ECE232E Project3

Hengyu Lou (005035476)
Yanzhe Xu (404946757)
Yucong Wang (305036163)
Yutong Lu (005027184)

May 22, 2018

1 Introduction

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 re- ward 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.

2 Reinforcement learning (RL)

The two main objects in Reinforcement learning are:
– Agent
– Environment
In this project, we will learn the optimal policy of a single agent navigating in a 2-D environment. In this project, we assume that the environment of the agent is modeled by a Markov Decision Process (MDP). In a MDP, agents occupy a state of the environment and perform actions to change the state they are in. After taking an action, they are given some representation of the new state and some reward value associated with the new state.
An MDP formally is a tuple $(S, A, P^a_{ss'}, R^a_{ss'}, \gamma)$ where:

- S is a set of states
- A is a set of actions
- $P^a_{ss'}$ is a set of transition probabilities.
  $-P^a_{ss'} = P(s_{t+1} = s'|s_t = s, a_t = a)$
- $R^a_{ss'}$ is a set of next state reward functions.
  $-R^a_{ss'} = E(r_{t+1}|s_t = s, a_t = a, s_{t+1} = s')$
- γ is a discount factor.

Question 1

For visualization purpose, we generate heat maps of Reward function 1 and Reward function 2. Here we use the pcolor function in the matplotlib package.

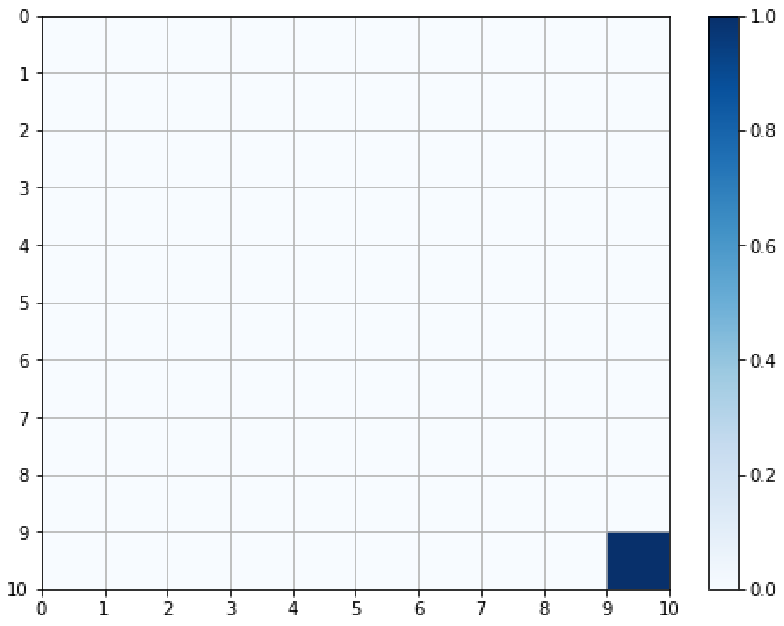


Figure 1: Reward 1 heatmap

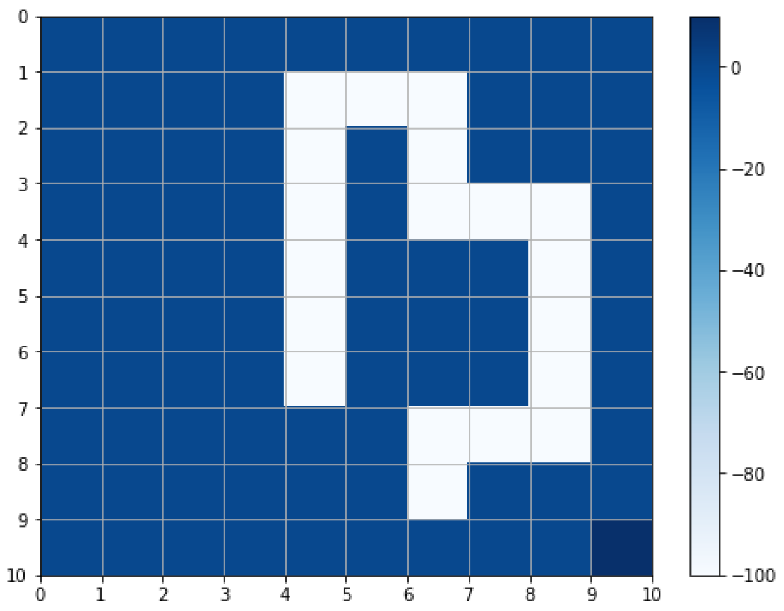


Figure 2: Reward 2 heatmap

Here we could see that the most critical value in the map is at the right down corner of each reward function, and reward 1 has no cost on the way to the best value, but in reward 2 we have to avoid some bad values to make a proper way.

