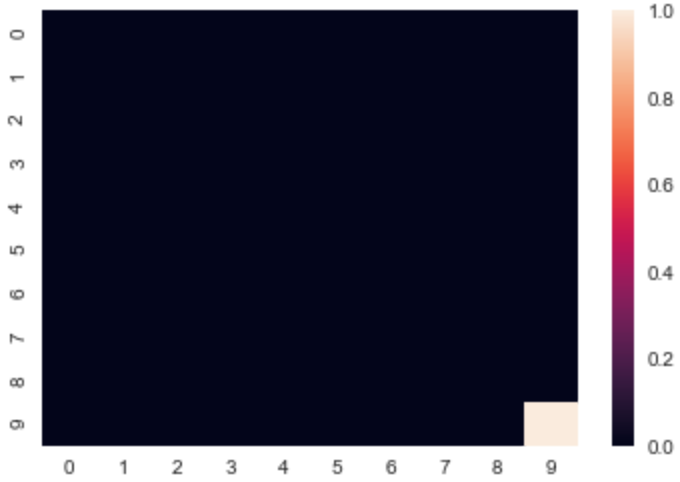
EE232 Project 3
Social Network Mining

Team members:
Sonali Garg (104944076)
Shweta Sood(905029230)
Karan Sanwal(205028682)
Ashish Shah(804946005)

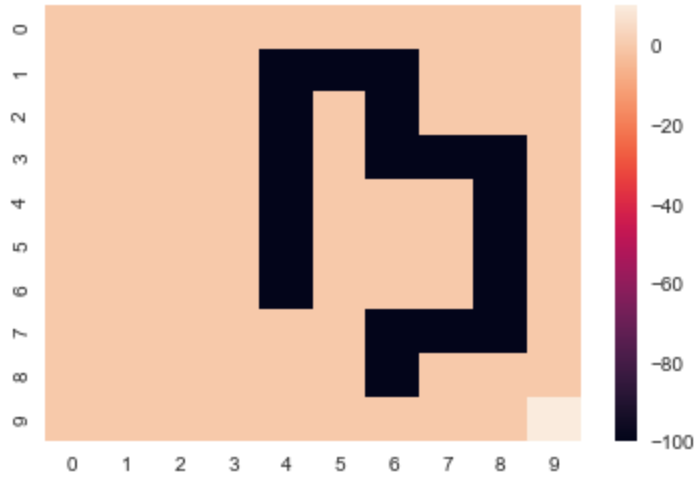
Question 1: (10 points) For visualization purpose, generate heat maps of Reward function 1 and Reward function 2. For the heat maps, make sure you display the coloring scale. You will have 2 plots for this question

Reward Function 1: The heat map visualization of the reward function shows the reward values on a gradient. It clearly reflects the provided reward function has the highest value on the bottom right corner and everything else has equal intensity.



	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 2: Just as reward function 1, the heat map visualization of reward function 2 reflects the function. We can see the intensity of states with the smallest reward value is the darkest, whereas that of the highest reward value is the brightest. The bottom right corner is the brightest again as it has the highest reward value.



	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

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 E = 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.

