

ECE 232E: Large-Scale Social and Complex Networks: Models and Algorithms

# Project 3: Reinforcement Learning and Inverse Reinforcement Learning

---

Akshay Sharma (504946035)

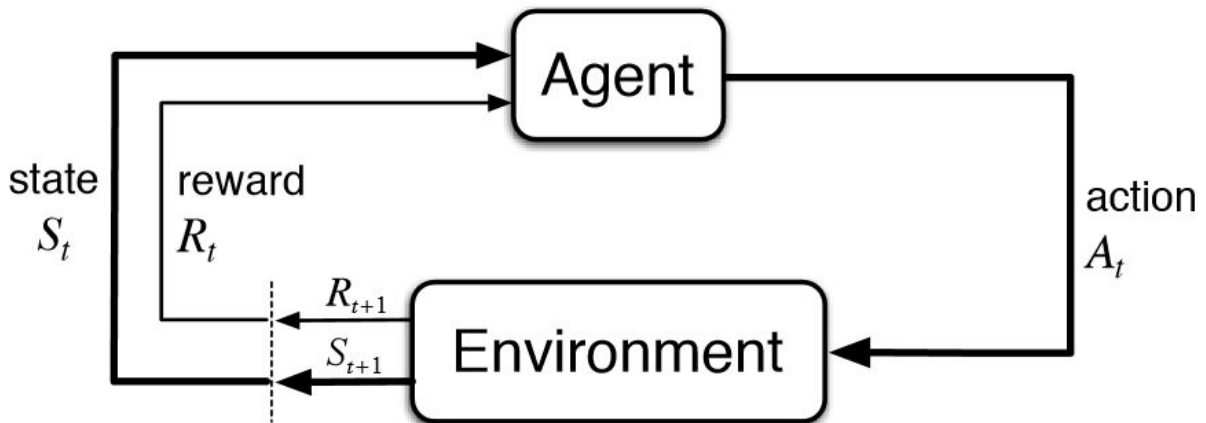
Anoosha Sagar (605028604)

Nikhil Thakur(804946345)

Rahul Dhavalikar (205024839)

---

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



Reinforcement Learning (RL) is a machine learning technique that enables an agent to learn in an interactive environment by trial and error using feedback from its own actions and experiences. The image above represents the basic idea and the elements involved in a reinforcement learning model.

The main components in Reinforcement Learning are:

- **Agent** - an agent takes actions
- **Environment** - the world through which an agent moves. The environment takes the agent's current state and action as input, and returns as output the agent's reward and next state.
- **Action (A)** - it is a set of all possible moves that an agent can make.
- **State (S)** - it is a concrete and immediate situation in which the agent finds itself
- **Reward (R)** - it is the feedback by which we measure the success or failure of an agent's actions.
- **Policy ( $\pi$ )** - it is the strategy that the agent employs to determine the next action based on the current state
- **Discount Factor** - the discount factor is multiplied with future rewards as discovered by the agent in order to dampen their effect on the agent's choice of action. It makes future rewards worth less than immediate rewards.
- **Value** - the expected long-term return with discount, as opposed to the short term reward R.

Environments are functions that transform an action taken in the current state into the next state and a reward and agents are functions that transform the new state and reward into the next action.

For this project, the state space is a 2-D square grid with 100 states. The 2-D square grid along with the numbering of states is shown below:

	0	1	2	3	4	5	6	7	8	9
0	0.0	10.0	20.0	30.0	40.0	50.0	60.0	70.0	80.0	90.0
1	1.0	11.0	21.0	31.0	41.0	51.0	61.0	71.0	81.0	91.0
2	2.0	12.0	22.0	32.0	42.0	52.0	62.0	72.0	82.0	92.0
3	3.0	13.0	23.0	33.0	43.0	53.0	63.0	73.0	83.0	93.0
4	4.0	14.0	24.0	34.0	44.0	54.0	64.0	74.0	84.0	94.0
5	5.0	15.0	25.0	35.0	45.0	55.0	65.0	75.0	85.0	95.0
6	6.0	16.0	26.0	36.0	46.0	56.0	66.0	76.0	86.0	96.0
7	7.0	17.0	27.0	37.0	47.0	57.0	67.0	77.0	87.0	97.0
8	8.0	18.0	28.0	38.0	48.0	58.0	68.0	78.0	88.0	98.0
9	9.0	19.0	29.0	39.0	49.0	59.0	69.0	79.0	89.0	99.0

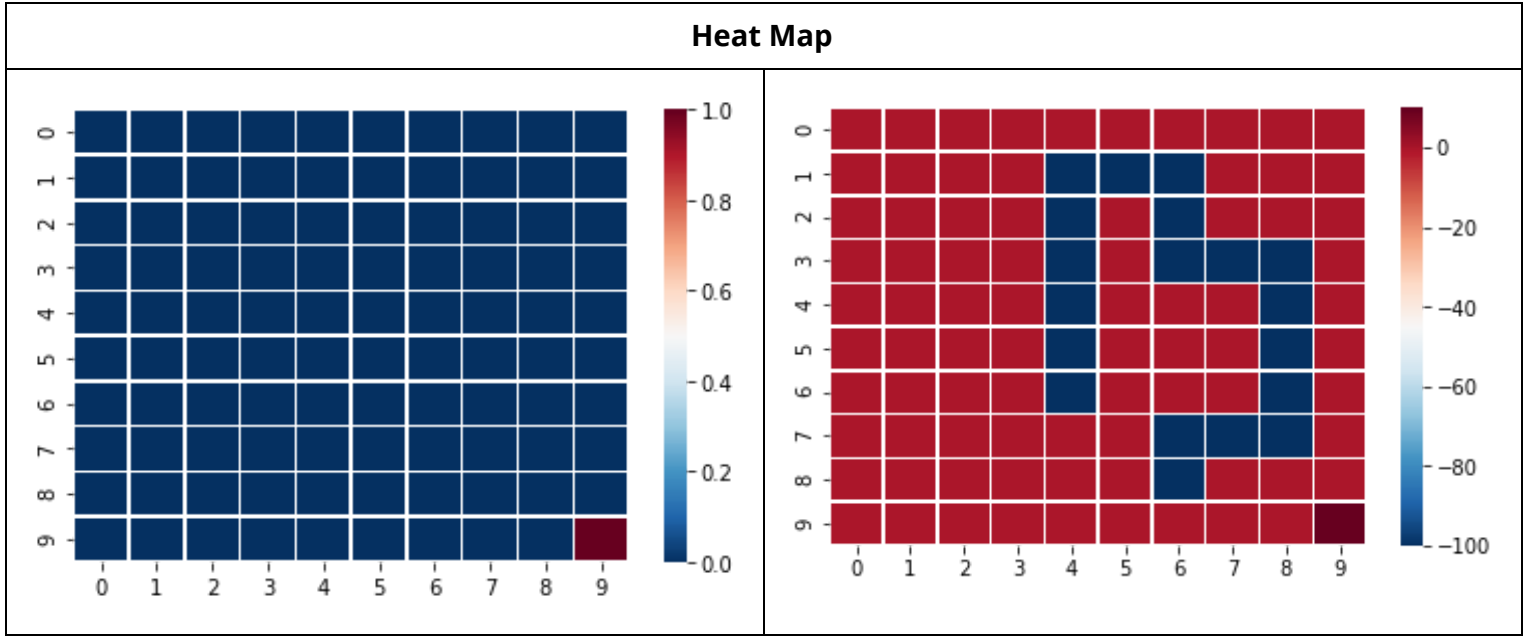
#### Reward Function 1

#### Reward Function 2

#### Matrix Representation

	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	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0

	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	-100.0	-100.0	-100.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	-100.0	0.0	-100.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	-100.0	0.0	-100.0	-100.0	-100.0	0.0
4	0.0	0.0	0.0	0.0	-100.0	0.0	0.0	0.0	-100.0	0.0
5	0.0	0.0	0.0	0.0	-100.0	0.0	0.0	0.0	-100.0	0.0
6	0.0	0.0	0.0	0.0	-100.0	0.0	0.0	0.0	-100.0	0.0
7	0.0	0.0	0.0	0.0	0.0	0.0	-100.0	-100.0	-100.0	0.0
8	0.0	0.0	0.0	0.0	0.0	0.0	-100.0	0.0	0.0	0.0
9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	10.0



For our problem, we are given two different types of reward functions. We have generated the heat maps for both of them and displayed them in the table above.

In reward function 1, we observe that all the rewards except the last state are zero. The last state has a positive reward of 1. An environment with this reward can be compared to driver driving in an open field with a goal towards the other diagonal end. Apart from the last state, all states have the same reward and hence the agent can go anywhere.

In reward function 2, we see that some states have a negative reward of -100 and a majority of states having zero reward. We again observe that the final state has the maximum reward. The presence of a negative reward can be thought of as having an obstacle in the environment which would enable the agent to get a better understanding of the environment with well defined paths. This can be thought of a scenario of a driver driving in the field with obstacles or in the city.

2. **Create the environment of the agent using the information provided in section 2. To be specific, create the MDP by setting up the state-space, action set, transition probabilities, discount factor, and reward function. For creating the environment, use the following set of parameters:**

**Number of states = 100 (state space is a 10 by 10 square grid as displayed in figure 1)**

---

**Number of actions = 4 (set of possible actions is displayed in figure 2)**

**w = 0.1**

**Discount factor = 0.8**

**Reward function 1**

**After you have created the environment, then write an optimal state-value function that takes as input the environment of the agent and outputs the optimal value of each state in the grid. For the optimal state-value function, you have to implement the Initialization (lines 2-4) and Estimation (lines 5-13) steps of the Value Iteration algorithm. For the estimation step, use  $\epsilon = 0.01$ . For visualization purpose, you should generate a figure similar to that of figure 1 but with the number of state replaced by the optimal value of that state. In this question, you should have 1 plot.**

In this and the following questions, we implement the Value Iteration algorithm for finding the optimal policy. Value iteration starts at the 'end' and then works backwards, refining an estimate of either  $Q^*$  or  $V^*$ . The algorithm looks as follows:

---

```
1: procedure VALUE ITERATION( $\mathcal{P}_{ss'}^a, \mathcal{R}_{ss'}^a, \mathcal{S}, \mathcal{A}, \gamma$ ):
2:   for all  $s \in \mathcal{S}$  do                                     ▷ Initialization
3:      $V(s) \leftarrow 0$ 
4:   end for
5:    $\Delta \leftarrow \infty$ 
6:   while  $\Delta > \epsilon$  do                                     ▷ Estimation
7:      $\Delta \leftarrow 0$ 
8:     for all  $s \in \mathcal{S}$  do
9:        $v \leftarrow V(s)$ ;
10:       $V(s) \leftarrow \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a [\mathcal{R}_{ss'}^a + \gamma V(s')]$ ;
11:       $\Delta \leftarrow \max(\Delta, |v - V(s)|)$ ;
12:    end for
13:  end while
14:  for all  $s \in \mathcal{S}$  do                                     ▷ Computation
15:     $\pi(s) \leftarrow \arg \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a [\mathcal{R}_{ss'}^a + \gamma V(s')]$ ;
16:  end for
17: end procedure return  $\pi$ 
```

---

---

	0	1	2	3	4	5	6	7	8	9
0	0.04	0.06	0.09	0.13	0.17	0.22	0.29	0.38	0.49	0.61
1	0.06	0.09	0.12	0.16	0.22	0.29	0.38	0.49	0.63	0.79
2	0.09	0.12	0.16	0.22	0.29	0.38	0.49	0.64	0.82	1.02
3	0.13	0.16	0.22	0.29	0.38	0.49	0.64	0.82	1.05	1.32
4	0.17	0.22	0.29	0.38	0.49	0.64	0.82	1.05	1.35	1.70
5	0.22	0.29	0.38	0.49	0.64	0.82	1.05	1.35	1.73	2.18
6	0.29	0.38	0.49	0.64	0.82	1.05	1.35	1.73	2.22	2.81
7	0.38	0.49	0.64	0.82	1.05	1.35	1.73	2.22	2.84	3.61
8	0.49	0.63	0.82	1.05	1.35	1.73	2.22	2.84	3.63	4.63
9	0.61	0.79	1.02	1.32	1.70	2.18	2.81	3.61	4.63	4.70

For this problem we have implemented the initialization and estimation steps of the algorithm. The obtained optimal state values are in the image above.

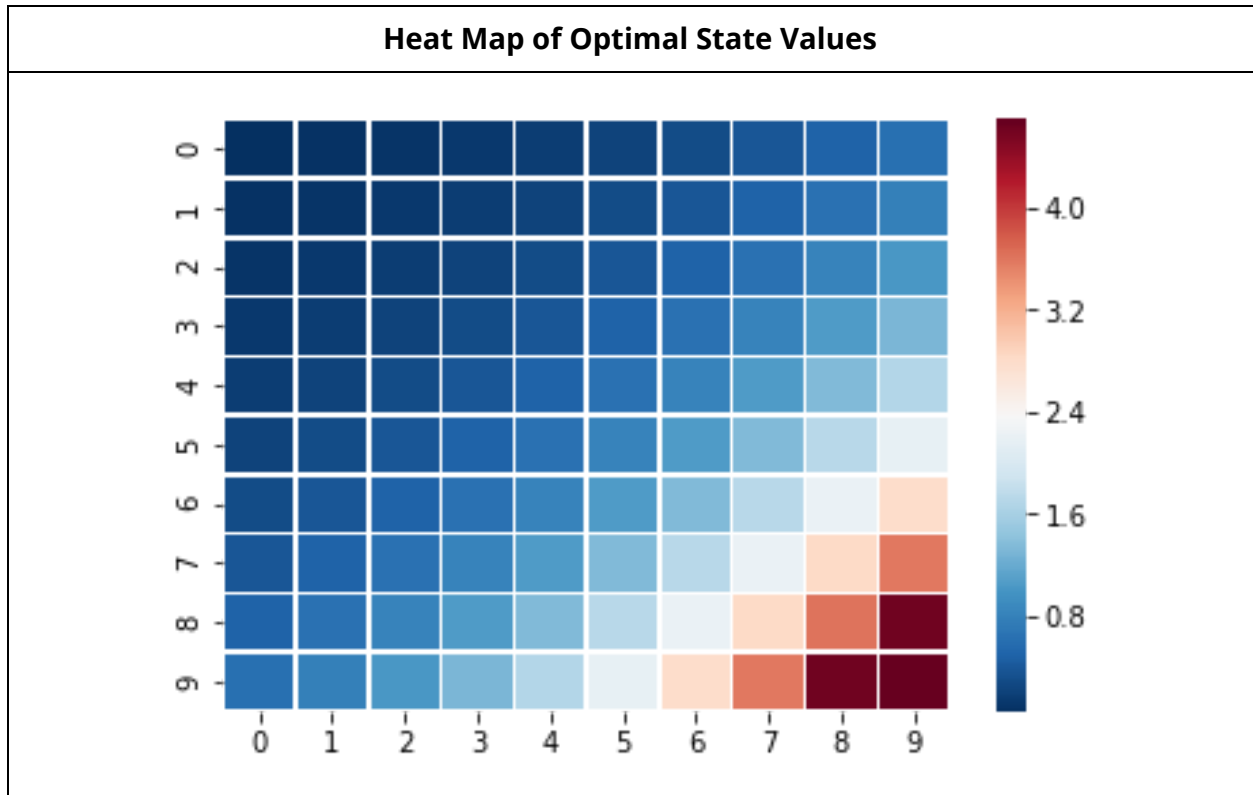
We first generated the transition probability matrix of size (100,100,4) which could store the transition probabilities from any state to any other state by taking a certain action. We had to take care of special cases of edge states which had actions that would make the agent go out of the grid (or remain at the same state) and the corner states as well.

