ECE 232E Project 3

Yifan Shu, Chengshun Zhang, Xuan Yang, Yifan Zhang

Introduction

Reinforcement Learning (RL) is the task of learning from interaction to achieve a goal. The learner and the decision maker is called the agent. The thing it interacts with, comprising everything outside the agent, is called the environment. These interact continually, the agent selecting actions and the environment responding to those actions by presenting rewards and new states.

In the first part of the project, we will learn the optimal policy of an agent navigating in a 2-D environment. We will implement the Value iteration algorithm to learn the optimal policy.

Inverse Reinforcement Learning (IRL) is the task of extracting an expert's reward function by observing the optimal policy of the expert. In the second part of the project, we will explore the application of IRL in the context of apprenticeship learning.

Part 1: Reinforcement learning (RL)

In this project, we will learn the optimal policy of a single agent navigating in a 2-D environment.

Question 1.

To visualize reward function 1 and reward function 2, we generated two heat maps shown in Fig 1 and Fig 2. The coloring scale is shown in the sidebar.

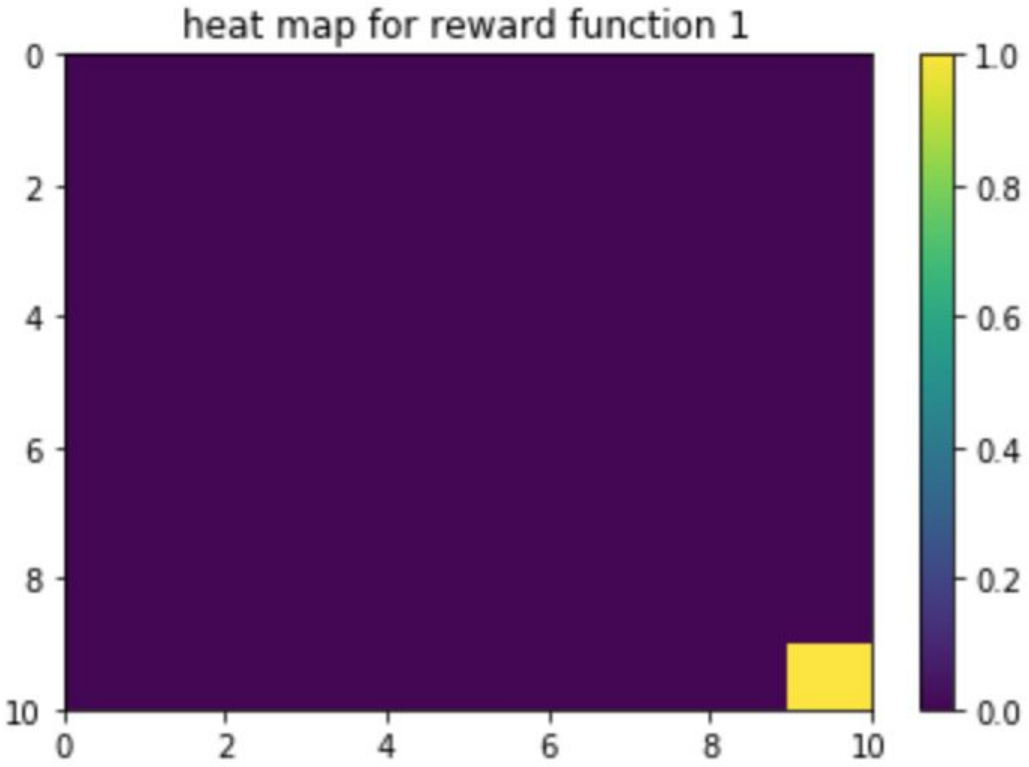

Figure 1. Heat Map for Reward Function 1

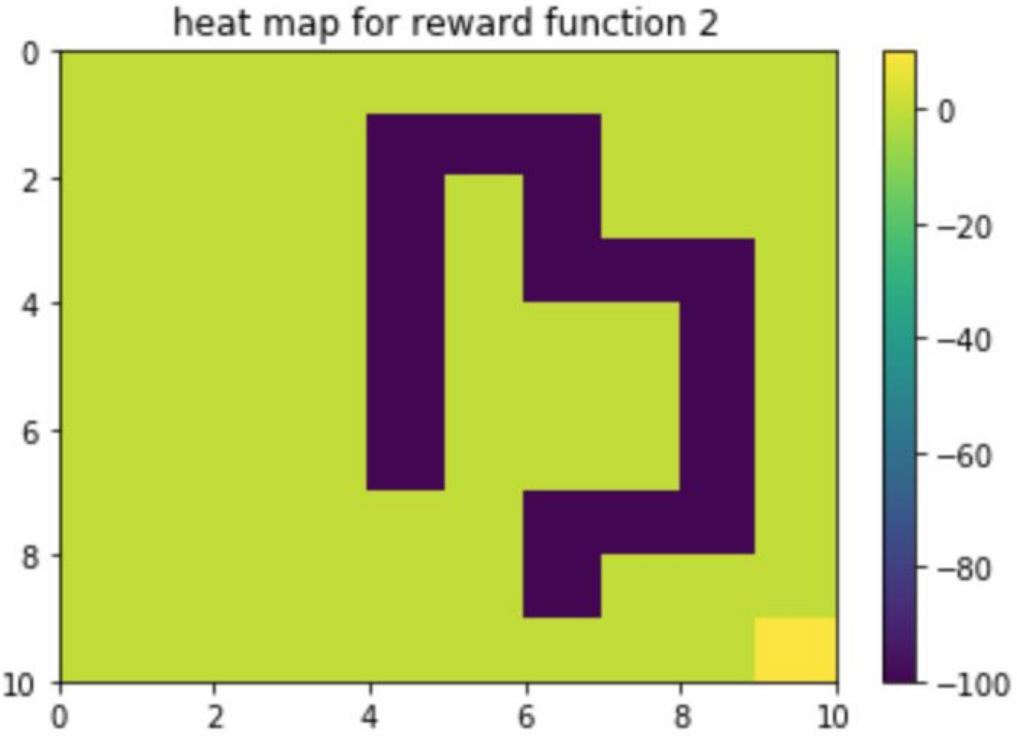

Figure 2. Heat Map for Reward Function 2

Question 2.

We created the environment of the agent with the following parameters:

- Number of states = 100 (state space is a 10 by 10 square grid as displayed in heat maps)
- Number of actions = 4 (set of possible actions is up, down, left and right)
- $w = 0.1$
- Discount factor $\gamma = 0.8$
- Reward function 1
- $\varepsilon = 0.01$

After the creation of the environment, we implemented the Initialization and Estimation phases of Value Iteration algorithm by writing a function *optimal_state_value* that takes these environment parameters as input and outputs the optimal value for each state. We also encapsulated a function *compute_max* to calculate the maximum value in line 10 of the algorithm.

For visualization purpose, we generated a value grid which displays the optimal value for each state. The result is shown in Fig 3.

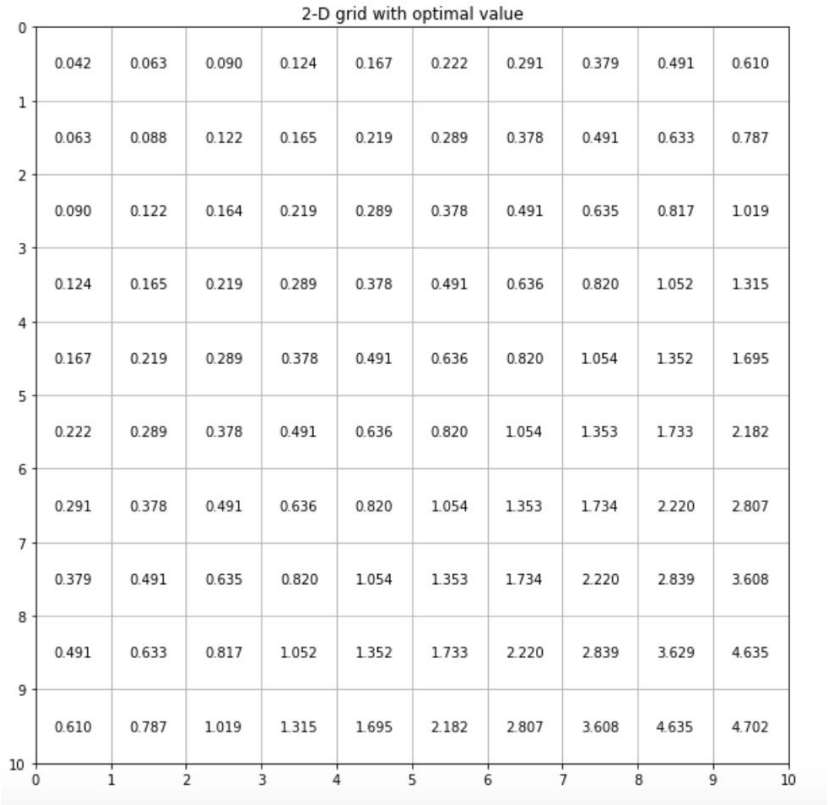

Figure 3. Grid with Optimal Values Using Reward Function 1

Question 3.

We then generated the heat map corresponding to the optimal values shown in Fig 3 for better and perceptual visualization. The heat map is shown in Fig 4.