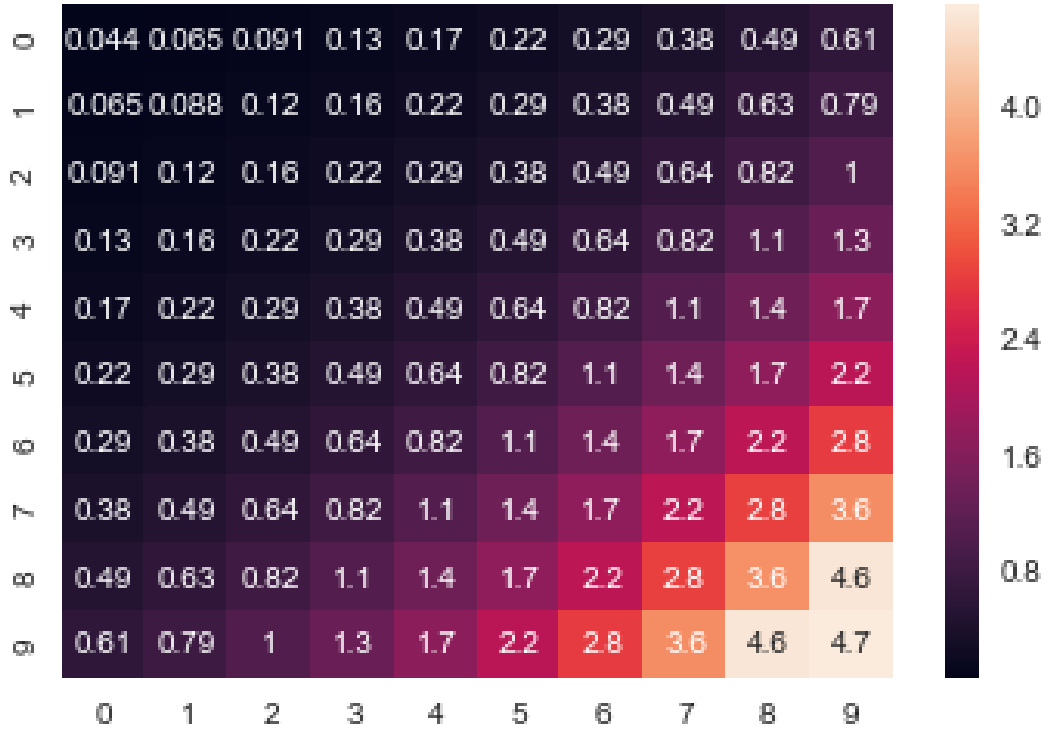
An MDP was set up with the aforementioned parameters. Optimal state value function was created optimizing over the equation:

$$V(s) \leftarrow \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} \mathcal{P}^{a}_{ss'} [\mathcal{R}^{a}_{ss'} + \gamma V(s')]$$

From the above expression, we can see that the action-value function is the expected discounted return for taking action a in state s and thereafter following policy π. This value is a measure of goodness of taking action a in state s.
The goal of the agent is to choose action a such that the expected return is maximized.

	0	1	2	3	4	5	6	7	8	9
0	0.044240	0.064574	0.091184	0.125041	0.168078	0.222693	0.291988	0.379880	0.491349	0.610011
1	0.064574	0.088334	0.121853	0.164761	0.219453	0.289178	0.378043	0.491228	0.633384	0.787519
2	0.091184	0.121853	0.164590	0.219304	0.289070	0.377989	0.491317	0.635651	0.817516	1.018758
3	0.125041	0.164761	0.219304	0.289065	0.377987	0.491320	0.635761	0.819770	1.052352	1.315219
4	0.168078	0.219453	0.289070	0.377987	0.491320	0.635764	0.819856	1.054426	1.351732	1.695233
5	0.222693	0.289178	0.377989	0.491320	0.635764	0.819857	1.054482	1.353485	1.733341	2.182392
6	0.291988	0.378043	0.491317	0.635761	0.819856	1.054482	1.353511	1.734622	2.219687	2.806963
7	0.379880	0.491228	0.635651	0.819770	1.054426	1.353485	1.734622	2.220362	2.839418	3.607787
8	0.491349	0.633384	0.817516	1.052352	1.351732	1.733341	2.219687	2.839418	3.628974	4.634708
9	0.610011	0.787519	1.018758	1.315219	1.695233	2.182392	2.806963	3.607787	4.634708	4.701701

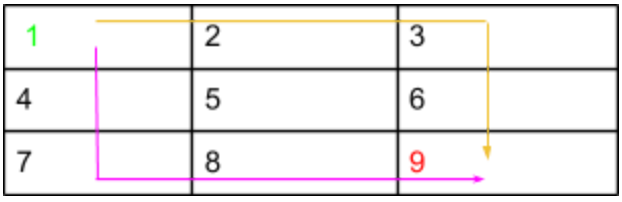
Question 3: (5 points) Generate a heat map of the optimal state values across the 2-D grid. For generating the heat map, you can use the same function provided in the hint earlier (see the hint after question 1).