After the transition probability matrix was generated, we used this and the reward function matrix in the algorithm to calculate values for each state ( $V$ ). The algorithm tries to modify  $V$  at each step for each state to its neighboring states while using the transition probability, reward function and values  $V$  from the previous iteration. A batch update for all maximum values is performed at the end of each step. This algorithm is run until the changes in the values of  $V$  drop below a certain threshold.

**3. Generate a heat map of the optimal state values across the 2-D grid. For generating the heat map, you can use the same function provided in the hint earlier (see the hint after question 1).**

The heat map of the optimal state values obtained in the previous question is shown in the table below.

We observe that as we move closer to the last state, the values increase steadily reaching the maximum value at the last state.



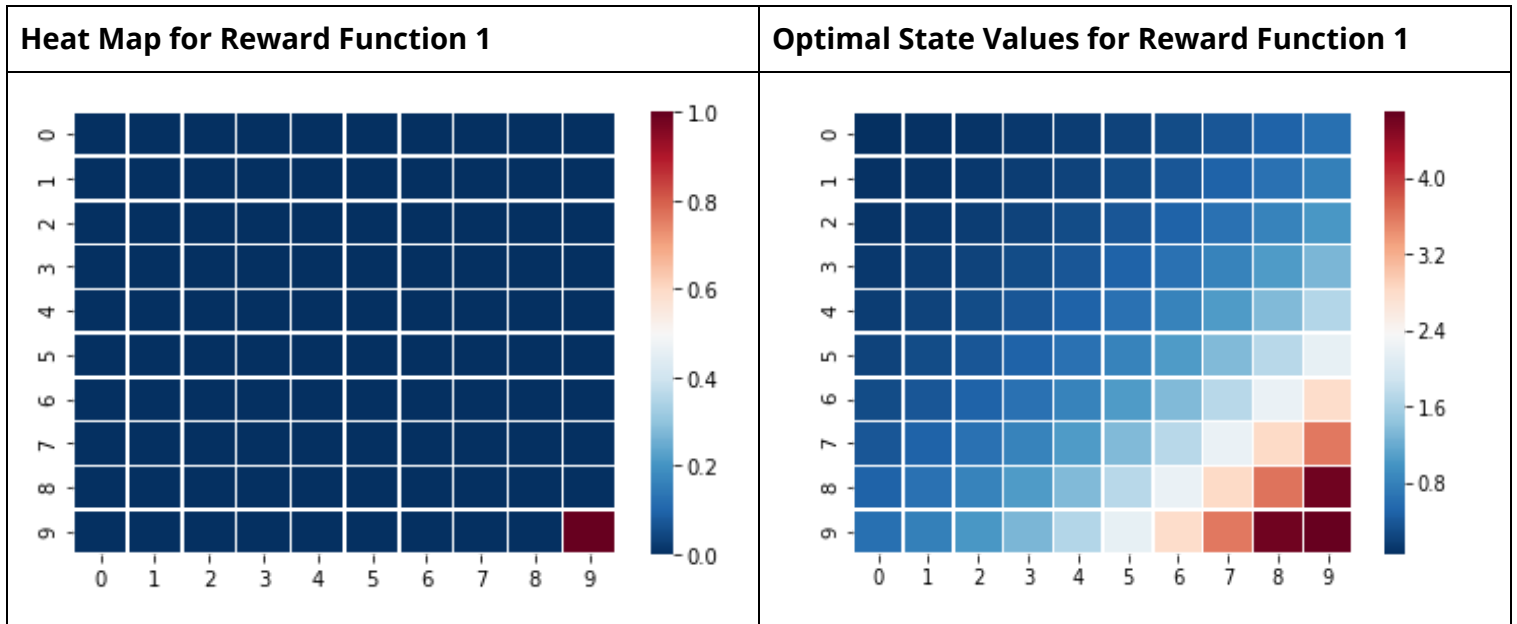
**4. Explain the distribution of the optimal state values across the 2-D grid. (Hint: Use the figure generated in question 3 to explain)**

The state values obtained from Value Iteration basically give the information as to how beneficial it is for the agent to be in that state.

So from the heatmap obtained, we can see that the states closer to the last state have larger values and are more beneficial to be in than the states farther away.

From the heatmap and the corresponding matrix, we observe that the values are symmetric, which is expected as we do not have any obstacles in the environment.

Additionally, as only the final state has a positive reward and the rest have zero rewards, we observe a steady increase in the optimal state values.



- Implement the computation step of the value iteration algorithm (lines 14-17) to compute the optimal policy of the agent navigating the 2-D state-space. For visualization purpose, you should generate a figure similar to that of figure 1 but with the number of state replaced by the optimal action at that state. The optimal actions should be displayed using arrows. Does the optimal policy of the agent match your intuition? Please provide a brief explanation. Is it possible for the agent to compute the optimal action to take at each state by observing the optimal values of its neighboring states? In this question, you should have 1 plot.

In this problem, we implement the computation step of value iteration to compute the actions which should be taken at each state. The actions obtained are reported in the table below (left image). Additionally, we also wanted to see the actions computed by the agent based on observing the optimal values of its neighboring states. The actions obtained from this are also reported in the table below (right image).



Optimal Actions Obtained by Optimal Policy	Optimal Action Based on Observing Optimal Values of Neighboring States																																																																																																																																																																																																																																																		
<table><tr><th></th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th></tr><tr><th>0</th><td>↓</td><td>→</td><td>→</td><td>→</td><td>→</td><td>→</td><td>→</td><td>→</td><td>↓</td><td>↓</td></tr><tr><th>1</th><td>↓</td><td>→</td><td>→</td><td>→</td><td>→</td><td>→</td><td>↓</td><td>↓</td><td>↓</td><td>↓</td></tr><tr><th>2</th><td>↓</td><td>↓</td><td>↓</td><td>→</td><td>→</td><td>↓</td><td>↓</td><td>↓</td><td>↓</td><td>↓</td></tr><tr><th>3</th><td>↓</td><td>↓</td><td>↓</td><td>→</td><td>↓</td><td>↓</td><td>↓</td><td>↓</td><td>↓</td><td>↓</td></tr><tr><th>4</th><td>↓</td><td>↓</td><td>↓</td><td>→</td><td>→</td><td>↓</td><td>↓</td><td>↓</td><td>↓</td><td>↓</td></tr><tr><th>5</th><td>↓</td><td>↓</td><td>→</td><td>→</td><td>→</td><td>→</td><td>↓</td><td>↓</td><td>↓</td><td>↓</td></tr><tr><th>6</th><td>↓</td><td>→</td><td>→</td><td>→</td><td>→</td><td>→</td><td>↓</td><td>↓</td><td>↓</td><td>↓</td></tr><tr><th>7</th><td>↓</td><td>→</td><td>→</td><td>→</td><td>→</td><td>→</td><td>→</td><td>→</td><td>↓</td><td>↓</td></tr><tr><th>8</th><td>→</td><td>→</td><td>→</td><td>→</td><td>→</td><td>→</td><td>→</td><td>→</td><td>→</td><td>↓</td></tr><tr><th>9</th><td>→</td><td>→</td><td>→</td><td>→</td><td>→</td><td>→</td><td>→</td><td>→</td><td>→</td><td>→</td></tr></table>		0	1	2	3	4	5	6	7	8	9	0	↓	→	→	→	→	→	→	→	↓	↓	1	↓	→	→	→	→	→	↓	↓	↓	↓	2	↓	↓	↓	→	→	↓	↓	↓	↓	↓	3	↓	↓	↓	→	↓	↓	↓	↓	↓	↓	4	↓	↓	↓	→	→	↓	↓	↓	↓	↓	5	↓	↓	→	→	→	→	↓	↓	↓	↓	6	↓	→	→	→	→	→	↓	↓	↓	↓	7	↓	→	→	→	→	→	→	→	↓	↓	8	→	→	→	→	→	→	→	→	→	↓	9	→	→	→	→	→	→	→	→	→	→	<table><tr><th></th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th></tr><tr><th>0</th><td>↓</td><td>→</td><td>→</td><td>→</td><td>→</td><td>→</td><td>→</td><td>→</td><td>↓</td><td>↓</td></tr><tr><th>1</th><td>↓</td><td>→</td><td>→</td><td>→</td><td>→</td><td>→</td><td>↓</td><td>↓</td><td>↓</td><td>↓</td></tr><tr><th>2</th><td>↓</td><td>↓</td><td>↓</td><td>→</td><td>→</td><td>↓</td><td>↓</td><td>↓</td><td>↓</td><td>↓</td></tr><tr><th>3</th><td>↓</td><td>↓</td><td>↓</td><td>→</td><td>↓</td><td>↓</td><td>↓</td><td>↓</td><td>↓</td><td>↓</td></tr><tr><th>4</th><td>↓</td><td>↓</td><td>↓</td><td>→</td><td>→</td><td>↓</td><td>↓</td><td>↓</td><td>↓</td><td>↓</td></tr><tr><th>5</th><td>↓</td><td>↓</td><td>→</td><td>→</td><td>→</td><td>→</td><td>↓</td><td>↓</td><td>↓</td><td>↓</td></tr><tr><th>6</th><td>↓</td><td>→</td><td>→</td><td>→</td><td>→</td><td>→</td><td>↓</td><td>↓</td><td>↓</td><td>↓</td></tr><tr><th>7</th><td>↓</td><td>→</td><td>→</td><td>→</td><td>→</td><td>→</td><td>→</td><td>→</td><td>↓</td><td>↓</td></tr><tr><th>8</th><td>→</td><td>→</td><td>→</td><td>→</td><td>→</td><td>→</td><td>→</td><td>→</td><td>→</td><td>↓</td></tr><tr><th>9</th><td>→</td><td>→</td><td>→</td><td>→</td><td>→</td><td>→</td><td>→</td><td>→</td><td>→</td><td>←</td></tr></table>		0	1	2	3	4	5	6	7	8	9	0	↓	→	→	→	→	→	→	→	↓	↓	1	↓	→	→	→	→	→	↓	↓	↓	↓	2	↓	↓	↓	→	→	↓	↓	↓	↓	↓	3	↓	↓	↓	→	↓	↓	↓	↓	↓	↓	4	↓	↓	↓	→	→	↓	↓	↓	↓	↓	5	↓	↓	→	→	→	→	↓	↓	↓	↓	6	↓	→	→	→	→	→	↓	↓	↓	↓	7	↓	→	→	→	→	→	→	→	↓	↓	8	→	→	→	→	→	→	→	→	→	↓	9	→	→	→	→	→	→	→	→	→	←
	0	1	2	3	4	5	6	7	8	9																																																																																																																																																																																																																																									
0	↓	→	→	→	→	→	→	→	↓	↓																																																																																																																																																																																																																																									
1	↓	→	→	→	→	→	↓	↓	↓	↓																																																																																																																																																																																																																																									
2	↓	↓	↓	→	→	↓	↓	↓	↓	↓																																																																																																																																																																																																																																									
3	↓	↓	↓	→	↓	↓	↓	↓	↓	↓																																																																																																																																																																																																																																									
4	↓	↓	↓	→	→	↓	↓	↓	↓	↓																																																																																																																																																																																																																																									
5	↓	↓	→	→	→	→	↓	↓	↓	↓																																																																																																																																																																																																																																									
6	↓	→	→	→	→	→	↓	↓	↓	↓																																																																																																																																																																																																																																									
7	↓	→	→	→	→	→	→	→	↓	↓																																																																																																																																																																																																																																									
8	→	→	→	→	→	→	→	→	→	↓																																																																																																																																																																																																																																									
9	→	→	→	→	→	→	→	→	→	→																																																																																																																																																																																																																																									
	0	1	2	3	4	5	6	7	8	9																																																																																																																																																																																																																																									
0	↓	→	→	→	→	→	→	→	↓	↓																																																																																																																																																																																																																																									
1	↓	→	→	→	→	→	↓	↓	↓	↓																																																																																																																																																																																																																																									
2	↓	↓	↓	→	→	↓	↓	↓	↓	↓																																																																																																																																																																																																																																									
3	↓	↓	↓	→	↓	↓	↓	↓	↓	↓																																																																																																																																																																																																																																									
4	↓	↓	↓	→	→	↓	↓	↓	↓	↓																																																																																																																																																																																																																																									
5	↓	↓	→	→	→	→	↓	↓	↓	↓																																																																																																																																																																																																																																									
6	↓	→	→	→	→	→	↓	↓	↓	↓																																																																																																																																																																																																																																									
7	↓	→	→	→	→	→	→	→	↓	↓																																																																																																																																																																																																																																									
8	→	→	→	→	→	→	→	→	→	↓																																																																																																																																																																																																																																									
9	→	→	→	→	→	→	→	→	→	←																																																																																																																																																																																																																																									

Intuitively, on looking at the state values obtained in question 2 and subsequently observing the heat map generated for the same, we believe that the agent should take actions which lead it to states which are beneficial with respect to leading to the final state. Hence, the actions should move in the direction of larger state values ultimately tracing a path to the last state.

The results obtained are in accordance with our intuition, i.e., we clearly observe that we are moving towards more beneficial states from each state and finally reaching the final state.

We also observe that the optimal actions obtained by the optimal policy are same as the optimal actions obtained based on observing the optimal values of the neighboring states except for the last state. We get this difference because at each state, we check the optimal state value matrix for all valid neighbors. Since the right and down arrows are not valid for the last state and we do not have a stay action which helps us remain at the same state, only the left and up actions are available and as our action assignment assigns 0 value to left and 1 to up and we chose the first action in case of multiple actions with the same priority, left is chosen as the action for the last state.

- 6. Modify the environment of the agent by replacing Reward function 1 with Reward function 2. Use the optimal state-value function implemented in question 2 to compute the optimal value of each state in the grid. For visualization purpose, you should generate a figure**

---

**similar to that of figure 1 but with the number of state replaced by the optimal value of that state. In this question, you should have 1 plot.**

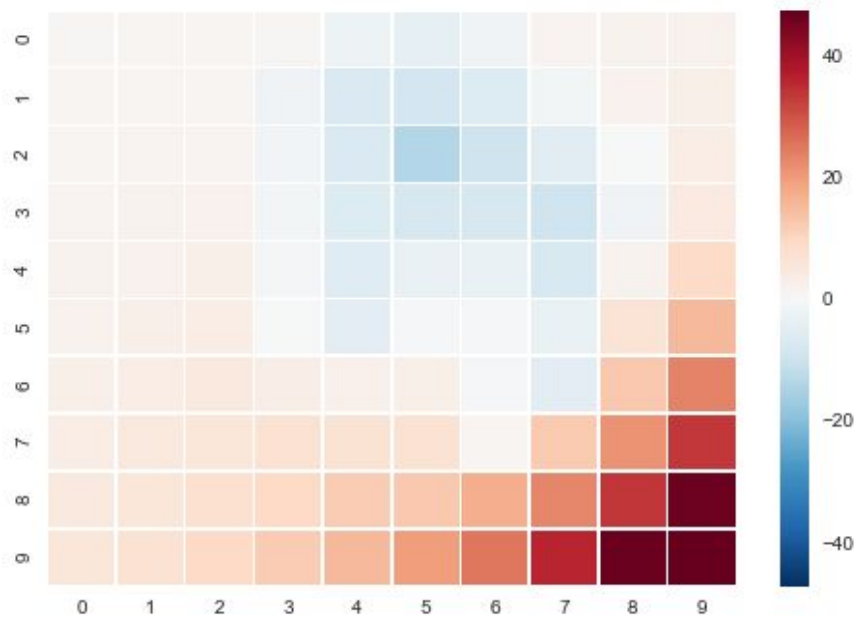
In reward function 2, we see that some states have a negative reward of -100 and a majority of states having zero reward. We again observe that the final state has the maximum reward. The presence of a negative reward can be thought of as having an obstacle in the environment which would enable the agent to get a better understanding of the environment with well defined paths. This can be thought of a scenario of a driver driving in the field with obstacles or in the city.