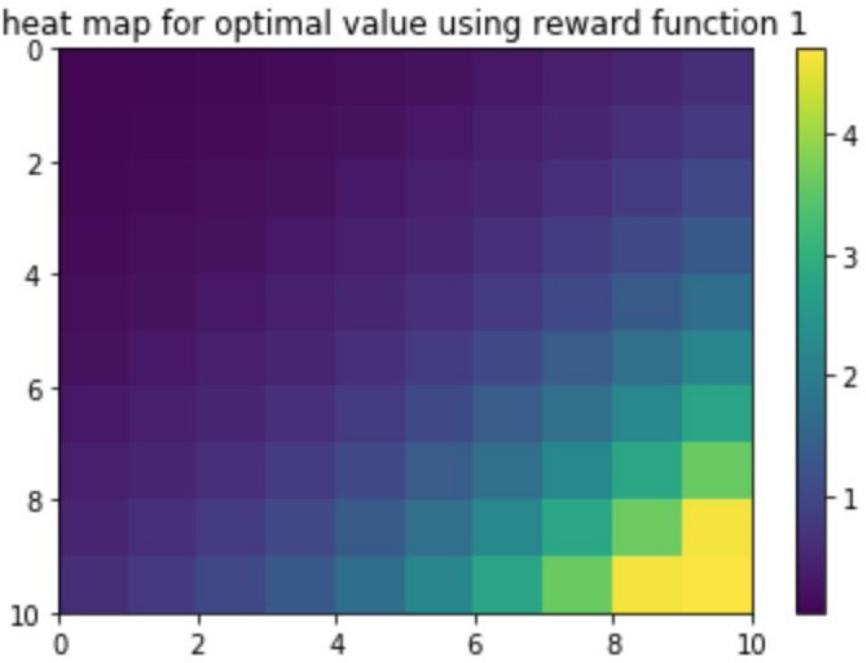

Figure 4. Heat Map for Optimal Values with Reward Function 1

Question 4.

By observing the heat map in Fig 4, we find that the value distribution is symmetric about the diagonal. The values at upper left part are smaller than those at lower right part, with the smallest value at (0, 0) position being 0.042 and largest value at (9, 9) being 4.702. This is because we only have reward function value 1 at the (9, 9) position and the rest 0. At each state, to achieve the best reward, the agent would move toward the lower right corner and gets larger rewards as it is closer to the corner. We can view it in this way for some better intuition. The lower right corner is a signal source and emits signal with certain strength outwards and the closer we are to the source, the stronger signal we receive. Each circle with the source as the center has the same strength of the signal.

Question 5.

Similar to the Estimation phase in Value Iteration algorithm, we implemented the Computation phase by writing a *compute_argmax* function, which corresponds to line 15 in the algorithm. So we got the optimal policy, i.e., the action taken by the agent at each state. We plotted the figure with an arrow in each grid showing the action taken at that state, which is shown in Fig 5.

As we can see in the figure, the action at each state is going right or going down, towards the lower right corner, which has a reward value 1. As we have explained in previous question, this matches our intuition exactly.

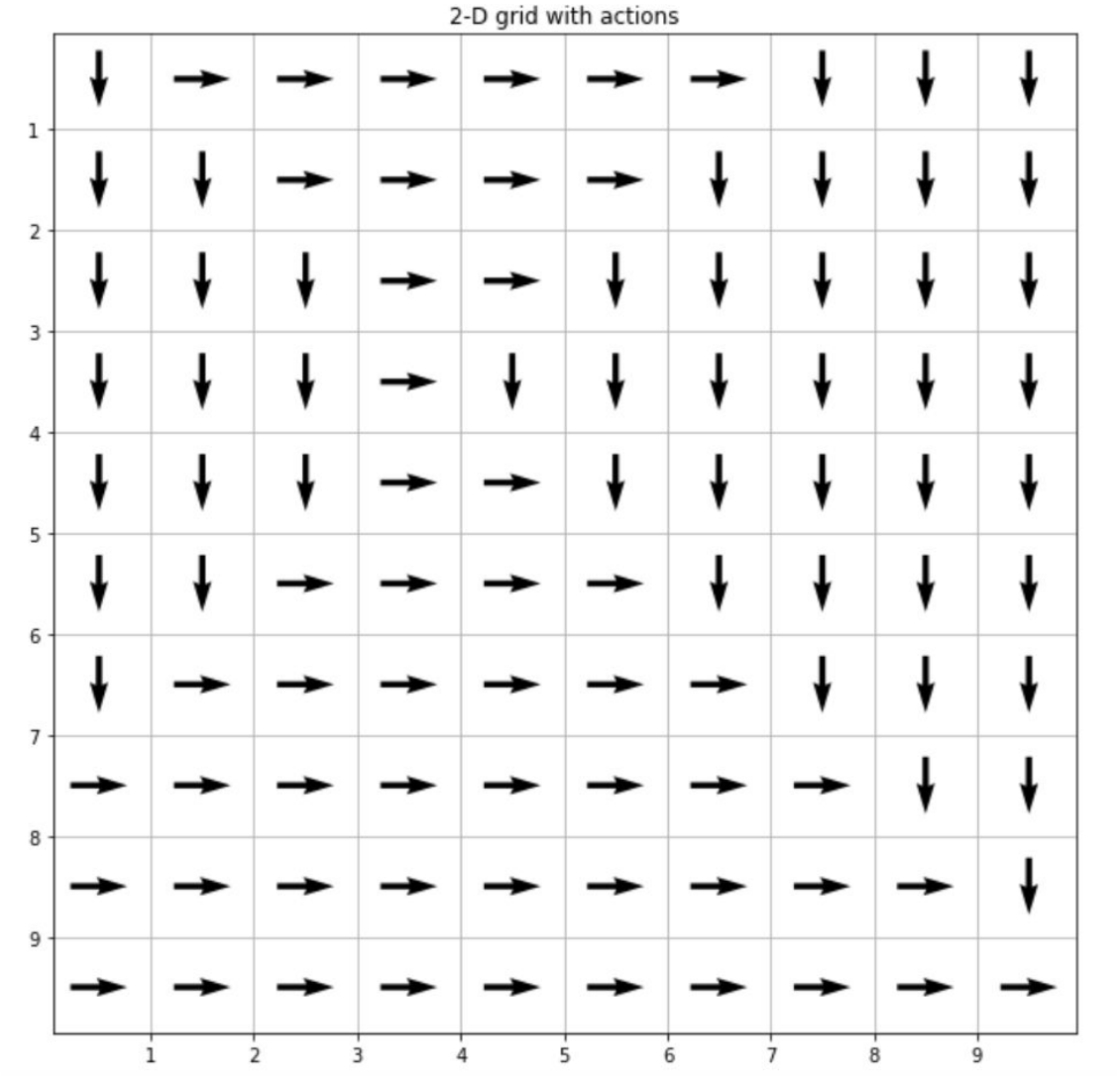

Figure 5. Optimal Action taken at Each State with Reward Function 1

It is also possible for the agent to calculate the optimal action by looking at the optimal value of its neighboring grids. The agent just needs to go toward the state with highest value, either going right or going down.

Question 6.

With all the other settings the same as in Question 2, we used the reward function 2 instead and similarly computed the optimal value for each state. The result is shown in Fig 6.

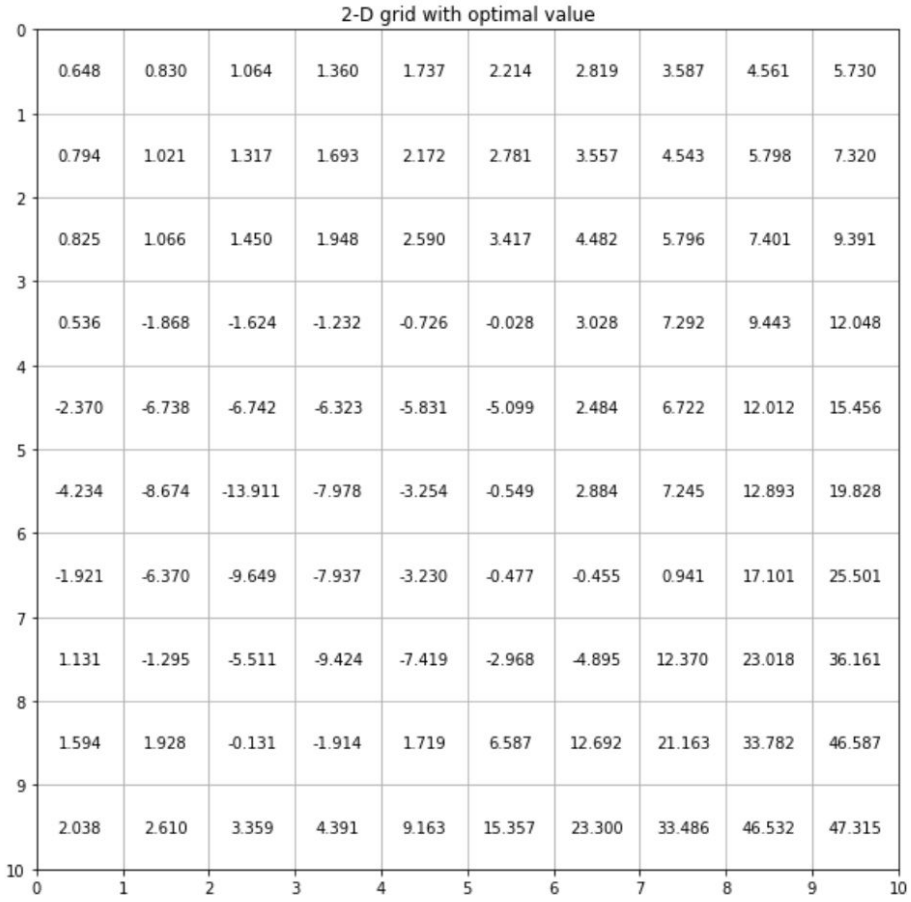

Figure 6. Grid with Optimal Values Using Reward Function 2

Question 7.

