Yajie Wang

Miles Abel

8/13/2019

Project 3

1. Introduction

Reinforcement Learning and Inverse Reinforcement Learning are explored in Project 3. Reinforcement Learning is learning from interaction to achieve a goal, i.e., learn the optimal policy of an agent in a given environment. Whereas, Inverse Reinforcement Learning is the process of learning an expert's reward function by observing the optimal policy of the expert.

2. Reinforcement Learning

	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

Figure 1: 2-D grid with state numbering

Question 1

We generate heat maps to represent Reward Function 1 and Reward Function 2. The reward values are the rewards for transitioning to the new state, s'. Figure 2 shows Reward Function 1 has all rewards of 0 except for a reward of 1 for transitioning to state 99. Similarly, Figure 3 shows transitioning to most of the states has a reward of 0 (darker yellow squares); transitioning to state 99 has a reward value of 10 (bright yellow square); and a small grouping of blue squares have rewards of -100 for transitioning to these states.

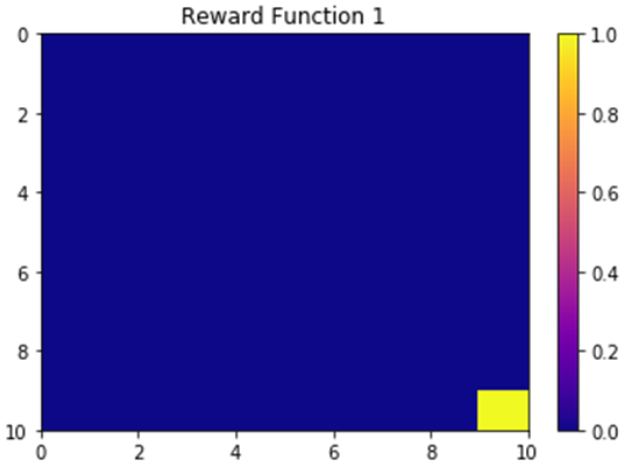

Figure 2: Heat map of Reward Function 1

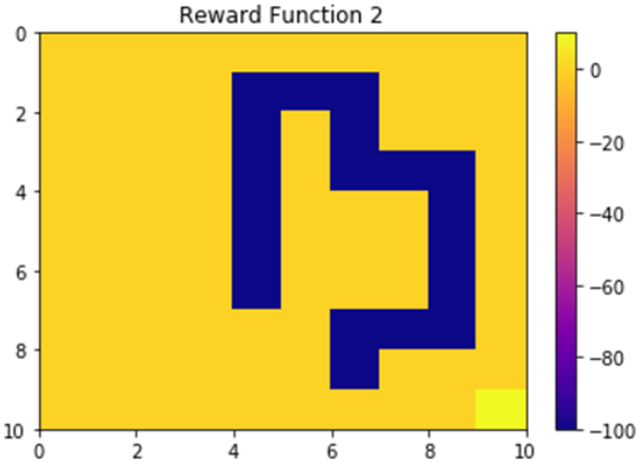

Figure 3: Heat map of Reward Function 2

3. Optimal policy learning using RL algorithms

Question 2

We create and set up the Markov Decision Process (MDP) environment with the following criteria: 1) number of states = 100 (10 x 10 square grid) 2) number of actions is 4 (left, right, up, down) 3) w = 0.1 4) discount factor is 0.8 5) using Reward Function 1. After the generation of the environment, we write code to implement the Initialization and Estimation portions of the value iteration algorithm, shown in Figure 4.

1: **procedure** VALUE ITERATION($\mathcal{P}^a_{ss'}$, $\mathcal{R}^a_{ss'}$, $\mathcal{S}$, $\mathcal{A}$, γ):
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: $V(s) \leftarrow \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} \mathcal{P}^a_{ss'}[\mathcal{R}^a_{ss'} + \gamma V(s')]$;
11: $\Delta \leftarrow \max(\Delta, |v - V(s)|)$;
12: **end for**
13: **end while**
14: **for all** $s \in \mathcal{S}$ **do** ▷ Computation
15: $\pi(s) \leftarrow \arg\max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} \mathcal{P}^a_{ss'}[\mathcal{R}^a_{ss'} + \gamma V(s')]$;
16: **end for**
17: **end procedure** return π

Figure 4: Value iteration algorithm

Using a threshold of $\epsilon = 0.01$, we generate a plot of the optimal value for each state of the 2D grid, shown in Figure 5. Observing patterns in Figure 2, we notice that the smallest state values are those that are furthest away from the single positive reward state, state 99, in the bottom right corner. Furthermore, the state values increase along any path that moves closer to the positive reward.