In this problem, we make use of reward function 2 instead of 1 and have obtained the following state values:

	0	1	2	3	4	5	6	7	8	9
0	0.65	0.79	0.82	0.53	-2.39	-4.24	-1.92	1.13	1.59	2.03
1	0.83	1.02	1.06	-1.88	-6.75	-8.68	-6.37	-1.30	1.92	2.61
2	1.06	1.31	1.45	-1.64	-6.76	-13.92	-9.65	-5.51	-0.13	3.36
3	1.36	1.69	1.94	-1.24	-6.34	-7.98	-7.95	-9.43	-1.92	4.39
4	1.73	2.17	2.59	-0.74	-5.85	-3.26	-3.24	-7.43	1.72	9.16
5	2.21	2.78	3.41	-0.04	-5.11	-0.55	-0.49	-2.98	6.58	15.35
6	2.82	3.55	4.48	3.02	2.48	2.88	-0.47	-4.91	12.69	23.30
7	3.58	4.54	5.79	7.29	6.72	7.24	0.93	12.37	21.16	33.48
8	4.56	5.79	7.40	9.44	12.01	12.89	17.10	23.01	33.78	46.53
9	5.73	7.32	9.39	12.04	15.45	19.82	25.50	36.16	46.58	47.31

- 7. Generate a heat map of the optimal state values (found in question 6) across the 2-D grid. For generating the heat map, you can use the same function provided in the hint earlier.**

The heat map for the state values obtained in the previous question are as shown:



**8. Explain the distribution of the optimal state values across the 2-D grid. (Hint: Use the figure generated in question 7 to explain)**

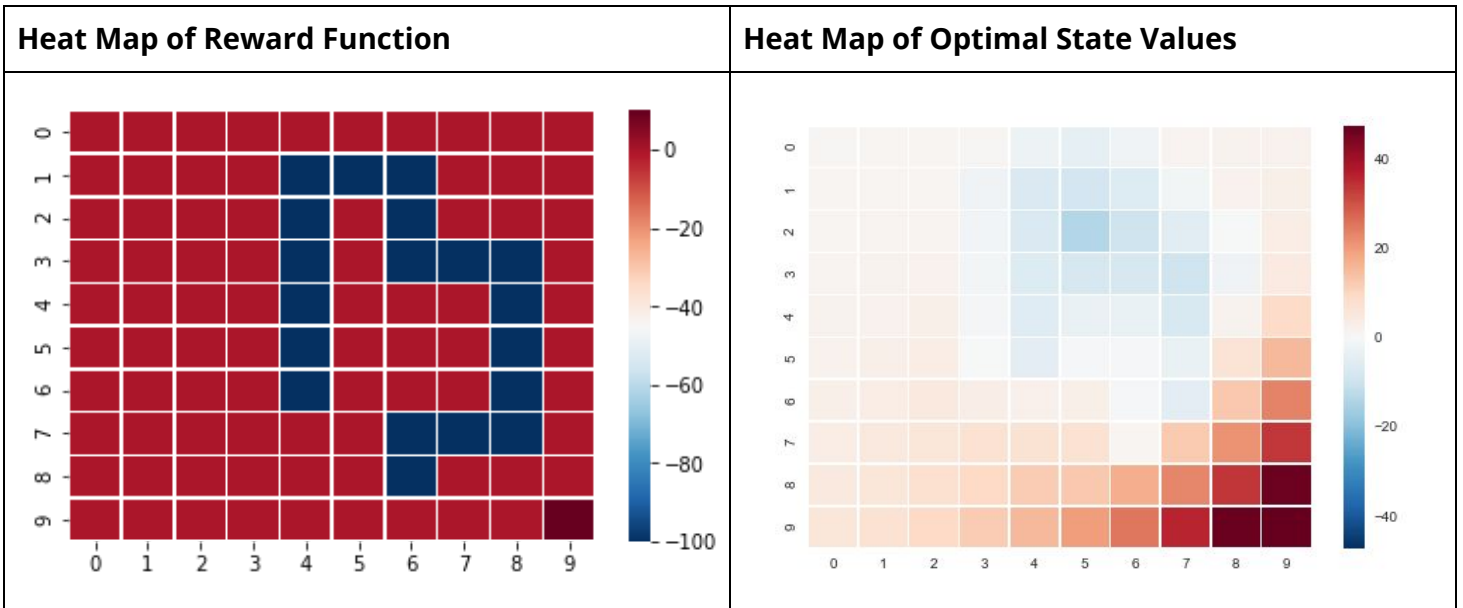
Similar to the heat map for reward function 1, we observe that as we move closer towards the last state, the values increase steadily. We also observe a set of states where the values are very low in the middle of the environment.

This region of low values is due to the very low values of rewards set in that region. From the reward function 2 we see that there is a set of states with negative reward of -100 (refer reward heatmap below). This implies that it is not beneficial to go to those states. Additionally, as the negative reward states nearly form a closed boundary enclosing a region of zero reward states, we can infer that if one is in any of the states in the said region, it is highly penalizing as one would have to cross one or more negative reward states to move closer towards the final state. Hence, the state values for these states are low.

We observe that the approximate shape of the low values of optimal state values is similar to the shape of the area enclosed by the negative reward region in the reward heatmap.

An analogy to understand this better can be a driver in an open field with obstacles. The driver will try to avoid the obstacles and stay as far away as possible while trying to reach its destination.

Due to the nature of the reward function, the distribution is no longer symmetric.



- Implement the computation step of the value iteration algorithm (lines 14-17) to compute the optimal policy of the agent navigating the 2-D state-space. For visualization purpose, you should generate a figure similar to that of figure 1 but with the number of state replaced by the optimal action at that state. The optimal actions should be displayed using arrows. Does the optimal policy of the agent match your intuition? Please provide a brief explanation. In this question, you should have 1 plot.

On implementing the computation step, the following actions are obtained (see image below). Intuitively, we expect the agent to take actions avoiding the obstacles to reach the final state. Hence, we expect to see well defined paths. Our intuition is confirmed when we observe the optimal actions obtained using the optimal policy obtained using value iteration.

We observe that the actions for the negatively rewarded states and its surroundings are such that they avoid that region and move towards better states with better optimal values with the goal of moving towards the last state which has the maximum reward assigned to it. Furthermore, we see a set of well defined paths from the start to the end states which we were not observing with reward function 1.

Given only the actions we can infer that there is a region of negative reward. On referencing the reward heatmap, we see that there is a negatively rewarded region in column 4. The agent hence takes actions to avoid this column by moving away from this and its adjoining states.

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

**10. Express  $\mathbf{c}$ ,  $\mathbf{x}$ ,  $\mathbf{D}$  in terms of  $\mathbf{R}$ ,  $P_a$ ,  $P_{a_1}$ ,  $t_i$ ,  $\mathbf{u}$ ,  $\lambda$  and  $R_{max}$**

**Ans:** The Linear Programming Formulation we have initially is as below.

$$\begin{aligned}
 & \underset{\mathbf{R}, t_i, u_i}{\text{maximize}} && \sum_{i=1}^{|\mathcal{S}|} (t_i - \lambda u_i) \\
 & \text{subject to} && [(\mathbf{P}_{a_1}(i) - \mathbf{P}_a(i))(\mathbf{I} - \gamma \mathbf{P}_{a_1})^{-1} \mathbf{R}] \geq t_i, \quad \forall a \in \mathcal{A} \setminus a_1, \forall i \\
 & && (\mathbf{P}_{a_1} - \mathbf{P}_a)(\mathbf{I} - \gamma \mathbf{P}_{a_1})^{-1} \mathbf{R} \succeq 0, \quad \forall a \in \mathcal{A} \setminus a_1 \\
 & && -\mathbf{u} \preceq \mathbf{R} \preceq \mathbf{u} \\
 & && |\mathbf{R}_i| \leq R_{max}, \quad i = 1, 2, \dots, |\mathcal{S}|
 \end{aligned}$$

In the above equations,

- 't' and 'u' are slack or auxiliary variables
- ' $\gamma$ ' is the discount factor
- 'R' is the reward function for which we want to find an optimal value
- ' $R_{\max}$ ' is the maximum of the absolute reward
- ' $\lambda$ ' is the regularization parameter
- ' $P_{a1}$ ' represents the transition probabilities using the optimal action and ' $P_a$ ' represents the transition probabilities using the actions other than the optimal action

The above LP formulation is then represented as below,

$$\begin{aligned} & \underset{\mathbf{x}}{\text{maximize}} && \mathbf{c}^T \mathbf{x} \\ & \text{subject to} && \mathbf{D}\mathbf{x} \preceq 0, \quad \forall a \in \mathcal{A} \setminus a_1 \end{aligned}$$

We will consider 'X' as a 301 x 1 matrix with the following:

- 'Ri' for each of the 100 states in the first 100 rows
- ' $t_i$ ' for each of the 100 states in the next 100 rows
- ' $u_i$ ' for each of 100 states in the next 100 rows and
- Rmax in the last row

Now, for the expression to **Maximize**  $\sum_{i=1}^S t_i - \lambda * u_i$ , with the above 'X',

'C' is a 301 x 1 matrix with the following:

- zeros in first 100 rows (corresponding to coefficients for 'Ri's)
- '1' in the next 100 rows (corresponding to coefficients for 'ti's)
- ' $-\lambda$ ' in the next 100 rows (corresponding to coefficients for 'ui's) and
- 0 in the last row (corresponding to coefficient for Rmax)

D is a big matrix consisting of different parts that make up the constraints defined in the LP formulation. We have a  $\leq$  constraint, have made RHS of all constraints as 0 and so the constant terms are also incorporated in the LHS term which is product of D and x.

D is overall a 1000 x 301 matrix with each row corresponding to 1 constraint and the 301 columns corresponding to the coefficients of 'X' with

- First 100 columns: Coefficient corresponding to R for each state
- Next 100 columns: Coefficient corresponding to  $t_i$  for each state
- Next 100 columns: Coefficient corresponding to  $u_i$  for each state
- Last column: Coefficient corresponding to Rmax

To explain better, we have divided the matrix D into blocks - I, II, III, etc corresponding to each constraint (along with their dimensions).

I	<p>Dimension: 300 x 301</p> <p>Constraint:</p> $[(\mathbf{P}_{a_1}(i) - \mathbf{P}_a(i))(\mathbf{I} - \gamma\mathbf{P}_{a_1})^{-1}\mathbf{R}] \geq t_i, \quad \forall a \in \mathcal{A} \setminus a_1, \forall i$
II	<p>Dimension: 300 x 301</p> <p>Constraint:</p> $(\mathbf{P}_{a_1} - \mathbf{P}_a)(\mathbf{I} - \gamma\mathbf{P}_{a_1})^{-1}\mathbf{R} \succeq 0, \quad \forall a \in \mathcal{A} \setminus a_1$
III	<p>Dimension: 100 x 301</p> <p>Constraint:</p> $-\mathbf{u} \preceq \mathbf{R}$
IV	<p>Dimension: 100 x 301</p> <p>Constraint:</p> $\mathbf{R} \preceq \mathbf{u}$
V	<p>Dimension: 100 x 301</p> <p>Constraint:</p> $ \mathbf{R}_i  \leq R_{max}, \quad i = 1, 2, \dots,  \mathcal{S} $ $(-R_{max} \leq R_i)$
VI	<p>Dimension: 100 x 301</p> <p>Constraint:</p> $ \mathbf{R}_i  \leq R_{max}, \quad i = 1, 2, \dots,  \mathcal{S} $ $(R_i \leq +R_{max})$

## Block I

---

The 300 rows in Block I are for all states and for each state their sub-optimal actions, hence we will have  $100 \times 3 = 300$  constraints corresponding to the constraint equation:

$$[(\mathbf{P}_{a_1}(i) - \mathbf{P}_a(i))(\mathbf{I} - \gamma\mathbf{P}_{a_1})^{-1}\mathbf{R}] \geq t_i, \quad \forall a \in \mathcal{A} \setminus a_1, \forall i$$

Each row in the first 300 rows of D contains the following:

- Coefficient corresponding to R for each state, for each sub-optimal action which will be equal to

$$-[(\mathbf{P}_{a_1}(i) - \mathbf{P}_a(i))(\mathbf{I} - \gamma\mathbf{P}_{a_1})^{-1}] \quad \text{for the particular state } i \text{ and}$$

sub-optimal action a. Rest 99 columns will be 0.

- Coefficient corresponding to  $t_i$  for each state, for each sub-optimal action will be equal to +1 for the particular state  $i$  and sub-optimal action  $a$ . Rest 99 columns will be 0
- Coefficient corresponding to  $u_i$  for each state, for each sub-optimal action will be equal to 0 as  $u_i$  is not present in this constraint
- Coefficient corresponding to  $R_{\max} = 0$  as  $R_{\max}$  is not present in this constraint.

## Block II

The next important component specifies the following constraints,

$$(\mathbf{P}_{a_1} - \mathbf{P}_a)(\mathbf{I} - \gamma\mathbf{P}_{a_1})^{-1}\mathbf{R} \succeq 0, \quad \forall a \in \mathcal{A} \setminus a_1$$

The 300 rows in Block II are for all states and for each state their sub-optimal actions, hence we will have  $100 \times 3 = 300$  constraints corresponding to the constraint equation.

Each row in the first 300 rows of D contains the following:

- Coefficient corresponding to R for each state, for each sub-optimal action which will be equal to

$$-[(\mathbf{P}_{a_1}(i) - \mathbf{P}_a(i))(\mathbf{I} - \gamma\mathbf{P}_{a_1})^{-1}] \quad \text{for the particular state } i \text{ and}$$

sub-optimal action a. Rest 99 columns will be 0.

- Coefficient corresponding to  $t_i$  for each state, for each sub-optimal action will be equal to 0 as  $t_i$  is not present in this constraint



- Coefficient corresponding to  $u_i$  for each state, for each sub-optimal action will be equal to 0 as  $u_i$  is not present in this constraint
- Coefficient corresponding to  $R_{max} = 0$  as  $R_{max}$  is not present in this constraint.

### Block III

Corresponds to the equation:  $-\mathbf{u} \preceq \mathbf{R}$

Each of the 100 rows correspond to each of the 100 states. Each row contains:

- Coefficient corresponding to  $R_i$  for the particular state  $i$  will be -1. Coefficient for rest of the states will be 0
- Coefficient corresponding to  $t_i$  for each state will be equal to 0 as  $t_i$  is not present in this constraint
- Coefficient corresponding to  $u_i$  for particular state  $i$  will be -1. For the rest of the states, coefficients will be 0.
- Coefficient corresponding to  $R_{max} = 0$  as  $R_{max}$  is not present in this constraint.

### Block IV

Corresponds to the equation:  $\mathbf{R} \preceq \mathbf{u}$

Each of the 100 rows correspond to each of the 100 states. Each row contains:

- Coefficient corresponding to  $R_i$  for the particular state  $i$  will be +1. Coefficient for rest of the states will be 0.
- Coefficient corresponding to  $t_i$  for each state will be equal to 0 as  $t_i$  is not present in this constraint
- Coefficient corresponding to  $u_i$  for particular state  $i$  will be -1. For the rest of the states, coefficients will be 0.
- Coefficient corresponding to  $R_{max} = 0$  as  $R_{max}$  is not present in this constraint.