3 Optimal policy learning using RL algorithms

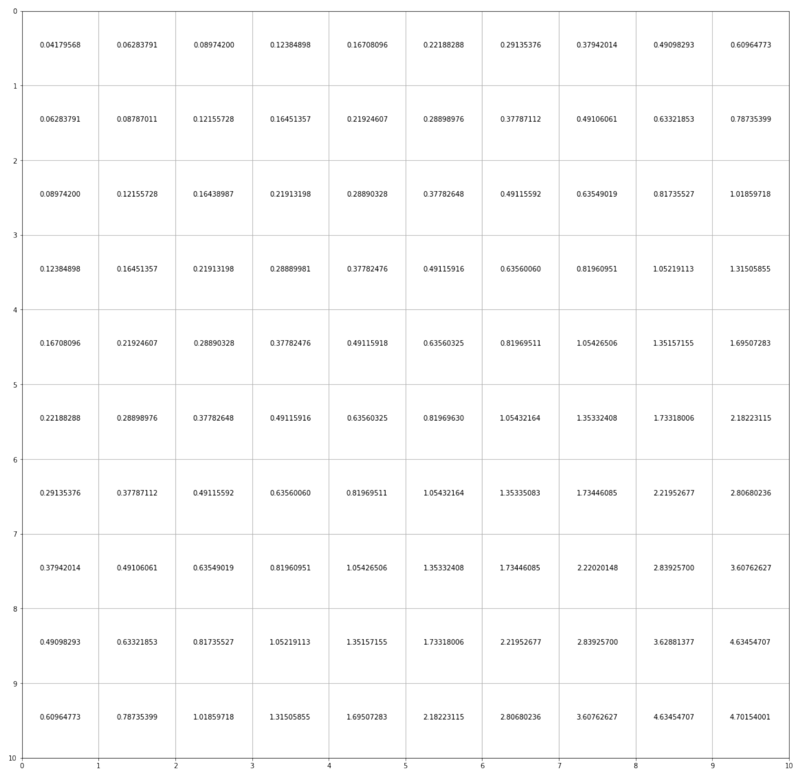
In this part of the project, we will use reinforcement learning (RL) algorithm to find the optimal policy. The main steps in RL algorithm are:
–Find optimal state-value or action-value
–Use the optimal state-value or action-value to determine the deterministic optimal policy

Question 2

We will just reproduce the algorithm as the pseudocode described for the ease of implementation. We created the environment of the agent using the in- formation provided in section 2. We create the MDP by setting up the state-space, action set, transition probabilities, discount factor, and reward function. For creating the environment, the parameters setting are as follows.

- Number of states = 100 (state space is a 10 by 10 square grid as displayed)
- Number of actions = 4
- w = 0.1
- Discount factor = 0.8
- use reward function 1

After we have created the environment, then we 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, we implement the Initialization and Estimation steps of the Value Iteration algorithm. For the estimation step, use $\sigma = 0.01$. We generate a figure of optimal value of each state.



	0	1	2	3	4	5	6	7	8	9
0	0.04179568	0.06283791	0.08974200	0.12384898	0.16708096	0.22188288	0.29135376	0.37942014	0.49098293	0.60964773
1	0.06283791	0.08787011	0.12155728	0.16451357	0.21924607	0.28898976	0.37787112	0.49106061	0.63321853	0.78735399
2	0.08974200	0.12155728	0.16438987	0.21913198	0.28890328	0.37782648	0.49115592	0.63549019	0.81735527	1.01859718
3	0.12384898	0.16451357	0.21913198	0.28889981	0.37782476	0.49115916	0.63560060	0.81960951	1.05219113	1.31505855
4	0.16708096	0.21924607	0.28890328	0.37782476	0.49115918	0.63560325	0.81969511	1.05426506	1.35157155	1.69507283
5	0.22188288	0.28898976	0.37782648	0.49115916	0.63560325	0.81969630	1.05432164	1.35332408	1.73318006	2.18223115
6	0.29135376	0.37787112	0.49115592	0.63560060	0.81969511	1.05432164	1.35335083	1.73446085	2.21952677	2.80680236
7	0.37942014	0.49106061	0.63549019	0.81960951	1.05426506	1.35332408	1.73446085	2.22020148	2.83925700	3.60762627
8	0.49098293	0.63321853	0.81735527	1.05219113	1.35157155	1.73318006	2.21952677	2.83925700	3.62881377	4.63454707
9	0.60964773	0.78735399	1.01859718	1.31505855	1.69507283	2.18223115	2.80680236	3.60762627	4.63454707	4.70154001

Figure 3: Optimal value for each state – reward 1

Question 3

For this question, we generate a heat map of the optimal state values across the 2-D grid. For generating the heat map, we use the same function provided in matplotlib.

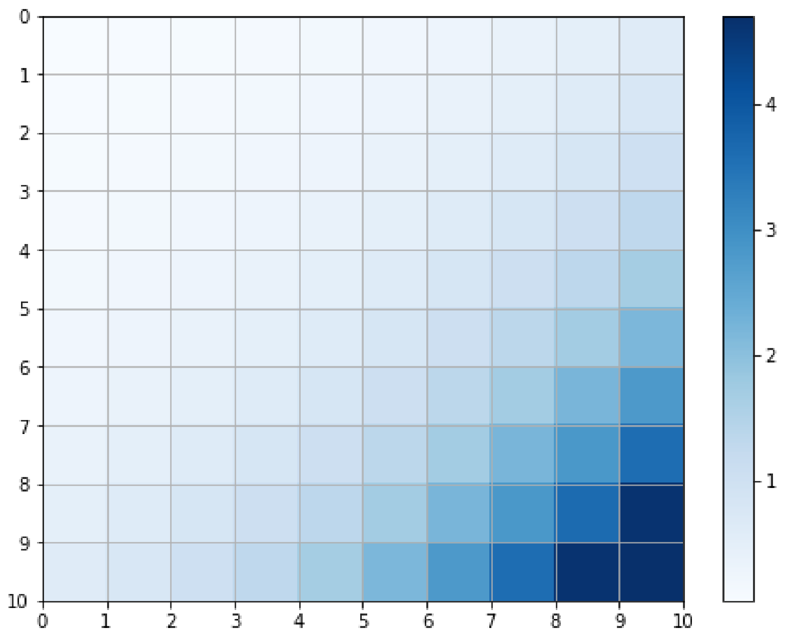


Figure 4: Heatmap of optimal state value – reward 1

Question 4

Explain the distribution of the optimal state values across the 2-D grid. From this heatmap, we can see the optimal state value is get larger when it becomes more closer to the best value at the lower right corner, which is reasonable since the optimal state is to find a path to get the best value at last.