	0	1	2	3	4	5	6	7	8	9
0	0.042	0.063	0.09	0.124	0.167	0.222	0.291	0.379	0.491	0.61
1	0.063	0.088	0.122	0.165	0.219	0.289	0.378	0.491	0.633	0.787
2	0.09	0.122	0.164	0.219	0.289	0.378	0.491	0.635	0.817	1.019
3	0.124	0.165	0.219	0.289	0.378	0.491	0.636	0.82	1.052	1.315
4	0.167	0.219	0.289	0.378	0.491	0.636	0.82	1.054	1.352	1.695
5	0.222	0.289	0.378	0.491	0.636	0.82	1.054	1.353	1.733	2.182
6	0.291	0.378	0.491	0.636	0.82	1.054	1.353	1.734	2.22	2.807
7	0.379	0.491	0.635	0.82	1.054	1.353	1.734	2.22	2.839	3.608
8	0.491	0.633	0.817	1.052	1.352	1.733	2.22	2.839	3.629	4.635
9	0.61	0.787	1.019	1.315	1.695	2.182	2.807	3.608	4.635	4.702

Figure 5: State value function values for each state

Question 3

Figure 6 displays the heat map of optimal state values for Reward Function 1.

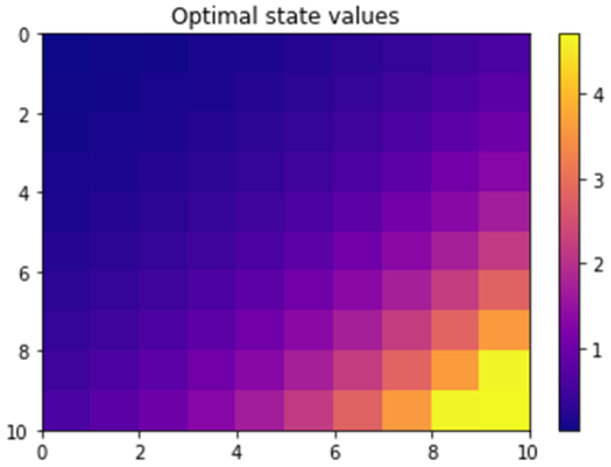

Figure 6: Heat map of the optimal state values of Reward Function 1

Question 4

The optimal state values increase as the agent approaches the state with highest reward, i.e., state 99 (referring to the state numbering in Figure 1). Put another way, the lower the number of actions required to move from a state to state 99 the higher the optimal state value. Figure 6 shows this pattern. As less actions are required to reach state 99, the color of the squares turn from dark purple to purple to pink to orange and finally yellow. State 89 and 98 are the only states requiring one action to reach state 99, and they are the second most yellow colors on the heat map. Similarly, states 79, 88, and 97 are the only states requiring one action to reach state 99, and they are the only states that are light orange. That is, as the color of the squares becomes warmer, the number of actions required to reach the reward decreases – see the legend in Figure 6 for the scale from cool colors to bright colors. Therefore, the optimal state values increase as the minimum number of actions to the state with highest reward, state 99, decreases.

Question 5

The optimal actions match intuition for Reward Function 1. The arrows all lead directly to the reward, state 99. That is, no arrow is pointing to the left or up; and all arrows are pointing down or right and follow a direct path to state 99. It is possible for the agent to compute the optimal action to take at each state by observing the optimal values of its neighboring states. Doing so takes the agent towards the neighbor that has the highest state value.

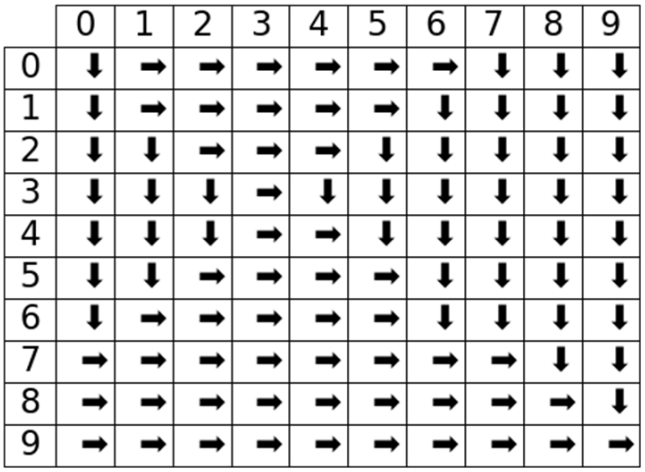

Figure 7: Optimal actions displayed as arrows for Reward Function 1

Question 6

Figure 8 displays the optimal state value plot for Reward Function 2. The highest optimal state value corresponds to the highest reward, which is for state 99. However, this time we see that there are negative optimal state values for some states, which correspond to the states with -100 reward values or states that are surrounded by the -100 reward states.

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

Figure 8: Optimal state value 2-D grid for Reward Function 2

Question 7

Figure 9 displays the heat map of the optimal state values for Reward Function 2.