We similarly generated a heat map corresponding to the optimal values shown in Fig 6. The resulting figure is shown in Fig 7.

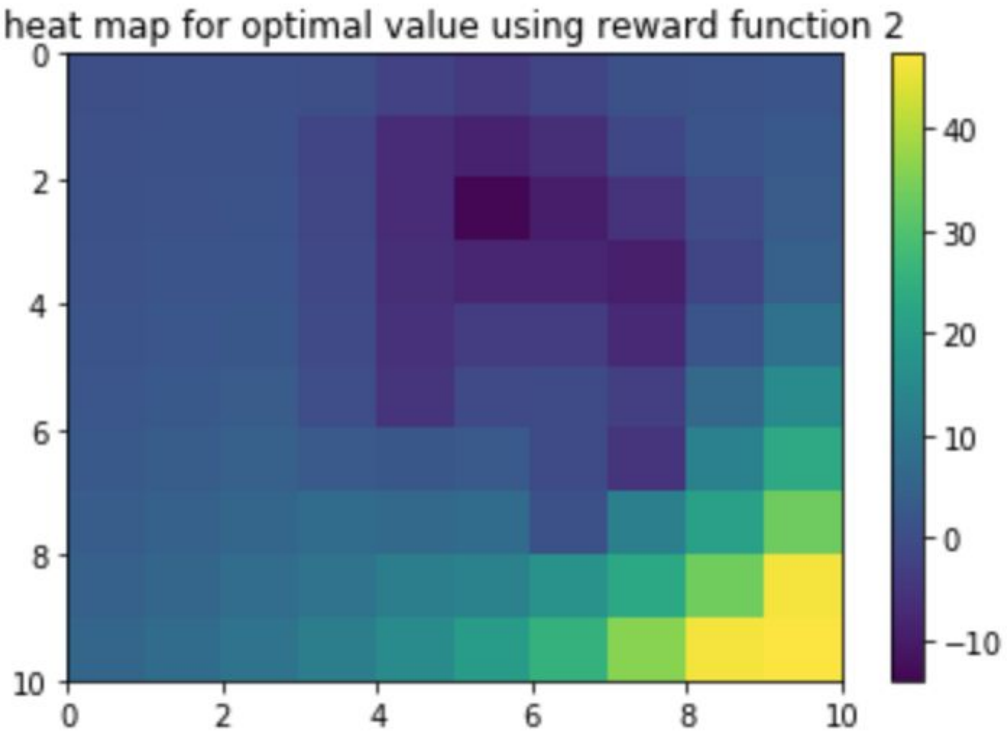

Figure 7. Heat Map for Optimal Values with Reward Function 2

Question 8.

The optimal values at the bottom right are the largest and the values at thumbs up region are the smallest because the reward value is 10 at the bottom right corner and the values around the thumbs up region are -100. The value distribution basically is that the state closer to the bottom right, it has larger optimal value; the closer to the -100 states, it has smaller optimal value. And the values are not symmetric anymore as the reward values are not.

Question 9.

We implemented the Computation phase of the Value Iteration algorithm to calculate the optimal policy taken at each state, similar to Question 5, with the reward function 1 replaced by reward function 2. Again, we plotted a figure with arrows in each grid to visualize the action taken at that state. The result is shown in Fig 8.

By intuition, the actions taken at each state should be toward bottom right corner and outward -100 positions. As we can see from the figure, all the states, except those at the position of -100 reward values, either move down or right to approach the bottom right corner. The states whose

reward values are -100 all move outward to stay away from very negative rewards. So this matches our intuition.

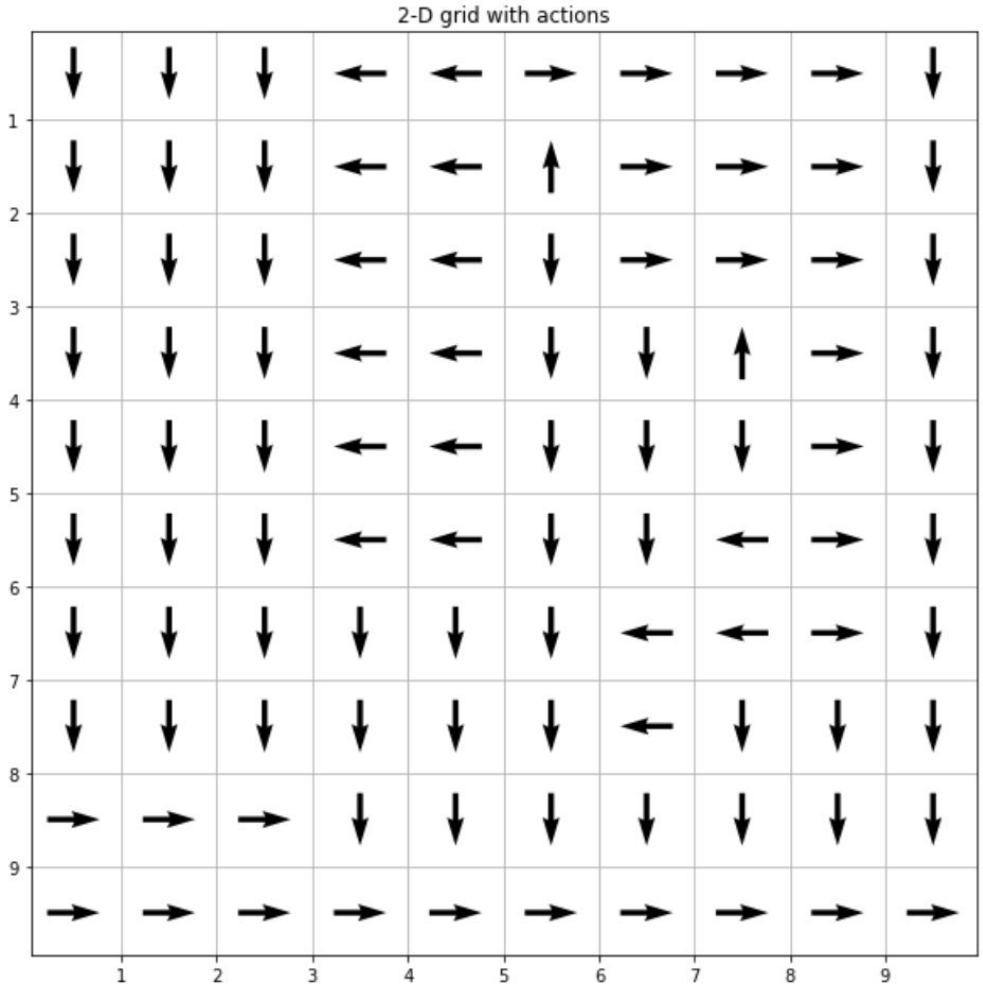

Figure 8. Optimal Action taken at Each State with Reward Function 2

Question 10.

The matrix of **c**, **x**, **D** could be expressed like this:

$$\mathbf{c} = \begin{bmatrix} 0_{100\times1} \\ I_{100\times1} \\ -\lambda_{100\times1} \\ 0_{100\times1} \end{bmatrix}$$

$$\mathbf{x} = \begin{bmatrix} R_{100\times1} \\ T_{100\times1} \\ U_{100\times1} \\ Rmax_{100\times1} \end{bmatrix}$$

$$\mathbf{D} = \begin{bmatrix} -[(P_{a1}(i) - P_a(i))(I - \gamma P_{a1})^{-1}]_{300\times100} & I_{300\times100} & 0_{300\times100} & 0_{300\times100} \\ -[(P_{a1} - P_a)(I - \gamma P_{a1})^{-1}]_{300\times100} & 0_{300\times100} & 0_{300\times100} & 0_{300\times100} \\ I_{100\times100} & 0_{100\times100} & -I_{100\times100} & 0_{100\times100} \\ -I_{100\times100} & 0_{100\times100} & -I_{100\times100} & 0_{100\times100} \\ I_{100\times100} & 0_{100\times100} & 0_{100\times100} & I_{100\times100} \\ -I_{100\times100} & 0_{100\times100} & 0_{100\times100} & I_{100\times100} \end{bmatrix}$$

Where the subscript is the dimension of the matrix.

Question 11.

