EE232E PROJECT 3

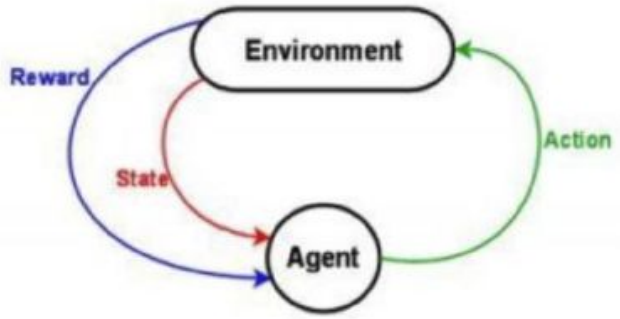
REINFORCEMENT LEARNING AND INVERSE REINFORCEMENT LEARNING

Devanshi Patel	504945601	devanshipatel@cs.ucla.edu
Ekta Malkan	504945210	emalkan@cs.ucla.edu
Pratiksha Kap	704944610	pratikshakap@cs.ucla.edu
Sneha Shankar	404946026	snehashankar@cs.ucla.edu

INTRODUCTION

Learning by observing the environment is an innate human characteristic. When a small child plays, runs, falls down, touches fire, etc. he learns by his senses. The child takes the fun of playing, the pain of falling, the hurt of touch a flame etc. as rewards from the environment and bases his actions based on these experiences. Reinforcement learning is learning what to do—how to map situations to actions—so as to maximize a numerical reward signal. The learner is not told which actions to take, but instead must discover which actions yield the most reward by trying them. In the most interesting and challenging cases, actions may affect not only the immediate reward but also the next situation and, through that, all subsequent rewards. These two characteristics—trial-and-error search and delayed reward—are the two most important distinguishing features of reinforcement learning.

A learning agent must be able to sense the state of its environment to some extent and must be able to take actions that affect the state. The agent also must have a goal relating to the state of the environment. Markov decision processes are intended to include just these three aspects—sensation, action, and goal.



As a human learner, many a times we do not have an answer to consequences of our actions. Nevertheless, we have to make a choice and believe that it would somehow lead us to our goal. Similarly in reinforcement learning, a learning agent needs to explore, which means it might take certain actions and fail, and then try again next time reconsider his past experiences as reward functions. The agent would consider taking actions that produced higher rewards, and would avoid making past mistakes to reach his goal.

In this project,we are provided with reward functions for simplification. We will learn the optimal policy of an agent for two different reward functions:

	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

Reward function 1

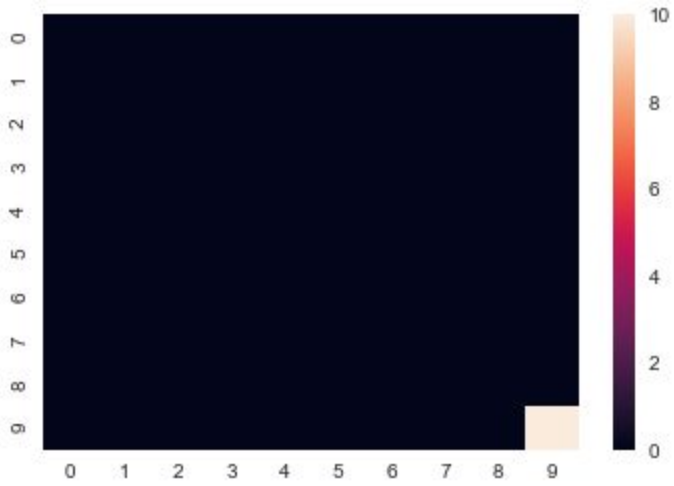
	0	1	2	3	4	5	6	7	8	9
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	-100.0	-100.0	-100.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	-100.0	0.0	-100.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	-100.0	0.0	-100.0	-100.0	-100.0	0.0
4	0.0	0.0	0.0	0.0	-100.0	0.0	0.0	0.0	-100.0	0.0
5	0.0	0.0	0.0	0.0	-100.0	0.0	0.0	0.0	-100.0	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

Reward function 2

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

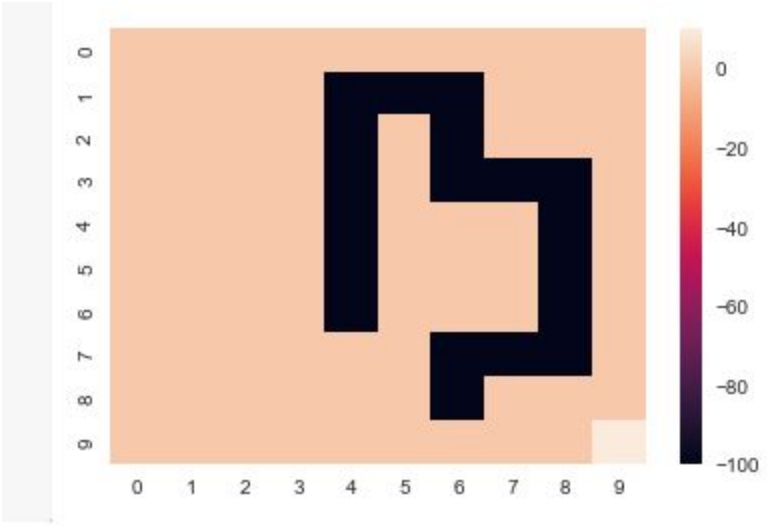
Answer:

Reward Function 1



A heat map is a graphical representation of data where the individual values contained in a matrix are represented as colors. In the heat map above, we can see only two colours. The dark colors are for states with reward functions of value 0 and the white color represents the state with maximum reward function here, which is 1 in this case.

Reward Function 2



In the above heat map for reward function2, we can see the black (dark) colored squares in the states where reward function values are the lowest, (-100 in this case) . Also, we see a lighter shade of colour for all the states where reward function is 0. The value of the 99th state with maximum reward function i.e. 10 is the brightest (almost white).

Thus we observe here that heat maps are a very useful tool in analyzing results of the reward function quickly. Just by a glimpse, we can tell apart a higher reward state from the lower rewarding state. This would help us analyze how an agent should progress towards a maximum reward. Hence, we will be

using heat maps in this project for analysis and comparisons of different reward functions and their corresponding algorithms.

Optimal policy learning using RL algorithms

State is a summary (also called state-space) of all information needed to determine what happens next. you don't need past states to do a optimal decision, all you need is the current state. This is because you could encode on your current state everything you need from the past to take a good decision.
To simplify our universe we imagine the grid world, here our agent's objective is to arrive on the block that gives maximum reward, and avoid the blocks that reduce rewards. The four available actions are: $\uparrow, \downarrow, \leftarrow, \rightarrow$

In this project, we analyze the optimal policy of a single agent in a 2D environment of 10*10 state space modelled by Markov Decision Process. Markov Decision process(MDP) is a framework used to help to make decisions on a stochastic environment. A stochastic environment means that the agent could pursue different actions with unequal probabilities (as opposed to a deterministic environment in which the agent pursues the next action probability of 1).

Our goal is to find a policy, which is a map that gives us all optimal actions on each state on our environment. The only problem is that we don't know which states are good or what the actions to perform. The only way to learn those things is to try them out and learn from our mistakes. MDP is somehow more powerful than simple planning, because your policy will allow you to do optimal actions even if something went wrong along the way. We're looking for a policy , which means a map that gives us optimal actions for every state.

We force the MDP to have good rewards as soon as possible by giving some discount reward over time.