### Block V

Corresponds to the equation:  $|\mathbf{R}_i| \leq R_{max}, i = 1, 2, \dots, |S|$  or  $-R_{max} \leq R_i$

Each of the 100 rows correspond to each of the 100 states. Each row contains:

- Coefficient corresponding to  $R_i$  for the particular state  $i$  will be -1. Coefficient for rest of the states will be 0.
- Coefficient corresponding to  $t_i$  for each state will be equal to 0 as  $t_i$  is not present in this constraint

- Coefficient corresponding to  $u_i$  for each state  $i$  will be 0 as  $u_i$  is not present in the constraint
- Coefficient corresponding to  $R_{max} = -1$

### Block VI

Corresponds to the equation:  $|\mathbf{R}_i| \leq R_{max}, i = 1, 2, \dots, |S|$  or  $R_i \leq +R_{max}$

Each of the 100 rows correspond to each of the 100 states. Each row contains:

- Coefficient corresponding to  $R_i$  for the particular state  $i$  will be +1. Coefficient for rest of the states will be 0.
- Coefficient corresponding to  $t_i$  for each state will be equal to 0 as  $t_i$  is not present in this constraint
- Coefficient corresponding to  $u_i$  for each state will be 0 as  $u_i$  is not present in the constraint
- Coefficient corresponding to  $R_{max} = -1$

**11. Sweep  $\lambda$  from 0 to 5 to get 500 evenly spaced values for  $\lambda$ . For each value of  $\lambda$  compute  $O_A(s)$  by following the process described above. For this problem, use the optimal policy of the agent found in question 5 to fill in the  $O_E(s)$  values. Then use equation 3 to compute the accuracy of the IRL algorithm for this value of  $\lambda$ . You need to repeat the above process for all 500 values of  $\lambda$  to get 500 data points. Plot (x-axis) against Accuracy (y-axis). In this question, you should have 1 plot.**

**Ans:**

In this and the following questions we implement Inverse Reinforcement Learning (IRL) which is basically the field of learning an agent's objectives, values, or rewards by observing its behavior.

In IRL, we are given an agent's policy or a history of behavior and we try to find a reward function that explains the given behavior. Under the assumption that our agent acted optimally, we try to estimate a reward function that could have led to this behavior.

There are three major IRL algorithms, one for each of the following scenarios:

- Optimal policy is known and we have a small state space
- Optimal policy is known and we have a large or infinite state space
- Optimal policy is unknown

---

Some modern approaches in IRL are:

- Apprenticeship learning via IRL
- Bayesian IRL
- Maximum Entropy IRL
- Maximum Causal Entropy IRL
- Maximum Entropy Deep IRL

For our implementation of IRL,

$\mathbf{O}_A(\mathbf{s})$  represents optimal action of the agent at state  $\mathbf{s}$

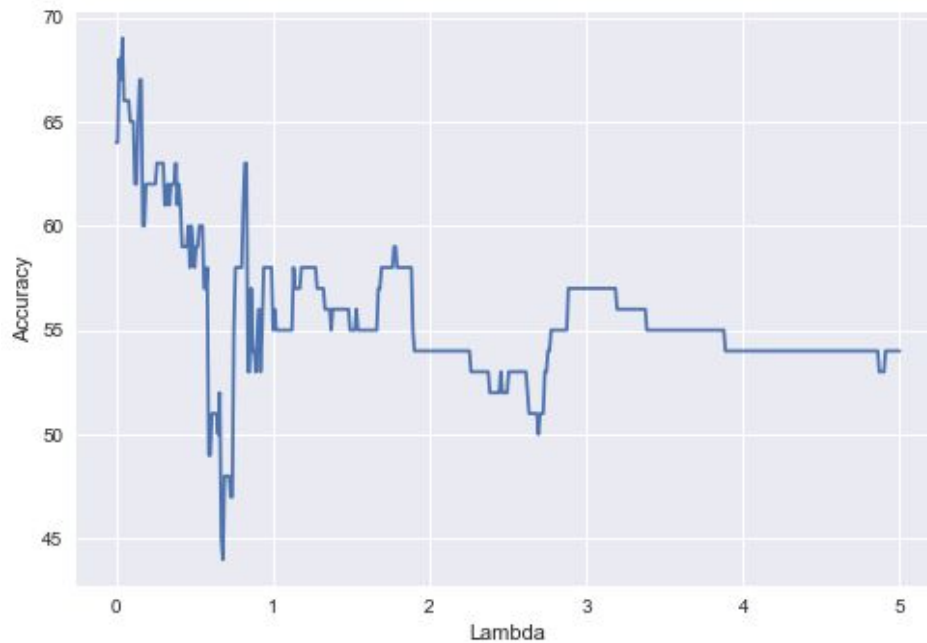
$\mathbf{O}_E(\mathbf{s})$  represents optimal action of the expert at state  $\mathbf{s}$

The accuracy of the agent is given by,  $\sum_i \frac{m(S_i)}{|S|}$

where,

$$m(s) = \begin{cases} 1, & O_A(s) = O_E(s) \\ 0, & \text{else} \end{cases}$$

We varied the regularization parameter from 0 to 5 to get 500 values and calculated the accuracy for each one of them. Below is the graph for the same.



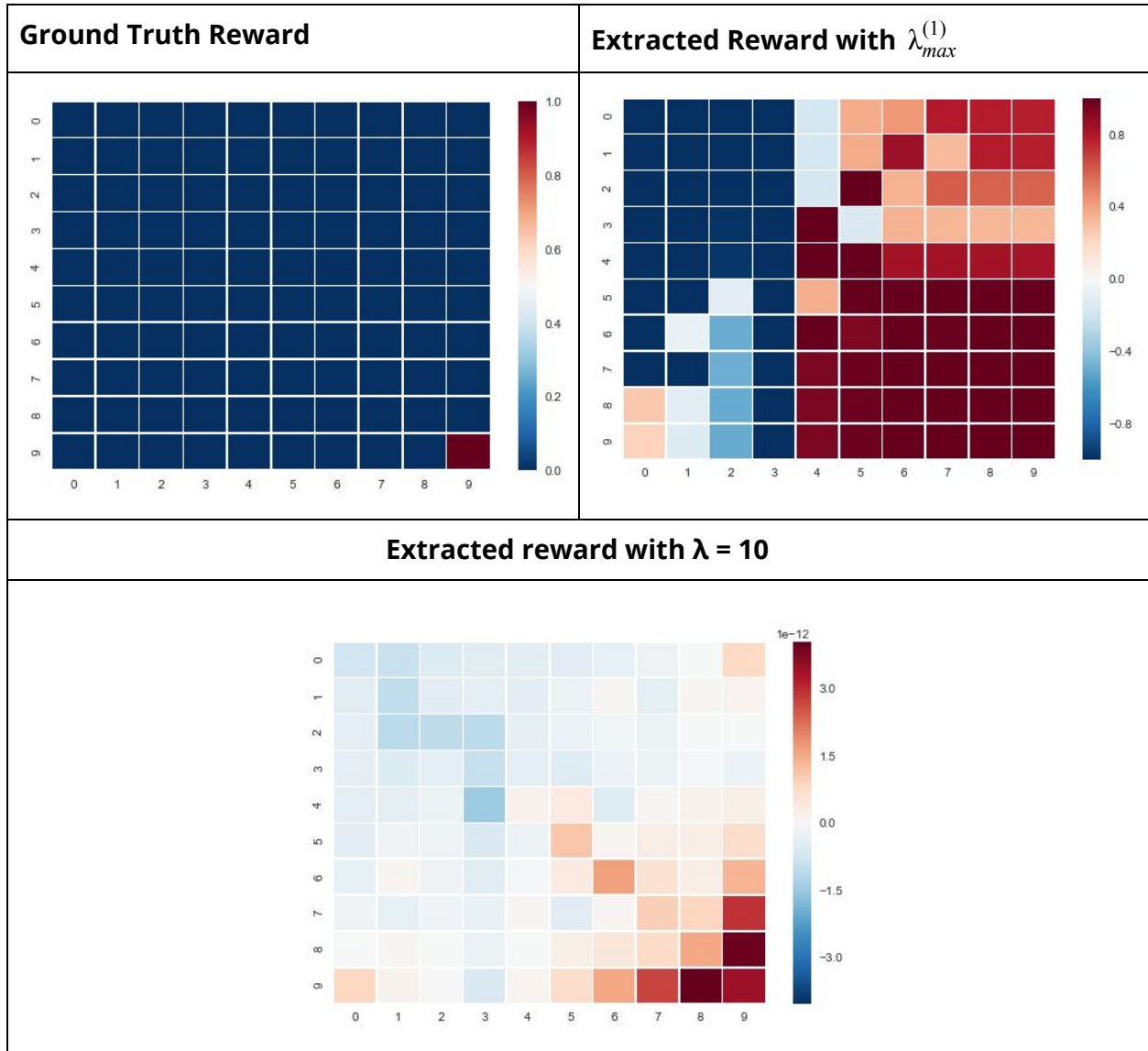
From the accuracies we observe an interesting trend. We observe the highest accuracy value for a very small value of lambda (~0.05). After this value of lambda, we see a lot of variation in the accuracy values. Some spikes are obtained before the accuracy takes a sharp dip for lambda around 0.8. After that, it makes a recovery and we again observe some spikes and variation in the values before the accuracy begins stabilizing for lambdas of 3.5 and above. This stable accuracy is significantly (~54%) than the maximum accuracy obtained (~69%).

We also believe that increasing the regularization parameter above a certain threshold leads to a negative effect and reduces the overall accuracy as giving a larger weight to the negative term in the function  $\sum_{i=1}^S t_i - \lambda * u_i$  may lead it to move away from the global maximum.

**12. Use the plot in question 11 to compute the value of for which accuracy is maximum. For future reference we will denote this value as  $\lambda_{max}^{(1)}$ . Please report  $\lambda_{max}^{(1)}$**

Max Accuracy	69
$\lambda_{max}^{(1)}$	0.04008016032064128

13. For  $\lambda_{max}^{(1)}$ , generate heat maps of the ground truth reward and the extracted reward. Please note that the ground truth reward is the Reward function 1 and the extracted reward is computed by solving the linear program given by equation 2 with the parameter set to  $\lambda_{max}^{(1)}$ . In this question, you should have 2 plots.



We can clearly see that if the regularization parameter  $\lambda$  is closer to 0, we get large regions of similar extracted reward values (both negative and positive). However, as we increase the regularization parameter  $\lambda = 10$ , we clearly see that the extracted

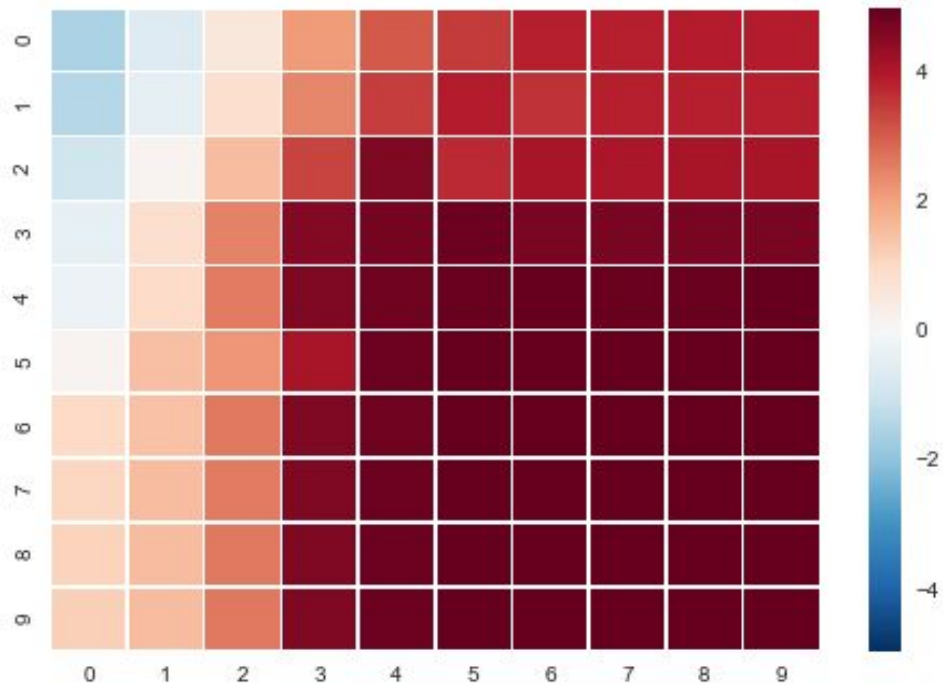
reward function is much better with higher values closer to the state with maximum reward. We also stop getting regions of similar values which would be helpful for the agent to make better policies.

**14. Use the extracted reward function computed in question 13, to compute the optimal values of the states in the 2-D grid. For computing the optimal values you need to use the optimal state-value function that you wrote in question 2. For visualization purpose, generate a heat map of the optimal state values across the 2-D grid (similar to the figure generated in question 3). In this question, you should have 1 plot.**

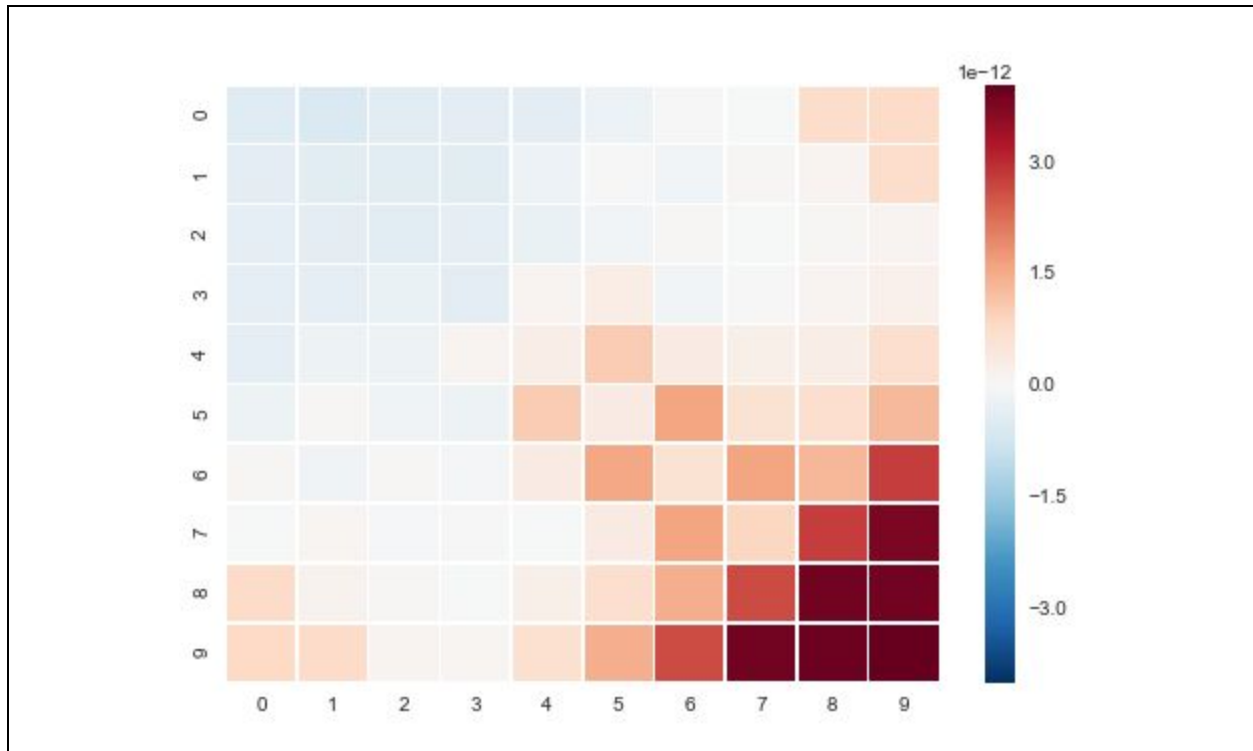
The optimal state values computed from the extracted reward function are as follows:

Optimal Values of States										
	0	1	2	3	4	5	6	7	8	9
0	-1.57	-0.63	0.57	2.11	3.04	3.47	3.85	3.86	3.88	3.88
1	-1.41	-0.44	0.82	2.42	3.43	3.91	3.60	3.87	3.87	3.88
2	-0.97	0.13	1.55	3.36	4.64	3.72	4.09	4.06	4.08	4.07
3	-0.41	0.85	2.47	4.57	4.75	4.87	4.71	4.71	4.71	4.71
4	-0.31	0.96	2.57	4.65	4.84	4.91	4.93	4.92	4.92	4.93
5	0.13	1.51	2.18	4.11	4.87	4.94	4.96	4.95	4.95	4.96
6	0.99	1.47	2.62	4.65	4.82	4.96	4.96	4.96	4.96	4.96
7	1.09	1.51	2.59	4.65	4.88	4.96	4.96	4.96	4.96	4.96
8	1.14	1.52	2.62	4.65	4.88	4.96	4.96	4.96	4.96	4.96
9	1.17	1.54	2.61	4.65	4.88	4.96	4.96	4.96	4.96	4.96