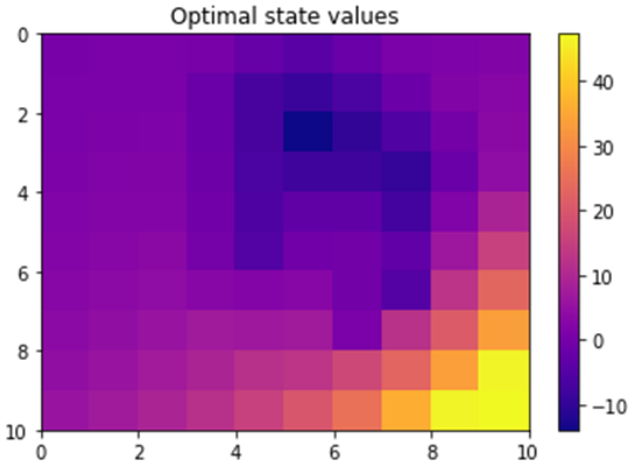

Figure 9: Heat map of the optimal state values for Reward Function 2

Question 8

Figure 9 displays the heat map for the optimal state value for Reward Function 2. The darkest states on the grid reflect the same general shape as the -100 reward states make. The bottom right corner is where the reward of 10 at state 99 is; and we see that this area is the bright spot of the grid, which matches our expectation. The general flow of the heat map, starting in the upper left and moving to the bottom right, is that the grid is cool and continues to heat up as the states approach the highest reward state, state 99, except for a large obstruction in the middle, upper middle, and middle-upper right portion of the grid. The obstruction is the darkest or coolest area of the grid and represents the -100 reward states and those states surrounded by these negative reward states.

Question 9

Figure 10 displays the optimal action value arrow map for Reward Function 2. The optimal policy matches intuition in general with a few exceptions. That is, the arrows at states 30 and 40 should face right instead of left. However, the arrows in general point towards the shortest path to the reward, and those states surrounded by -100 rewards have arrows that point out of this barrier and directly to the reward of 10 in state 99. It is possible for the agent to compute the optimal action to take at each state by observing the optimal values of the neighboring state.

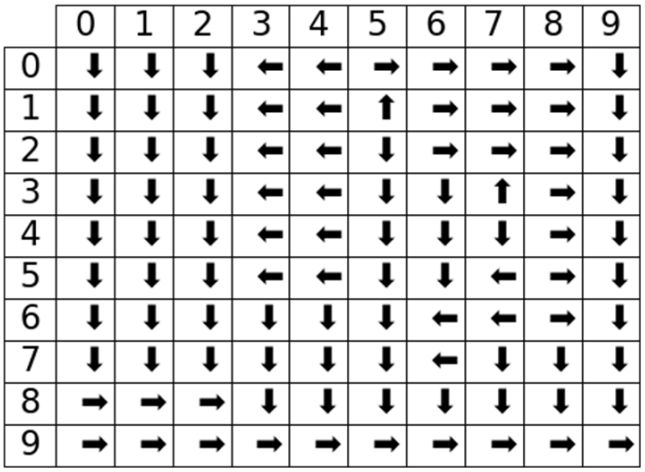

Figure 10: Optimal action state arrow map for Reward Function 2

Inverse Reinforcement learning

Question 10

The Inverse Reinforcement Learning (IRL) Linear Programming (LP) formulation is given by

$$\underset{\boldsymbol{R},t_i,u_i}{\text{maximize}} \sum_{i=1}^{|S|}(t_i - \lambda u_i)$$

$$\begin{aligned} subject\ to \quad & \left[\left(\boldsymbol{P}_{a1}(i) - \boldsymbol{P}_a(i)\right)(\boldsymbol{I} - \gamma \boldsymbol{P}_{a1})^{-1}\boldsymbol{R}\right] \geq t_i, \forall a \ni \boldsymbol{A}\backslash a_1, \forall i \\ & (\boldsymbol{P}_{a1} - \boldsymbol{P}_a)(\boldsymbol{I} - \gamma \boldsymbol{P}_{a1})^{-1}\boldsymbol{R} \geq 0, \forall a \in \boldsymbol{A}\backslash a_1 \\ & -\boldsymbol{u} \leq \boldsymbol{R} \leq \boldsymbol{u} \\ & |\boldsymbol{R_i}| \leq R_{max}, \ \ i = 1, 2, \dots, |S| \end{aligned} \tag{1}$$

For ease of implementation, we rewrite equation (1) into an equivalent form using block matrices.

$$\begin{aligned} \underset{x}{\text{maximize}} \quad & \boldsymbol{c}^T\boldsymbol{x} \\ subject\ to \quad & \boldsymbol{D}\boldsymbol{x} \leq \boldsymbol{b}, \ \ \forall a \in \boldsymbol{A}\backslash a_1 \end{aligned} \tag{2}$$

We then express the elements of equation (2), **c, x, D,** and **b**, in terms of the elements of equation (1), **R, P**a, **P**a1, ti, u, λ, and Rmax.