Here we swept parameter λ from 0 to 5 and get 500 evenly spaced values of λ. Here we used the optimal policy of the agent found in question 5 to fill in the OE(s) values. Then used question 3 to compute the accuracy of the IRL algorithm for this value of λ. The plot of λ against accuracy is as below:

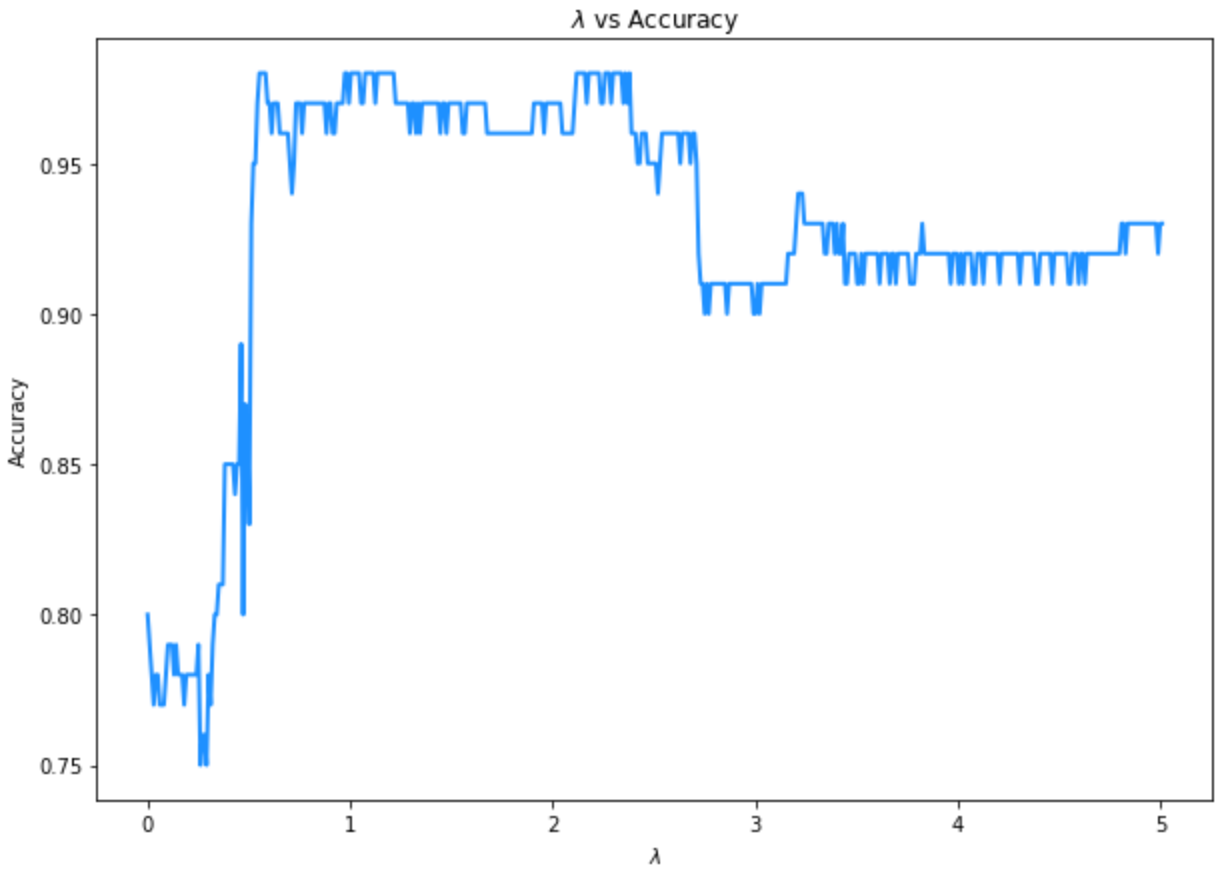

Figure 9. The Action Accuracy Against ƛ Using OE(s) Values Found in Question 5

Question 12.

We used plot in question 11 to compute the value of λ which the accuracy is the maximum. And the λ_{max} is:

$$\lambda_{max} = 0.5522044088176352$$

Question 13.

Using the λ_{max}, we generated heat maps of the ground truth reward and the extracted reward, which are as below:

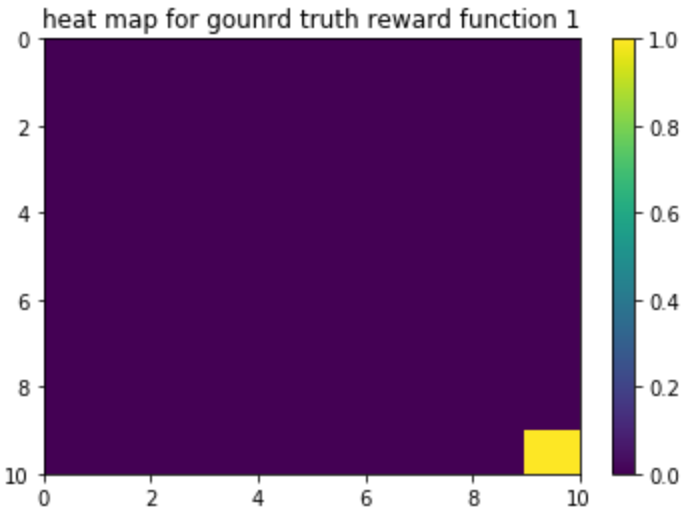

Figure 10. Heat Map of Ground Truth Reward Reward Function 1

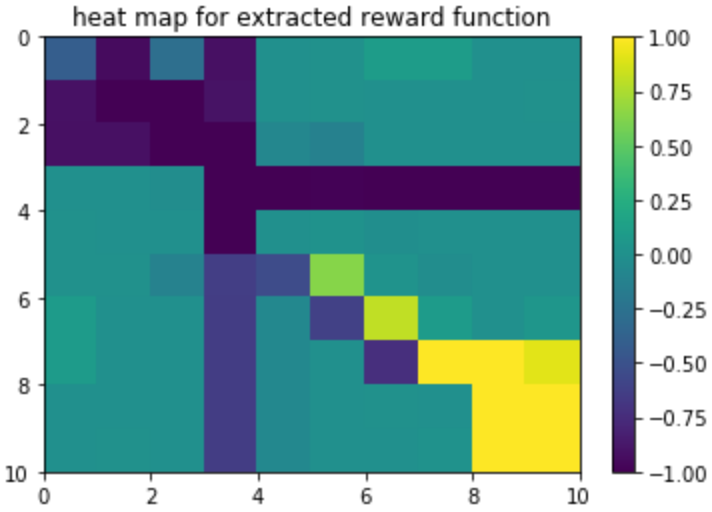

Figure 11. Heat Map of Extracted Reward Function 1

Question 14.

We used the extracted reward function computed in question 13 to compute the optimal values of states in the 2-D grid. And the heat map of the optimal state values in 2-D grid is shown as below:

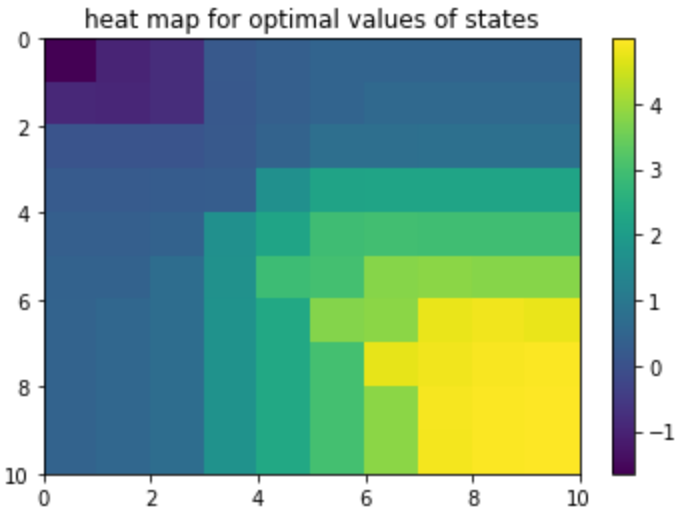

Figure 12. Heat Map of Optimal Values of States Using Extracted Reward Function 1

Question 15.

Comparing the two heat maps of optimal values of states in question 14 (Fig.12) and question 3 (Fig.4), we could find that both of them gets larger optimal values of states at the bottom right and smaller optimal values of states at the top-left corner. But there are also some differences, the heat map in Fig.12 shows larger values of optimal values of states at the bottom right than the heat map in Fig.4. The state values from top-left corner to bottom right corner increases faster in Fig.12 than that in Fig.4.

Question 16.

Here we used the extracted reward function computed in question 13 to compute the optimal policy of the agent. We used the function in question 5 to compute the optimal policy of agent. The plot of the arrows showing the optimal policy of agent in each state in 2-D grid shows as below:

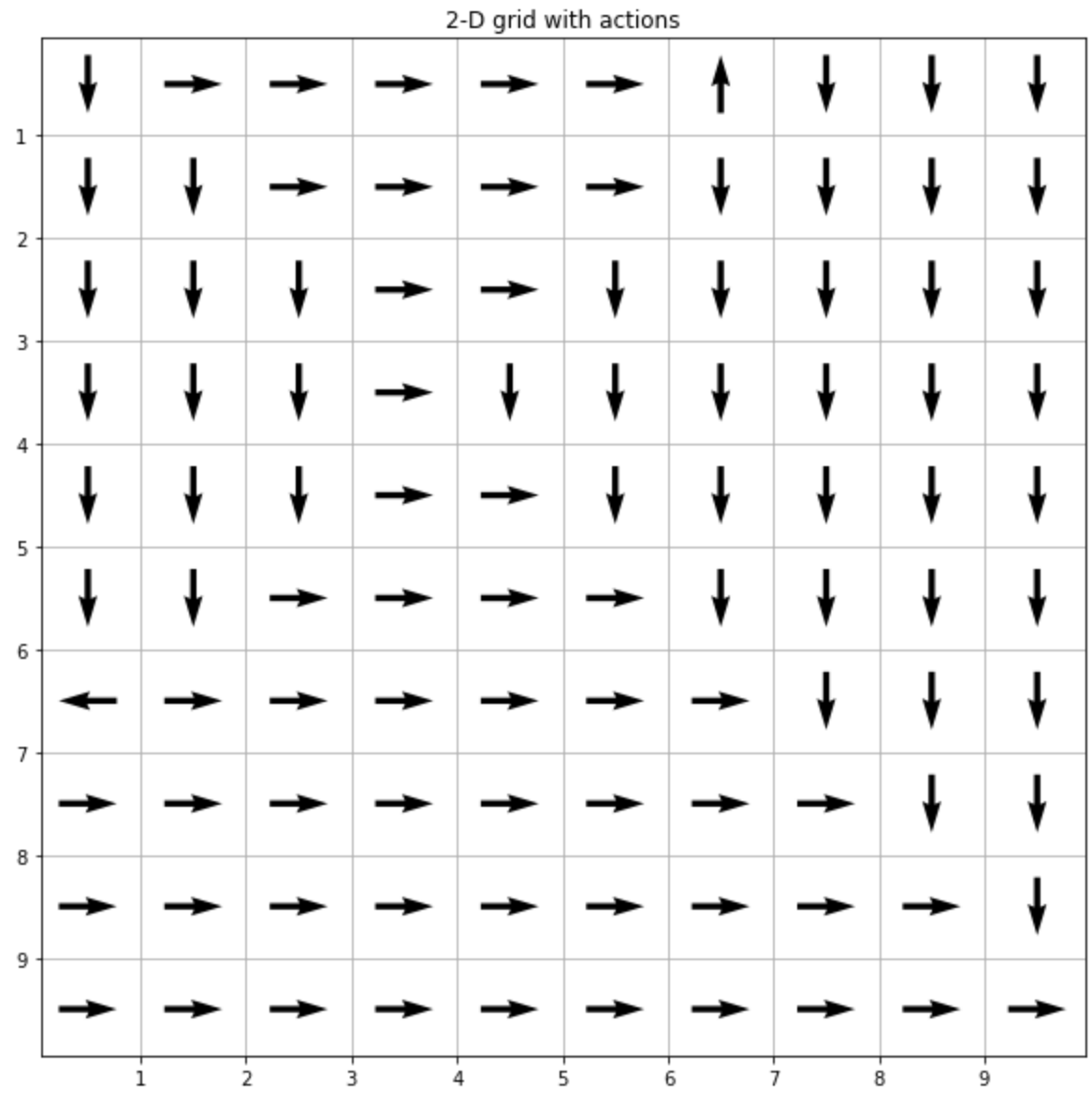

Figure 13. Arrows of Optimal Policy of Agent Using IRL

Question 17.

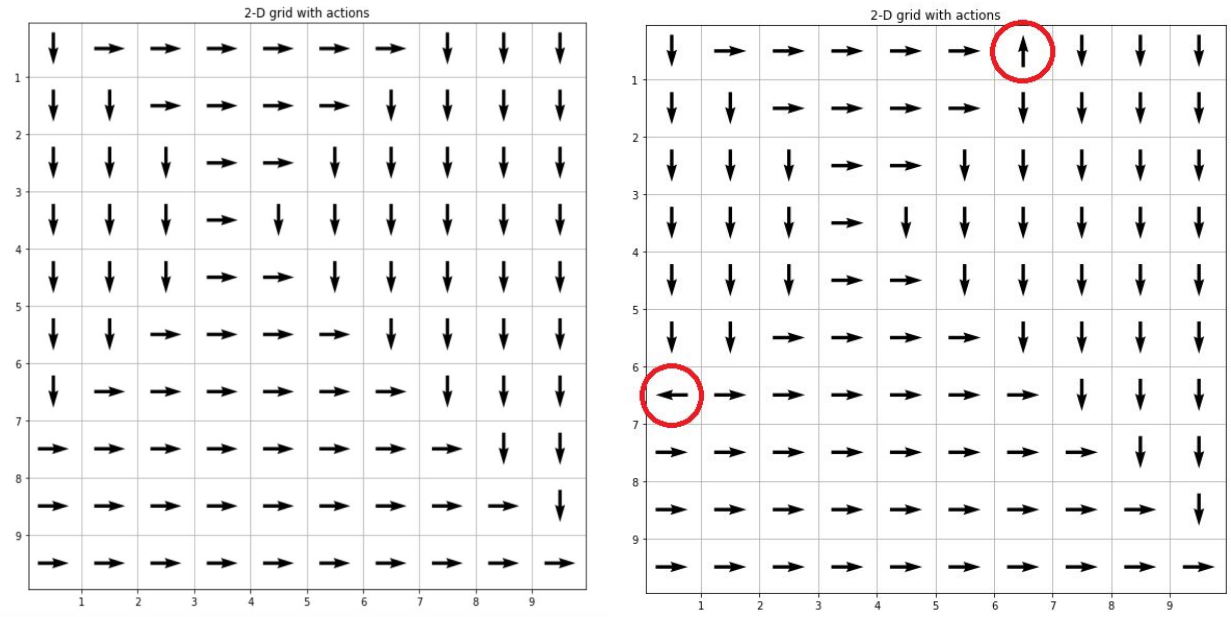

Figure 14. Difference Between the Optimal Policy Using the Extracted Reward Function 1 (Right) and the Optimal Policy Using the Ground Truth Reward Function 1 (Left)

From Fig.14, we could find that most of the arrows shown in these 2 pictures are almost the same, keeping moving right and moving down. But some of the arrows on the edge are pointing out of the edge in Fig.13, which is in the opposite direction compared to that in Fig.5.

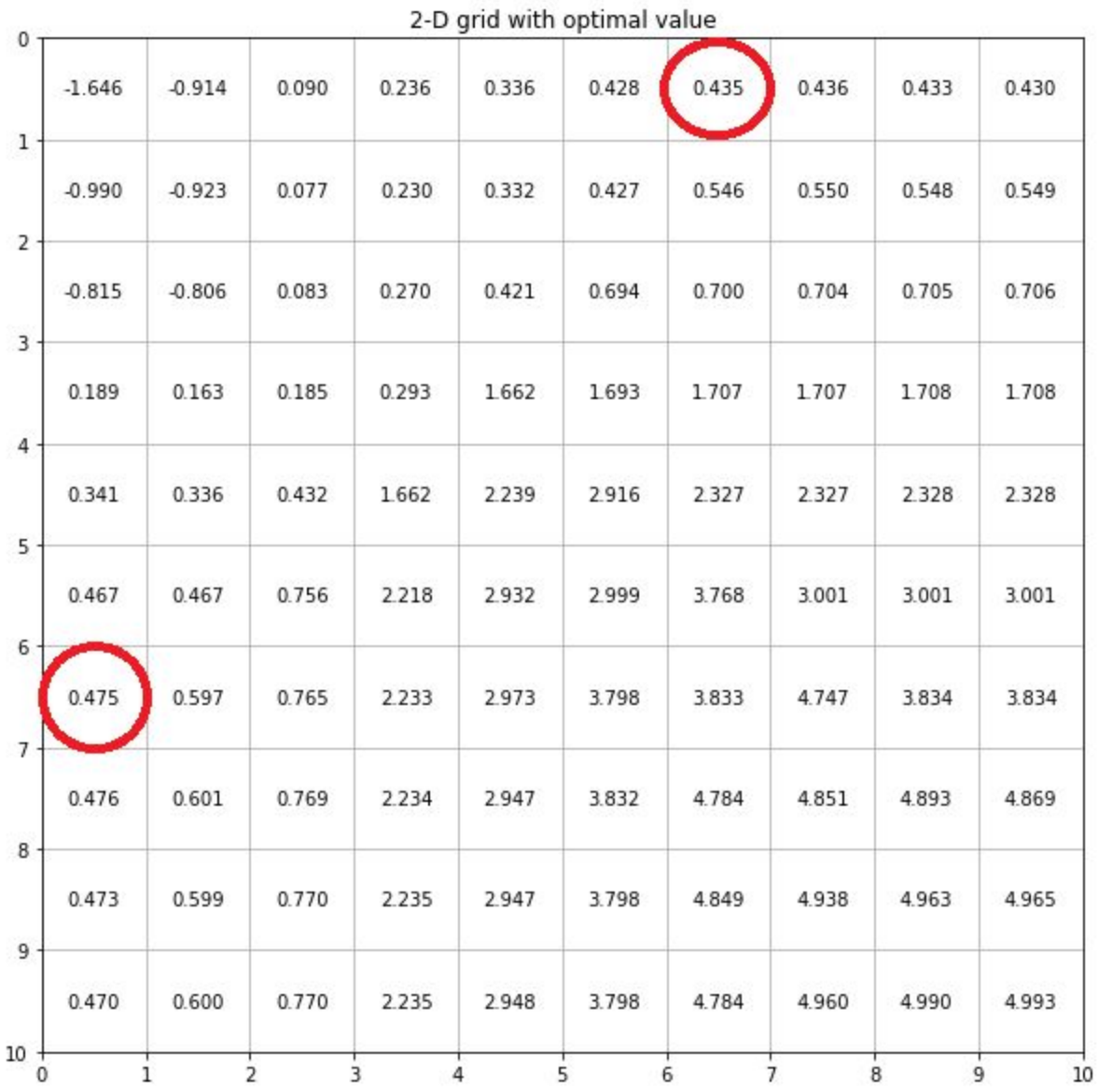

Figure 15. Optimal Values of States Using Extracted Reward Function 1

By checking the optimal value of states using extracted reward function 1 (Fig.15), we found that because the difference between the state values of the red circled states (which are the places where the prediction go wrong) and that of their neighbours is so small, thus, even some small change in reward function can result in different output. And from previous experiment, we can see that the extracted reward function solved by LP problem can vary a little, which may result in the different prediction policy.

Question 18.