Optimal state value function with  $\lambda_{\max}$



Optimal state value function with  $\lambda = 10$

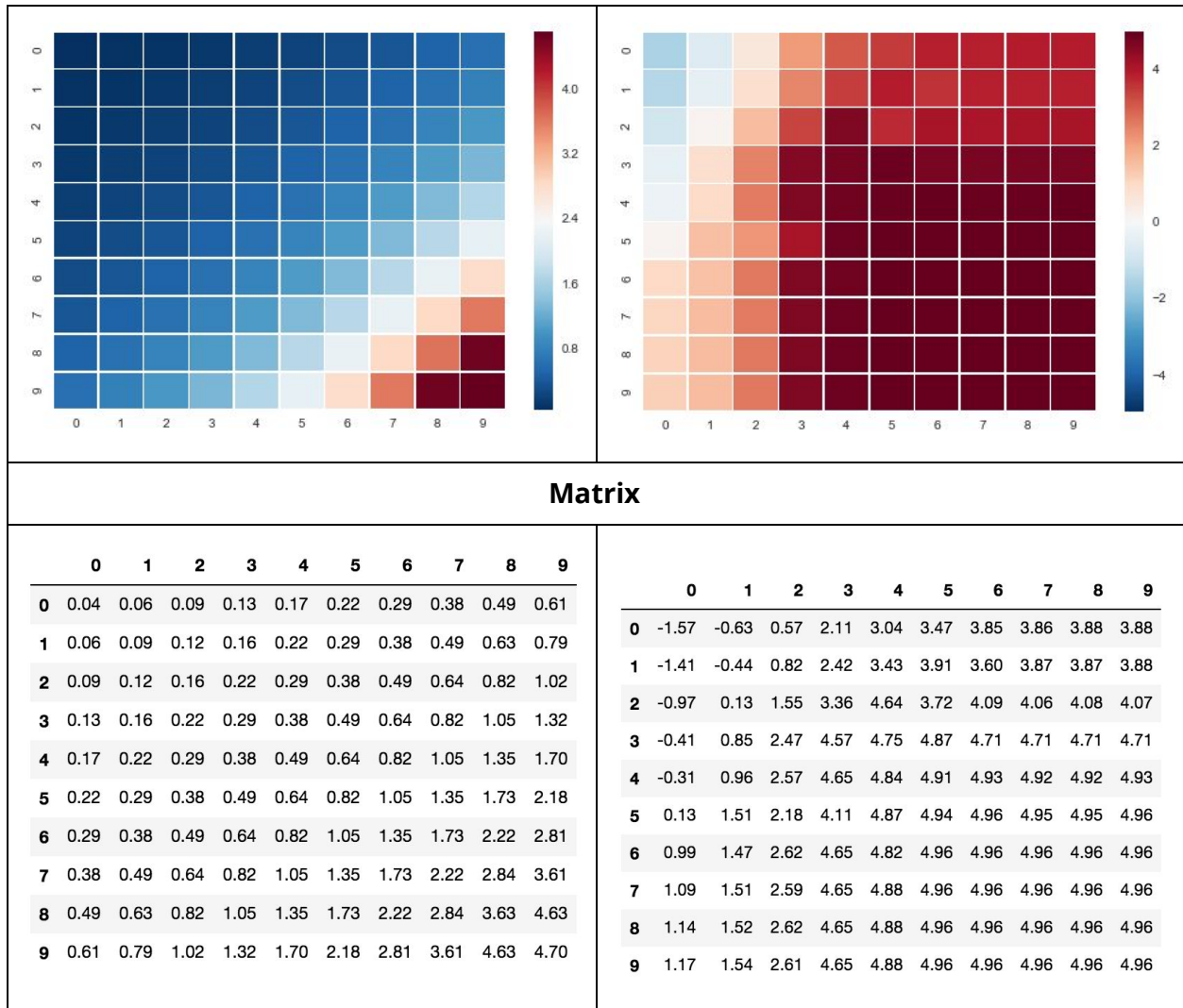


We can clearly see that if the regularization parameter  $\lambda$  is closer to 0, we get large regions of similar values in the optimal state value function. However, as we increase the regularization parameter  $\lambda = 10$ , we clearly see that the distribution is much better with higher values closer to the state with maximum reward. We also stop getting regions of similar values which would be helpful for the agent to make better policies.

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

Question 3	Question 14
Heat Maps	





In question 3, we computed the optimal state values for the given reward function. In question 14, we calculate the optimal state values using the extracted reward function.

A general trend we observe is that the values increase as we move from the start state to the final state. However, the rate of increase is faster in the values obtained using the extracted reward. From the minimum value of -1.57 at the start state, we quickly reach values in the range of 4 in the 3rd row. On the contrary, we start from 0 at the start state and only arrive at values in the range of 4 when we are closer to the destination.

We also observe that in the values obtained using the extracted reward, we have multiple states having the maximum possible value whereas this is not the case for the values obtained using the given reward function - only the last state has the maximum state value in this case.

16. Use the extracted reward function found in question 13 to compute the optimal policy of the agent. For computing the optimal policy of the agent you need to use the function that you wrote in question 5. For visualization purpose, you should generate a figure similar to that of figure 1 but with the number of state replaced by the optimal action at that state. The actions should be displayed using arrows. In this question, you should have 1 plot.

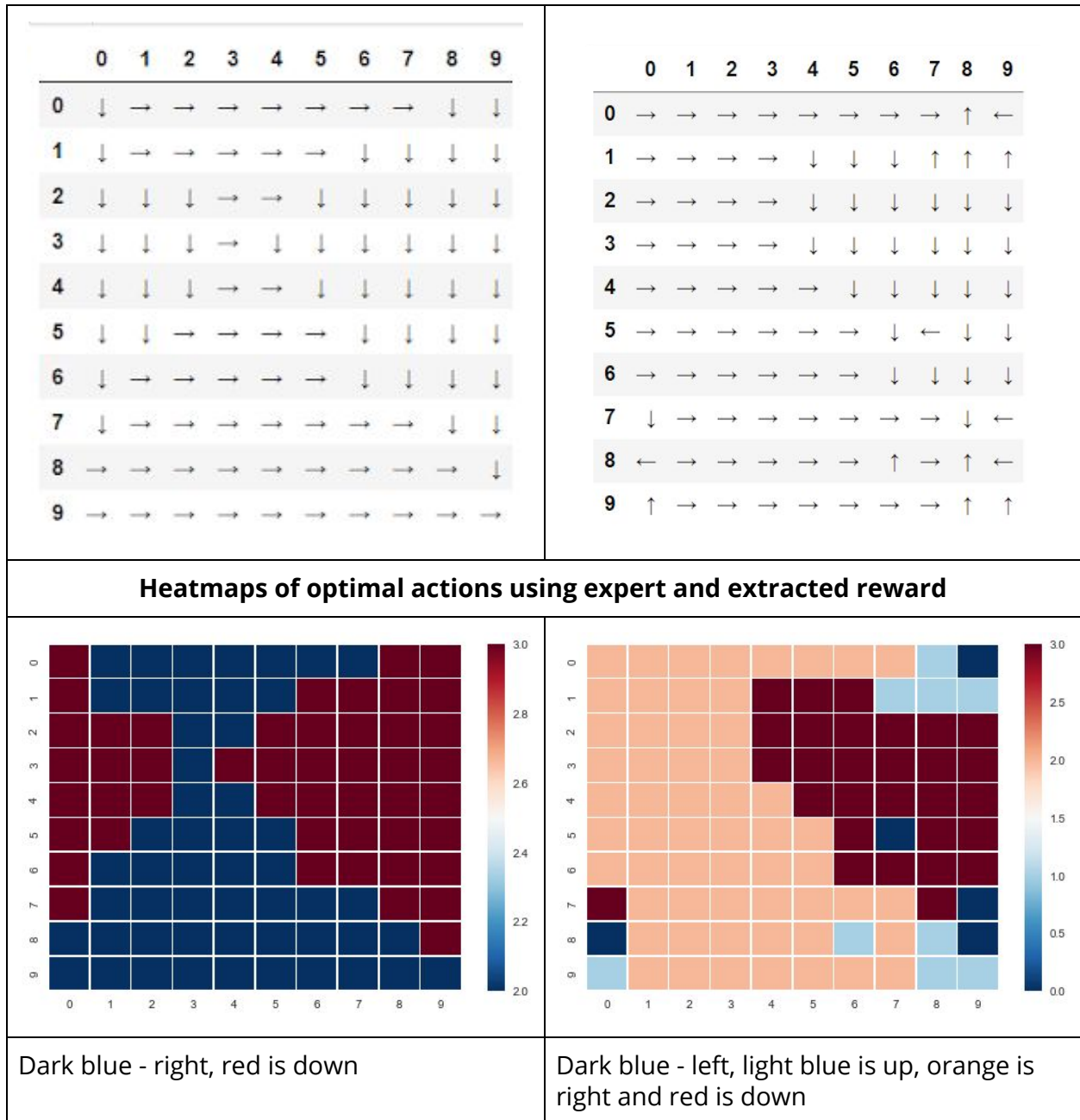
The optimal actions obtained using the extracted reward are given below. We observe that there are some states where the actions suggest that the agent should move out of the environment which is equivalent to staying at the same state.

Additionally, we also observe instances of deadlocks where we have opposing actions in adjacent states which prevent us from ever reaching the goal state.

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

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

Question 5	Question 16
Optimal Actions	



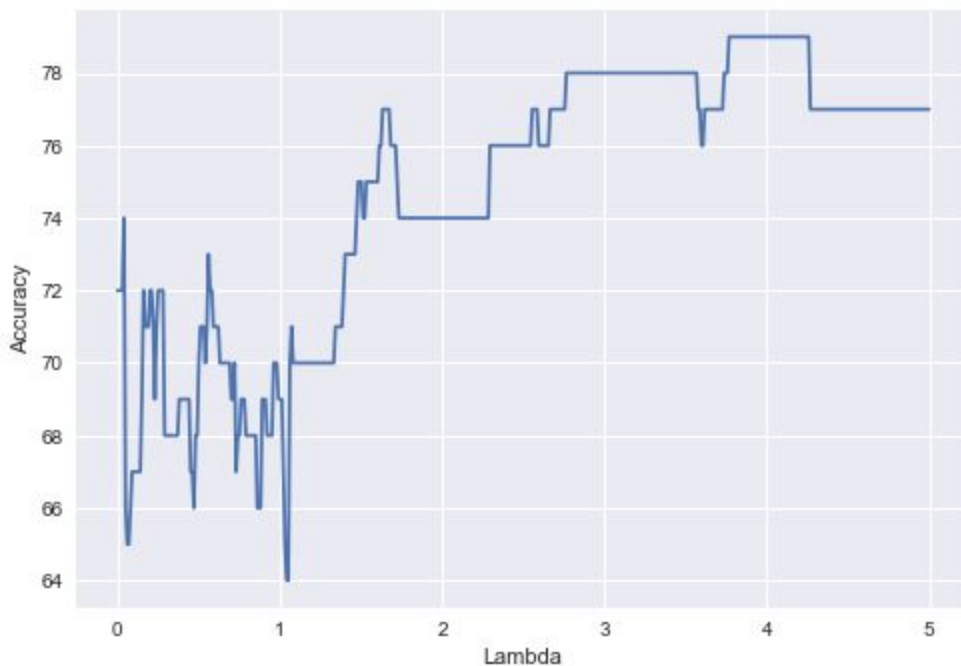
We can see that a sizeable chunk of actions (dark blue in left image and orange in right image, dark red in left and right images) match, which is why we get a good accuracy.

As explained above, we observe that the actions obtained using the extracted reward contain some actions which lead the agent out of the environment and some combinations of actions which lead us to deadlocks. Due to these two factors, we cannot find a definitive path from the start to the end state .

---

On the contrary, there are no deadlock conditions occurring in the optimal actions obtained using the given reward and there are no actions which are leading the agent to leave the environment. As a result, there are multiple paths from the start to the end state.

**18. Sweep  $\lambda$  from 0 to 5 to get 500 evenly spaced values for  $\lambda$ . For each value of  $\lambda$  compute OA(s) by following the process described above. For this problem, use the optimal policy of the agent found in question 9 to 11 in the OE(s) values. Then use equation 3 to compute the accuracy of the IRL algorithm for this value of  $\lambda$ . You need to repeat the above process for all 500 values of  $\lambda$  to get 500 data points. Plot  $\lambda$  (x-axis) against Accuracy (y-axis). In this question, you should have 1 plot.**



The accuracies obtained for various values of lambda have been plotted and displayed above. Initially, we see many sharp spikes in a very short range of lambda (0 - 1.1). In this range the accuracy ranges from 64% (minimum) to 74% (maximum). After this, there is a steady rise in the accuracy values after which we observe intermittent plateaus of rising and falling accuracies. The maximum accuracy is obtained in one such plateau which starts with lambda around 3.7 and goes on till

---

4.3. The accuracy obtained in this region is approximately 78%. After this, we again observe a dip in accuracy which continues till  $\lambda = 5$ .

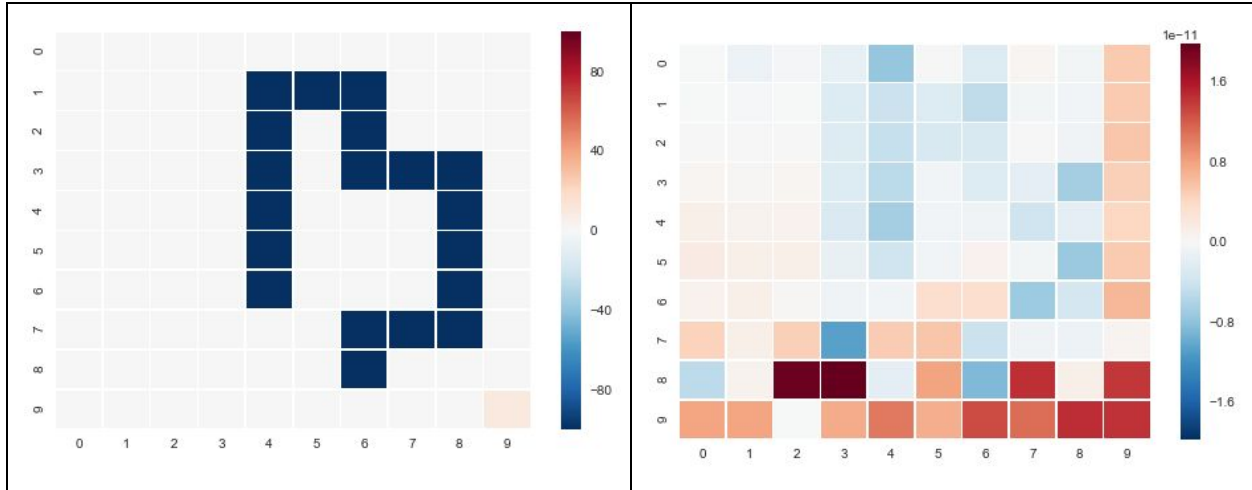
We also believe that increasing the regularization parameter helps increasing the accuracy in this case because the agent may have been overfitting to the paths leading to the goal state while avoiding the obstacle, but now could be generalizing better, hence increasing the accuracy.