Question 5

Here we implement the computation step of the value iteration algorithm to compute the optimal policy of the agent navigating the 2-D state-space. We generate a figure for the optimal action at that state. And the optimal actions are displayed using arrows.

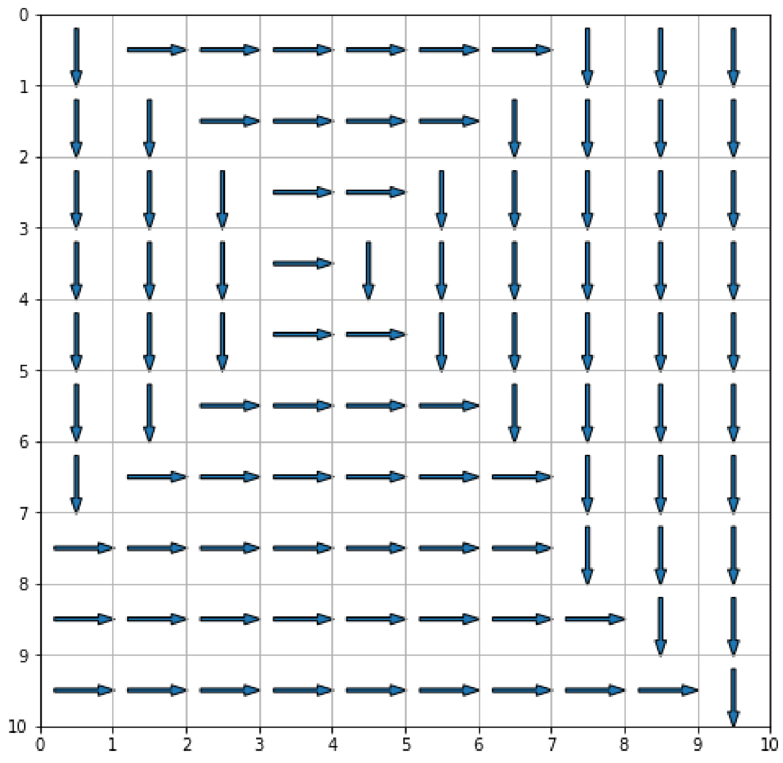
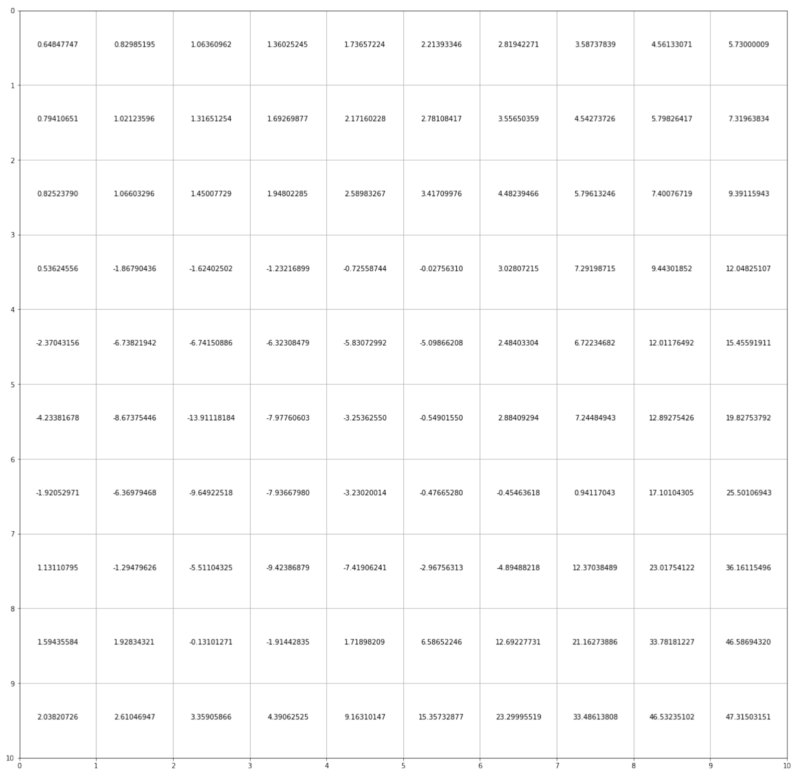


Figure 5: Optimal actions – reward 1

We could observe that the optimal policy of the agent match our intuition, because the agent manage its way to get close to the best value at lower right corner. From the figure we can know that almost half the actions are going down and half are going right, which is reasonable since the reward function is symmetrical. And the agent compute the optimal action to take at each state by observing the optimal values of it's neighboring states. Two or three arrows around (row 4, column 3) and (row 1, column 5) are pointing to different directions in the two figures. But the two are really similar.

Question 6

In this problem, we just replace reward function 1 with reward function 2 and keep the same environment as question 2. And then repeat steps in question 2. The result is as following.