$$\sum_{i=1}^{|S|}(t_i - \lambda u_i) = \begin{bmatrix}\mathbf{1} & -\boldsymbol{\lambda} & \mathbf{0}\end{bmatrix}\begin{bmatrix}\boldsymbol{t}\\ \boldsymbol{u}\\ \boldsymbol{R}\end{bmatrix},$$

where **1**, $\boldsymbol{\lambda}$, **0**, **t, u, R** are 100 x 1 vectors.

$$\begin{cases}(\boldsymbol{P_{a1}}(i) - \boldsymbol{P_a}(i))(I - \gamma \boldsymbol{P_{a1}})^{-1}\boldsymbol{R} \geq t_i \\ (\boldsymbol{P_{a1}} - \boldsymbol{P_a})(I - \gamma \boldsymbol{P_{a1}})^{-1}\boldsymbol{R} \geq 0 \\ -\boldsymbol{u} \leq \boldsymbol{R} \leq \boldsymbol{u} \\ |\boldsymbol{R}_i| \leq R_{max}\end{cases} \Rightarrow \begin{cases}(\boldsymbol{P_{a1}}(i) - \boldsymbol{P_a}(i))(I - \gamma \boldsymbol{P_{a1}})^{-1}\boldsymbol{R} \leq -t_i \\ (\boldsymbol{P_{a1}} - \boldsymbol{P_a})(I - \gamma \boldsymbol{P_{a1}})^{-1}\boldsymbol{R} \leq 0 \\ \boldsymbol{R} \leq \boldsymbol{u} \\ -\boldsymbol{u} \leq \boldsymbol{R} \\ \boldsymbol{R} \leq R_{max} \\ -\boldsymbol{R} \leq R_{max}\end{cases}$$

$$\begin{cases}t_i + \left(\boldsymbol{P_{a1}}(i) - \boldsymbol{P_a}(i)\right)(I - \gamma \boldsymbol{P_{a1}})^{-1}\boldsymbol{R} \leq \mathbf{0} \\ (\boldsymbol{P_{a1}} - \boldsymbol{P_a})(I - \gamma \boldsymbol{P_{a1}})^{-1}\boldsymbol{R} \leq 0 \\ -\boldsymbol{u} + \boldsymbol{R} \leq \mathbf{0} \\ -\boldsymbol{u} - \boldsymbol{R} \leq \mathbf{0} \\ \boldsymbol{R} \leq R_{max} \\ -\boldsymbol{R} \leq R_{max}\end{cases}$$

$$\Rightarrow \begin{cases}\begin{bmatrix}\boldsymbol{I} & \mathbf{0} & (\boldsymbol{P_{a1}} - \boldsymbol{P_a})(I - \gamma \boldsymbol{P_{a1}})^{-1} \\ \mathbf{0} & \mathbf{0} & (\boldsymbol{P_{a1}} - \boldsymbol{P_a})(I - \gamma \boldsymbol{P_{a1}})^{-1} \\ \mathbf{0} & -\boldsymbol{I} & \boldsymbol{I} \\ \mathbf{0} & -\boldsymbol{I} & -\boldsymbol{I} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{I} \\ \mathbf{0} & \mathbf{0} & -\boldsymbol{I}\end{bmatrix}\begin{bmatrix}\boldsymbol{t}\\ \boldsymbol{u}\\ \boldsymbol{R}\end{bmatrix} = \begin{bmatrix}\mathbf{0}\\ \mathbf{0}\\ \mathbf{0}\\ \mathbf{0}\\ R_{max}\\ R_{max}\end{bmatrix}$$

Therefore, we find the expressions **c, x, D,** and **b** with $(\boldsymbol{P_{a1}}(i) - \boldsymbol{P_a}(i))$ as a 1 x 100 row vector and $(I - \gamma \boldsymbol{P_{a1}})^{-1}$ as a 100 x 100 matrix as follows:

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

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

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

$$\boldsymbol{b} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ R_{max} \\ R_{max} \end{bmatrix}.$$

Note that matrix **c** is a 300 x 1 matrix; matrix **D** is a 1000 x 300 matrix; matrix **b** is a 1000 x 1 matrix; and matrix **x** is a 300 x 1 matrix.

Question 11

We now carry out the Inverse Reinforcement Learning (IRL) algorithm to extract Reward Function 1, using the optimal policy previously found to define the optimal action of the expert at each state s, $O_E(s)$; and compare $O_E(s)$ to the calculated optimal actions of the agent, $O_A(s)$. To calculate the optimal actions of the agent at each state s, we solve the linear program given in equation (2) above, using cvxopt solvers. The solver allows us to extract the reward function to compute the optimal policy of the agent using the value iteration algorithm. Finally, we sweep over the adjustable penalty coefficient, λ, from 0 to 5 in 500 evenly spaced steps, to find the optimal combination of lambda to maximize the accuracy.