We can infer from the heat map that the values converge to the optimal state. The intensity is lowest farthest away from the goal state and slowly intensity increases as we reach the optimal goal state. This implies that the heat map reflects the optimal policy

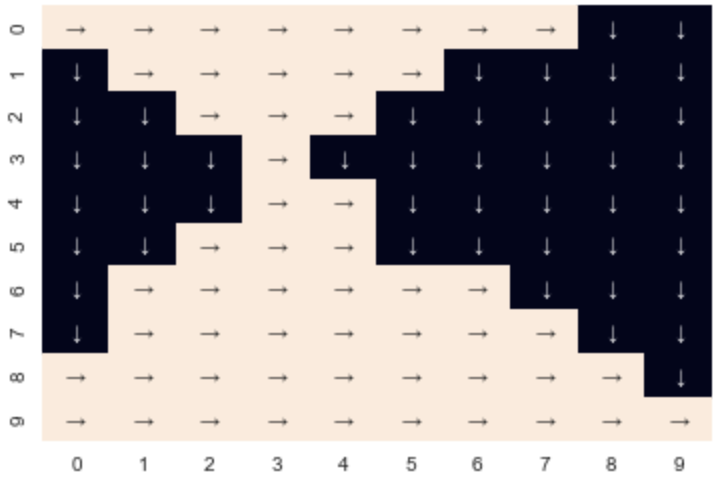
Question 4: (15 points) Explain the distribution of the optimal state values across the 2-D grid. (Hint: Use the figure generated in question 3 to explain)

The optimal state values across the 2-D grid are symmetric along the diagonal of the matrix. Also, these values increase from top left (farthest from goal state) to bottom right (goal state). This is expected as the reward of the goal state is maximum and the reward value of all states is the same. Thus, from every point in the matrix path, we can trace two paths of equal length to reach the goal state. This makes the optimal value matrix symmetric.



For example in the above square matrix, there are 2 equal paths from state 1 to state 9, highlighted in yellow and pink. Thus values will be same for corresponding states along the two paths.

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 its neighboring states? In this question, you should have 1 plot.



The optimal action represented in the figure above matches intuition. Since the actions from every non goal state are such that they form a path to the goal state, or in other words, converge to the goal state, the optimal action makes sense. Here, the only actions are bottom and right; as the top and left actions would move the agent away from the goal state. Moreover, all the states have equal reward value, thus, the agent doesn't have to avoid any state as the utility is same.

For the given reward function, It is possible for the agent to compute the optimal action at each state by observing the optimal values of its neighboring states. This is because the reward function has equal values for all the states except the goal state.
However, in general circumstances where we can have any reward function and transition probability matrix, this should not be true.

Value iteration includes: finding optimal value function + one policy extraction. There is no repeat of the two because once the value function is optimal, then the policy out of it should also be optimal (i.e. converged). Every iteration updates both the values and (implicitly) the policy
We don't track the policy, but taking the max over actions implicitly recomputes it.

Policy iteration includes: policy evaluation + policy improvement, and the two are repeated

iteratively until policy converges. We do several passes that update utilities with fixed policy. After the policy is evaluated, a new policy is chosen. The new policy will be better (or we're done).
Thus to recompute a new policy/ improve over the policy we need to find the argmax over actions based on the following equation:

$$\pi(s) \leftarrow \arg\max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} \mathcal{P}^{a}_{ss'} [\mathcal{R}^{a}_{ss'} + \gamma V(s')]$$

