Project 3: Reinforcement learning and Inverse Reinforcement learning

May 7,2018

Due on Monday, May 21, 2018 by 11:59 PM

1 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.

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.

2.1 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 $(\mathcal{S}, \mathcal{A}, \mathcal{P}^a_{ss'}, \mathcal{R}^a_{ss'}, \gamma)$ where:

- $\mathcal{S}$ is a set of **states**
- $\mathcal{A}$ is a set of **actions**
- $\mathcal{P}^a_{ss'}$ is a set of **transition probabilities**, where $\mathcal{P}^a_{ss'}$ is the probability of transitioning from state $s \in \mathcal{S}$ to state $s' \in \mathcal{S}$ after taking action $a \in \mathcal{A}$
  - $\mathcal{P}^a_{ss'} = \mathbb{P}(s_{t+1} = s' | s_t = s, a_t = a)$
- Given any current state and action, s and a, together with any next state, s', the expected value of the next reward is $\mathcal{R}^a_{ss'}$
  - $\mathcal{R}^a_{ss'} = \mathbb{E}(r_{t+1} | s_t = s, a_t = a, s_{t+1} = s')$
- $\gamma \in [0, 1)$ is the discount factor, and it is used to compute the present value of future reward
  - If γ is close to 1 then the future rewards are discounted less
  - If γ is close to 0 then the future rewards are discounted more

In the next few subsections, we will discuss the parameters that will be used to generate the environment for the project.

2.1.1 State space

In this project, we consider the state space to be a 2-D square grid with 100 states. The 2-D square grid along with the numbering of the states is shown in figure 1

	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 square grid with state numbering

2.1.2 Action set

In this project, we consider the action set($\mathcal{A}$) to contain the 4 following actions:

- Move Right
- Move Left

- Move Up
- Move Down

The 4 types of actions are displayed in figure 2

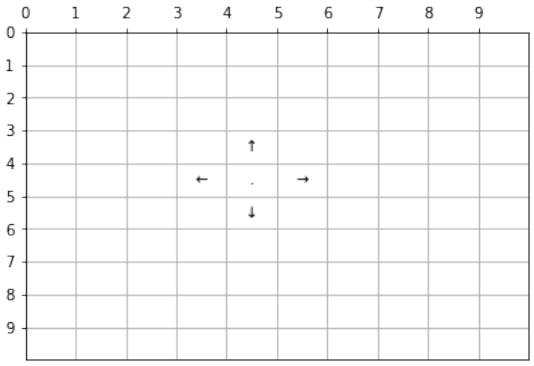


Figure 2: 4 types of action

From the above figure, we can see that the agent can take 4 actions from the state marked with a dot.

2.1.3 Transition probabilities

In this project, we define the transition probabilities in the following manner:

1. If state s' and s are not neighboring states in the 2-D grid, then

$$\mathbb{P}(s_{t+1} = s' | s_t = s, a_t = a) = 0$$

   s' and s are neighbors in the 2-D grid if you can move to s' from s by taking an action a from the action set $\mathcal{A}$. We will consider a state s to be a neighbor of itself. For example, from figure 1 we can observe that states 1 and 11 are neighbors (we can transition from 1 to 11 by moving right) but states 1 and 12 are not neighbors.

2. Each action corresponds to a movement in the intended direction with probability $1 - w$, but has a probability of w of moving in a random direction instead due to wind. To illustrate this, let's consider the states shown in figure 3

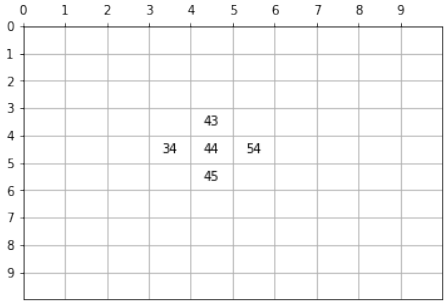


Figure 3: Inner grid states (Non-boundary states)

The transition probabilities for the non-boundary states shown in figure 3 are given below:

第一个才是正确的向上走所到达的 — intended direction

$$\mathbb{P}(s_{t+1} = 43 | s_t = 44, a_t = \uparrow) = 1 - w + \frac{w}{4}$$

$$\mathbb{P}(s_{t+1} = 34 | s_t = 44, a_t = \uparrow) = \frac{w}{4}$$

$$\mathbb{P}(s_{t+1} = 54 | s_t = 44, a_t = \uparrow) = \frac{w}{4}$$

$$\mathbb{P}(s_{t+1} = 45 | s_t = 44, a_t = \uparrow) = \frac{w}{4}$$

From the above calculation it can be observed that if the agent is at a non-boundary state then it has 4 neighbors excluding itself and the probability w is uniformly distributed over the 4 neighbors. Also, if the agent is at a non-boundary state then it transitions to a new state after taking an action ($\mathbb{P}(s_{t+1} = 44 | s_t = 44, a_t = \uparrow) = 0$)

non-bounded 的state下一次一定会到其他位置，不会停在自己

3. If the agent is at one of the four corner states (0,9,90,99), the agent stays at the current state if it takes an action to move off the grid or is blown off the grid by wind. The actions can be divided into two categories:

corner state的位置有可能留在原位，概率即为其跳出地图的概率

   - Action to move off the grid
   - Action to stay in the grid

To illustrate this, let's consider the states shown in figure 4

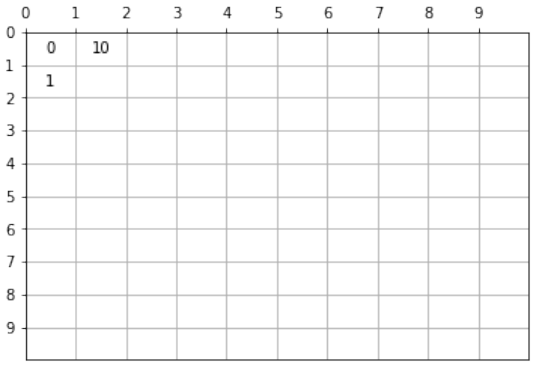


Figure 4: Corner states

The transition probabilities for taking an action to move off the grid are given below:

$$\mathbb{P}(s_{t+1} = 10 | s_t = 0, a_t = \uparrow) = \frac{w}{4}$$
$$\mathbb{P}(s_{t+1} = 1 | s_t = 0, a_t = \uparrow) = \frac{w}{4}$$
$$\mathbb{P}(s_{t+1} = 0 | s_t = 0, a_t = \uparrow) = 1 - w + \frac{w}{4} + \frac{w}{4}$$

因为向上其实是跳出map了，所以向上的intended direction应该是stay at current state

The transition probabilities for taking an action to stay in the grid are given below:

$$\mathbb{P}(s_{t+1} = 10 | s_t = 0, a_t = \rightarrow) = 1 - w + \frac{w}{4}$$
$$\mathbb{P}(s_{t+1} = 1 | s_t = 0, a_t = \rightarrow) = \frac{w}{4}$$
$$\mathbb{P}(s_{t+1} = 0 | s_t = 0, a_t = \rightarrow) = \frac{w}{4} + \frac{w}{4}$$

At a corner state, you can be blown off the grid in two directions. As a result, we have $\mathbb{P}(s_{t+1} = 0 | s_t = 0, a_t = \rightarrow) = \frac{w}{4} + \frac{w}{4}$ since we can be blown off the grid in two directions and in both the cases we stay at the current state.