With accuracy defined as

$$Accuracy = \frac{\sum_{s \in S} m(s)}{|S|} \tag{3}$$

and

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

Figure 11 displays the accuracy vs lambda plot for all 500 data points.

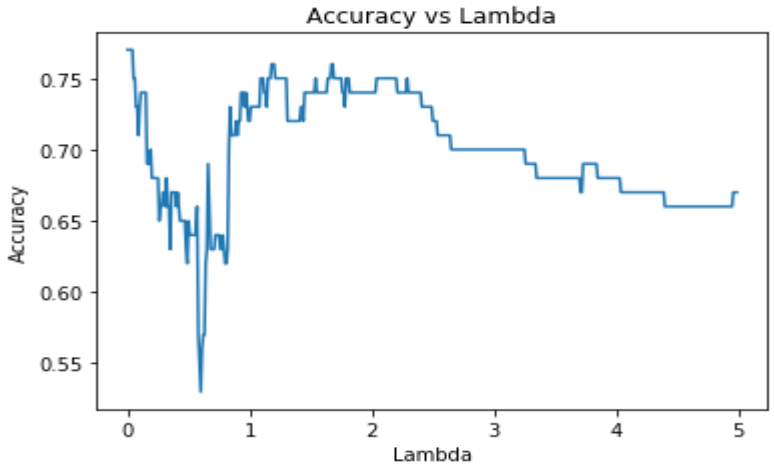

Figure 11: Extracted Reward Function 1 accuracy vs lambda plot

Question 12

A maximum accuracy of 77% is achieved for 5 lambda values:

$\lambda_{max}^{(1)} = 0,\ 0.01, 0.02, 0.03, 0.04.$

Question 13

We compare the heat map of Reward Function 1 to that of the heat map generated from using only the $\lambda_{max}^{(1)}$ parameter set to create the reward function. Although there are many differences, there are some general similarities to the heat maps of the two. Figure 12 shows that the reward function generated from the $\lambda_{max}^{(1)}$ set has its lowest reward value in the upper left corner and highest reward value in the bottom right corner. However, there are clearly major errors. For instance, more than half of the extracted reward function graph has the highest or at least very high reward values, indicated by the bright colors. Originally, all of the states are the same value except the positive reward state. Also, the heat scale doubles in range to cover the interval [-1, 1] instead of [0,1].

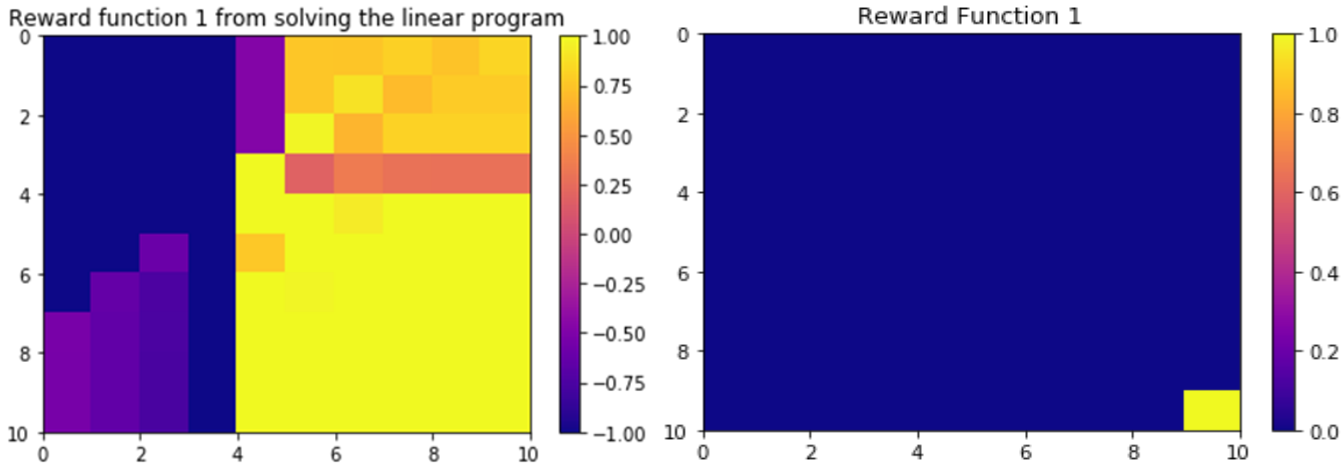

Figure 12: Extracted Reward Function 1 generated from $\lambda_{max}^{(1)}$ (left), and Reward Function 1 (right)

Question 14

The optimal state values heat map generated from the extracted reward function that used only the $\lambda_{max}^{(1)}$ parameter set is displayed in Figure 13.