Markov Decision Processes

Here are the most important parts of MDP :
States: A set of possible states S
Model: $T(s,a,s') \therefore P(s'|s,a)$ Probability to go to state s' when you do the action a while you were on state s , is also called transition model.
Action: A(s) , things that you can do on a particular state s
Reward: R(s) , scalar value that you get for being on a state.
Policy: $\Pi(s) \rightarrow a$, our goal, is a map that tells the optimal action for every state
Optimal policy: $\Pi*(s) \rightarrow a$, is a policy that maximizes your expected reward R(s)

The main steps in RL algorithm are:

1. Find optimal state-value or action-value functions
2. Use the optimal state-value or action-value functions to determine the optimal policy

In order to understand how to get an optimal policy, we first need to realize how exactly to define the value $V^*(s)$ of a state, which in other words, is the expected utility we will get if we are at state s and play optimally. We have used Bellman Optimality Equation for finding optimal value functions.

For solving the Bellman Optimality equation, the following iterative solution methods exists :

- Value iteration
- Policy iteration
- Q-learning
- Sarsa

Value Iteration is an iterative algorithm that computes values of states indexed by a k, as in $V^k(s)$), which can also be thought of the best value of state ss assuming the game ends in k time-steps. This is not the actual policy itself, but these values are used to determine the optimal policy.

Starting with V(s)=0 for all s, value iteration performs the following update:

$$V_{k+1}(s) = \max_{a \in A} \sum_{s'} \mathcal{P}^a_{ss'}[\mathcal{R}^a_{ss'} + \gamma V_k(s')]$$

and it repeats until we tell it to stop.

To make this intuitive, during each step, we have a vector of $V^k(s)$ values, and then we do a one-ply expectimax computation to get the next vector. Note that expectimax could compute all of the values we need, but it would take too long due to repeated and infinite depth state trees. Due to the fact that γ^k, as long as $\gamma \neq 1$, will go down to zero, we can prove that the value iteration algorithm converges.

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

Number of states = 100 (state space is a 10 by 10 square grid as displayed in figure 1)
Number of actions = 4 (set of possible actions is displayed in figure 2)
w = 0.1
Discount factor = 0.8
Reward function 1

After you have created the environment, then write an optimal state-value function that takes as input the environment of the agent and outputs the optimal value of each state in the grid. For the optimal state-value function, you have to implement the Initialization (lines 2-4) and Estimation (lines 5-13) steps of the Value Iteration algorithm. For the estimation step, use ε = 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.

Answer:

We have used the following specifications to setup our MDP,

- We have an array ss which is a 10x10 matrix storing all 100 states
- We have created 4 transition matrices of size 100x100. Each matrix will have the transition probability of every state to every other state depending upon the action. An *UP* matrix will have probabilities of state transition when you are using the UP action. Similarly, we have matrices for DOWN, LEFT and RIGHT. Each cell (i,j) in each of these matrices signifies a movement from ith state to the jth state of ss.
- We have two reward functions, rf1 and rf2. These are 10x10 matrices storing reward for all 100 states.

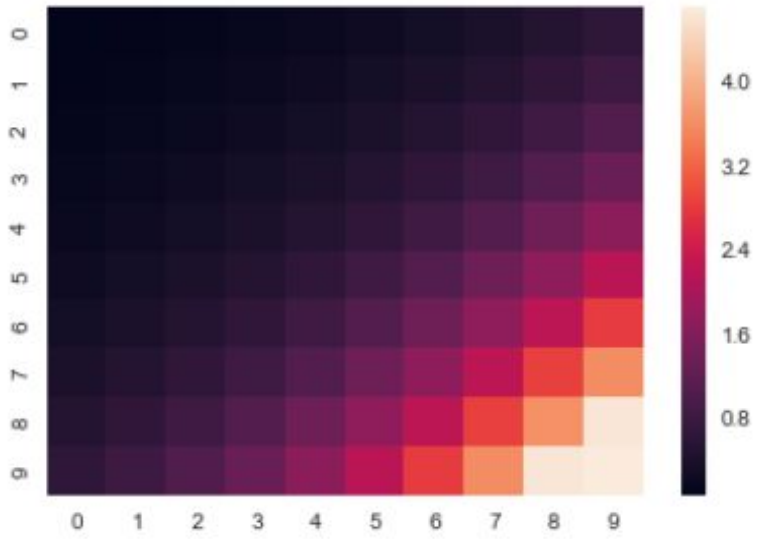
We then used the value iteration algorithm to get the optimal values for first reward function.

Optimal Values for RF1										
	0	1	2	3	4	5	6	7	8	9
0	0.044	0.065	0.091	0.125	0.168	0.223	0.292	0.380	0.491	0.610
1	0.065	0.088	0.122	0.165	0.219	0.289	0.378	0.491	0.633	0.788
2	0.091	0.122	0.165	0.219	0.289	0.378	0.491	0.636	0.818	1.019
3	0.125	0.165	0.219	0.289	0.378	0.491	0.636	0.820	1.052	1.315
4	0.168	0.219	0.289	0.378	0.491	0.636	0.820	1.054	1.352	1.695
5	0.223	0.289	0.378	0.491	0.636	0.820	1.054	1.353	1.733	2.182
6	0.292	0.378	0.491	0.636	0.820	1.054	1.354	1.735	2.220	2.807
7	0.380	0.491	0.636	0.820	1.054	1.353	1.735	2.220	2.839	3.608
8	0.491	0.633	0.818	1.052	1.352	1.733	2.220	2.839	3.629	4.635
9	0.610	0.788	1.019	1.315	1.695	2.182	2.807	3.608	4.635	4.702

From the above plot we understand that the values in the initial states is low (close to 0), and the values gradually increase as we reach the states with the maximum reward value.

Question 3: Generate a heat map of the optimal state values across the 2-D grid.

Heat map of optimal values is:



Question 4: Explain the distribution of the optimal state values across the 2-D grid.

Answer:

From the above diagram, ,we can observe a pattern in the heat map. The value at the state 99 is the maximum . This happens because the reward function gives maximum reward to state 99.As you move away from state 99, the optimal state values decreases gradually. This is because the reward at all other state is 0. In other words, the optimal state value increases gradually as the agent moves towards state 99; which is desirable. As we go close to the exit, the value increases. when you are far away form the exit, the values are low. And because of this, the heat map which we obtained is almost symmetrical across the left diagonal.

Question 5: Implement the computation step of the value iteration algorithm (lines 14-17) to compute the optimal policy of the agent navigating the 2-D state-space. For visualization purpose, you should generate a figure similar to that of figure 1 but with the number of state replaced by the optimal action at that state. The optimal actions should be displayed using arrows. Does the optimal policy of the agent match your intuition? Please provide a brief explanation. Is it possible for the agent to compute the optimal action to take at each state by observing the optimal values of its neighboring states? In this question, you should have 1 plot.

Actions for optimal policy for RF1										
	0	1	2	3	4	5	6	7	8	9
0	→	→	→	→	→	→	→	→	↓	↓
1	↓	→	→	→	→	→	↓	↓	↓	↓
2	↓	↓	→	→	→	↓	↓	↓	↓	↓
3	↓	↓	↓	→	↓	↓	↓	↓	↓	↓
4	↓	↓	↓	→	→	↓	↓	↓	↓	↓
5	↓	↓	→	→	→	→	↓	↓	↓	↓
6	↓	→	→	→	→	→	→	↓	↓	↓
7	↓	→	→	→	→	→	→	→	↓	↓
8	→	→	→	→	→	→	→	→	→	↓
9	→	→	→	→	→	→	→	→	→	→

Observations and Analysis:

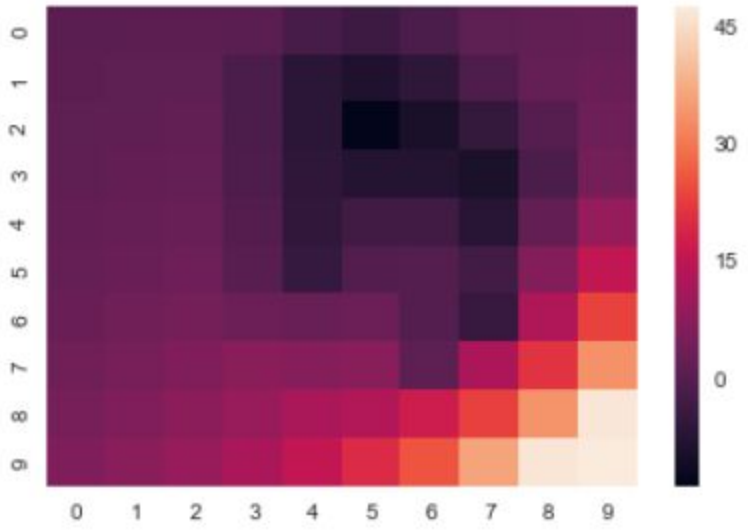
Here, we observe that the arrows are all leading or converging to state 99. Intuitively, this is what which was supposed to be observed as the reward for state 99 is the highest and the agent should eventually find a path to that state. This can also be verified from the optimal values and the heat map of the optimal values which we depicted in the previous questions. Also, in this question where agent relies on reward function 1, it only gets a reward for the state 99. Reward for all other states is 0. Hence, according to the computation step of the Value-Iteration algorithm, the optimal policy will solely depend on the transition probabilities, discount factor and the optimal value of its neighbouring states. However, in this case, we are not varying gamma i.e. the discount factor. Thus, the two factors which affect the optimal policy computation are the transition probabilities and the optimal value of the neighbouring states. The algorithm then takes the max of all directions by considering the optimal values and the transition probabilities of the respective neighbouring states. Thus, it first considers the optimal

values of the neighbouring states and then chooses an optimum value for itself. It is in this manner that the entire optimum policy is built. Also, the agent is likely to move in the direction which gives the best optimal value. Thus we conclude that the agent computes the optimal action at each state by observing the optimal values of its neighboring states and deciding the best among them.

Question 6: Modify the environment of the agent by replacing Reward function 1 with reward function 2. Use the optimal state-value function implemented in question 2 to compute the optimal value of each state in the grid. For visualization purpose, you should generate a figure similar to that of figure 1 but with the number of state replaced by the optimal value of that state. In this question, you should have 1 plot.

Optimal Values for RF2										
	0	1	2	3	4	5	6	7	8	9
0	0.6467	0.7908	0.8208	0.5251	-2.3865	-4.2369	-1.9234	1.1281	1.5912	2.0348
1	0.8277	1.0177	1.0616	-1.8792	-6.7547	-8.6837	-6.3735	-1.2984	1.9248	2.6069
2	1.0613	1.3130	1.4458	-1.6352	-6.7578	-13.9166	-9.6532	-5.5148	-0.1346	3.3555
3	1.3578	1.6892	1.9439	-1.2432	-6.3392	-7.9828	-7.9473	-9.4345	-1.9182	4.3870
4	1.7339	2.1681	2.5859	-0.7365	-5.8467	-3.2584	-3.2411	-7.4345	1.7152	9.1595
5	2.2111	2.7776	3.4133	-0.0381	-5.1141	-0.5534	-0.4875	-2.9835	6.5827	15.3538
6	2.8164	3.5530	4.4788	3.0244	2.4802	2.8802	-0.4655	-4.9105	12.6885	23.2964
7	3.5842	4.5392	5.7926	7.2884	6.7188	7.2411	0.9309	12.3664	21.1592	33.4826
8	4.5580	5.7947	7.3972	9.4395	12.0082	12.8892	17.0973	23.0140	33.7783	46.5288
9	5.7266	7.3161	9.3876	12.0447	15.4524	19.8240	25.4975	36.1576	46.5834	47.3115

Question 7: Generate a heat map of the optimal state values (found in question 6) across the 2-D grid.



Question 8: Explain the distribution of the optimal state values across the 2-D grid.

Answer:

In this case, since the reward function has some rewards to states in addition to the last state i.e. state 99, it is obvious that the optimal state values obtained in this case will not be same as when using

reward function 1. In reward function 2, we observe that some of the states have a negative reward i.e. -100. This means that if the agent moves to that state, it will be getting a negative reward, which is obviously not desirable! Hence, the agent should try to avoid these states. Essentially this means that such states should have lesser optimal state values that the states to which the agent actually wants to go. These states mostly beget a negative optimum state value. This is what is depicted in the above heat map of optimal state values. Here too, the reward for reaching the 99th state is the highest. Thus, the agent will always try to get to this state by avoiding the states as much as possible which give negative reward. Such states with negative optimal values are depicted in darker shades in the heat map. This is very prominently observed in the 52nd state of the grid which begets the darkest colour. The darkest colour signifies that it has the least optimal value. This can be justified because this state is surrounded by states on three sides (up: 51st state, left: 42nd state, right: 62nd state) which have -100 reward. That is why the optimal value of this state is very low, because it is highly possible that if the agent comes to this state then it will most likely land up in one of the neighbouring states which have -100 reward. Hence, the agent should avoid this state. That is why it is justified for this state to have a very low optimal state value. Likewise, for all other states which have dark colour in the above heat map, their optimal state values are also low as compared to the states where the agent will actually benefit. Other states which have also been shown in the shades of purple colour (but a lighter shade) signify that the optimal values is close to 0 but positive. This is starkly observed in the upper left part of the grid. If the agent lands up in some state in the bottom right part of the grid, then it will be easier for him to go to the 99th state which has the highest reward. Therefore, these states in the bottom right part of the grid are signified by a high optimal value and a light shade which essentially means that it is beneficial for the agent to visit these states. Such is the distribution of the optimal state values across the 2-D grid.

Question 9: 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 offi 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.

Actions for optimal policy for RF2										
	0	1	2	3	4	5	6	7	8	9
0	↓	↓	↓	←	←	→	→	→	→	↓
1	↓	↓	↓	←	←	↑	→	→	→	↓
2	↓	↓	↓	←	←	↓	→	→	→	↓
3	↓	↓	↓	←	←	↓	↓	↑	→	↓
4	↓	↓	↓	←	←	↓	↓	↓	→	↓
5	↓	↓	↓	←	←	↓	↓	←	→	↓
6	↓	↓	↓	↓	↓	↓	←	←	→	↓
7	↓	↓	↓	↓	↓	↓	←	↓	↓	↓
8	→	→	→	↓	↓	↓	↓	↓	↓	↓
9	→	→	→	→	→	→	→	→	→	→

Observations and Analysis:
In this part of the project, since the reward function begets some negative value, it will have some role to play in the estimation and computation step. On the basis of previous results and also intuition, the agent will try to avoid the states which give a negative reward. This is depicted in the grid above where we can see that the arrows are diverging from the state which give a negative reward. For example, if we consider the same example of the 52nd state, then we can see that the arrow in the 52nd state is pointing down, i.e. towards the 53rd state. This is correct because the remaining 3 neighbouring states have a negative reward which should be avoided. These 3 states also point in a direction away from the enter i.e. the 52nd state. This is desirable, because if not there would have been a chance that the agent would have got locked in the 52nd state. In other states surrounding the states with reward -100, we observe that the arrows do not point to these negatively rewarding states. Again, this is desirable, because we do not want the agent to go to these states. We also observe in the bottom right part of the grid that almost all the arrows are leading to the 99th state. The agent will only move to the state which has the maximum optimum value among all the neighbouring states. Again, all this is intuitive as well as justified by our optimal values obtained previously. Thus, the actions of the agent as shown in this grid is validated and justified by the optimal values and the heat map obtained previously.

