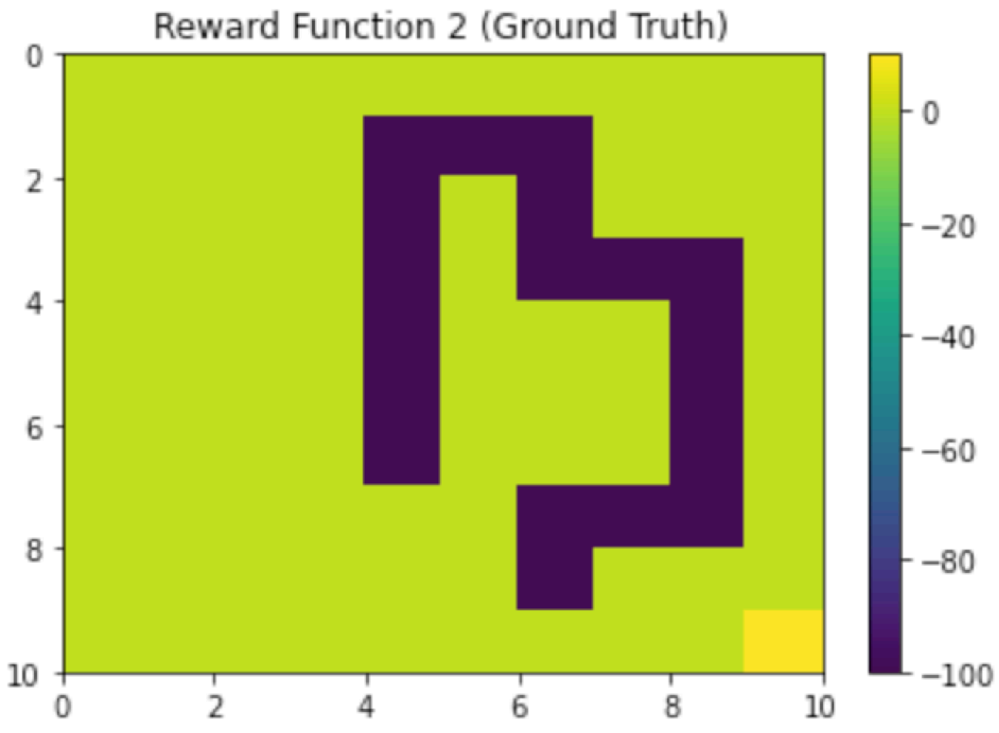


Figure 3.9. Heat map of the ground truth reward

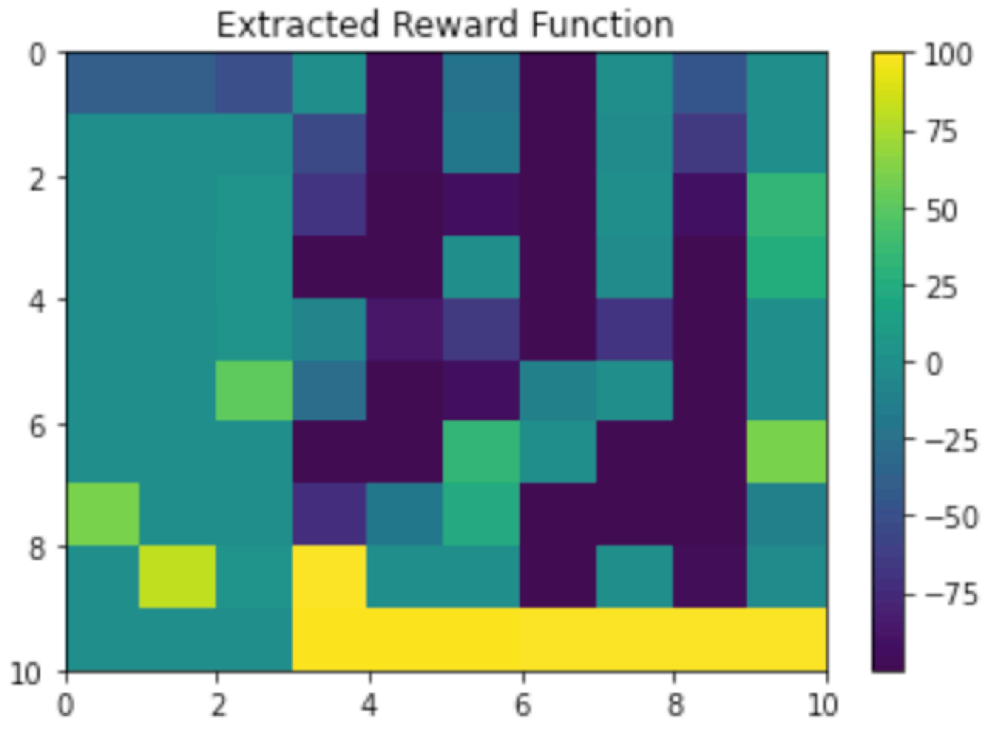


Figure 3.10. Heat map of the extracted reward

Question 21:
We use the extracted reward function computed in question 13 to compute the optimal values of the states

in the 2-D grid, then we generate a heat map of the optimal state values across the 2-D grid as follows:

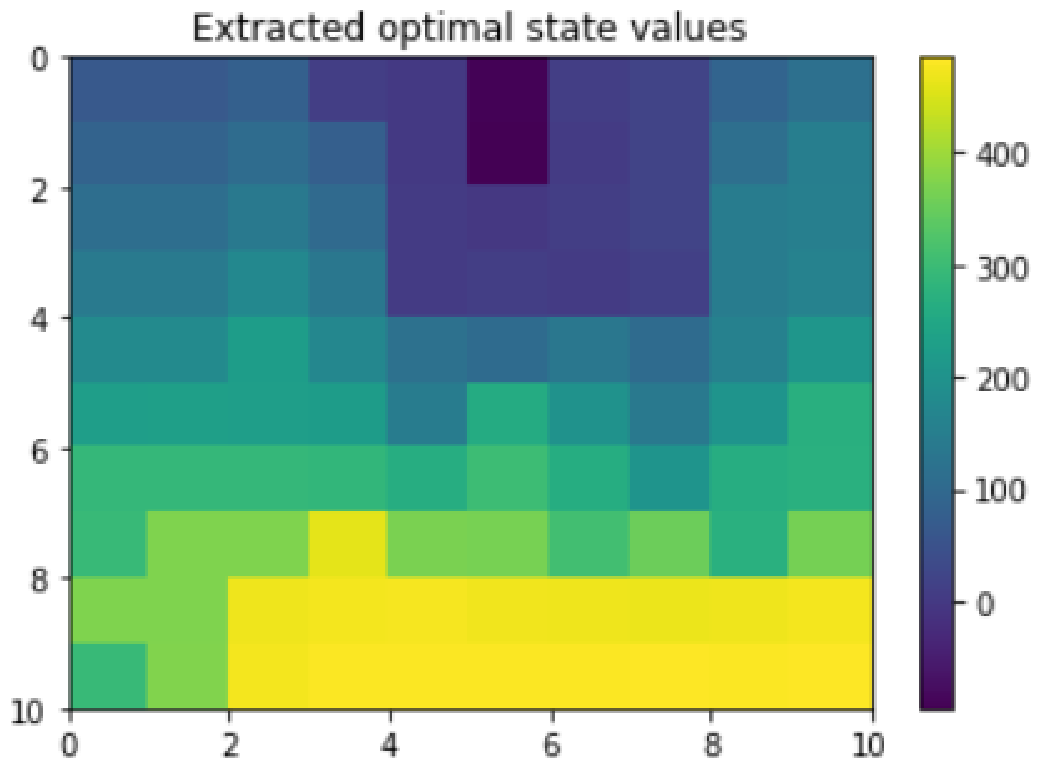
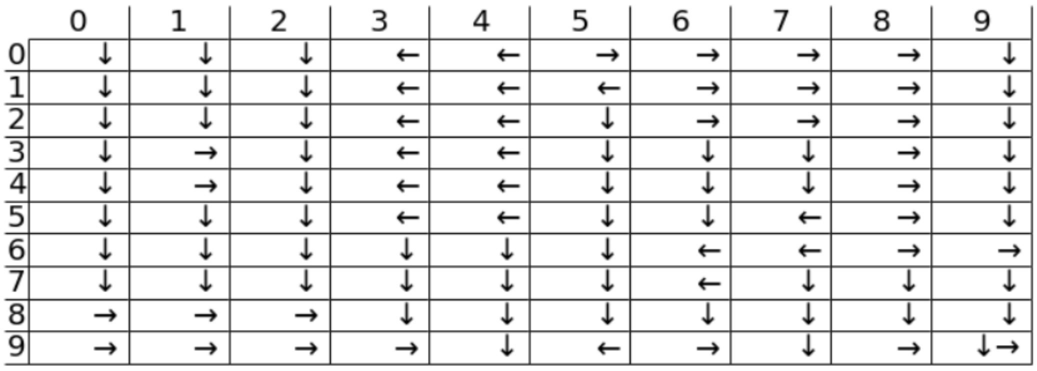


Figure 3.11. Heat map of the optimal state values across the 2-D grid

Question 22:
Similarities: The heat map from the extracted reward function and the ground truth reward function has similar trend: both have high state values at the bottom-right of the heat map, and low state values at the top-left.
Differences:
The difference between the two heat maps is quite obvious. First, we can notice that scale of the state values calculated from the extracted reward function is larger than that from the ground truth reward function. This is caused by the higher upper bound of the extracted reward function in linear programming. Second, the heat map of the ground truth reward function decrease from bottom-right to top-left, while this trend in the heat map of the extracted reward function is not so obvious. Instead, it decrease from bottom to top of the heat map, which is caused by large reward in the last row of the extracted reward function.