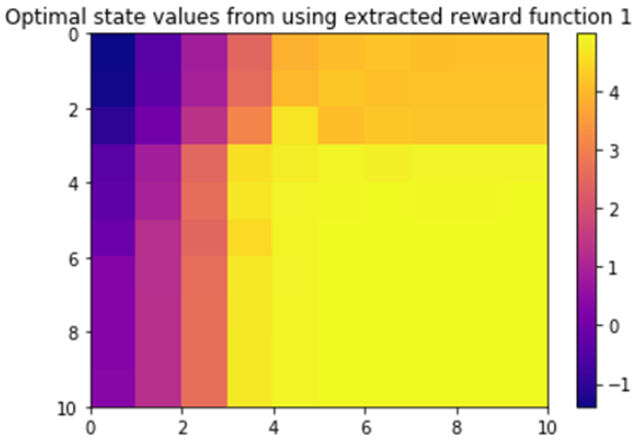

Figure 13: Optimal values heat map generated from extracted reward function generated from the $\lambda_{max}^{(1)}$ parameter set

Question 15

Although there are significant differences, a sense of the general flow of the optimal state value heat map shown in Figure 6 (Question 3) can be found in Figure 13 (Question 14). That is, we see that the coolest colors of the heat map are in the upper left corner, state 0, and the map generally heats up as the states approach the bottom right, state 99. However, there is significant distortion in the entire bottom right half and part of the middle of the grid all have the hottest colors. Also, the upper right corner is hotter than the bottom left instead of displaying the symmetric increase in state value as the states approach the reward state. This blurs the boundary between the reward state and the non-reward states.

Question 16

Figure 14 displays the optimal policy, i.e., the optimal action at each state, for the IRL extracted version of Reward Function 1.

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

Figure 14: Optimal policy with IRL extracted Reward Function 1

Question 17

Comparing Figure 7 (Question 5) and Figure 14 (Question 16) we find that in general the two optimal policy arrow maps have the same actions indicated. However, there are a few differences. States 52, 61, 62, 70, 71, and 75 all point in a non-optimal direction, i.e., away from the reward. In general, the optimal policy map, using the IRL algorithm extracted reward function, indicates the same actions at each state as the original Reward Function 1 with the six non-optimal cases mentioned above and a few states have actions pointing in the other optimal direction than specified in Figure 7. Note, an action to the right or down at any state that does not cause the agent to leave the grid, is an optimal action. This causes the agent to take unnecessary steps and increases the number of steps. Finally, there are states, such as state 0, that have no path to the reward state.

Question 18

We now carry out the Inverse Reinforcement Learning (IRL) algorithm to extract Reward Function 2, using the optimal policy previously found to define the optimal action of the expert at each state s, $O_E(s)$; and compare $O_E(s)$ to the calculated optimal actions of the agent, $O_A(s)$. To calculate the optimal actions of the agent at each state s, we solve the linear program given in equation (2) (Question 11), using cvxopt solvers. The solver allows us to extract the reward function to compute the optimal policy of the agent using the value iteration algorithm. Finally, we sweep over the adjustable penalty coefficient, λ, from 0 to 5 in 500 evenly spaced steps, to find the optimal combination of lambda to maximize the accuracy. The accuracy is defined the same as above in equation (3).

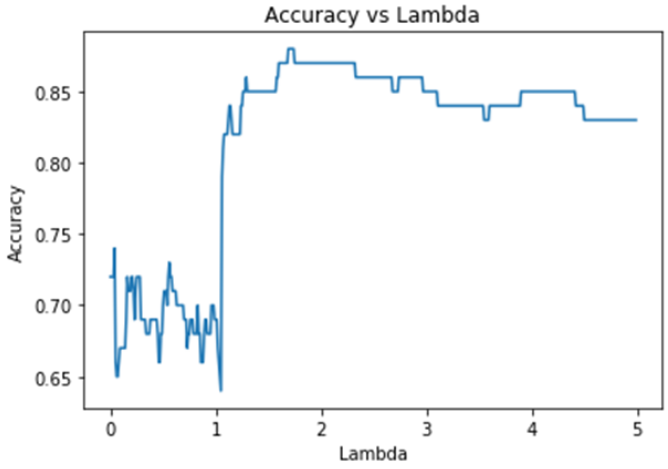

Figure 15: Extracted Reward Function 2 accuracy vs lambda plot

Question 19

A maximum accuracy of 88% is achieved for six values of lambda:

$\lambda_{max}^{(2)} = 1.69, 1.7, 1.71, 1.72, 1.73, 1.74$.

Question 20

The ground truth Reward Function 2 and the extracted Reward Function 2 generated from $\lambda_{max}^{(2)}$ are shown in Figure 16. A blurry version of the original is visible but with much distortion in particular in the lower portion of the grid. The location of the highest reward state has been lost, as the two highest reward states occur now at state 28 and state 78. Also, note the heat scale legend for the extracted reward function has reduced in span by an order of magnitude.