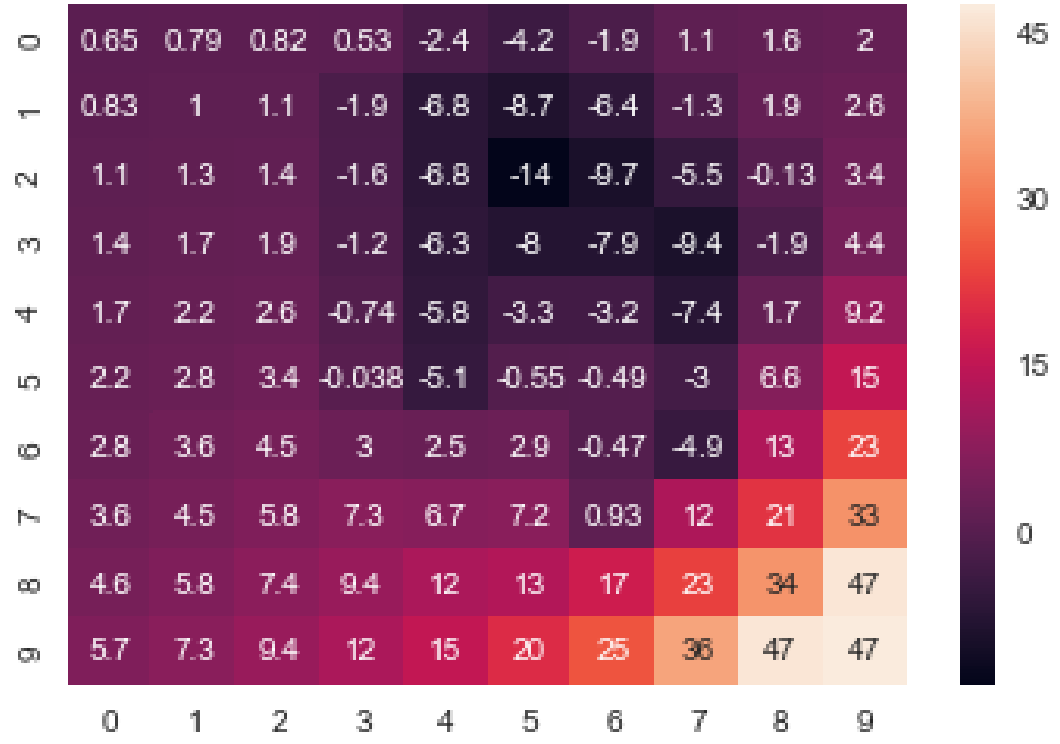
Thus, we see we need not just the optimal values of the neighboring states, but also probability transition for neighboring states and reward values.

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.

	0	1	2	3	4	5	6	7	8	9
0	0.646710	0.790798	0.820813	0.525100	-2.386476	-4.236930	-1.923374	1.128097	1.591162	2.034827
1	0.827724	1.017733	1.061580	-1.879217	-6.754669	-8.683714	-6.373483	-1.298361	1.924775	2.606893
2	1.061312	1.313001	1.445788	-1.635211	-6.757757	-13.916635	-9.653202	-5.514815	-0.134600	3.355478
3	1.357786	1.689182	1.943907	-1.243217	-6.339216	-7.982771	-7.947292	-9.434452	-1.918155	4.387045
4	1.733934	2.168079	2.585895	-0.736490	-5.846727	-3.258403	-3.241071	-7.434498	1.715158	9.159524
5	2.211119	2.777553	3.413345	-0.038140	-5.114112	-0.553380	-0.487536	-2.983515	6.582692	15.353757
6	2.816429	3.552963	4.478824	3.024358	2.480209	2.880151	-0.465523	-4.910549	12.688464	23.296389
7	3.584202	4.539186	5.792569	7.288420	6.718773	7.241102	0.930890	12.366437	21.159165	33.482577
8	4.557967	5.794705	7.397203	9.439455	12.008201	12.889183	17.097345	23.013970	33.778251	46.528791
9	5.726633	7.316079	9.387596	12.044687	15.452355	19.823974	25.497506	36.157594	46.583383	47.311471

Having replaced reward function 1 by reward function 2, the optimal value of the states reflect the changes as per the values of the new reward function. We notice that states (and their neighbors) that are negatively penalized in reward function have large negative values as opposed to other states. This means, we want to discourage the agent from exploring paths that goes via these states.

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.