In the question, we swept parameter λ from 0 to 500 as we did in question 11. But here we used the optimal policy of agent found in question 9 to fill the OE(s) values. The used the same algorithm as in question 11 to get the accuracy of IRL and get the plot of λ against accuracy here. The plot is as below:

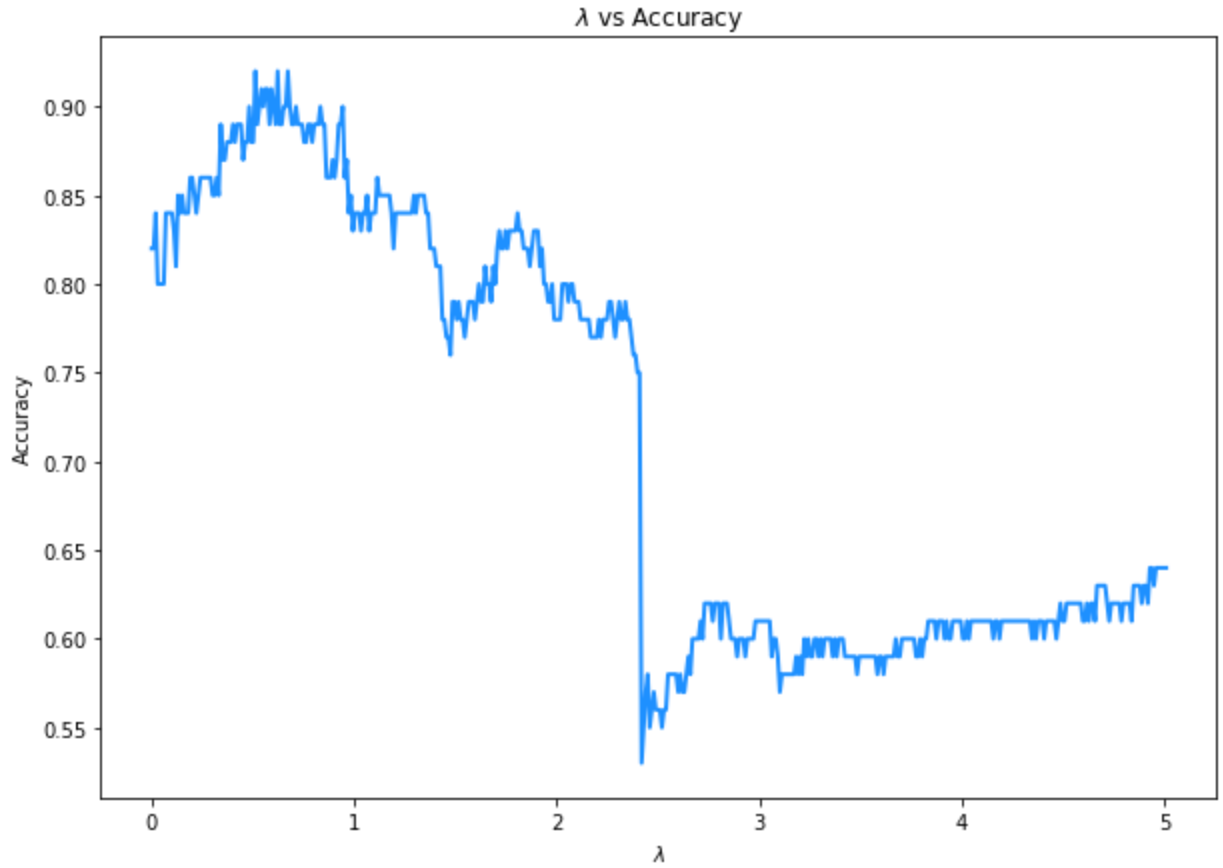

Figure 16. The Action Accuracy Against λ Using OE(s) Values Found in Question 9

Question 19.

Here we used the plot of Fig.16 to find out the λ_{max} which the accuracy attains the maximum. And the λ_{max} is:

$$\lambda_{max} = 0.5120440881763526$$

Question 20.

Using the λ_{max} we found in question 19, we generated heat maps of the ground truth reward and the extracted reward. The plots are as below:

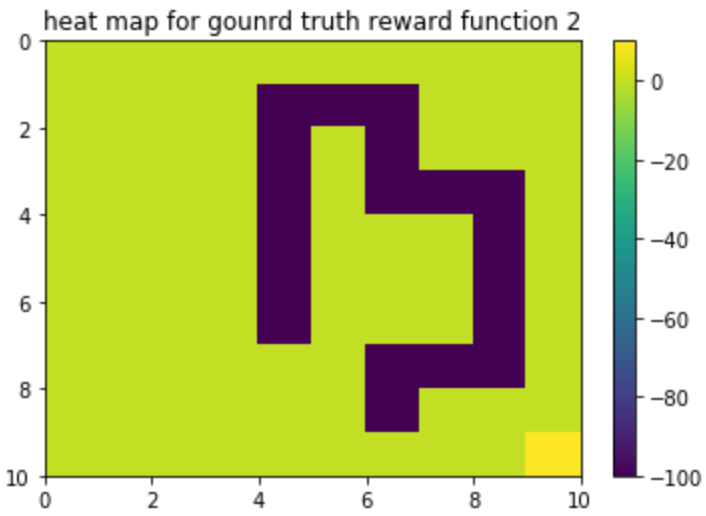

Figure 17. Heat Map of Ground Truth Reward Using Reward Function 2

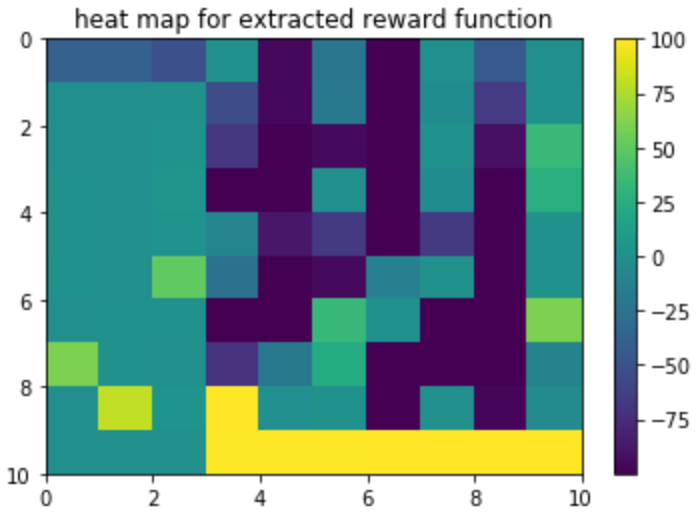

Figure 18. Heat Map of Extracted Reward Function 2

Question 21.

We used the extracted reward function computed in question 20 to compute the optimal values of the states in a 2-D grid. The heat map of the optimal values of the states is as below:

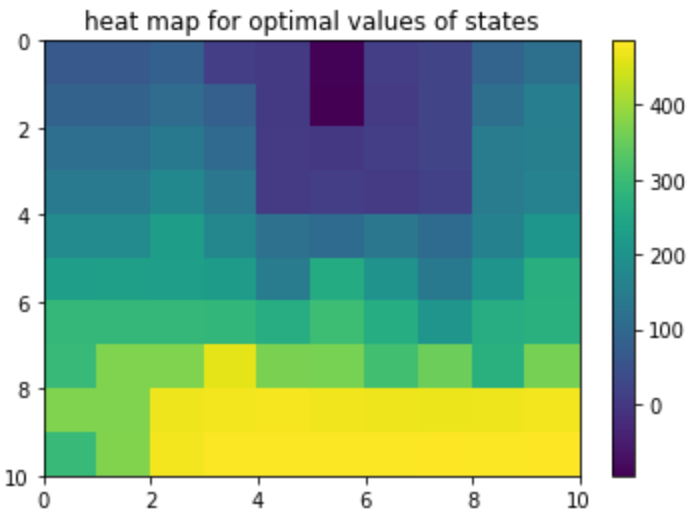

Figure 19. Heat Map of Optimal Values of States Using the Extracted Reward Function 2

Question 22.

Comparing the heat maps of optimal values of states in question 21 and question 7, we could find that both of them get larger optimal values of states at the right bottom corner and smaller optimal values of states on the top near the middle. But the differences are that Fig.19 has more larger optimal values of states at the bottom extending to the left than that in Fig.7, and that most of the states show brighter color no matter on the top or at the bottom than that in Fig.7.

Question 23.