**19. Use the plot in question 18 to compute the value of  $\lambda$  for which accuracy is maximum. For future reference we will denote this value as  $\lambda_{max}^{(2)}$ . Please report  $\lambda_{max}^{(2)}$**

<b>Max Accuracy</b>	79
$\lambda_{max}^{(2)}$	3.7675350701402803

**20. For  $\lambda_{max}^{(2)}$ , generate heat maps of the ground truth reward and the extracted reward. Please note that the ground truth reward is the Reward function 2 and the extracted reward is computed by solving the linear program given by equation 2 with the parameter set to  $\lambda_{max}^{(2)}$ . In this question, you should have 2 plots.**

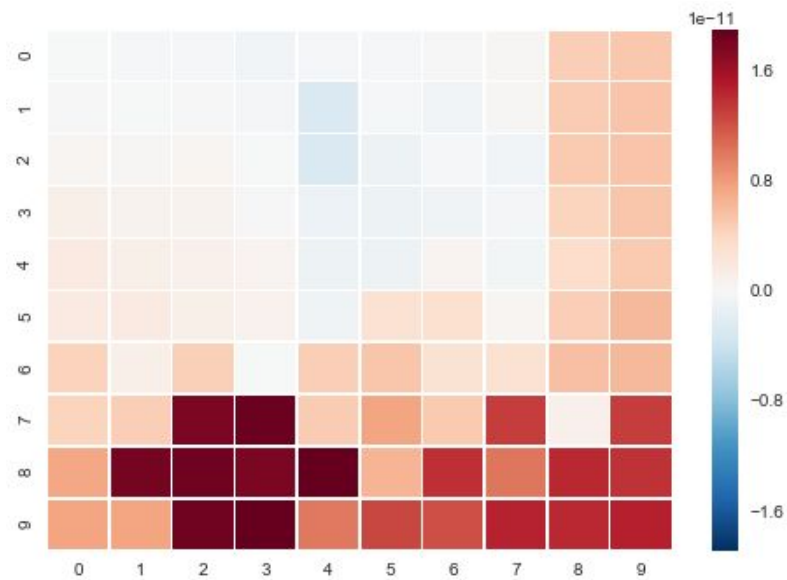
<b>Ground Truth Reward</b>	<b>Extracted Reward</b>
----------------------------	-------------------------



In the extracted reward, we can clearly see that it is negative or closer to zero in states which have a negative reward value in the ground truth reward function. We also observe a higher positive value in the extracted goal state. There are clear well defined paths towards the goal from the start state to the goal states via the edges. However, we came across an interesting observation that the highest value in the extracted reward is obtained at two adjoining states (8,2) and (8,3). This is because the agent gets stuck between trying to exit the obstacle and moving away from it and moving closer to the goal state. This results in a deadlock state in the adjoining states in the highest reward.

**21. Use the extracted reward function computed in question 20, to compute the optimal values of the states in the 2-D grid. For computing the optimal values you need to use the optimal state-value function that you wrote in question 2. For visualization purpose, generate a heat map of the optimal state values across the 2-D grid (similar to the gure generated in question 7). In this question, you should have 1 plot.**

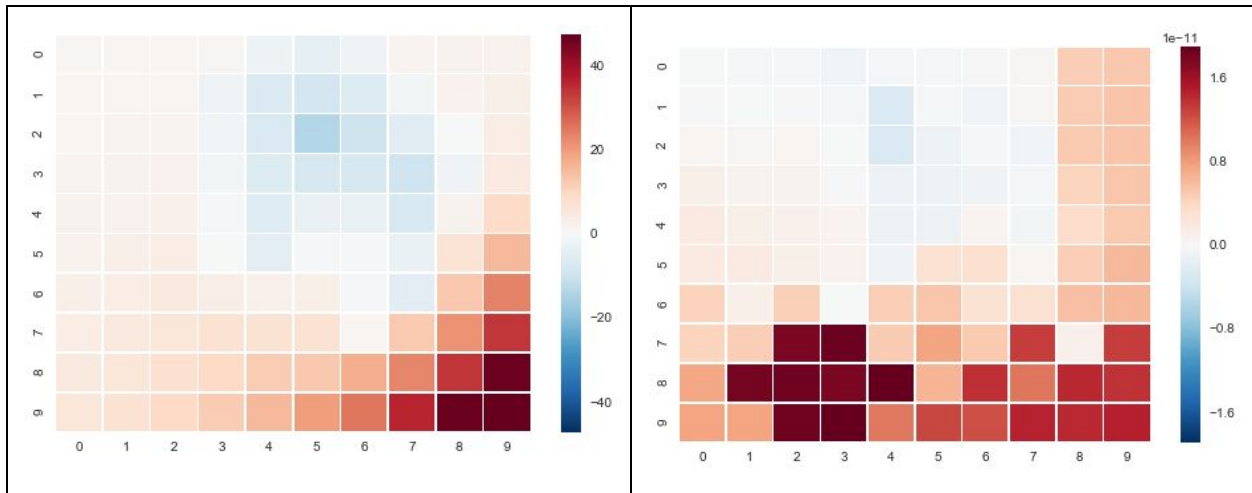
	0	1	2	3	4	5	6	7	8	9
0	-0.01	-0.02	-0.02	-0.07	-0.02	-0.02	0.01	0.02	0.47	0.51
1	0.01	-0.00	0.00	-0.03	-0.29	-0.02	-0.07	0.02	0.48	0.54
2	0.04	0.03	0.03	-0.01	-0.28	-0.10	-0.02	-0.06	0.49	0.53
3	0.11	0.07	0.07	0.01	-0.11	-0.09	-0.09	-0.03	0.42	0.52
4	0.16	0.11	0.10	0.05	-0.10	-0.09	0.06	-0.06	0.34	0.49
5	0.17	0.16	0.11	0.08	-0.08	0.29	0.30	0.03	0.46	0.61
6	0.44	0.11	0.45	-0.01	0.46	0.52	0.28	0.29	0.57	0.61
7	0.42	0.46	1.77	1.85	0.48	0.75	0.50	1.31	0.10	1.31
8	0.72	1.81	1.84	1.78	1.89	0.65	1.38	1.01	1.42	1.37
9	0.75	0.74	1.84	1.88	0.99	1.26	1.22	1.45	1.42	1.46



The high value states can be seen in the same place as the high reward patch from the extracted reward. This is because it is favorable for the agent to be in states where the reward is high. Similarly, the places where there are negative rewards are not good states to be in and this have a low state value.

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

Question 7	Question 21
------------	-------------



Similarities	Differences
<ol style="list-style-type: none"> <li>1. We can see that the states closer to the start state (0,0) have a value closer to zero.</li> <li>2. We also see that the optimal values of states from extracted rewards have a negative value at states where the ground truth reward had negative values.</li> <li>3. We see that the values across the edges on paths from start state to goal state is similar.</li> </ol>	<ol style="list-style-type: none"> <li>1. The highest optimal values of states from the extracted reward is obtained at around adjoining states (8,2) and (8,3), where the extracted reward has maximum values. This is because the agent gets stuck between trying to exit the obstacle and moving away from it and moving closer to the goal state. We observed a larger value in the neighboring states of these two states.</li> <li>2. The values around states (8,2) and (8,3) is higher than the goal state. In an ideal case, the goal state should have the maximum value.</li> </ol>

**23. Use the extracted reward function found in question 20 to compute the optimal policy of the agent. For computing the optimal policy of the agent you need to use the function that you wrote in question 9. For visualization purpose, you should generate a figure similar to that of figure 1 but with the number of state replaced by the optimal action at that state. The actions should be displayed using arrows. In this question, you should have 1 plot.**

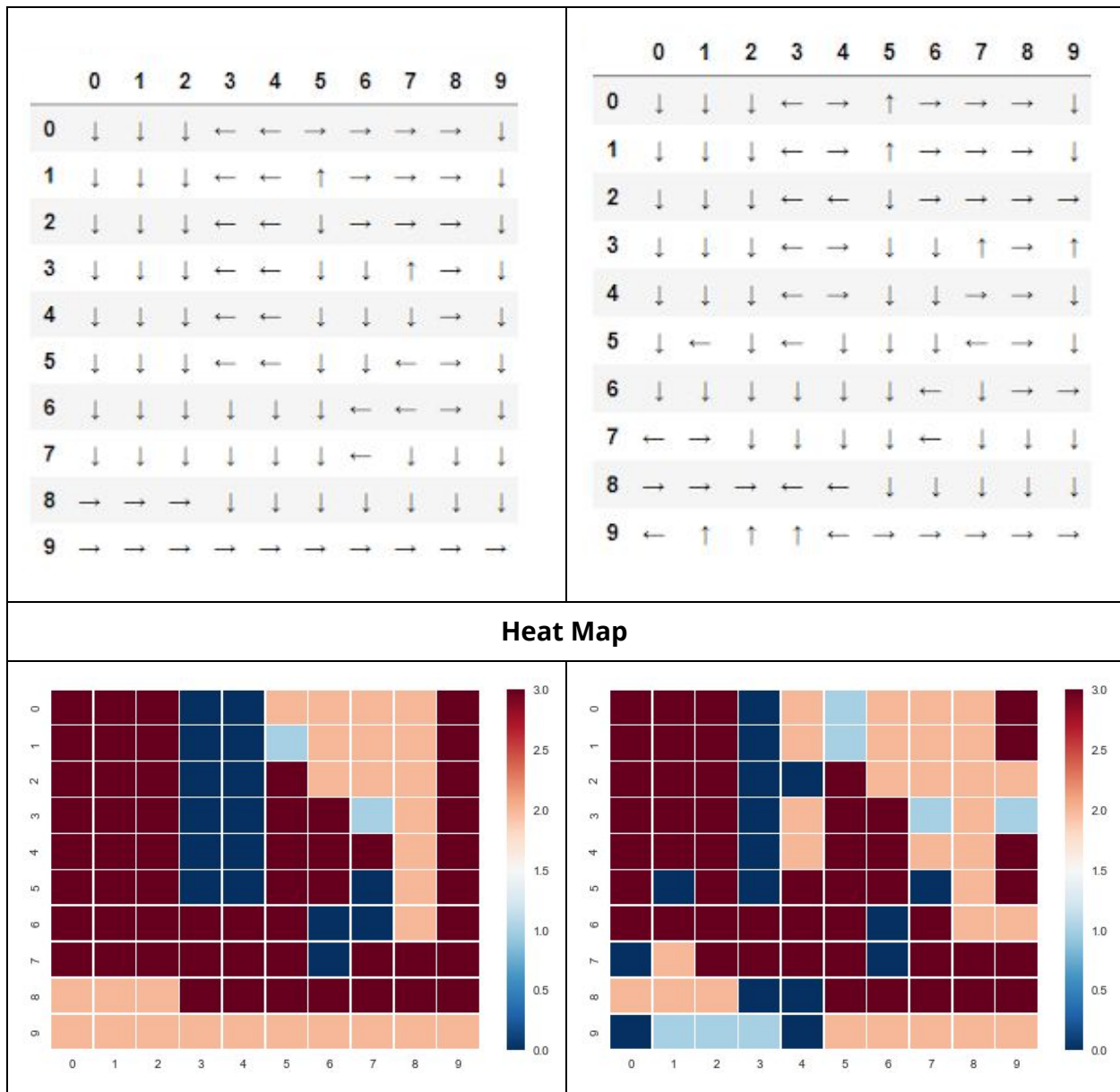


	0	1	2	3	4	5	6	7	8	9
0	↓	↓	↓	←	→	↑	→	→	→	↓
1	↓	↓	↓	←	→	↑	→	→	→	↓
2	↓	↓	↓	←	←	↓	→	→	→	→
3	↓	↓	↓	←	→	↓	↓	↑	→	↑
4	↓	↓	↓	←	→	↓	↓	→	→	↓
5	↓	←	↓	←	↓	↓	↓	←	→	↓
6	↓	↓	↓	↓	↓	↓	←	↓	→	→
7	←	→	↓	↓	↓	↓	←	↓	↓	↓
8	→	→	→	←	←	↓	↓	↓	↓	↓
9	←	↑	↑	↑	←	→	→	→	→	→

As expected, the agent tries to go from the top left(starting point) to the bottom right corner(where there is high reward point). Agent move away from negative reward regions and also move out from the region enclosed by these obstacles. We see that in some places, the agent prefers to move out of the grid and selects it as its optimal action. We also observe some deadlock scenarios where the agent actions will just cause the agent to move back and forth and not reach the actual goal.

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

Question 9	Question 23
Optimal Actions	

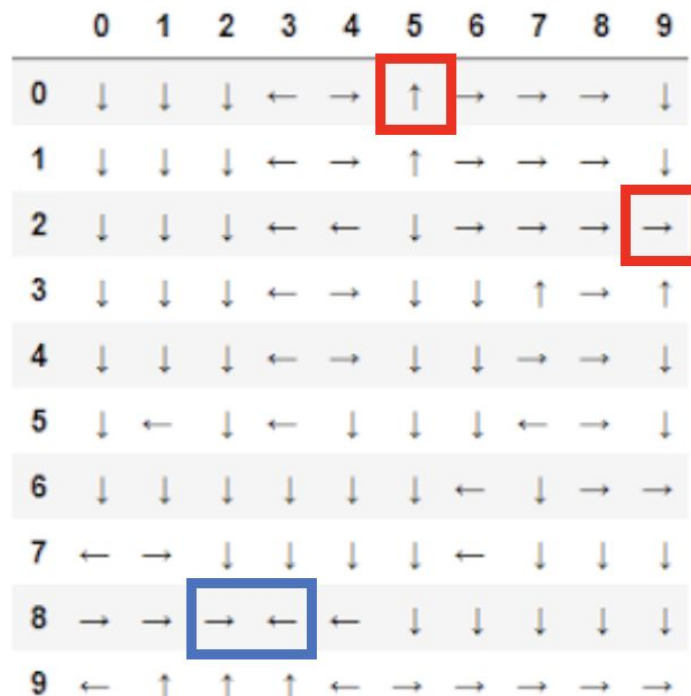


Similarities	Differences
<ol style="list-style-type: none"> <li>Both the expert and the agent, try to go from the top left to the bottom right corner.</li> <li>Both the expert and agent move away from negative reward regions and move out from the region enclosed by the obstacles</li> </ol>	<ol style="list-style-type: none"> <li>The expert does not ever have an optimal action of going out of the grid which we observe in some places for the agent</li> <li>For the expert actions, we can start from any point and reach the bottom right corner of high reward. For the agent however, we see local deadlocks in the agent actions</li> </ol>

which prevent the agent from reaching the high reward point in the bottom right corner.

25. From the figure in question 23, you should observe that the optimal policy of the agent has two major discrepancies. Please identify and provide the causes for these two discrepancies. One of the discrepancy can be fixed easily by a slight modification to the value iteration algorithm. Perform this modification and re-run the modified value iteration algorithm to compute the optimal policy of the agent. Also, recompute the maximum accuracy after this modification. Is there a change in maximum accuracy? The second discrepancy is harder to fix and is a limitation of the simple IRL algorithm. If you can provide a solution to the second discrepancy then we will give you a bonus of 50 points.

**Ans:** Below is the plot for Question 23. We have also highlighted the two discrepancies that we have found. The first discrepancy is highlighted in red and the second one in blue.



---

So, if we see the optimal policy obtained above, we can observe the following two discrepancies clearly and easily.