0.64847747	0.82985195	1.06360962	1.36025245	1.73657224	2.21393346	2.81942271	3.58737839	4.56133071	5.73000009
0.79410651	1.02123596	1.31651254	1.69269877	2.17160228	2.78108417	3.55650359	4.54273726	5.79826417	7.31963834
0.82523790	1.06603296	1.45007729	1.94802285	2.58983267	3.41709976	4.48239466	5.79613246	7.40076719	9.39115943
0.53624556	-1.86790436	-1.62402502	-1.23216899	-0.72558744	-0.02756310	3.02807215	7.29198715	9.44301852	12.04825107
-2.37043156	-6.73821942	-6.74150886	-6.32308479	-5.83072992	-5.09866208	2.48403304	6.72234682	12.01176492	15.45591911
-4.23381678	-8.67375446	-13.91118184	-7.97760603	-3.25362550	-0.54901550	2.88409294	7.24484943	12.89275426	19.82753792
-1.92052971	-6.36979468	-9.64922518	-7.93667980	-3.23020014	-0.47665280	-0.45463618	0.94117043	17.10104305	25.50106943
1.13110795	-1.29479626	-5.51104325	-9.42386879	-7.41906241	-2.96756313	-4.89488218	12.37038489	23.01754122	36.16115496
1.59435584	1.92834321	-0.13101271	-1.91442835	1.71898209	6.58652246	12.69227731	21.16273886	33.78181227	46.58694320
2.03820726	2.61046947	3.35905866	4.39062525	9.16310147	15.35732877	23.29995519	33.48613808	46.53235102	47.31503151

Figure 6: Optimal value for each state – reward 2

Question 7

For this question, we generate a heat map of the optimal state values across the 2-D grid. For generating the heat map, we use the same function provided in matplotlib.

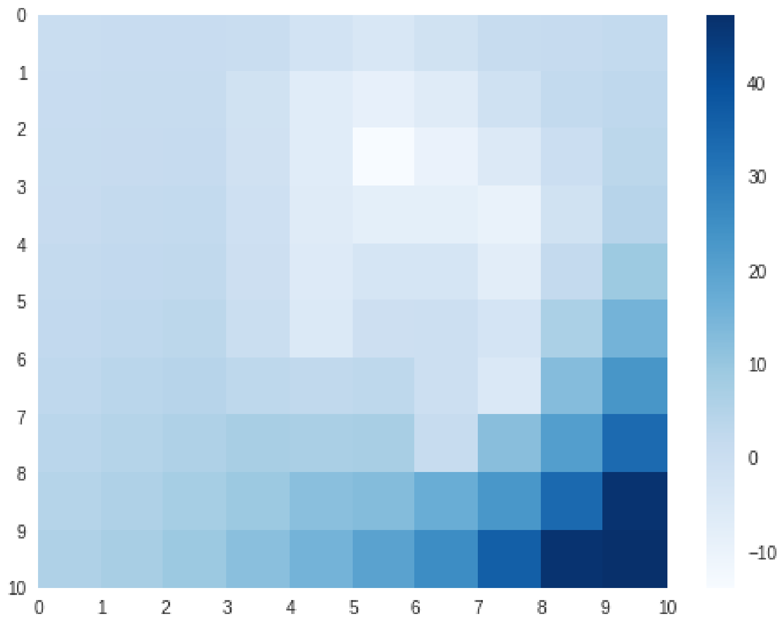


Figure 7: Heatmap of optimal state value – reward 2

Question 8

From the heatmap generated by question 7, we know states close to lower right corner is larger than other states which is reasonable because values of reward function near the corner are 0. It is obvious that middle area has lowest value in the map. This is reasonable since values of reward function in this area are -100 and much lower than other states.

Question 9

After we apply value iteration steps with reward function 2, we can find out optimal actions for each state. We generate the figure by using optimal actions set. Then the optimal actions are displayed using arrows.

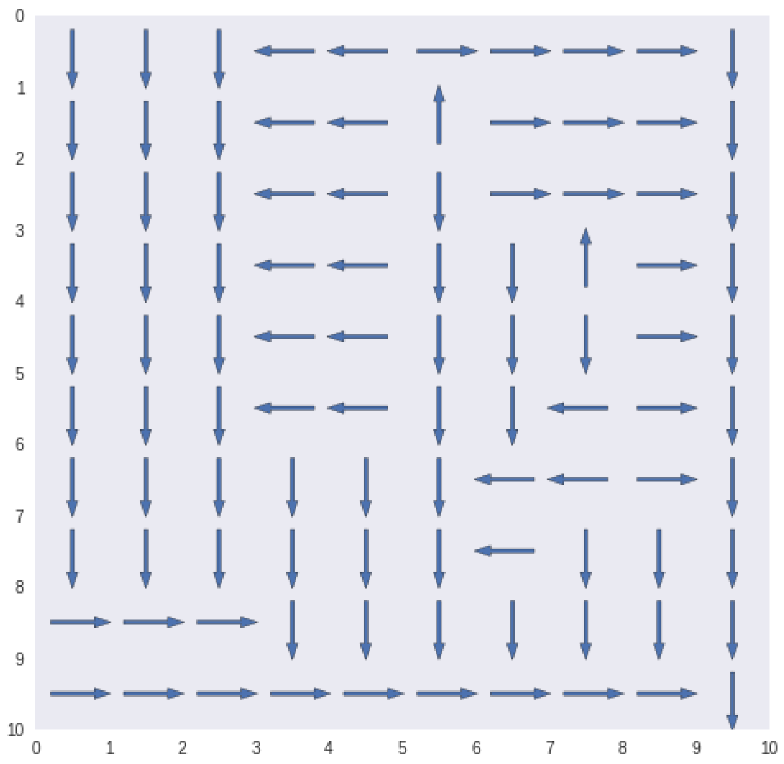


Figure 8: Optimal actions – reward 2

We could observe that the optimal policy of the agent match our intuition, because the agent manage its way to get close to the best value at lower right corner. From the figure we know the agent avoid taking actions to states which have -100 reward. In the area which has pretty bad reward, the actions taken by agent is to get out of this area as quickly as possible. Thus, this scenario matches our intuition.