Here we used the extracted reward function from question 20 to compute the optimal policy of agent. The plot of arrows showing the optimal policy of agent is as below:

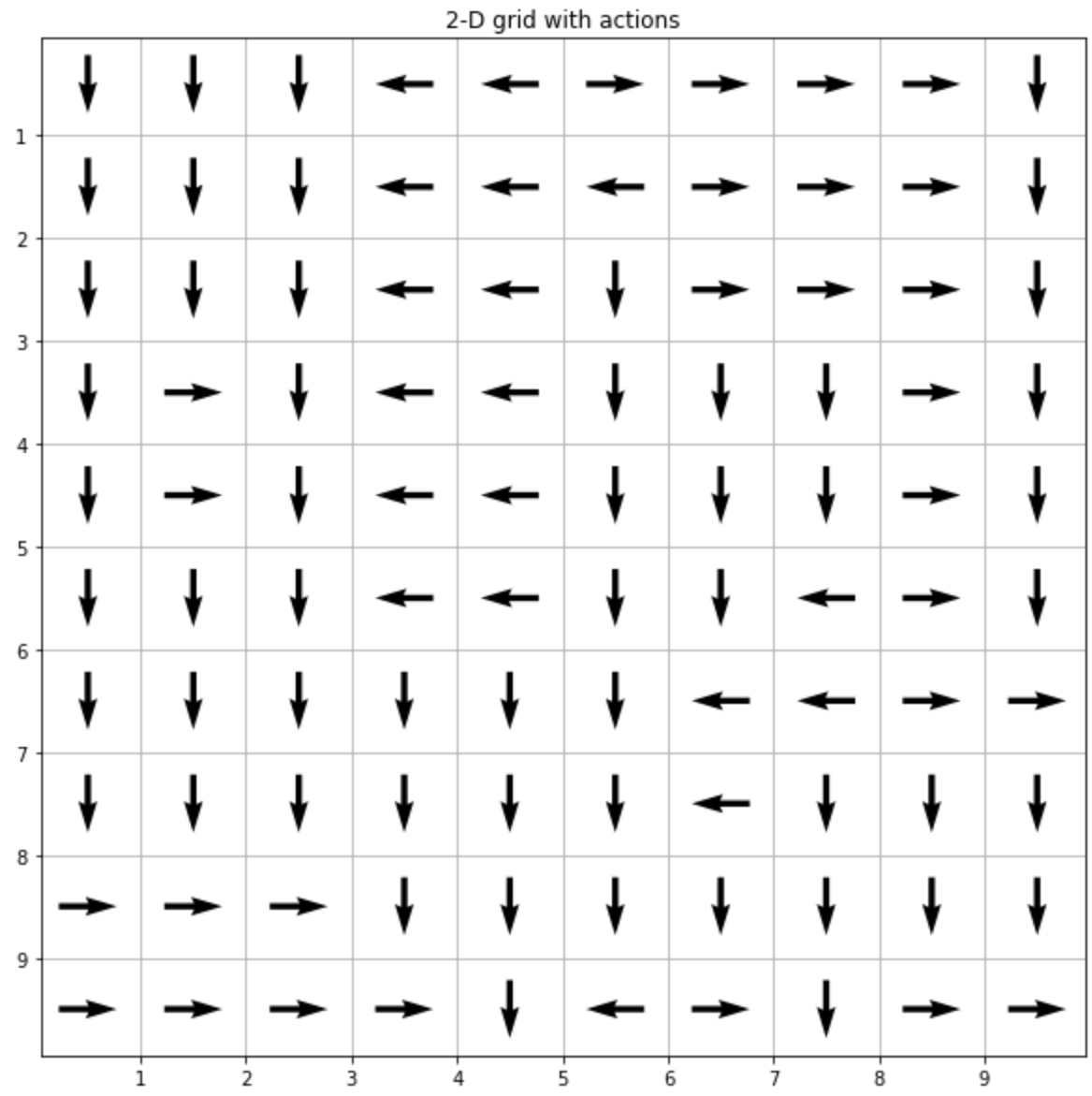

Figure 20. Arrows of Optimal Policy of Agent Using IRL

Question 24.

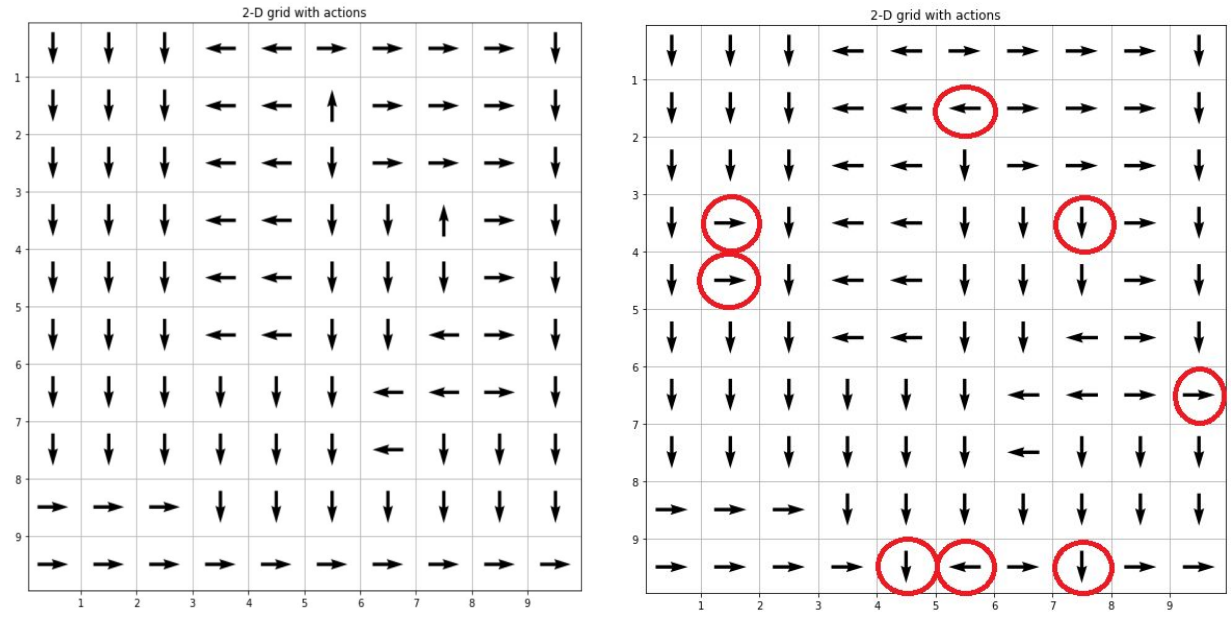

Figure 21. Difference Between the Optimal Policy Using the Extracted Reward Function 2 (Right) and the Optimal Policy Using the Ground Truth Reward Function 2 (Left)

Comparing both arrow plots of optimal policy of agent in question 23 and question 9, we could find that most of the arrows are almost the same in both pictures, pointing to the left, down and right. But there is no arrows pointing up in Fig.20 while there are few up-pointers in Fig.8. For the last row and the last column, some pointers turn to point out of the edge in Fig.20 while all the arrows in Fig.8 tend to point along the edge until the last corner state.