4. If the agent is at one of the edge states, the agent stays at the current state if it takes an action to move off the grid or is blown off the grid by wind. The actions can be divided into two categories:

   - Action to move off the grid
   - Action to stay in the grid

   To illustrate this, let's consider the states shown in figure 5

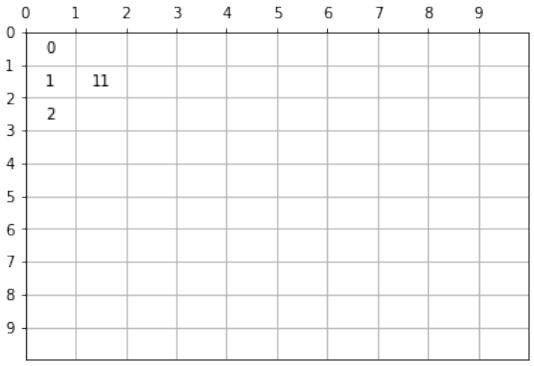


Figure 5: Edge states

The transition probabilities for taking an action to move off the grid are given below:

$$\mathbb{P}(s_{t+1} = 0 | s_t = 1, a_t = \leftarrow) = \frac{w}{4}$$
$$\mathbb{P}(s_{t+1} = 11 | s_t = 1, a_t = \leftarrow) = \frac{w}{4}$$
$$\mathbb{P}(s_{t+1} = 2 | s_t = 1, a_t = \leftarrow) = \frac{w}{4}$$
$$\mathbb{P}(s_{t+1} = 1 | s_t = 1, a_t = \leftarrow) = 1 - w + \frac{w}{4}$$

The transition probabilities for taking an action to stay in the grid are given below:

$$\mathbb{P}(s_{t+1} = 0 | s_t = 1, a_t = \uparrow) = 1 - w + \frac{w}{4}$$
$$\mathbb{P}(s_{t+1} = 11 | s_t = 1, a_t = \uparrow) = \frac{w}{4}$$
$$\mathbb{P}(s_{t+1} = 2 | s_t = 1, a_t = \uparrow) = \frac{w}{4}$$
$$\mathbb{P}(s_{t+1} = 1 | s_t = 1, a_t = \uparrow) = \frac{w}{4}$$

At an edge state, you can be blown off the grid in one direction. As a result, we have $\mathbb{P}(s_{t+1} = 1 | s_t = 1, a_t = \uparrow) = \frac{w}{4}$ since we can be blown off the grid in one direction and in that case we stay at the current state.

The main difference between a corner state and an edge state is that a corner state has 2 neighbors and an edge state has 3 neighbors.

2.1.4 Reward function

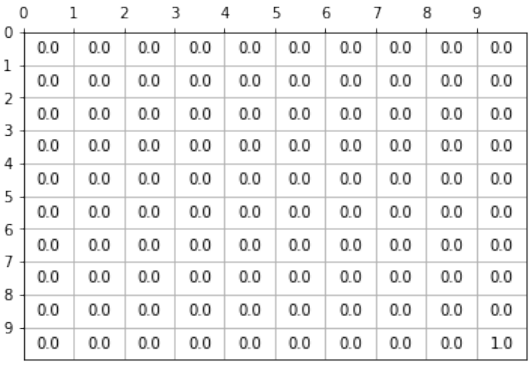
To simplify the project, we will assume that the reward function is independent of the current state (s) and the action that you take at the current state (a). To be specific, reward function only depends on the state that you transition to (s'). With this simplification, we have

$$\mathcal{R}^a_{ss'} = R(s')$$

In this project, we will learn the optimal policy of an agent for two different reward functions:

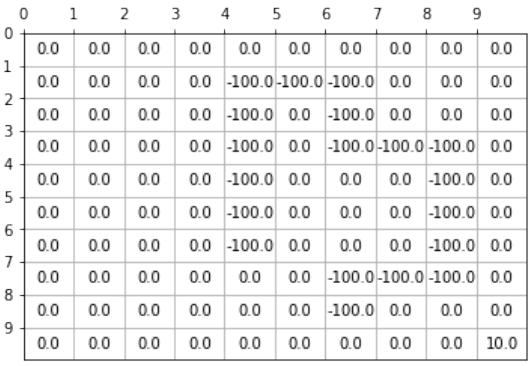
- Reward function 1
- Reward function 2

The two different reward functions are displayed in figures 6 and 7 respectively



	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

Figure 6: Reward function 1



	0	1	2	3	4	5	6	7	8	9
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	-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

Figure 7: Reward function 2

Question 1: (10 points) 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

For solving question 1, you might find the following function useful:

`https://matplotlib.org/api/_as_gen/matplotlib.pyplot.pcolor.html`

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

There are a couple of RL algorithms, but we will use the Value iteration algorithm since it was discussed in detail in the lecture. We will skip the derivation of the algorithm here because it was covered in the lecture (for the derivation details please refer to the lecture slides on Reinforcement learning). We will just reproduce the algorithm below for the ease of implementation:

---

procedure VALUE ITERATION($\mathcal{P}^a_{ss'}$, $\mathcal{R}^a_{ss'}$, $\mathcal{S}$, $\mathcal{A}$, γ):
  for all $s \in \mathcal{S}$ **do** ▷ Initialization
    $V(s) \leftarrow 0$
  end for
  $\Delta \leftarrow \infty$
  while $\Delta > \epsilon$ **do** ▷ Estimation
    $\Delta \leftarrow 0$
    for all $s \in \mathcal{S}$ **do**
      $v \leftarrow V(s)$;
      $V(s) \leftarrow \underset{a \in \mathcal{A}}{max} \sum_{s' \in \mathcal{S}} \mathcal{P}^a_{ss'}[\mathcal{R}^a_{ss'} + \gamma V(s')]$;
      $\Delta \leftarrow max(\Delta, |v - V(s)|)$;
    end for
  end while
  for all $s \in \mathcal{S}$ **do** ▷ Computation
    $\pi(s) \leftarrow \arg \underset{a \in \mathcal{A}}{max} \sum_{s' \in \mathcal{S}} \mathcal{P}^a_{ss'}[\mathcal{R}^a_{ss'} + \gamma V(s')]$;
  end for
end procedure return π

Question 2: (40 points) 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.

Question 3: (5 points) 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).

Question 4: (15 points) Explain the distribution of the optimal state values across the 2-D grid. (Hint: Use the figure generated in question 3 to explain)

Question 5: (30 points) 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 it's neighboring states? In this question, you should have 1 plot.

Question 6: (10 points) 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.

Question 7: (10 points) 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.

Question 8: (20 points) Explain the distribution of the optimal state values across the 2-D grid. (Hint: Use the figure generated in question 7 to explain)

Question 9: (20 points) 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.

4 Inverse Reinforcement learning (IRL)

Inverse Reinforcement learning (IRL) is the task of learning an expert's reward function by observing the optimal behavior of the expert. The motivation for IRL comes from apprenticeship learning. In apprenticeship learning, the goal of the agent is to learn a policy by observing the behavior of an expert. This task can be accomplished in two ways:

1. Learn the policy directly from expert behavior
2. Learn the expert's reward function and use it to generate the optimal policy

The second way is preferred because the reward function provides a much more parsimonious description of behavior. Reward function, rather than the policy, is the most succinct, robust, and transferable definition of the task. Therefore, extracting the reward function of an expert would help design more robust agents.

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.

4.1 IRL algorithm

For finite state spaces, there are a couple of IRL algorithms for extracting the reward function:

- Linear Programming (LP) formulation
- Maximum Entropy formulation

Since we covered LP formulation in the lecture and it is the simplest IRL algorithm, so we will use the LP formulation in this project. We will skip the derivation of the algorithm (for details on the derivation please refer to the lecture slides) here. The LP formulation of the IRL is given by equation 1

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

In the LP given by equation 1, $\mathbf{R}$ is the reward vector ($\mathbf{R}(i) = \mathbf{R}(s_i)$), $\mathbf{P}_a$ is the transition probability matrix, λ is the adjustable penalty coefficient, and t_i's and u_i's are the extra optimization variables (please note that $\mathbf{u}(i) = u_i$). Use the maximum absolute value of the ground truth reward as R_{max}. For the

ease of implementation, we can recast the LP in equation 1 into an equivalent form given by equation 2 using block matrices.