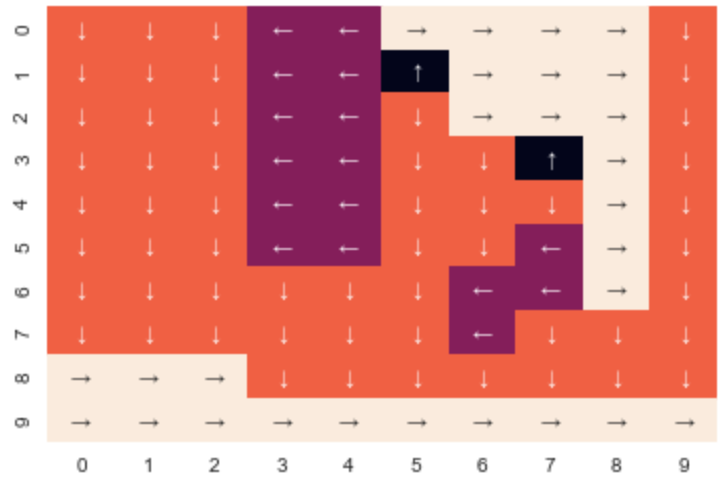


We can see the heat map reflects the optimal state values. The intensity is brighter for states closer to the goal state as opposed to the ones farther. Also, the states with negative rewards are much darker intensity compared to other states as we don't want the agent to explore a path via these states. As we move towards the goal state the intensity grows brighter.

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)

Looking at the distribution of the optimal state values across the 2-D grid. We can see the optimal values are highest for the goal state, lowest for the negatively penalized states. The values increases as we move towards the goal state. However, while exploring if we come close to negatively penalized states, the optimal value becomes negative as expected and starts increasing as we move away from there. Thus, the value iteration algorithm not only helps in reaching the goal state but helps in finding the optimal path towards the goal state based on what states are more economical to explore.

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.



Yes the optimal policy of the agent matches the intuition. We can see all the states have a path leading to the goal state (bottom right). Also, here we can see the actions are top, left along with bottom, right. This is around the negatively penalized states as once the agent is around those states, we would want the agent to move away from there using a top/ left action. For the other states, we have bottom and right actions to move towards the bottom right goal state.

Question 10: (10 points) Express c, x, D in terms of R, Pa, Pa1 , ti , u, λ and Rmax

Solution:

$$x = \begin{bmatrix} \mathbf{R} \\ \mathbf{t} \\ \mathbf{u} \\ 1 \end{bmatrix}$$

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

$$D = \begin{bmatrix} -(\mathbf{P}_{a1} - \mathbf{P}_a)(\mathbf{I} - \gamma \mathbf{P}_{a1})^{-1} & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ -(\mathbf{P}_{a1} - \mathbf{P}_a)(\mathbf{I} - \gamma \mathbf{P}_{a1})^{-1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{I} & \mathbf{0} & -\mathbf{I} & \mathbf{0} \\ -\mathbf{I} & \mathbf{0} & -\mathbf{I} & \mathbf{0} \\ \mathbf{I} & \mathbf{0} & \mathbf{0} & -Rmax\mathbf{1} \\ -\mathbf{I} & \mathbf{0} & \mathbf{0} & -Rmax\mathbf{1} \end{bmatrix}$$

Observation:
In our project, given 100 states and 4 actions:

In matrix x, R, u and t are vectors, with 100 rows each while 1 is a constant. Hence, x is 301 X 1 matrix.
In matrix c, the only the last 0 is scalar. The first 0 represents a vector, with 100 rows, each 0. Then 1 represents a vector, with 100 rows, each 1. Then the scalar -λ is multiplied by this 1 vector to get another 100 rows. Hence c is a matrix of dimensions 301 X 1.

In matrix D, Rmax is scalar, that is multiplied by the vector 1 of 100 rows and -1, to get 100 rows, each -Rmax.
The terms $-(P_{a1} - P_a)(I - \gamma P_{a1})^{-1}$ each represents a matrix 300 rows and 100 columns, and so do the corresponding 0 and I (except for the last column). This is because there are 100 states and we do this for each action except the optimal action(a1) i.e 3 other actions.
The remaining I and 0, except for the last column are 100 X 100 matrices.
Each 0s in the last column have only 1 column and number of rows is 300, 300,100 and 100 respectively.
So finally D is a 1000 X 301 matrix

Question 11: (30 points) Sweep λ from 0 to 5 to get 500 evenly spaced values for λ. For each value of λ compute OA(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 OE(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.