4 Inverse Reinforcement learning (IRL)

In this part of the project, we will use IRL algorithm to extract the reward function. We will use the optimal policy computed in the previous section as the expert behavior and use the algorithm to extract the reward function of the expert. Then, we will use the extracted reward function to compute the optimal policy of the agent. We will compare the optimal policy of the agent to the optimal policy of the expert and use some similarity metric between the two to measure the performance of the IRL algorithm.

Question 10

In this problem, we can put all of variables into one side and other constants into the other side, then we can have the following form of this linear problem.

$$max \begin{bmatrix} 0 & 1^T & -\lambda 1^T \end{bmatrix} \begin{bmatrix} R \\ t \\ u \end{bmatrix}$$

$$st : \begin{bmatrix} -(P_{a1} - P_a)(I - \gamma P_{a1})^{-1} & I & 0 \\ -(P_{a1} - P_a)(I - \gamma P_{a1})^{-1} & 0 & 0 \\ -I & 0 & -I \\ I & 0 & -I \\ I & 0 & 0 \\ -I & 0 & 0 \end{bmatrix} \begin{bmatrix} R \\ t \\ u \end{bmatrix} \leq \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ R_{max}1 \\ R_{max}1 \end{bmatrix}$$

From this new linear problem, we can easily conclude that:

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

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

$$D = \begin{bmatrix} -(P_{a1} - P_a)(I - \gamma P_{a1})^{-1} & I & 0 \\ -(P_{a1} - P_a)(I - \gamma P_{a1})^{-1} & 0 & 0 \\ -I & 0 & -I \\ I & 0 & -I \\ I & 0 & 0 \\ -I & 0 & 0 \end{bmatrix}$$

Question 11

It can be seen for reward 1 function, as λ gradually increase, the accuracy increases.

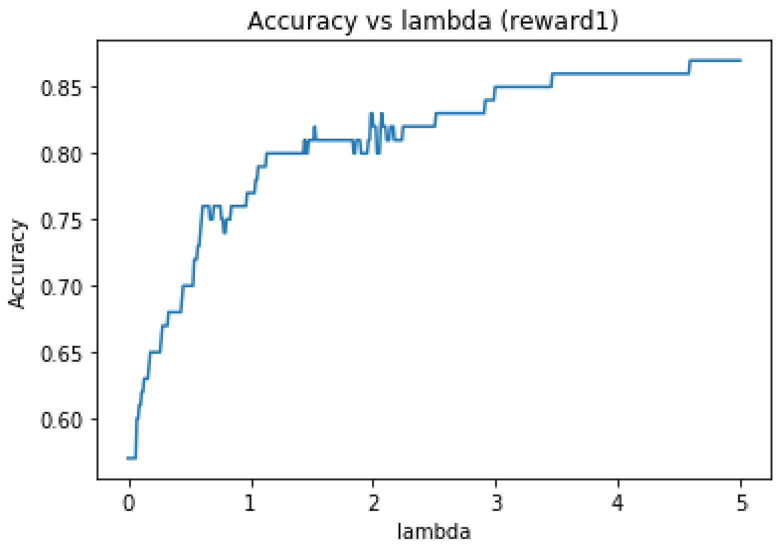


Figure 9: Accuracy vs λ (Reward 1)

Question 12

The maximum accuracy is 0.87 which can be achieved at $\lambda = 4.589178356713426$

Question 13

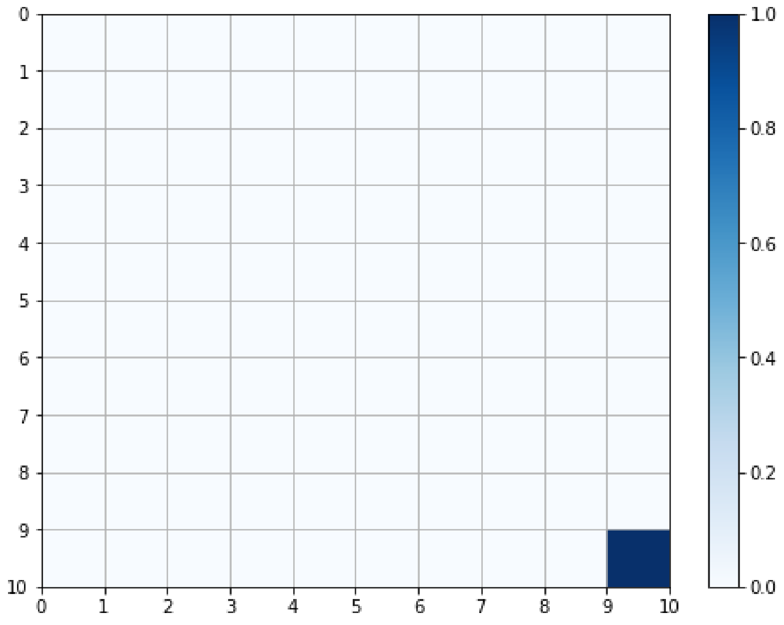


Figure 10: Heat Map of Reward 1)