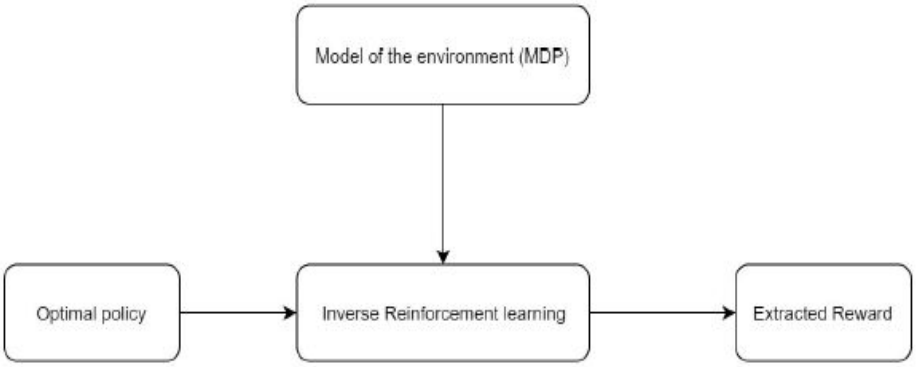
One of the problems of Reinforcement learning is the exploration vs exploitation dilemma.

- Exploitation: Make the best decision with the knowledge that we already know (ex: Choose the action with biggest Q-value)
- Exploration: Gather more information by doing different (stochastic) actions from known states.

Inverse Reinforcement learning (IRL)

Inverse reinforcement learning (IRL) is the process of deriving a reward function from observed optimal behavior of the expert. The motivation behind using inverse reinforcement learning is that in most of the cases the agent is not aware of the reward function in advance. Learning the reward function of the agent aids in designing more robust agents.

While ordinary "reinforcement learning" involves using rewards and punishments to learn behavior, in IRL the direction is reversed, and an agent observes a expert's behavior to figure out what goal that behavior is trying to achieve. In this part of the project, we use the optimal policy learnt in the previous section to extract reward function which is in turn used to compute optimal policy of the agent. In the coming questions, we will also compare the optimal policies of agent and expert.



Question 10: Express c, x, D in terms of R, P_a, P_{a1} , t_i, u, λ and R_{max}.

Answer: We want to choose R such that it makes optimal (by satisfying the constraints of the LP feasibility problem) and moreover we want to favor reward vectors R that make any single step-deviation from optimal policy as costly as possible.

We will use the LP formulation of the IRL as given in this equation:

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

We want to recast the above equation in this format:

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

Here, we consider the equation of the form $Dx \leq b$. The identity matrix has been denoted by I and the dimensions of each matrix has been mentioned as their subscript. After recasting, we get the below matrices:

c=

1	0	$-\lambda$

x=

T	R	U

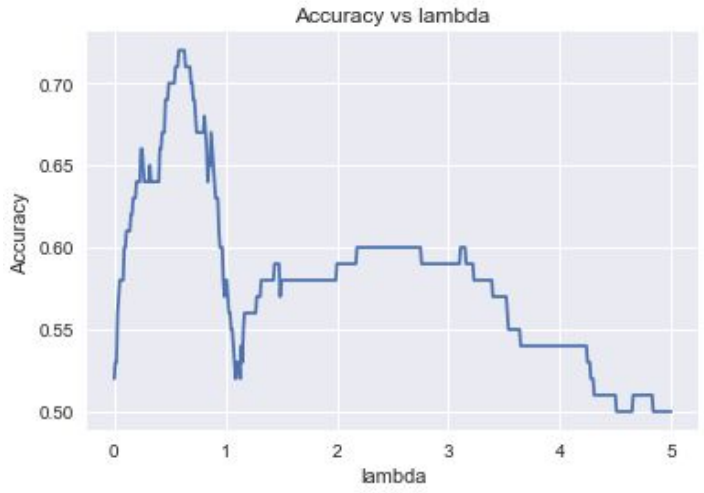
D =

$I_{300*100}$	$-[(Pa_1(i) - Pa(i))(I - \gamma\ Pa_1)^{-1}]_{300*100}$	$0_{300*100}$
$0_{300*100}$	$-[(Pa_1 - Pa))(I - \gamma\ Pa_1)^{-1}]_{300*100}$	$0_{300*100}$
$0_{100*100}$	$-I_{100*100}$	$-I_{100*100}$
$0_{100*100}$	$I_{100*100}$	$-I_{100*100}$
$0_{100*100}$	$I_{100*100}$	$0_{100*100}$
$0_{100*100}$	$-I_{100*100}$	$0_{100*100}$

b=

$0_{300*100}$
$0_{300*100}$
$0_{100*100}$
$0_{100*100}$
$(R_{max})_{100*100}$
$(R_{max})_{100*100}$

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



Answer:

Reward vectors with non-zero reward values at very few states are considered to be simpler. For favouring these simpler reward functions, we use L1 normalization to solve the equations mentioned earlier. λ is an adjustable penalty coefficient that promotes simpler reward vectors R.

λ needs to be tuned to achieve maximum accuracy. For this, we varied lambda from 0 to 5 in 0.01 steps, getting 500 λ values. For each of these values we calculated accuracy which can be seen in the plot above.

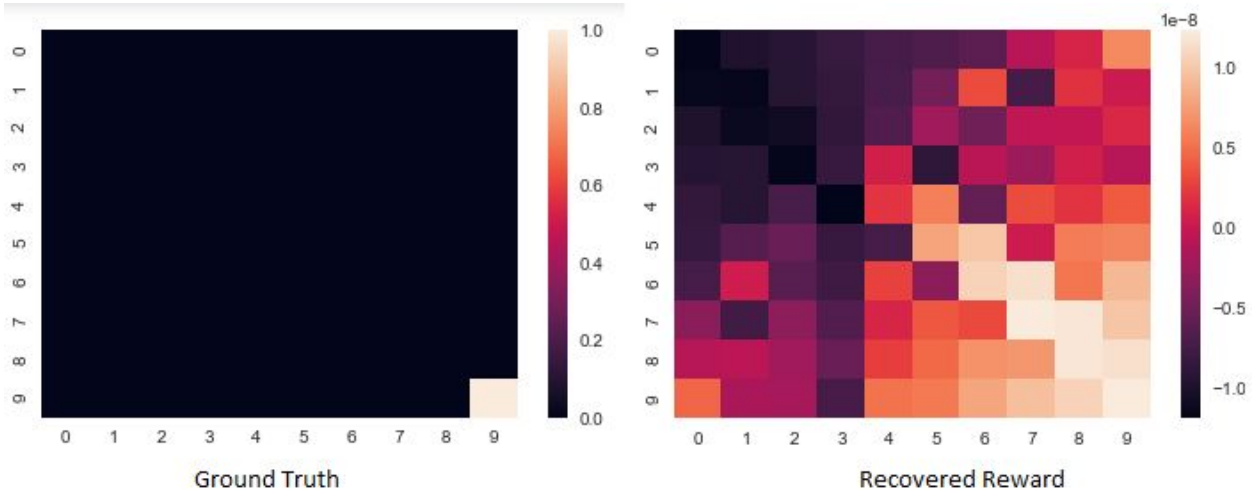
Initially when we increase lambda, the accuracy increases steeply, and reaches a optimal value. After this, further increase in lambda results in decrease in value of accuracy. Even further increase in lambda increases accuracy and achieves local maxima. But it fails to reach the global maxima attained earlier.

Question 12: 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 $\lambda_{max}^{(1)}$. Please report $\lambda_{max}^{(1)}$.

Answer: We sweep the value of λ from 0 to 5 in increments of 0.01 and compute accuracy for each value of λ. Upon observing these accuracies, it was observed that the maximum accuracy obtained is **0.72** and the corresponding λ_{max} is **0.58.**

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

Answer:



The heat map on the left represents the ground truth reward function 1 while the heat map on the right represents the recovered reward obtained by using linear programming.

Here we see that the ground truth reward function has black values for all states except the 99th state and has a reward value 0. A white state is shown for the 99th state that has the maximum reward value.