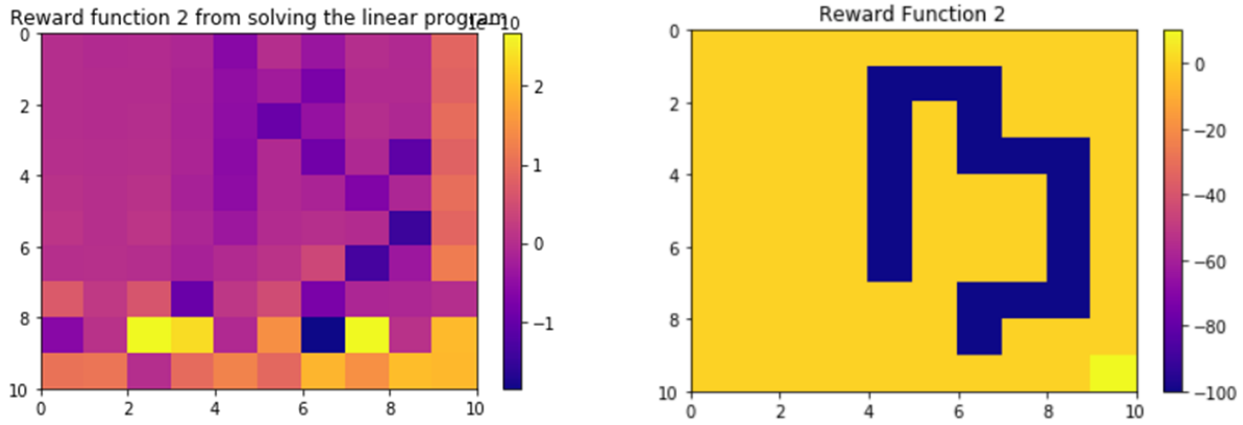

Figure 16: Extracted Reward Function 2 generated from $\lambda_{max}^{(2)}$ (left), and Reward Function 2 (right)

Question 21

Using the extracted reward function from Question 20, we compute the optimal values of the states of the 2-D grid. Again, we use the optimal state value algorithm used for Question 2. Figure 17 shows the heat map of the optimal state values for the extracted Reward Function 2.

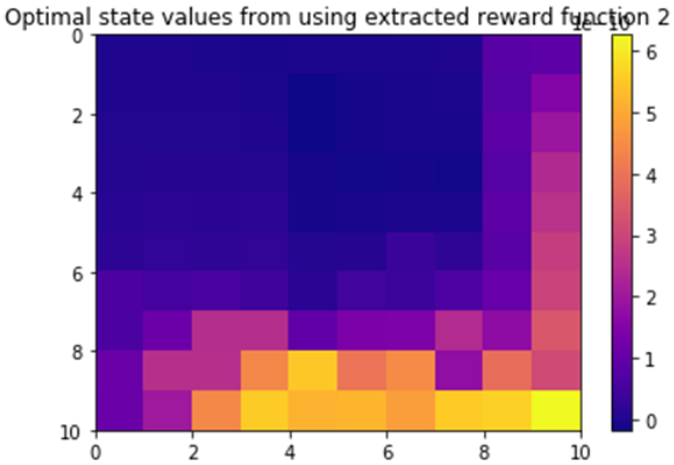

Figure 17: Optimal state values using extracted Reward Function 2

Question 22

Comparing the heat map in Figure 9 (Question 7) to the heat map of Figure 17 (Question 21), we see similarities and differences. Both heat maps indicate state 99 has the highest reward by making this the brightest state. Additionally, both heat maps have the bottom right corner as being a bright area of the grid. However, the heat map for the extracted reward function has some differences in that the bottom center and bottom left of center are significantly brighter than the original Reward Function 2. Furthermore, the fidelity of the states in the area of the negative reward states has substantially diminished. It requires some effort to distinguish the distinction between the purple of the surrounding states and those of the negative reward states. Also, the location and shape of the negative value states seems to have shifted and changed slightly as compared to the original in Figure 9.

Question 23

The optimal policy arrow plot using the IRL extracted Reward Function 2 is displayed in Figure 18.