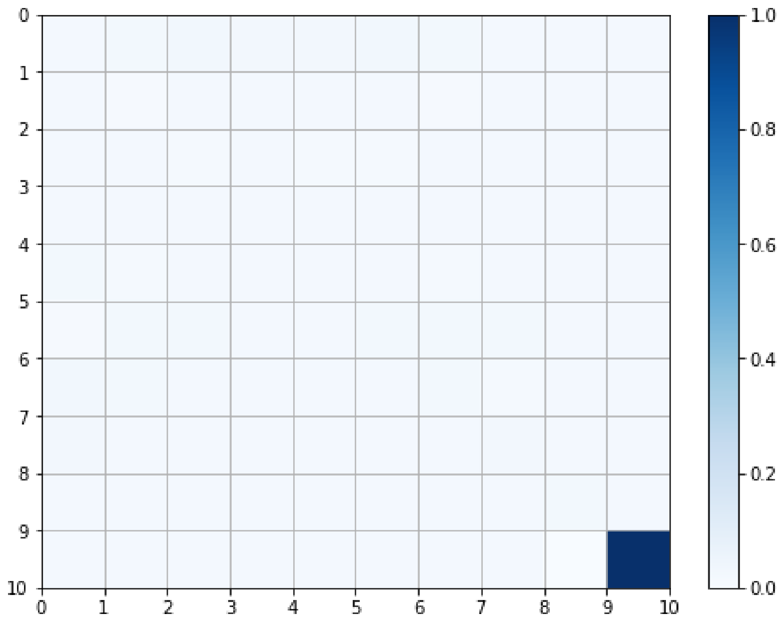


Figure 11: Heat Map of Extracted Reward 1)

The heat maps of the ground truth reward and the extracted reward are similar. Both of them have a very large optimal value at the right-bottom corner.

Question 14

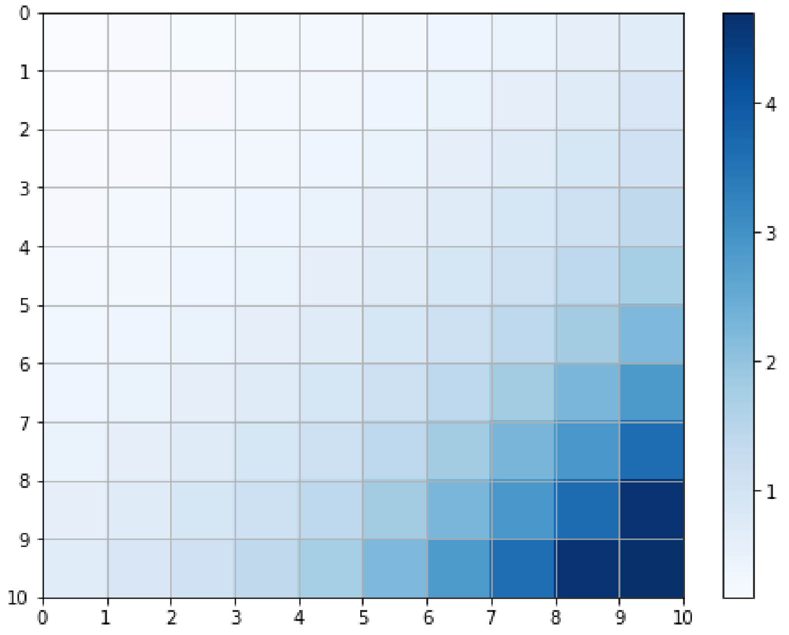


Figure 12: Optimal Values of States (Reward 1)

Question 15

The two both achieve the highest value at the right-bottom corner. The values in both heat maps gradually get larger as the indices of rows and columns become larger. But the heat map in Question 14 drops faster as the indices of the row and column become smaller.

Question 16

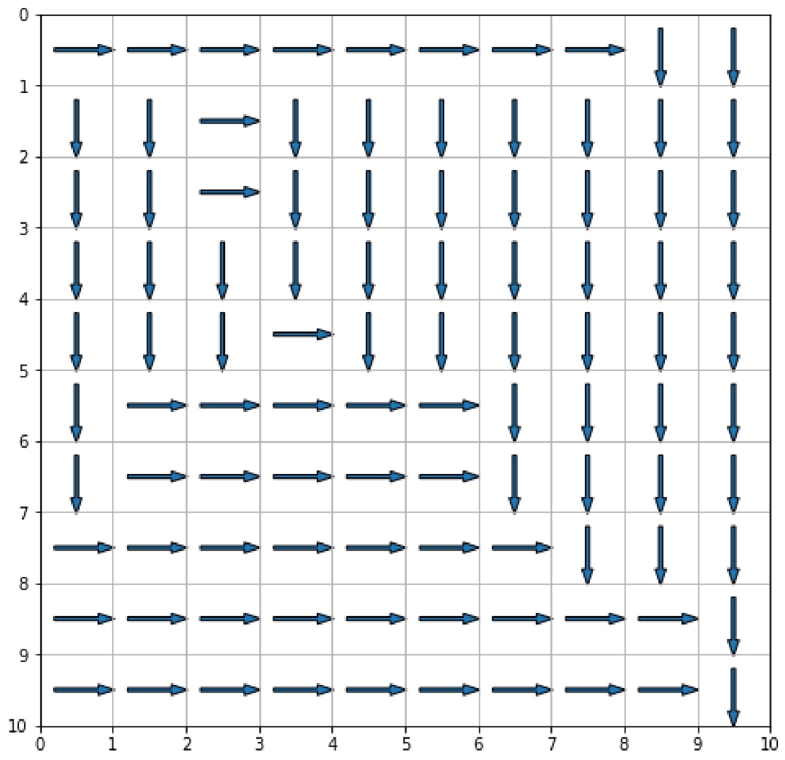


Figure 13: Optimal Policy of Agent (Reward 1)

Question 17

These two figures are similar. The arrows in the left part and right part of both figures are almost pointing down. And the arrows in the top part and bottom part of both figures are almost pointing to the right.