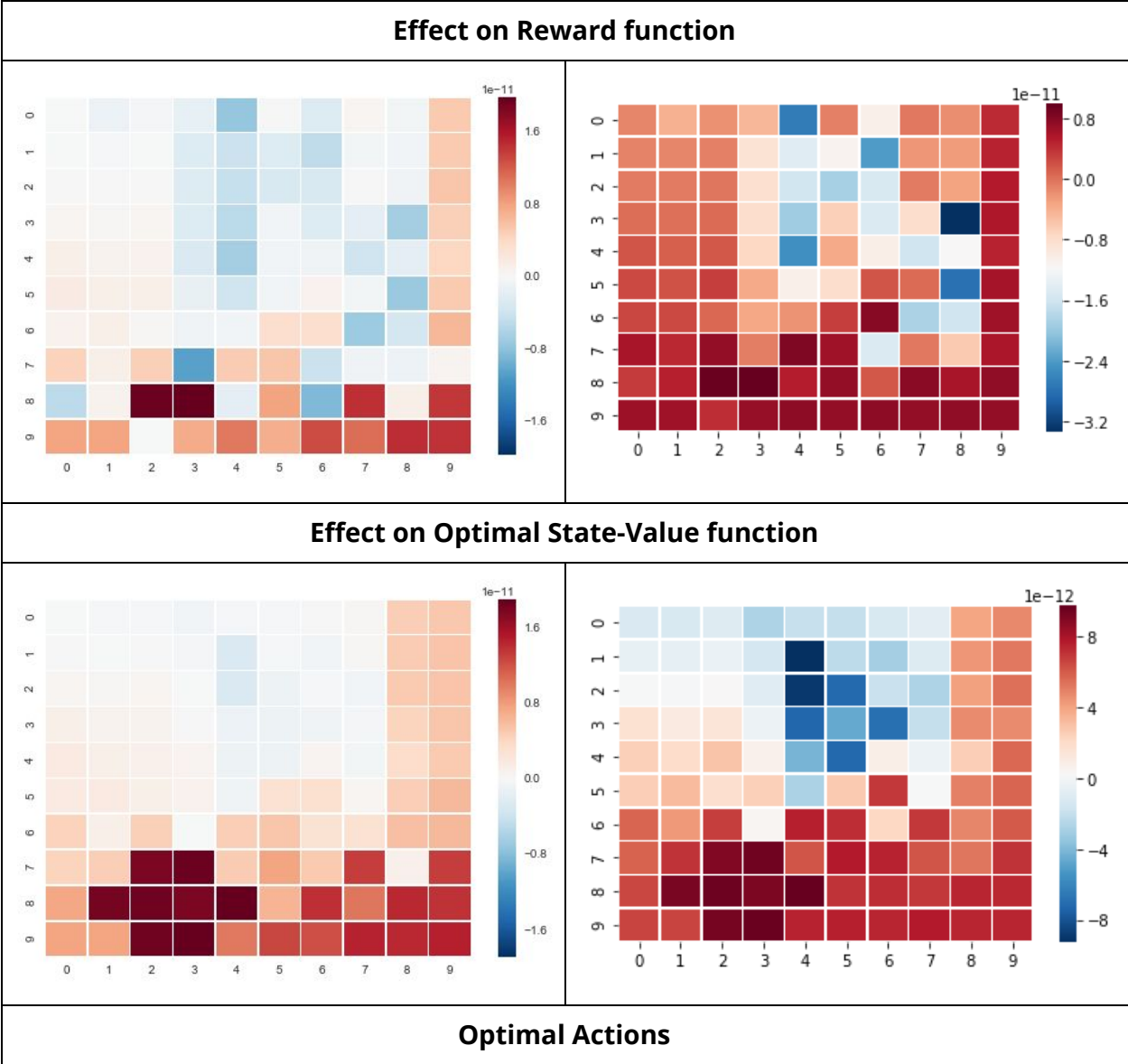
### **Discrepancy 1: Arrows leading agent out of bounds**

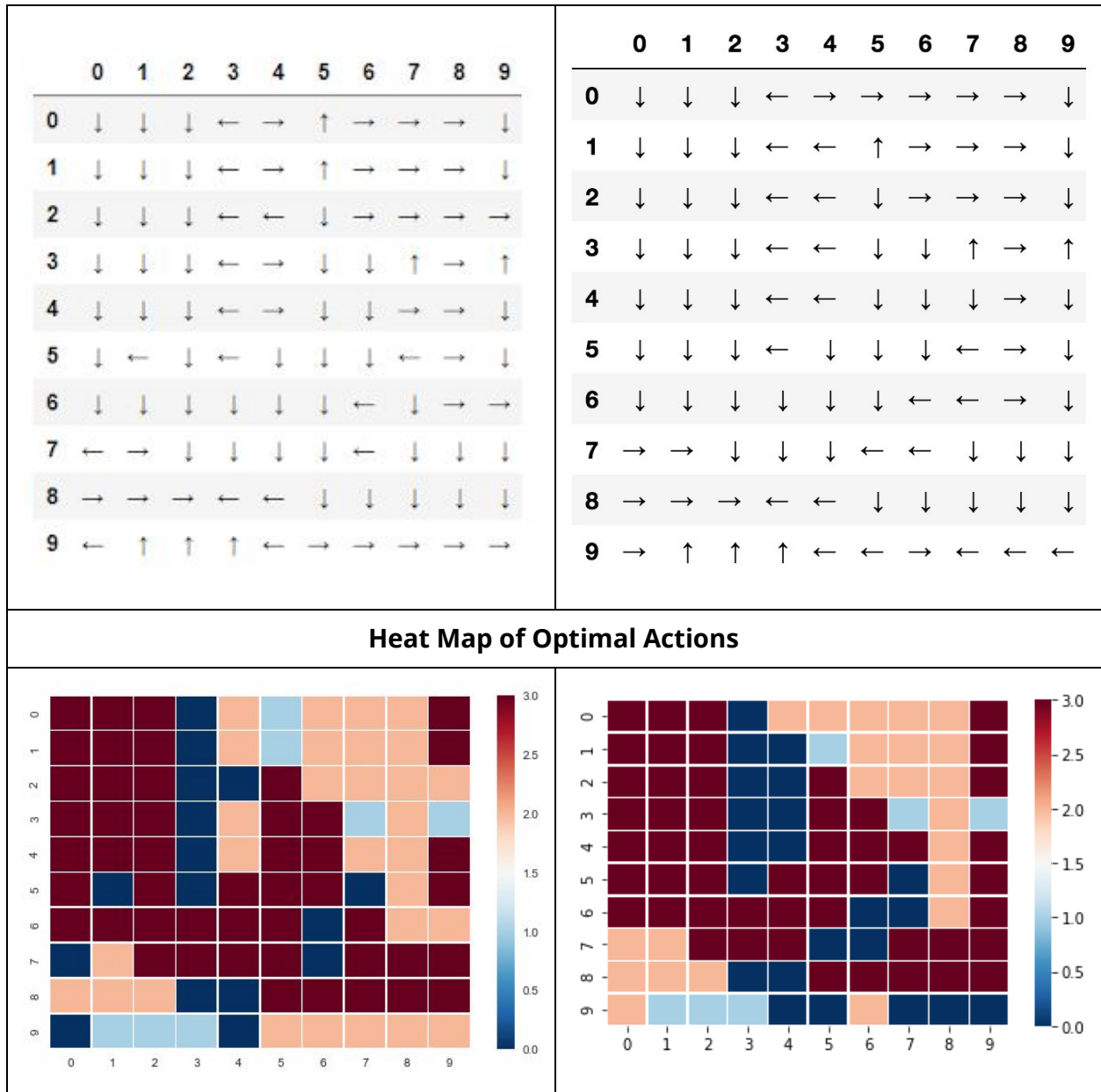
The first discrepancy that we think is the arrows going out of the grid on the boundary states. The meaning of this action on the boundary state represents that the agent will stay at the same state. We think that this is not the right action since the agent can't just stay at that state, rather it should be able to move to the the higher reward states by taking some beneficial action if not the best.

**Solution:** To fix this problem, we modify the computation part of Value Iteration Algorithm slightly. For the non-boundary states the computation remains the same. But, for the boundary states, we apply some modification. So, if for any boundary state, the action taken is outside the grid, then we use next best argmax from the actions. This easy fix enables us to avoid the above discrepancy.

On fixing this discrepancy we reach closer to the expert actions evident from both the accuracy value and agent optimal actions.

Comparison of results with/without discrepancy fixed	
Question 23: Results without any fix	Question 25: Discrepancy 1 fixed(agent going out of bounds)
Accuracy	
79	85





## Discrepancy 2: Deadlocks in optimal actions

We observed that there are some set of actions that lead to deadlocks and will not allow the agent to reach the points where the rewards are much higher. The deadlocks are due to a relative higher reward region in the area which we have got from the IRL algorithm. This local deadlock does not allow the agent to reach states where it can get much higher rewards.

The root cause of these deadlocks is this high reward part in the environment. The position of the patch is actually correlated with the opening in the bottom part of

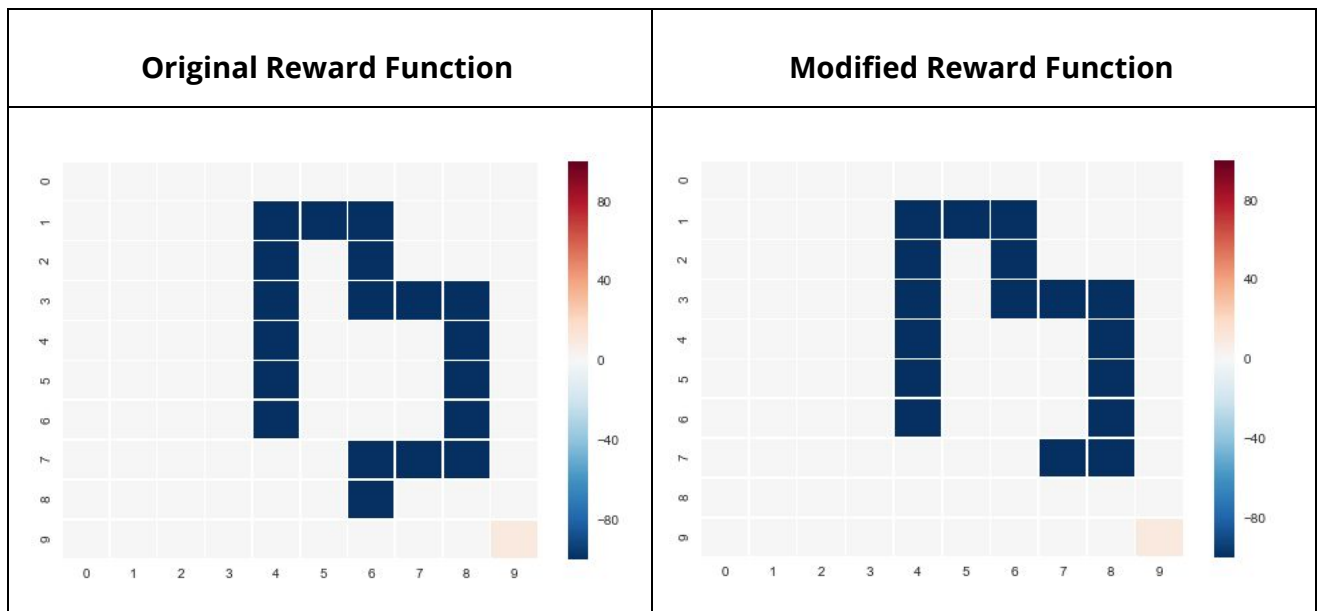
---

the heatmap of the reward function which we have explored later. The agent tries to go away from the obstacle and the original high reward point is on the other side of the obstacle.

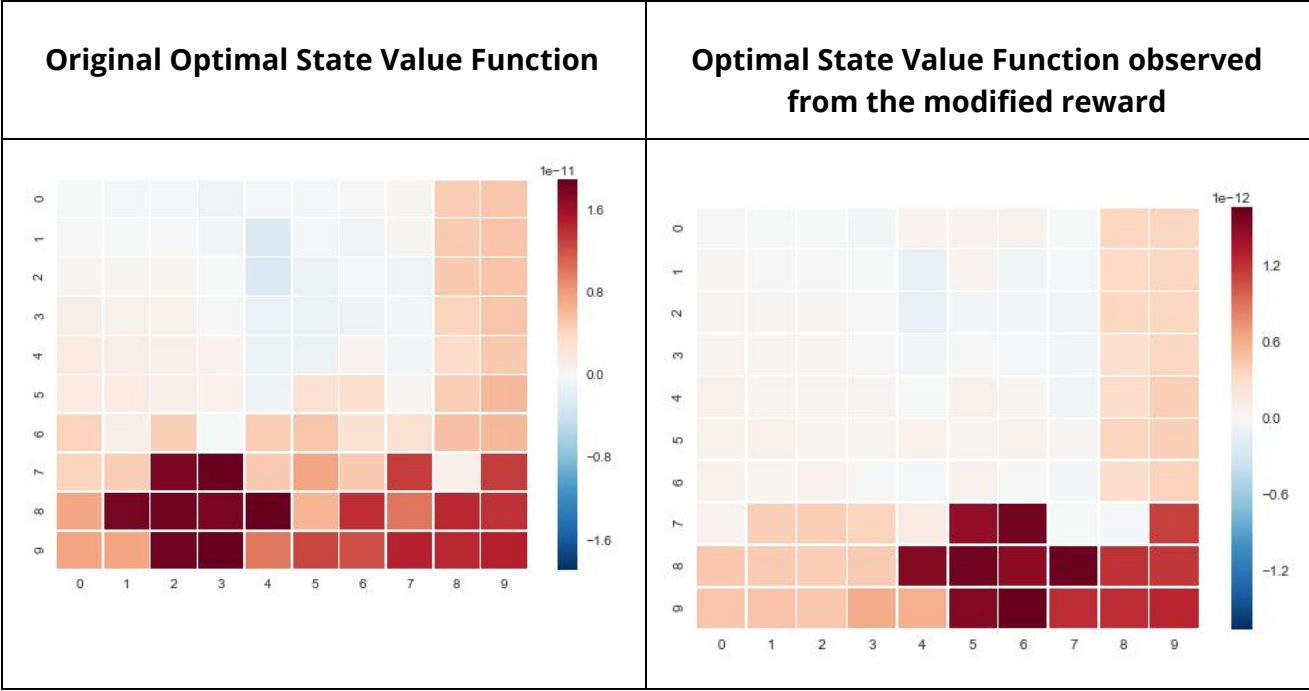
We observed that by removing some of the low rewards points or obstacles, the position of the deadlock shifts.