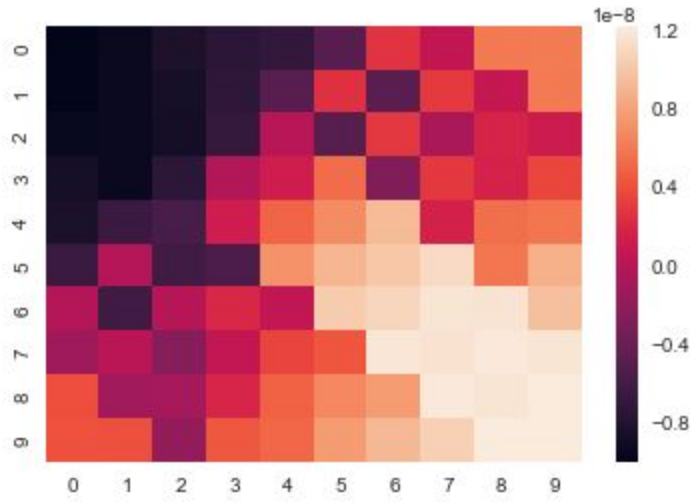
When we use Inverse Reinforcement learning, we forget about the ground truth reward functions and just use the expert's optimal policy that we derived using Reinforcement learning. We now learn the reward functions by formulating it as a Linear feasibility problem.

As we can see from the heat map in the right, the values close to the top left states (possibly where an agent might start his walk) has dark values. However, the states closer to the highest reward value of 1.0 has lighter shades.

While the ground truth reward function gives the agent many possible actions of reaching the destination, all being of the same colour. But for recovered reward functions, we see different shades from dark to bright colors, which gives the agent more information while choosing the optimal actions.

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

Answer:

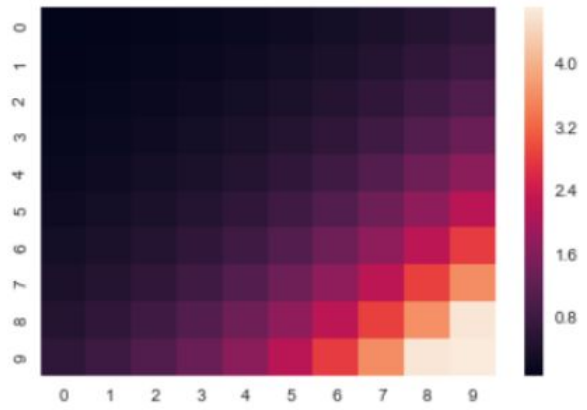


In the above figure , we have plot the heat map of optimal state values. We can see that similar to the reward function heat map in the previous question, there are dark values toward cells 0 and white values towards the cell 99. Also, we observe more dark and white cells as compared to the previous heat map of reward function only.

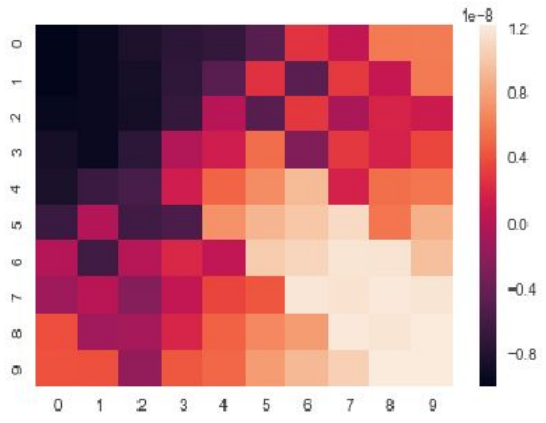
The optimal state value of a state is composed of many parameters like states (S), Actions, Transition Probabilities and Reward function values. Hence it includes the reward function as a component but inculcates more information than just the reward function heat map.

Although the reward value of a cell be 0 or negative, the optimal state value which includes other parameters might return a positive value. Hence we observe more whiter shades in the lower right cells in the above heat map. Also, this heat map is more informative than the previous heat map.

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



Question 3 (RF1 RL)



Question 14 (RF1 from IRL)

The heat map on the left represents the optimal state values computed using the ground truth reward function 1 (Question 3) while that on the right represents the optimal state values computed using the recovered reward (Question 14).

Similarities

- One of the most prominent similarity in both the heats maps is that the optimal value is highest for the 99th state. Hence that state and its neighboring ones are represented in lighter shades. This is pretty obvious and intuitive because both the ground truth and the recovered reward function have the highest reward for the 99th state. Hence its optimal state value ought to be the maximum and consecutively, the optimal state values of its neighbours should also be higher (i.e. close to maximum). Similarly, the optimal state values for the states in the top left section is lowest and hence, the heat map depicts that section in darker shades.
- In both the heat maps, the optimal state values gradually increase as the agent moves from top left portion of the grid to the bottom right portion of the grid. This is also because the reward and hence the optimal state value of the bottom right portion of the grid is the highest.

Differences

- One major difference between the two heat maps is that the optimal values in question 3 ranges from 0 to approximately 4.7, however in question 14, the optimal state values are all in the power of 10^-8 i.e. they are very close to 0. Additionally, they are not entirely positive; some values in the top left part of the grid are negative.
- This significant decrease of optimal state values in question 14 is because the recovered reward function from IRL itself has approximately the same range of values and that is reflected in the optimal state values of question 14.
- In question 13, we observe that the ground truth reward function and the recovered reward function are not very similar. This is because we have used a discount factor of 0.8. The more the discount factor is closer to 1, the more dissimilar the two functions will be. As a result of this, the optimal state values of both the questions are also different from each other.
- In question 14, the agent will get more rewards if it takes the path along the left diagonal since many lighter shade states (slightly larger optimal state values) are seen along this path. Whereas, in question 3, the optimal state values gradually increase from anywhere to the bottom-right corner of the grid.
- In question 3, the optimal state values monotonically increase from top left to bottom right. However, though the optimal state values in question 14 increase from top left to bottom right, they do not increase monotonically. This is because the reward function for question 3 has the same rewards for state 0 to 98 i.e. a reward of 0; while the reward function for question 14 does not have same reward for all the states.

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

Answer:

In this question, we use the extracted reward function to compute the optimal policy of the agent by using the value iteration algorithm implemented in the first part of the project. For better interpretation, we use this optimal policy to generate a tabular structure of optimal actions at each state as shown below.

	Actions for Optimal Policy of Recovered RF1									
	0	1	2	3	4	5	6	7	8	9
0	→	→	→	→	→	↓	→	→	→	→
1	↓	→	→	→	→	→	↓	→	↑	↑
2	↓	↓	→	→	↓	→	↑	→	↑	↑
3	↓	↓	→	→	↓	↓	↓	↓	↓	↓
4	↓	↓	↓	→	→	↓	↓	↓	↓	↓
5	→	↓	↓	→	→	→	↓	↓	↓	↓
6	↓	→	←	→	→	→	→	↓	↓	↓
7	↓	↓	↓	→	→	→	→	→	←	←
8	↓	←	←	→	→	→	→	↑	↑	↓
9	←	←	←	→	→	→	→	→	→	→

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

Answer: The table on the left (below) represents the optimal policy of the expert (Question 5) while that on the right represents the optimal policy of the agent(Question 16).