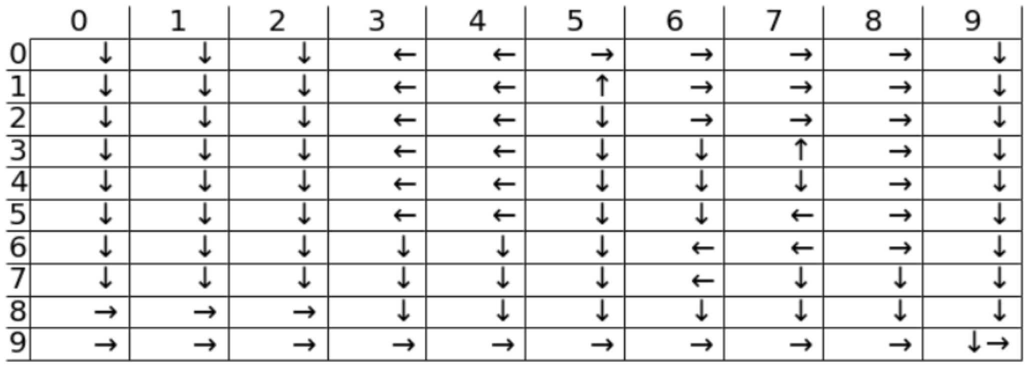
Question 23:



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

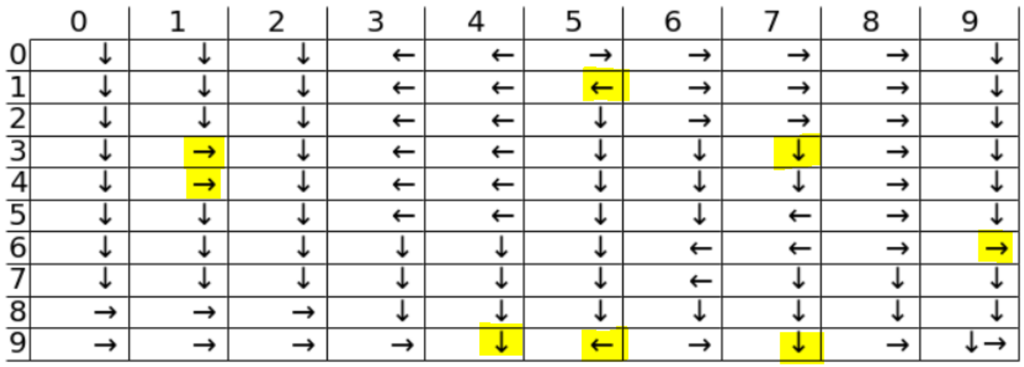
Figure 3.12. Optimal policy of the agent (Reward function 2)

Question 24:



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

Figure 3.13. Optimal policy for ground truth reward function 2

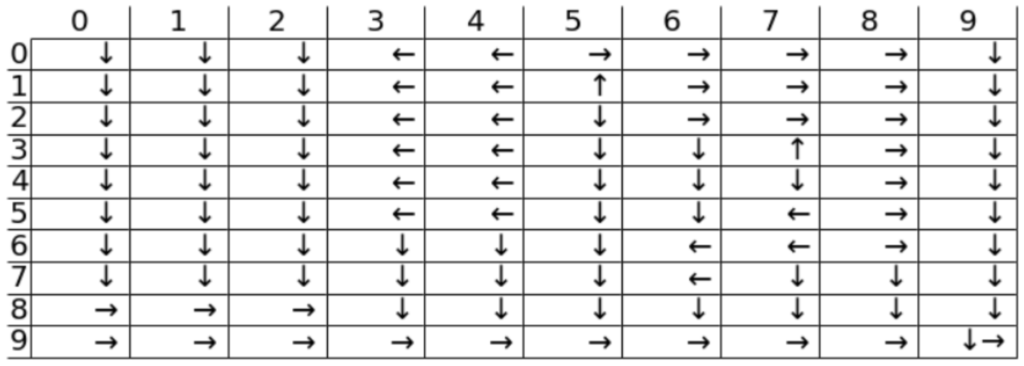


	0	1	2	3	4	5	6	7	8	9
0	↓	↓	↓	←	←	→	→	→	→	↓
1	↓	↓	↓	←	←	←	→	→	→	↓
2	↓	↓	↓	←	←	↓	→	→	→	↓
3	↓	→	↓	←	←	↓	↓	↓	→	↓
4	↓	→	↓	←	←	↓	↓	↓	→	↓
5	↓	↓	↓	←	←	↓	↓	←	→	↓
6	↓	↓	↓	↓	↓	↓	←	←	→	→
7	↓	↓	↓	↓	↓	↓	←	↓	↓	↓
8	→	→	→	↓	↓	↓	↓	↓	↓	↓
9	→	→	→	→	↓	←	→	↓	→	↓→