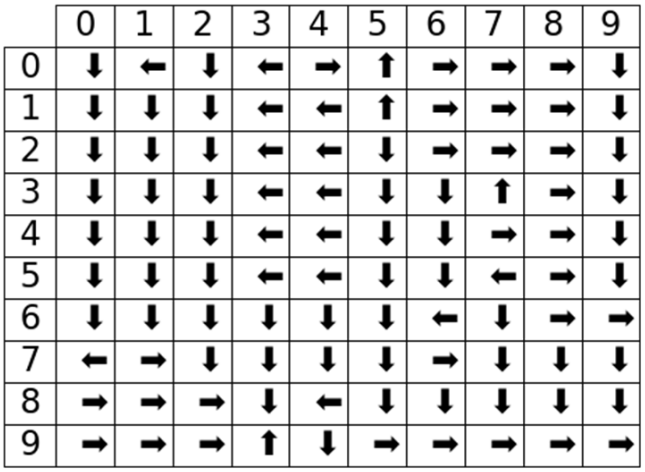

Figure 18: Optimal policy with IRL extracted Reward Function 2

Question 24

The overall general flow of the arrows is mainly preserved between the plot in Figure 10 and the plot in Figure 18. Both have an optimal policy that flows towards the bottom right corner, state 99. However, there no longer is a path from state 00 to state 99 in Figure 18. For instance, the directions of states 39 and 49 prohibit the optimal path completion. And, in general, the number of steps to reach a higher valued state increases because of the sub-optimal actions selected at some states.

Question 25

One discrepancy is that there is no longer a path from state 00 to state 99. Shown in Figure 19, the red circle centered around state 38 represents a discrepancy from the original optimal policy plot in Figure 10 that prevents the agent from reaching reward state 99. Whereas, the left arrow at state 10 just increases the total number of steps the agent if the agent could reach the reward state.

To correct the lack of path from state 00 to 99 and increased number of steps due to changed actions at specific states, we implemented several potential solutions. We focused our adjustments on line 15 of the Value Iteration algorithm:

$$\pi(s) \leftarrow \arg\max_{a \in A} \sum_{s' \in S} P_{ss'}^{a} \left[R_{ss'}^{a} + \gamma V(s') \right].$$

We tested Policy Iteration by implementing the following formula

$$V_{i+1}^{\pi_k}(s) \leftarrow \sum_{s'} \left[R(s, \pi_k(s), s') + P_{ss'}^{a} \gamma V_i^{\pi_k}(s') \right].$$

Additionally, we tested adjusting the discount factor, γ. Unfortunately, we were unsuccessful at reducing the discrepancies using either Policy Iteration or adjusting the discount factor. Using the Policy Iteration technique, we achieved an accuracy distribution as shown in Figure 20.

To address the second discrepancy, we implemented what is called the *costly single-step deviation*[1] that seeks to maximize the distance between the policy's Q-values and the other actions' Q-values given by:

$$maximize: \sum_{s \in S} \left(Q^{\pi}(s, \pi(s)) - \max_{a \in A \setminus \pi(s)} Q^{\pi}(s, a) \right).$$

Unfortunately, we were unsuccessful at reducing either one of the discrepancies using the methods described.

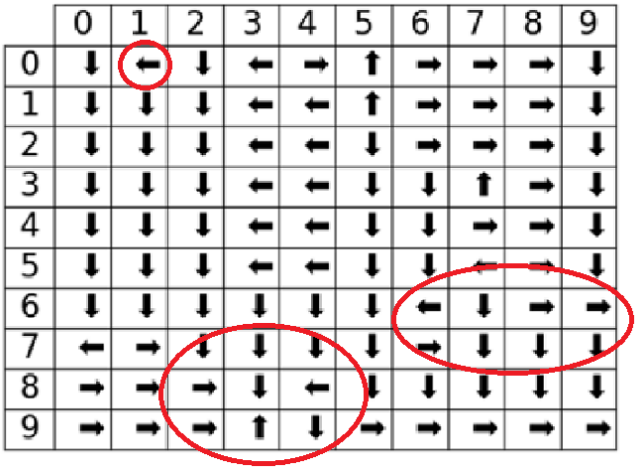

Figure 19: Identification of areas where Figure 18 diverged from Figure 10

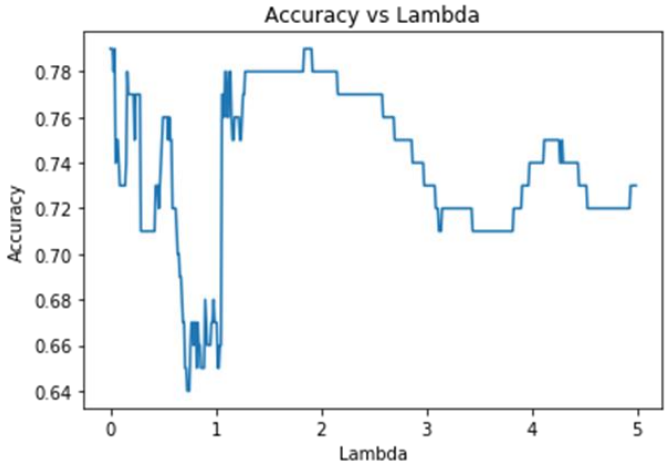

Figure 20: Policy Iteration improvement attempt

References

1. Heidecke, J., “Inverse Reinforcement Learning pt. I,” February 13, 2018, https://thinkingwires.com/posts/2018-02-13-irl-tutorial-1.html.