$$\begin{aligned} \underset{\mathbf{x}}{\text{maximize}} \quad & \mathbf{c}^T\mathbf{x} \\ \text{subject to} \quad & \mathbf{D}\mathbf{x} \preceq 0, \quad \forall a \in \mathcal{A} \setminus a_1 \end{aligned} \tag{2}$$

Question 10: (10 points) Express $\mathbf{c}, \mathbf{x}, \mathbf{D}$ in terms of $\mathbf{R}, \mathbf{P}_a, \mathbf{P}_{a_1}, t_i, \mathbf{u}, \lambda$ and R_{max}

4.2 Performance measure

In this project, we use a very simple measure to evaluate the performance of the IRL algorithm. Before we state the performance measure, let's introduce some notation:

- $O_A(s)$: Optimal action of the agent at state s
- $O_E(s)$: Optimal action of the expert at state s
-

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

Then with the above notation, accuracy is given by equation 3

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

Since we are using the optimal policy found in the previous section as the expert behavior, so we will use the optimal policy found in the previous section to fill the $O_E(s)$ values. Please note that these values will be different depending on whether we used Reward Function 1 or Reward Function 2 to create the environment.

To compute $O_A(s)$, we will solve the linear program given by equation 2 to extract the reward function of the expert. For solving the linear program you can use the LP solver in python (from cvxopt import solvers and then use solvers.lp). Then, we will use the extracted reward function to compute the optimal policy of the agent using the value iteration algorithm you implemented in the previous section. The optimal policy of the agent found in this manner will be used to fill the $O_A(s)$ values. Please note that these values will depend on the adjustable penalty coefficient λ. We will tune λ to maximize the accuracy.

Question 11: (30 points) Sweep λ from 0 to 5 to get 500 evenly spaced values for λ. For each value of λ 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 λ. You need to repeat the above process for all 500 values of λ to get 500 data points. Plot λ (x-axis) against Accuracy (y-axis). In this question, you should have 1 plot.

Question 12: (5 points) Use the plot in question 11 to compute the value of λ for which accuracy is maximum. For future reference we will denote this value as $\lambda^{(1)}_{max}$. Please report $\lambda^{(1)}_{max}$

Question 13: (15 points) For $\lambda^{(1)}_{max}$, 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 λ parameter set to $\lambda^{(1)}_{max}$. In this question, you should have 2 plots.

Question 14: (10 points) 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.

Question 15: (10 points) Compare the heat maps of Question 3 and Question 14 and provide a brief explanation on their similarities and differences.

Question 16: (10 points) 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.

Question 17: (10 points) Compare the figures of Question 5 and Question 16 and provide a brief explanation on their similarities and differences.

Question 18: (30 points) Sweep λ from 0 to 5 to get 500 evenly spaced values for λ. For each value of λ compute $O_A(s)$ by following the process described above. For this problem, use the optimal policy of the agent found in question 9 to fill in the $O_E(s)$ values. Then use equation 3 to compute the accuracy of the IRL algorithm for this value of λ. You need to repeat the above process for all 500 values of λ to get 500 data points. Plot λ (x-axis) against Accuracy (y-axis). In this question, you should have 1 plot.

Question 19: (5 points) Use the plot in question 18 to compute the value of λ for which accuracy is maximum. For future reference we will denote this value as $\lambda^{(2)}_{max}$. Please report $\lambda^{(2)}_{max}$

Question 20: (15 points) For $\lambda^{(2)}_{max}$, 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 λ parameter set to $\lambda^{(2)}_{max}$. In this question, you should have 2 plots.

Question 21: (10 points) Use the extracted reward function computed in ques-

tion 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 figure generated in question 7). In this question, you should have 1 plot.

Question 22: (10 points) Compare the heat maps of Question 7 and Question 21 and provide a brief explanation on their similarities and differences.

Question 23: (10 points) 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.

Question 24: (10 points) Compare the figures of Question 9 and Question 23 and provide a brief explanation on their similarities and differences.

Question 25: (50 points) 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.

5 Submission

Please submit a zip file containing your codes and report to CCLE. The zip file should be named as "Project2_UID1_..._UIDn.zip" where UIDx are student ID numbers of team members.