Figure 3.14. Optimal policy for extracted reward function 2

From the two plots above, we can see that the optimal policy from reward function 2 and the extracted reward function 2 are almost the same, which makes sense as the distribution of state values with the extracted reward function is very similar to that with the ground truth reward function, which has been stated in Question 22.
We can also see that there are 8 states where optimal policy is different, which matches the accuracy 0.92. The differences are marked in Figure 3.14. It can be explained by the difference of the extracted reward

function and the ground truth reward. From the heat map of the extracted reward function in Question 20 we can see that the states where the optimal policy is different have non-zero neighbors, which can explain why the optimal policy in these states are different from the original reward function.
Question 25:
From the plot in Question 23, we can see that there are several local optimalities. When the agent transits to a local optimal state, it will get stuck in that state and will not transit to the global optimal state (state 99). There are two discrepancies which might lead to the result. First, similar state values between two states could make the possibility of choosing the four actions become close, and meanwhile the algorithm could stop updating earlier than it should. This can be fixed by setting a smaller threshold ϵ in the value iteration algorithm.
We perform the modification by changing ϵ from 0.01 to 1×10^{-9}. We re-run the modified value iteration algorithm to compute the optimal policy of the agent with reward function 2. We still use $\lambda_{max}^{(1)}$=0.52. The optimal policy is shown below:



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

Figure 3.15. Modified optimal policy for extracted reward function 2

The modified accuracy is 1.0, which means the optimal optimal policy of the extracted reward function is exactly the same as that of the ground truth reward function.
We also plot the accuracy v.s. λ, which is shown below:

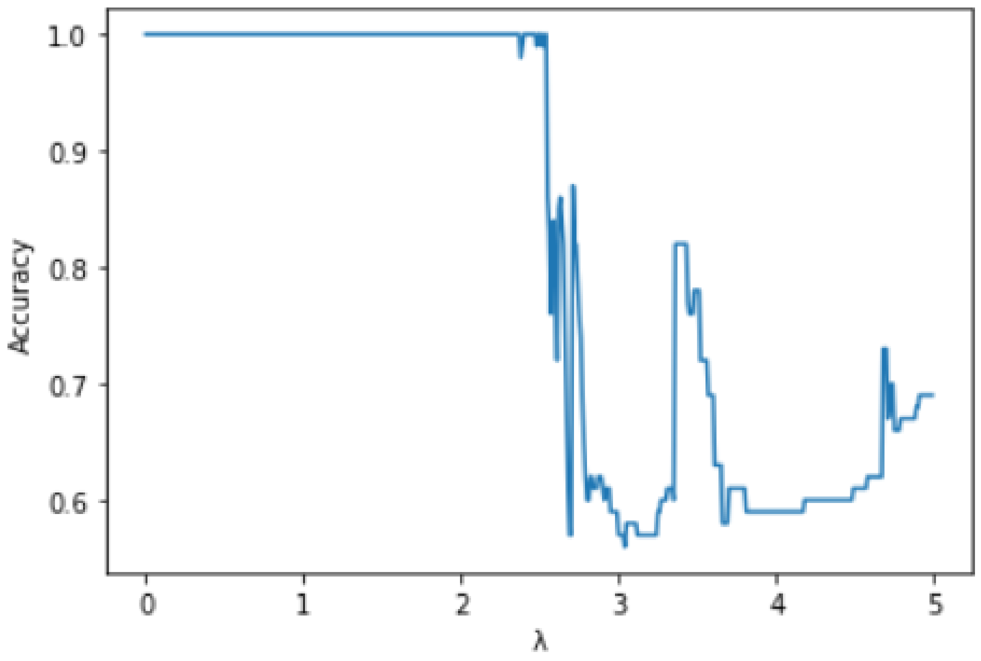


Figure 3.16. λ against Accuracy (Modified)

From the plot we can see that the accuracy reaches 1.0 when $\lambda<2$. The modification greatly improves the maximum accuracy.
Although we can see that modification of the value iteration algorithm can already improve the maximum accuracy to 1.0, we could still notice another discrepancy: the extracted reward function 2 have much larger values in the last row than other states, and this mistake could be another reason of the wrong actions. This discrepancy may be caused by the limitation of the simple IRL algorithm: the Linear Program maximizes the gap between the values of the optimal action and the suboptimal action, while this assumption might not be the real case.