Question 18

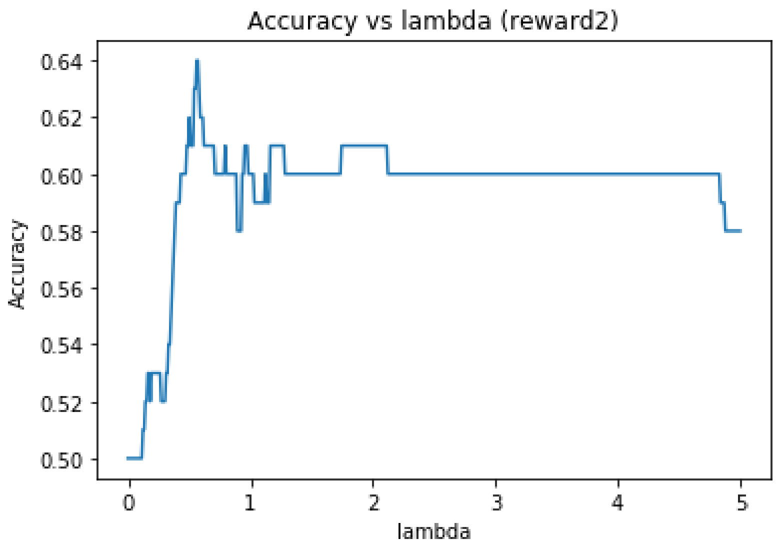


Figure 14: Accuracy vs λ (Reward 2)

Question 19

The maximum accuracy is 0.64 which can be achieved at $\lambda = 0.561122244488978$

Question 20

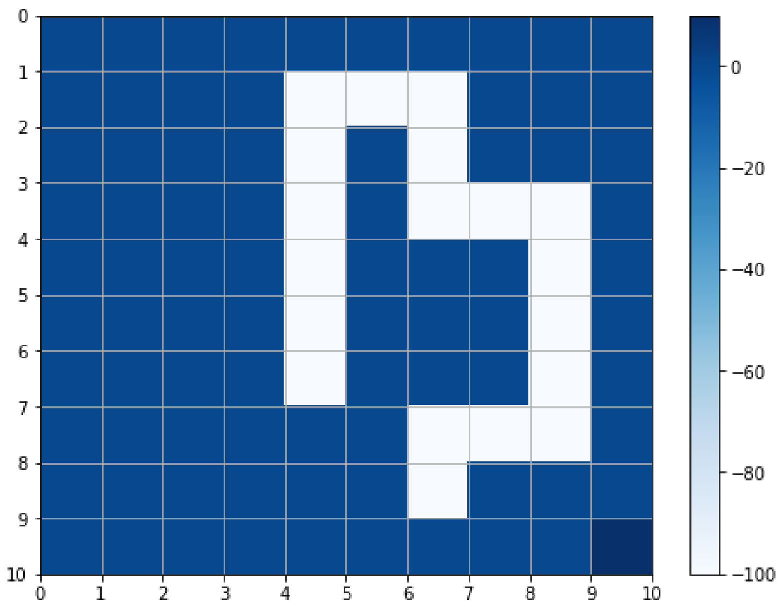


Figure 15: Heat Map of Reward 2)

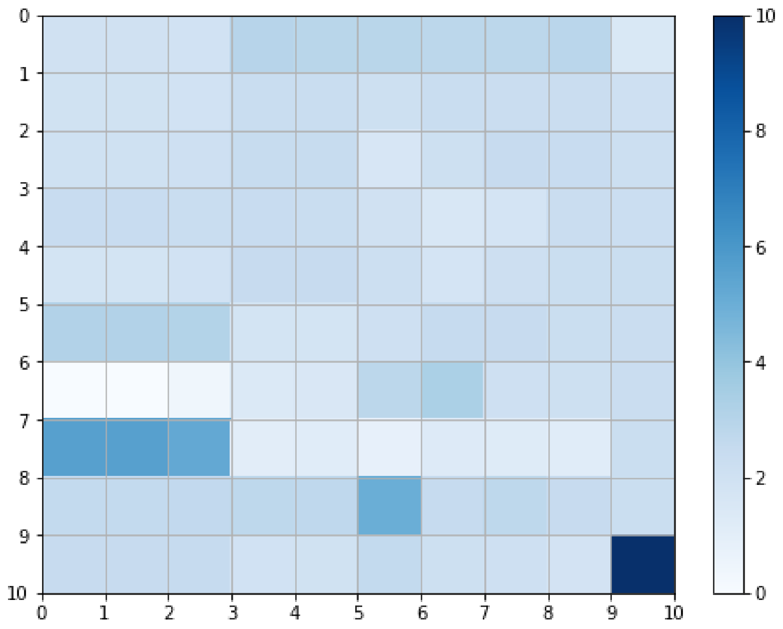


Figure 16: Heat Map of Extracted Reward 2)