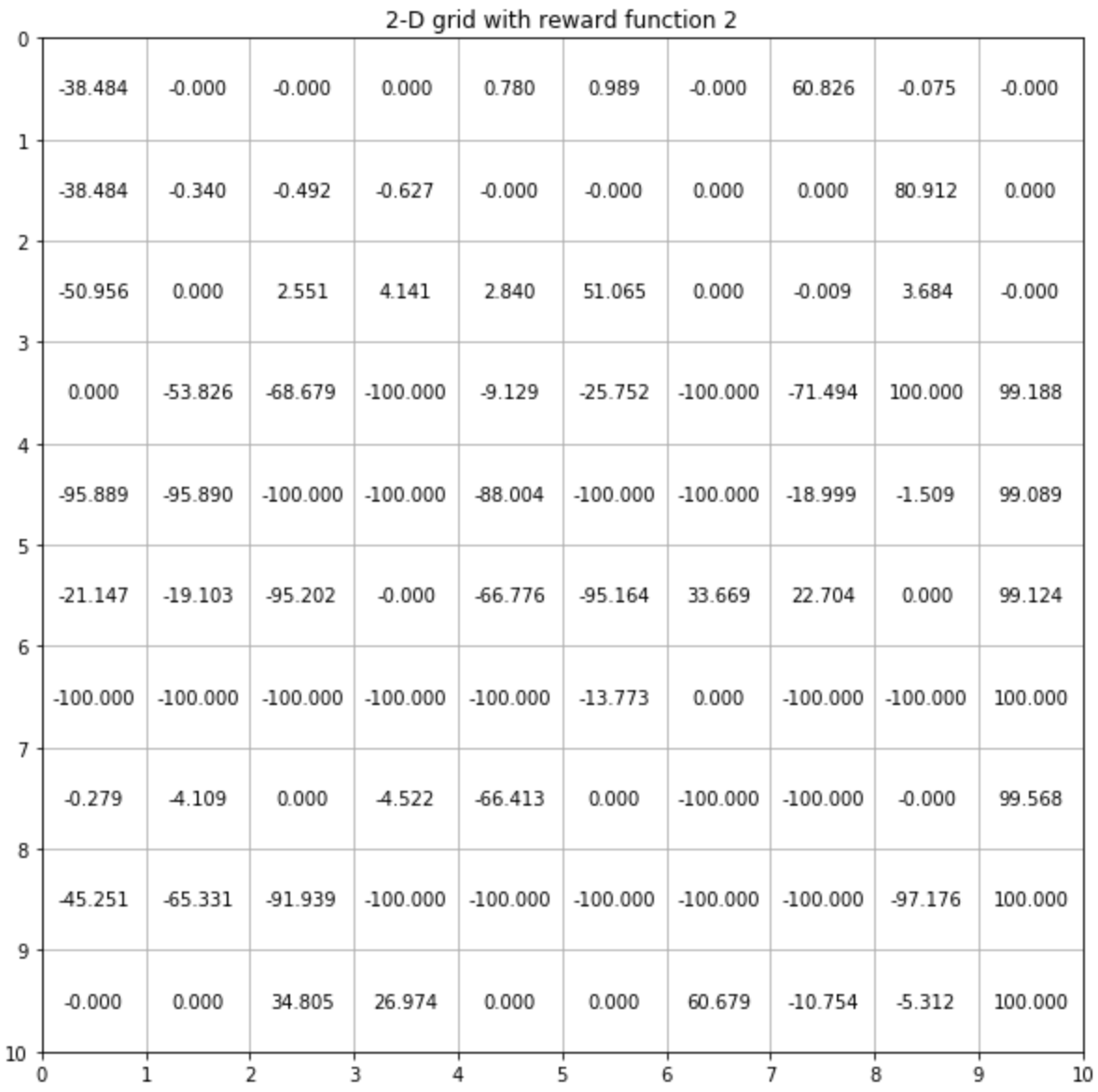

Figure 22. Extracted Reward Function 2

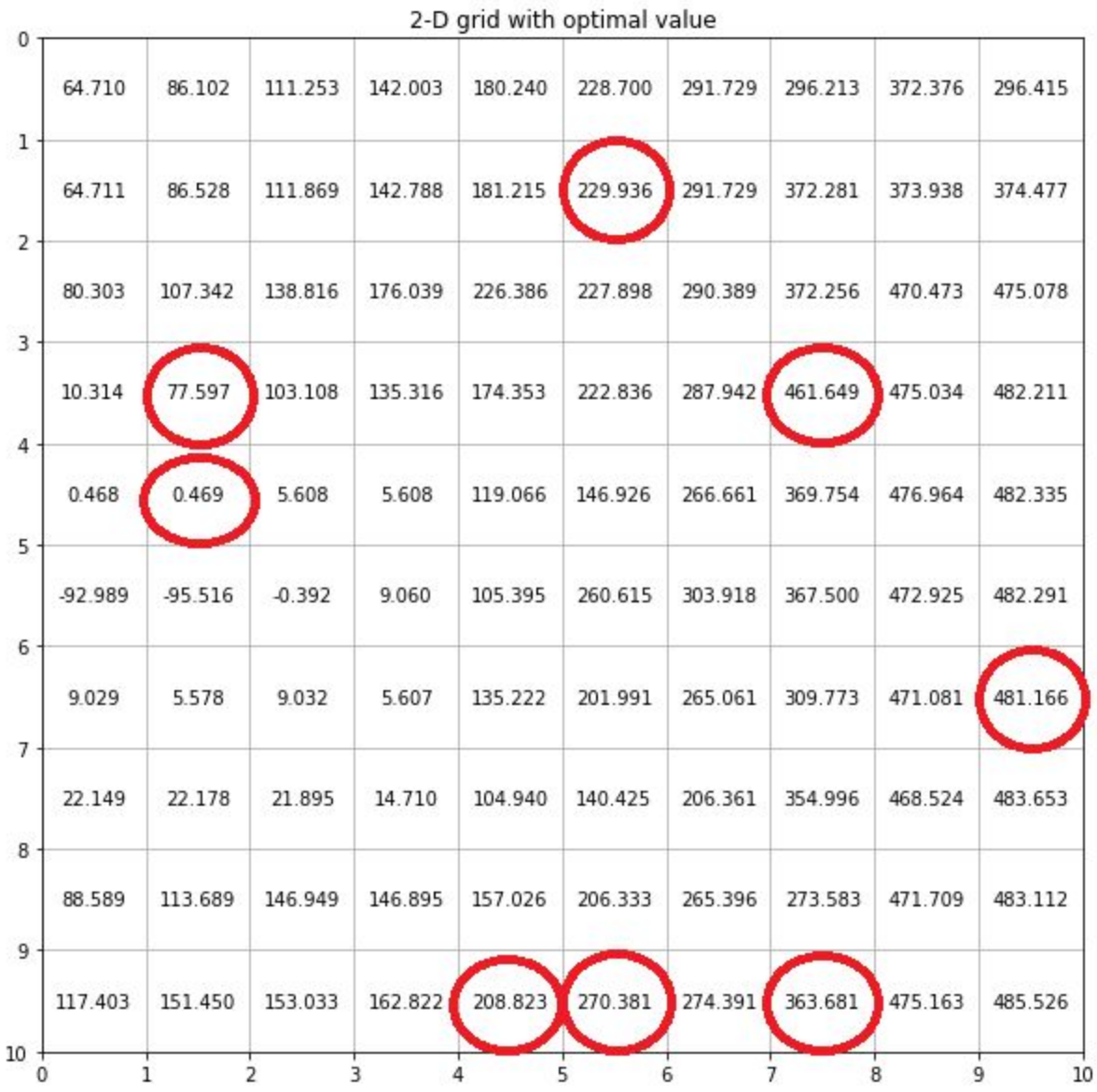

Figure 23. Optimal Values of States Using Extracted Reward Function 2

By carefully checking the extracted reward function and optimal values of states generated by it, we found that similar to Question 17, the extracted reward function and the optimal values of states are not smooth and contains noise, in this case, it may result in prediction error.

Question 25.

Through Question 17 and Question 24, we can know that the error policy prediction can be resulted from the noise, from the extracted function and the optimal state values. Thus, several solutions can be applied to make the algorithm better:

1. Make ε smaller so that the optimal state value converge to a much more stable states, thus make the performance better.
2. Encoding some prior to the LP problem, such as using some regularization to regularize the derivative of the reward.

First, we tried to change the ε from 0.01 to 1e-8. Then using the same hyperparameter $\lambda_{max} = 0.5522044088176352$, we can achieve an accuracy of 100%. The output policy can be shown in Fig.24 as follows:

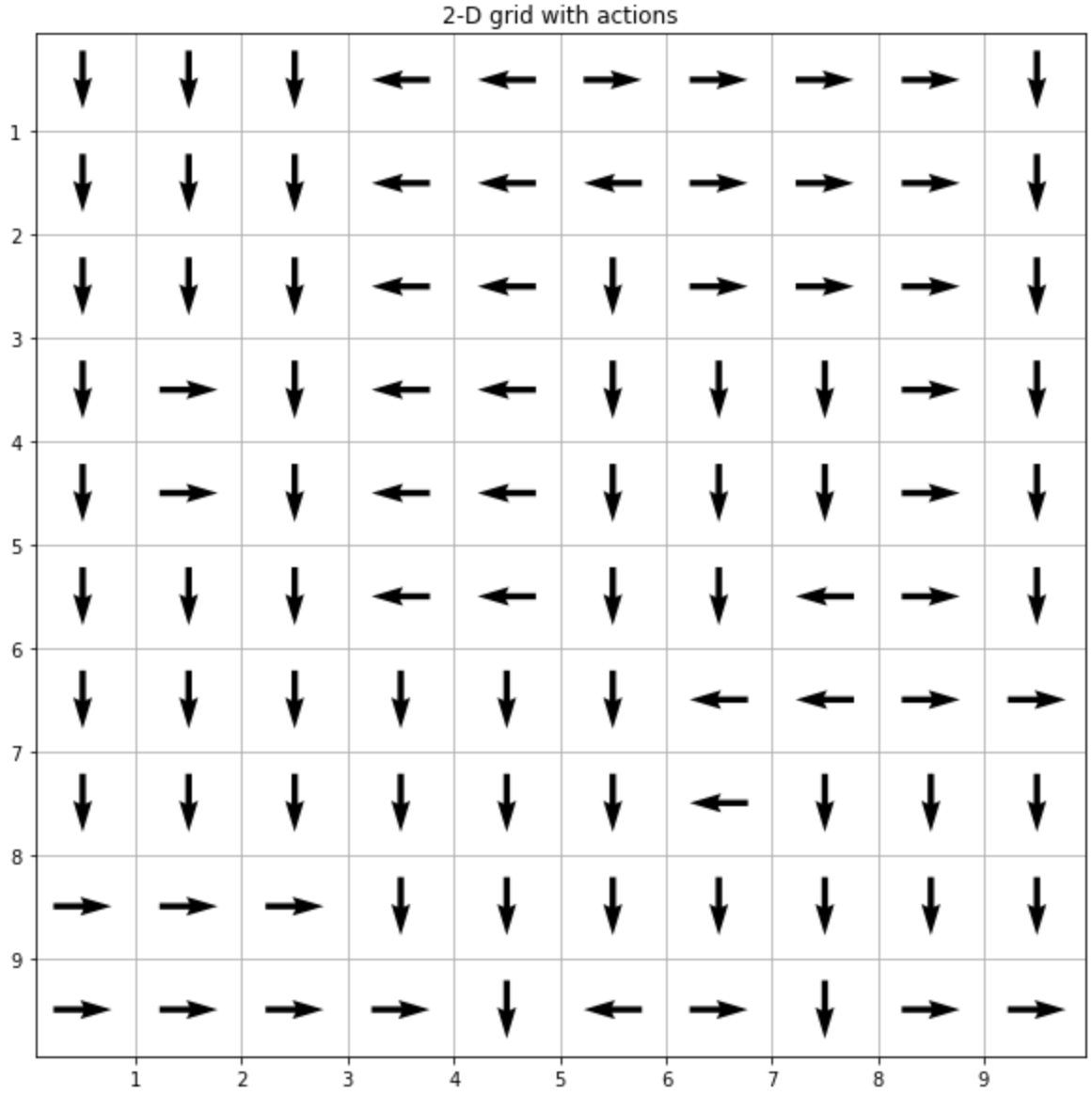

Figure 20. Arrows of Optimal Policy of Agent Using IRL (With ε=1e-8)

For the second option to make the performance better, we can add the following regularization to the LP problem:

$$|R(s^i) - R(s^j)| < p \ \forall \ s^i \in S \ and \ S^j \in Neighbour(S^i)$$

For this regularization term, p is a hyperparameter, and this term basically regularize the derivative of the reward in every state, thus make the reward function smoother and reduce some noise. However, due to time limitations, we do not have time to implement this regularization.