	Actions for optimal policy for RF1									
	0	1	2	3	4	5	6	7	8	9
0	→	→	→	→	→	→	→	→	↓	↓
1	↓	→	→	→	→	→	↓	↓	↓	↓
2	↓	↓	→	→	→	↓	↓	↓	↓	↓
3	↓	↓	↓	→	↓	↓	↓	↓	↓	↓
4	↓	↓	↓	→	→	↓	↓	↓	↓	↓
5	↓	↓	→	→	→	→	↓	↓	↓	↓
6	↓	→	→	→	→	→	→	↓	↓	↓
7	↓	→	→	→	→	→	→	→	↓	↓
8	→	→	→	→	→	→	→	→	→	↓
9	→	→	→	→	→	→	→	→	→	→

	Actions for Optimal Policy of Recovered RF1									
	0	1	2	3	4	5	6	7	8	9
0	→	→	→	→	→	↓	→	→	→	→
1	↓	→	→	→	→	→	↓	→	↑	↑
2	↓	↓	→	→	↓	→	↑	→	↑	↑
3	↓	↓	→	→	↓	↓	↓	↓	↓	↓
4	↓	↓	↓	→	→	↓	↓	↓	↓	↓
5	→	↓	↓	→	→	→	↓	↓	↓	↓
6	↓	→	←	→	→	→	→	↓	↓	↓
7	↓	↓	↓	→	→	→	→	→	←	←
8	↓	←	←	→	→	→	→	↑	↑	↓
9	←	←	←	→	→	→	→	→	→	→

Similarities

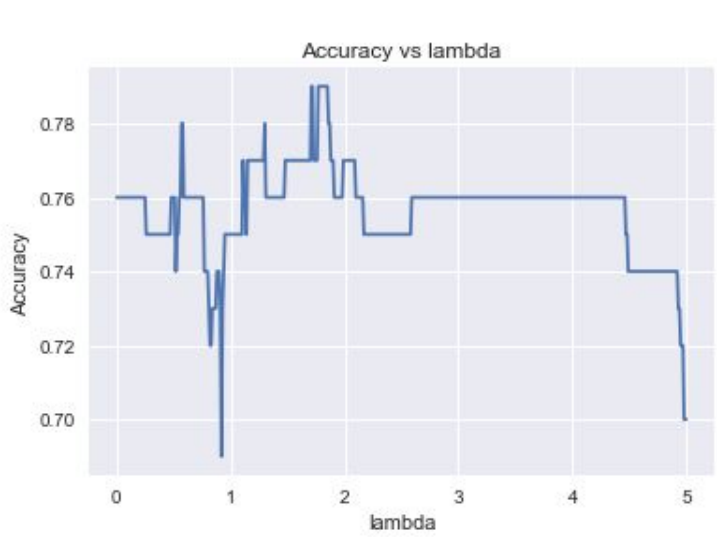
- The overall structure of both the optimal policies seems similar with the majority of actions involving going down or right. For both the optimal policies, the optimal value at the 99th state is maximum compared to all the other states. The optimal value is 4.70 in case of reward function 1 and the corresponding optimal value for the recovered reward function is 1.21e-08. Both of these values are the largest in their corresponding grids. This is because the reward function values are largest for the 99th state.

Differences

- One major difference between the two optimal policies is that in case of the optimal policy of the expert, all the arrows are going either down or right while for optimal policy of agent, the arrows can be seen pointing in all four directions. This is due to the fact that the optimal state values for reward function 1 are monotonically increasing while that for recovered reward function does not exhibit any such pattern. This is also evident from the fact that the arrows are pointing in all four directions.
- In case of optimal policy of the agent, there is presence of divergence i.e. the arrows in neighboring states are pointing away from each other. Moreover, in some cases, the directions of arrows are going off the grid. However, this behavior is not observed for optimal policy of the expert because the reward function values are same for all states except the last state. So, the agent tries to go either down or right from any randomly selected state.
- For the optimal policy of the agent, we can observe that states 16 and 26 are both pointing to each other. That means when the agent arrives at any of these states, it will simply keep on oscillating between the two states. This is nothing but the deadlock condition that causes the agent to get stuck once it enters either of those states. Additionally, there is a deadlock between states 61 and 62, and between states 77 and 87. The reason for this is because the optimal state values are influenced by that of the neighboring states. Intuitively, this is bound to happen if the optimal state values of the neighboring states differ by a very very small amount. In such a case, both will want to go to each other since that will be the best optimal value. This kind of situation is never encountered for optimal policy of the expert.
- When we observe the direction of arrows for optimal policy of the expert, we can see that there exists a path even when the agent starts from any random state. On the contrast, the optimal policy of the agent does not guarantee a path from a random state to the 99th state. The path exists only for certain states.

Question 18: Sweep λ from 0 to 5 to get 500 evenly spaced values for λ. For each value of 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 to get 500 data points. Plot (x-axis) against Accuracy (y-axis). In this question, you should have 1 plot.

Answer:



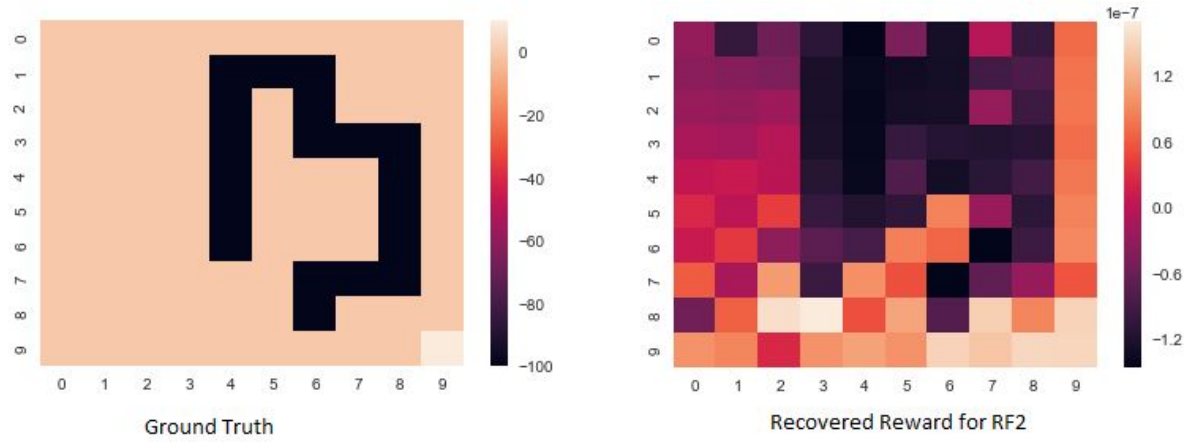
Reward vectors with non-zero reward values at very few states are considered to be simpler. For favouring these simpler reward functions, we use L1 regularization to solve the equations mentioned earlier.Here, λ is an adjustable penalty coefficient that balances between the goals of having small reinforcements and maximizing the sum of differences between quality of optimal action and the next best action.

When we vary λ from 0 to 5 in the increments of 0.01 and calculate the accuracy for each value of λ, we obtain a plot as shown above. It can be observed from the plot that initially when we increase λ, the accuracy actually decreases until it reaches a minima. After this, eventual increase in λ causes an improvement in accuracy. This point onwards, the accuracy increases with λ and finally reaches its peak value at λ =1.71 . Any increase in λ after this leads to degradation in accuracy.

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

Answer: We sweep the value of λ from 0 to 5 in increments of 0.01 and compute accuracy for each value of λ. Upon observing these accuracies, it was observed that the maximum accuracy obtained is **0.79** and the corresponding λ_{max} is **1.71.** When we compare this accuracy with the one obtained for reward function 1, it is quite clear the better accuracy is obtained for reward function 2. This is due to the fact that reward function is more informative than reward function 1.