1) Remove some of obstacles to show how the deadlock shifts

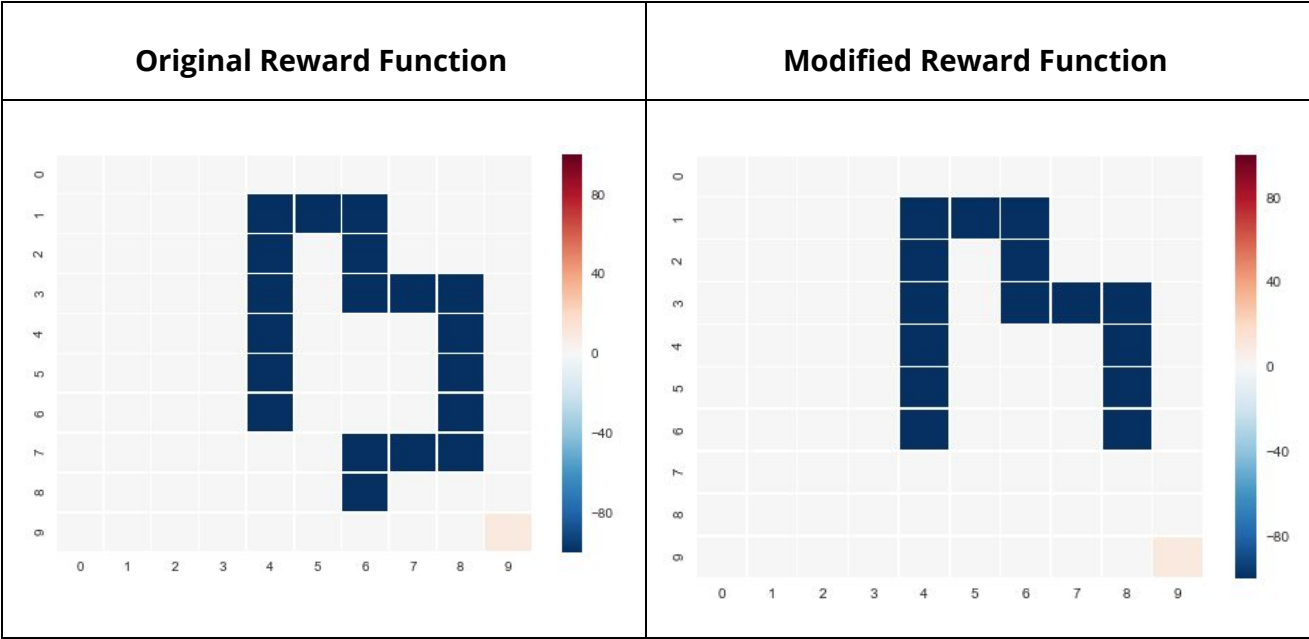
To try to analyze the deadlock we tried to do some experiments as to how does it relate with the reward function. So, we tried to modify the reward function as below,



After modifying the reward function we compared the changes in the optimal state value function as below

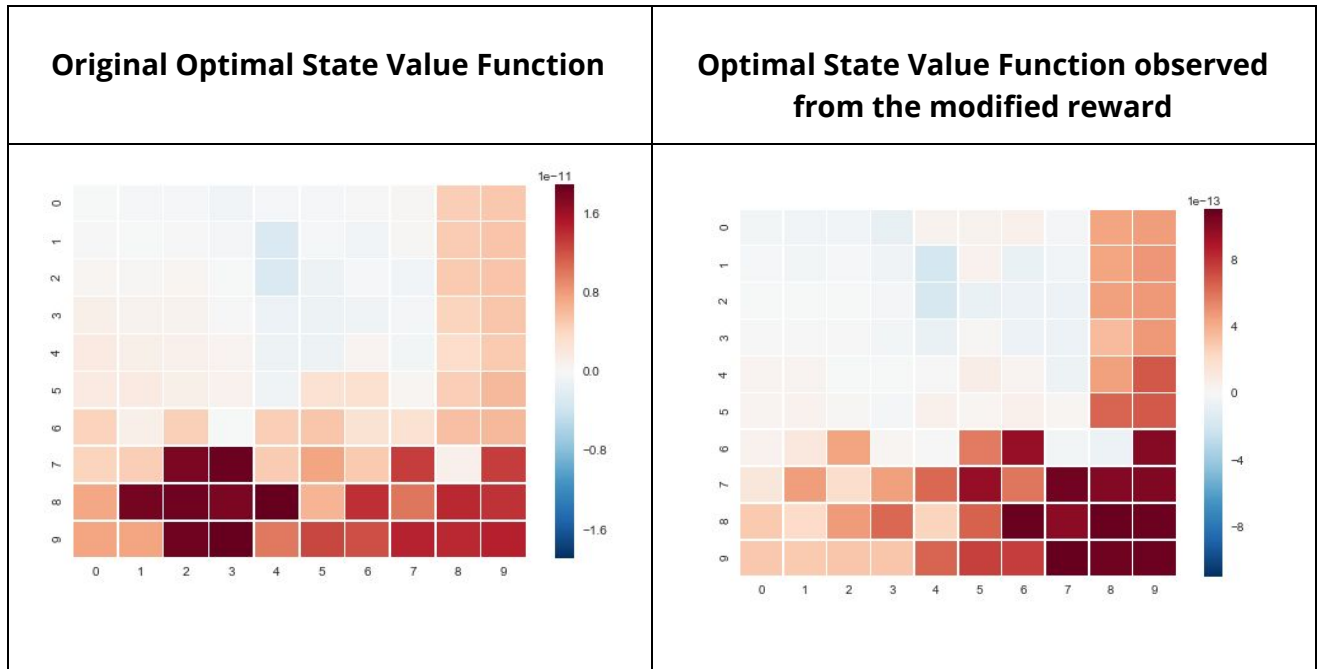


We further modified the reward function as below and correspondingly observed changes in the optimal state value function.





After modifying the reward function we compared the changes in the optimal state value function as below

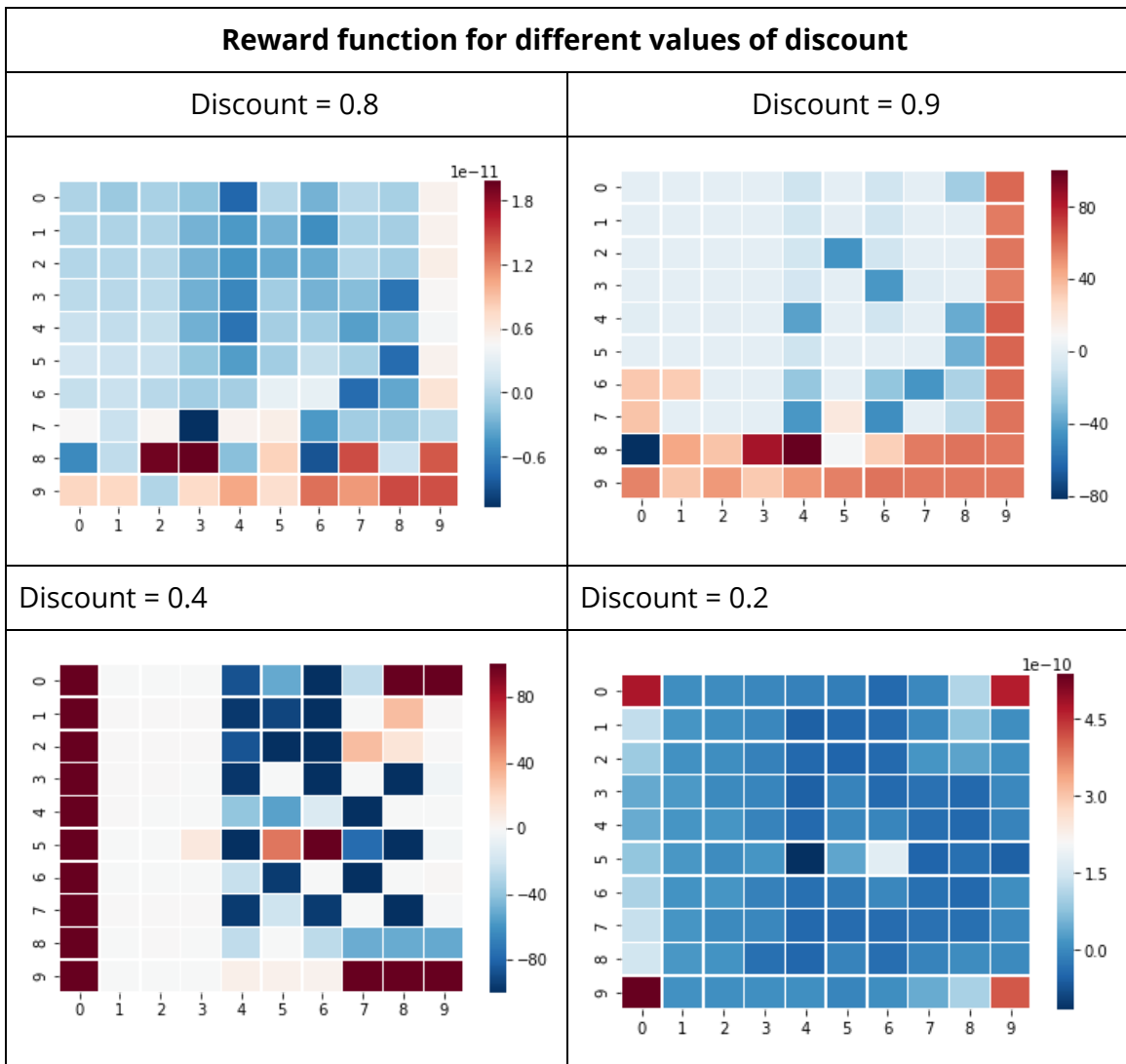


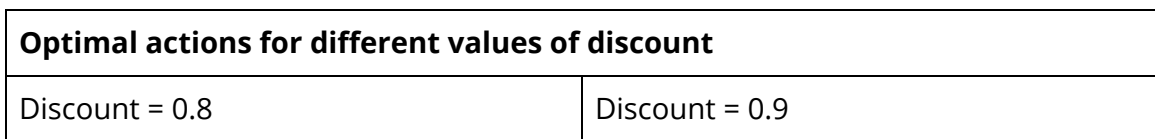
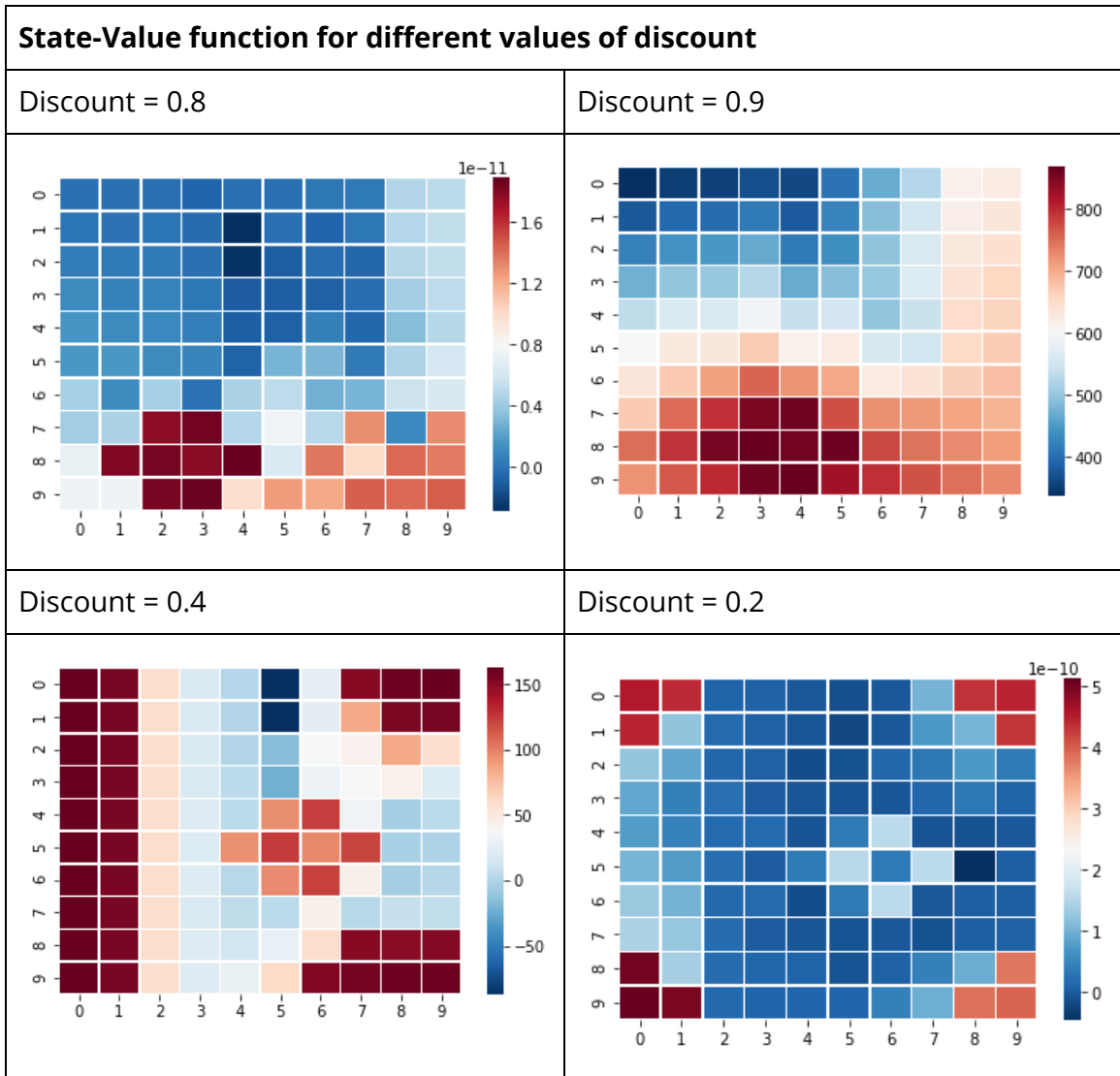
As we can clearly observe that in the optimal state value function, the red patch keeps shifting to the right. The position of the patch is actually correlated with the opening in the bottom part of the heatmap of the reward function. The agent tries to go away from the obstacle and the high reward point is on the other side of the obstacle. So when some parts of the obstacles were removed the high reward patch moved towards the actual high reward point.

To try to fix the deadlock situations, we also tried to experiment with the discount factor. The fact that the discount rate is bounded to be smaller than 1 is a mathematical trick to make an infinite sum finite. This helps proving the convergence of certain algorithms. In practice the discount factor could be used to model the fact that the decision maker is uncertain about if in the next decision instant the world is going to end.

For example, If the decision maker is a robot, the discount factor could be the probability that the robot is switched off in the next time instant (the world ends in the previous terminology). That is the reason why the robot is short sighted and does not optimize the sum reward but the discounted sum reward. We did some experiments by varying discounts to see some changes in the reward function 2. This however did not solve the deadlock situations. The results for this are listed below.

Accuracy for different values of discount	
Discount = 0.8	Discount = 0.9
79	73
Discount = 0.4	Discount = 0.2
94	96





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

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

Discount = 0.4

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

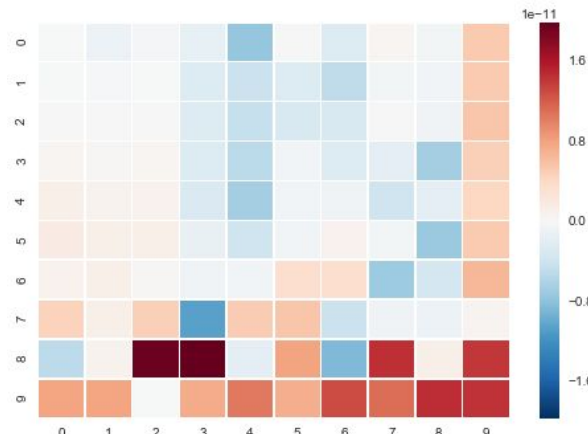
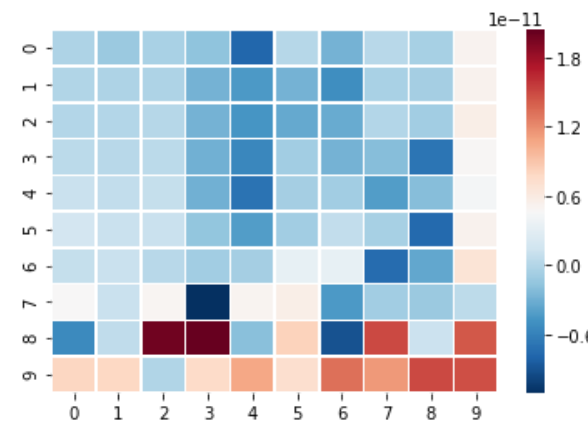
Discount = 0.2

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

**Solution:** To actually fix the deadlocks we have modified computation step of the value iteration algorithm. We extend this solution on top of the solution for discrepancy 1. We store all the actions instead of just the argmax. While considering an action for a state, if the action is causing the agent to go out of bound or it is leading to form a deadlock, that action is not chosen and we move to the next best. Thus, even though it the best possible action, as it might cause to go out of bounds

or deadlocks next best actions are considered and chosen. On fixing the deadlocks, we get slightly different different reward function and state value functions as well. On fixing the deadlock situations, we can start from any point in the environment and we will be able to reach the corner parts where the reward is high.

On fixing both the discrepancy we reach even closer to the expert actions evident from both the accuracy value and agent optimal actions.

Comparison of results with/without discrepancy fixed	
Question 23: Results without any fix	Question 25: Discrepancy 1 & 2 fixed(agent going out of bounds and deadlocks)
Accuracy	
79	87
Effect on Reward function	
	
Effect on Optimal State-Value function	