Question 21

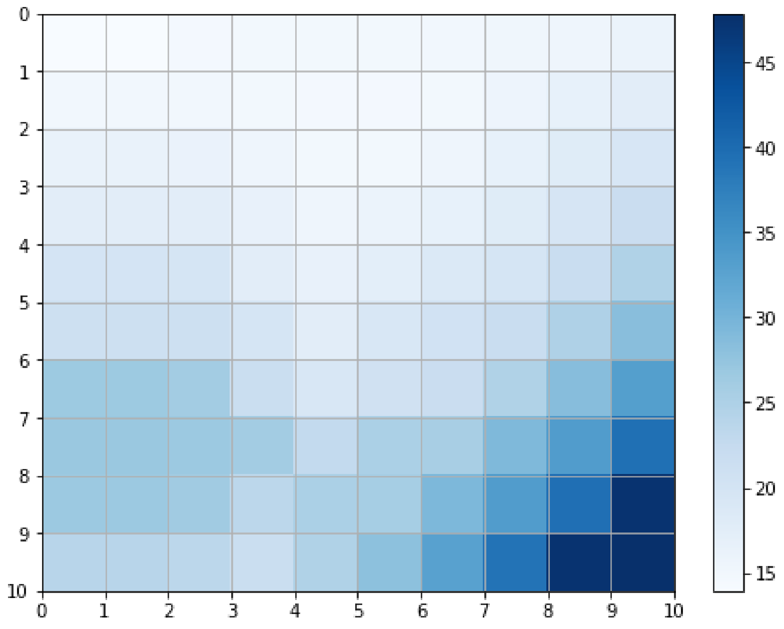


Figure 17: Optimal Values of States (Reward 2)

Question 22

The two figure both achieve highest values at the right-bottom corner. However, the figure of Question 7 has some shallow part around (row 2, row 6) which is because of the reward function. But the figure generated by us in Question 21 is not as shallow as that in Question 7.

Question 23

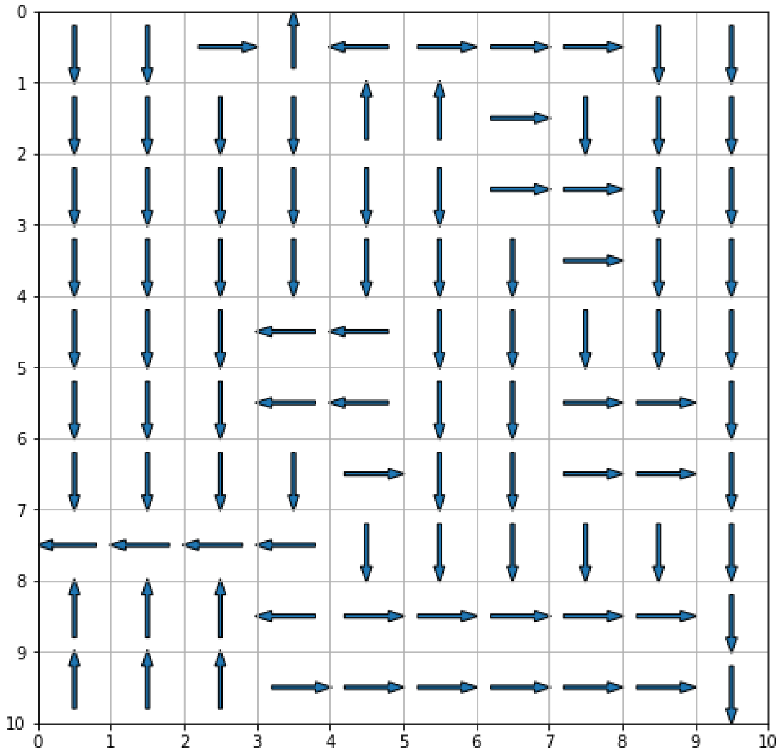


Figure 18: Optimal Policy of Agent (Reward 2)

Question 24

The optimal policy map looks similar to the policy map in Question in the most part. For example, for the area having largest steep, i.e. two adjacent box has a difference of 100, the policies are correct. However, there are some discrepancies here. For example, in Question 9, there is only one maximum point, and wherever the starting point is, it will end in the same position, which is the right lower corner. However, in Question 23, the heatmap can have several local optimal values, and from different starting points, it is possible to end in different position.

Question 25

There are major two discrepancies with our algorithm implementation. The first one is two araying in the adjacent boxes may be pointing each other, which indicates there is a local optima. If a point starts in either of these two positions, there will never go to an end. In this case, δ may be larger than the threshold for all time, therefore a possible solution is to modify the value iteration algorithm. We can limit the number of iterations together with checking δ is larger than the threshold.

Also, there may exist multiple reward functions for where π is optimal. For example, if $R(s)$ is a constant, then every policy is optimal.

Moreover, instead of having just one optimal value (ending point), the policy generated by the predicted reward function could lead to several optima. To be exactly, this means from different starting point, it may end in different ending point.

Last but not least, since for our project, the number of states could be 100 and the number of actions is 4, so that the transition probability matrix is of size 100*100*4, and it is suspected that for such a large state space, the formulation of LP may become intractable. In such circumstance, we may form the reward function $R(s)$ as a linear combination of feature vectors of the space.

Then, we could form a LP similar to that in Question 11 to solve the problem.

After modifying the value iteration algorithm, we can see there is a slight increase in the max accuracy.