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



Answer:

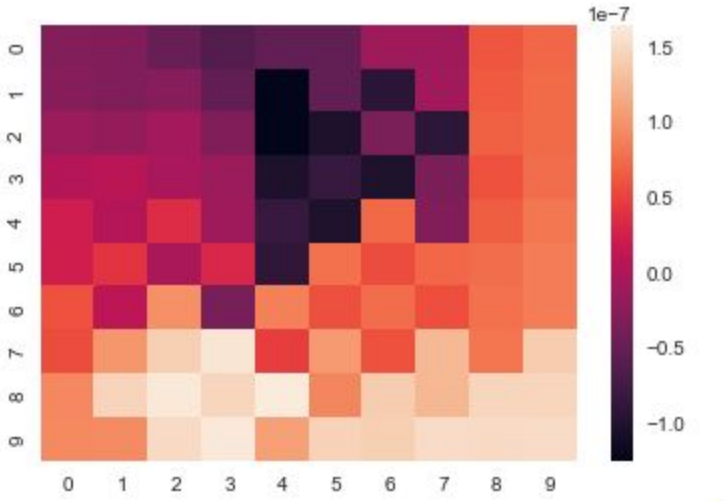
The heat map on the left represents the ground truth reward function 2 while the heat map on the right represents the recovered reward obtained by using linear programming.

Here we see that the ground truth reward function has black values for the states that have a reward value -100. A white state is shown for the 99th state that has the maximum reward value. The rest of the states having a zero reward are shown by the intermediate lighter shade.

When we use Inverse Reinforcement learning, we ignore the original reward functions and just use the expert's optimal policy derived using Reinforcement learning. We then learn the expert's reward functions by formulating it as a Linear feasibility problem.

The heat map in the right contains a wide range of colors which are an indication of variety of reward values at each state. Even in this heat map, the corresponding states that had negative reward values in the left heat map are shown by the darker shades and hence, the agent is still penalized heavily upon transitioning to those states. It can be also be seen that the 38th state is the brightest and has the largest reward value. This is contrary to our intuition that the 99th state should be possessing the largest reward. This is the reason that we don't rely solely on the reward function but we also consider the transition probabilities and the optimal values for neighboring states in order to select the next best action.

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



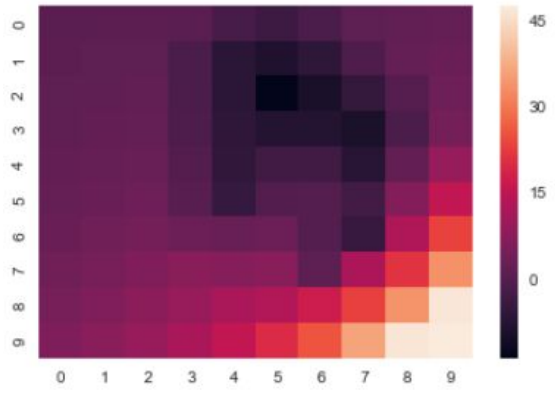
Answer:

In the above figure , we have plot the heat map of optimal state values. We can see that there are darker colors in the states that had negative values for ground truth reward. The states in the top left portion are more along the darker shades and also possess negative values for rewards. On the contrast, the states along the bottom right section are lighter and possess positive reward values indicating their desirability.

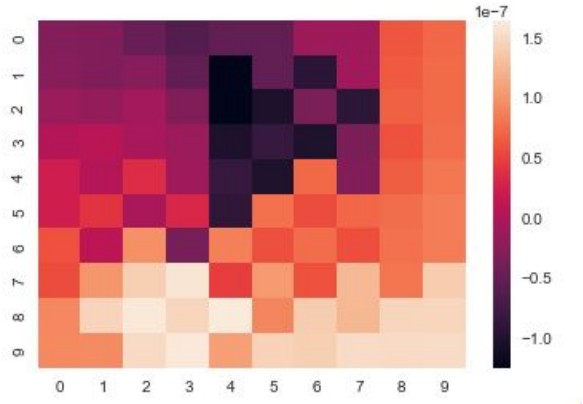
The optimal state value of a state is composed of many parameters like states, Actions, Transition Probabilities and Reward function values. Hence it includes the reward function as a component but inculcates more information than just the reward function heat map. This is why the heat map of optimal values does not completely resemble the heat map of reward function values.

It can be observed that although the range of values is similar in both the heat maps, this heat map is more informative than the previous heat map for extracted reward function.

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



Question 7



Question 21

Similarities:

- The most visible similarity in the above two heat maps is that the optimal value is the least in the upper middle area of the grid. This is because both the reward functions exhibit the least reward in this region. So the agent does not ideally want to go to these states and hence they have a very low optimal state value.

Differences:

- The most visible difference between the two heat maps is that in case of Question 7, the highest optimal state value is observed for the 99th state however for question 21, the highest optimal state value is observed for the 48th state. This is primarily because we see from question 20 that the recovered reward of getting to the 38th state (and not the 99th) is the maximum. So why is the 48th state having the most maximal optimum value and not the 38th state? This is because the optimal value of a state is calculated by taking into consideration the optimal values of its neighbors. Since the rank of state 38 is the highest, the agent will try to get to that state, however, the optimal value of 38 will now be the highest among all of the neighbours of state 48. Thus, for s=48, V(s') will now be the optimal value of state 38. To calculate V(s) for state 48, the reward of 48 as well as its transition probability will be considered in addition to V(s'). All this together will make the optimal value of state 48 more than that of state 38. However, this will be a very small difference. This can be verified from the heat map of question 21 plotted above.
- As mentioned in question 15, we observe the same difference here too i.e. the optimal values in question 7 ranges from -9.6 to approximately 47, however in question 21, the optimal state values are all in the power of 10^-8 i.e. they are very close to 0. Additionally, they are not entirely positive. Again, this is because of the difference in the reward functions of both the questions which is as a result of using a discounted factor of 0.8. The more the discounted factor is closer to 1, the lesser will the two reward functions be similar (and consecutively the optimal state values also).
- In some places along the edge/corners of question 21, we observe that the shade is slightly lighter than its neighbors. This means the probability of staying along that particular state on the edge or corner is more. Which means that in such cases it is possible for the agent to go out of the grid. This can also be verified in question 23 below. However, this is not observed anywhere in question 7; which means that the agent always tries to remain within the grid.

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

Answer:

In this question, we use the extracted reward function and compute the optimal policy of the agent by using the value iteration algorithm implemented in the first part of the project. For better interpretation, we use this optimal policy to generate a tabular structure of optimal actions at each state as shown below.

Actions for optimal policy for RF2										
	0	1	2	3	4	5	6	7	8	9
0	←	←	↓	←	→	↑	→	↑	→	↓
1	↓	↓	↓	←	←	↑	→	↑	→	↓
2	↓	↓	↓	←	←	→	→	→	→	→
3	↓	↓	↓	←	←	↓	↓	→	→	↓
4	↓	↓	↓	←	←	↓	↓	↓	→	↓
5	↓	↓	↓	←	↓	↓	↓	←	→	↓
6	↓	↓	↓	←	↓	↓	←	←	→	↓
7	←	→	↓	↓	↓	↓	←	↓	↓	↓
8	→	→	→	←	←	↓	↓	↓	↓	↓
9	←	↑	↑	↑	←	→	↓	→	→	→

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

Answer: The table on the left (below) represents the optimal policy of the expert (Question 9) while that on the right represents the optimal policy of the agent(Question 23).

Actions for optimal policy for RF2										
	0	1	2	3	4	5	6	7	8	9
0	↓	↓	↓	←	←	→	→	→	→	↓
1	↓	↓	↓	←	←	↑	→	→	→	↓
2	↓	↓	↓	←	←	↓	→	→	→	↓
3	↓	↓	↓	←	←	↓	↓	↑	→	↓
4	↓	↓	↓	←	←	↓	↓	↓	→	↓
5	↓	↓	↓	←	←	↓	↓	←	→	↓
6	↓	↓	↓	↓	↓	↓	←	←	→	↓
7	↓	↓	↓	↓	↓	↓	←	↓	↓	↓
8	→	→	→	↓	↓	↓	↓	↓	↓	↓
9	→	→	→	→	→	→	→	→	→	→

Actions for optimal policy for RF2										
	0	1	2	3	4	5	6	7	8	9
0	←	←	↓	←	→	↑	→	↑	→	↓
1	↓	↓	↓	←	←	↑	→	↑	→	↓
2	↓	↓	↓	←	←	→	→	→	→	→
3	↓	↓	↓	←	←	↓	↓	→	→	↓
4	↓	↓	↓	←	←	↓	↓	↓	→	↓
5	↓	↓	↓	←	↓	↓	↓	←	→	↓
6	↓	↓	↓	←	↓	↓	←	←	→	↓
7	←	→	↓	↓	↓	↓	←	↓	↓	↓
8	→	→	→	←	←	↓	↓	↓	↓	↓
9	←	↑	↑	↑	←	→	↓	→	→	→

Similarities

- The overall structure of both the optimal policies seems similar with the arrows pointing in all the four directions. This is in contrast to the optimal policy of the expert for ground truth reward function 1; in which case the arrows pointed either down or right.
- Moreover, for the states that have negative values in ground truth reward function, the arrows are pointing in directions away from those states in trying to prevent the agent from entering those states. For both the optimal policies, the optimal value at the 99th state is maximum compared to all the other states. This is because the reward function values are largest for the 99th state.
- In case of both the optimal policy of the agent and the optimal policy of the expert, there is presence of divergence i.e. the arrows in neighboring states are pointing away from each other.

Differences

- For the optimal policy of the agent, in some cases, the directions of arrows are going off the grid. However, this behavior is not observed for optimal policy of the expert because the reward

function values are not in a wide range and are either 0 for majority of states or negative for certain undesirable states.

- In the optimal policy of the agent, we can observe that states 28 and 38 both point at each other. This is nothing but the deadlock happening which causes the agent to get stuck once it enters either of these states. This is because the reward function value at the 38th state is higher than that of 28th state. So when the agent reaches the 28th state, it selects the 38th state as the next best action and assigns right as the optimal action. On the contrary, the optimal state value at the 28th state is higher. So, when the agent reaches 38th state, it picks 28th state as the next best action and assigns left movement as the optimal action. This causes the agent to get stuck when it transitions to either of these states. However, there is no such deadlock issue observed in case of optimal policy of the expert.
- There is another interesting phenomenon happening in the case of optimal policy of the agent. If we observe the states 18, 27, 29 and 38, we see that all of these states point to the 28th state. That is a convergence is happening at state 28. The same phenomena is observed at the 38th state. When we compare the optimal state values of states 28 and 38, we see that these values are greater than the optimal state value of 99th state. Therefore, it is possible that if the agent reaches any of the neighbouring states, it will converge to either the 28th or 38th state; which is not desirable. Such kind of convergence is also possible if there is a local maxima which need not be higher than the optimal state value of the desired state (i.e. the state with the highest reward). Such kind of a behaviour is not observed in the optimal policy of the expert.
- When we observe the direction of arrows for optimal policy of the expert, we can see that there exists a path even when the agent starts from any random state. On the contrast, the optimal policy of the agent does not guarantee a path from a random state to the 99th state. The path exists only for certain states.

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

Answer:

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

We found the following discrepancies in the optimal policy of agent:

1. The agent tries to jump out of the box. The states highlighted with yellow in the above figure shows the point where agent tries to go out of the grid.
2. There is deadlock of action between two neighbors . The states marked with red color show a deadlock between two states.
3. From the figure below, we can notice that there exists convergence of actions at a state which is not a final desirable state. States marked with green converge at a point which is not optimal. We can conclude that the algorithm has led to formation of a local maxima and all the neighboring states are pointing to that particular local maxima rather than going to the desirable state ,i.e. state 99.

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

4. We can observe that for most of the states, there is no path to the final destination. In the RL algorithm, there were paths from every state to the final (99) state, so the end state was reachable. But in this case, we can observe that reachability to the final state is not achieved. Some paths lead to deadlock whereas some paths causes the agent to go off the grid.

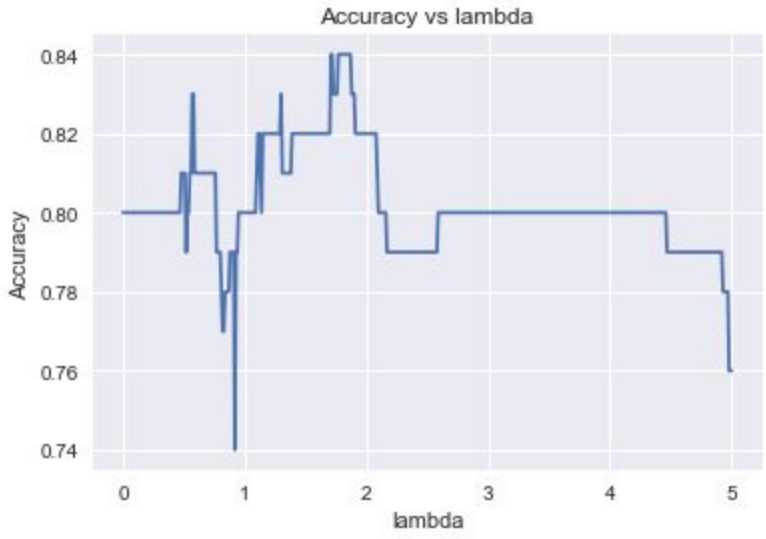
Our proposed solution:

To solve the first discrepancy(agent going off the grid), we modified the value iteration algorithm and made changes to avoid the agent from going off the grid. Suppose the agent is at the top row, then we

made changes so that the agent will choose max(left,right,down). Thus, we are restricting the set of actions depending on the current state of the agent. The action grid after making this change is as follows:

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

From the above figure, we can observe that the agent is not going off the grid. After implementing the changes to deal with this discrepancy, the maximum accuracy increased from 0.79 to 0.84 . We feel that this is because we are restricting the actions of the agent and not letting the agent go out of grid.



The following modification (consider only best valid) is made to the value iteration algorithm as a solution to this problem :

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

Here, instead of considering the maximum value in step 15, we first check whether the action corresponding maximum value is valid i.e. , does not lead to the agent falling off the grid which would have resulted in unreachability of the goal.

Also, the solution to the deadlock and convergence problem is hard to solve. So, we leave it as future scope.

Conclusion

In this project, we observed a few points due to the Linear Reinforcement algorithm used. Based on our observations we conclude the following:

1. MDPs make it easier to model the learning problem for the agent. MDPs assume that the complete state of the world is visible to the agent.However, this is clearly highly unrealistic (think of a robot in a room with enclosing walls: it cannot see the state of the world outside of the room).
2. The Value Iteration Algorithm for computing the optimal policy and optimal state values is highly dependent on the discount factor γ.
3. In Inverse Reinforcement Learning, we observe the behaviour of the expert and then extract the experts reward function. In this project we observed that the recovered reward function might be very different from the ground truth reward function, and may contain discrepancies like deadlock, convergence, falling-off the grid etc.
4. More sophisticated algorithms for both Reinforcement (e.g. Deep Reinforcement Learning, Q-Learning, etc.) and Inverse Reinforcement learning (e.g. Deep Maximum Entropy, Generative

Adversarial Networks etc.) exists which can be leveraged to tackle the discrepancies observed above.

References

[1] Markov Decision Processes and Reinforcement Learning https://danieltakeshi.github.io

[2] Kengo Katayama ,Takahiro Koshiishi, Hiroyuki Narihisa, Reinforcement Learning Agents with Primary Knowledge Designed by Analytic Hierarchy Process

[3] https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/reinforcement_learning.html

[4] https://en.wikipedia.org/wiki/Apprenticeship_learning#Via_inverse_reinforcement_learning

[5] Going Deeper Into Reinforcement Learning: Understanding Q-Learning and Linear Function Approximation. https://danieltakeshi.github.io