Solution:

Case1 :If we go right in case of equal probabilities of right and down:

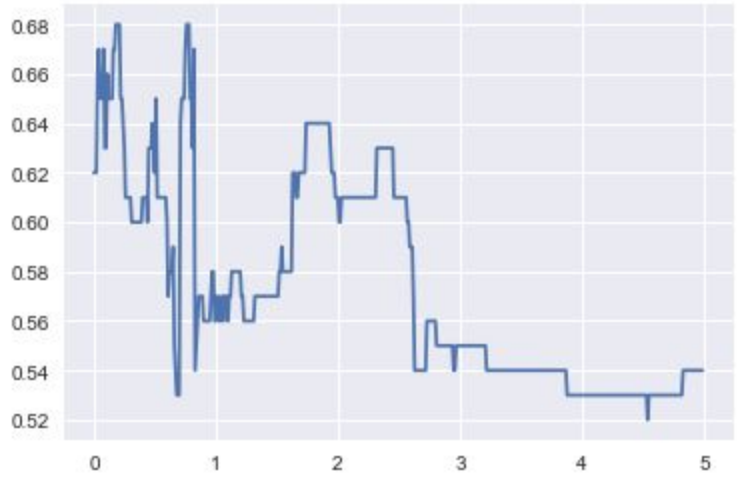


Fig: Plot of λ (X-axis) vs accuracy

Case2: If we go down in case of equal probabilities of right and down:

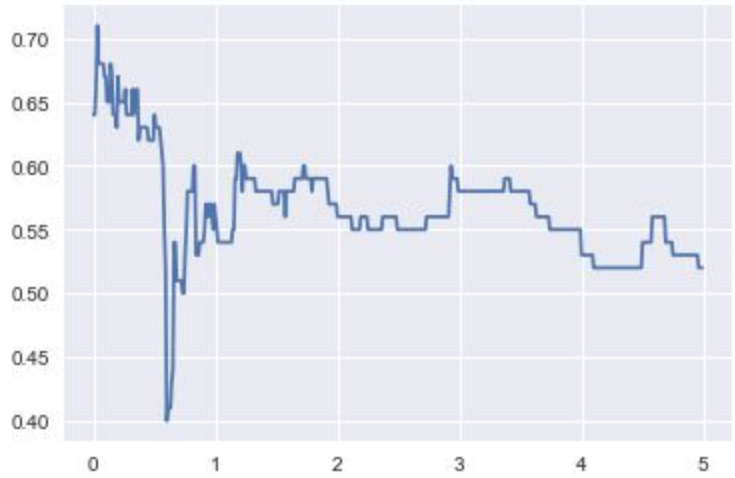


Fig: Plot of λ (X-axis) vs accuracy

Observation:
We shall be providing explanations for Case 1 only.
We sweep λ from 0 to 5 to get 500 evenly spaced values for λ. For each value of λ, we compute OA(s) by following the process describe and plot λ (x-axis) against Accuracy (y-axis).

We observe the accuracy ranges from 0.52 to 0.68 and that the peak value is reached fairly early(i.e. for a small λ). This could possibly be implying that not much regularization is needed(as possibly not much overfitting to begin with). This makes sense as ground truth reward function 1 and hence the optimum policy for it are simple and hence, not much overfitting is present(Basically, regularization leads to preferring simpler rewards and here, the reward is possibly simple to begin with).

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 λ (1) max. Please report λ (1) max

Solution:
Case1 :If we go right in case of equal probabilities of right and down:
max accuracy is:
0.68
best lambda is:
0.18

Case2: If we go down in case of equal probabilities of right and down:
max accuracy is:
0.71
best lambda is:
0.03

Observation:
We shall be providing explanations for Case 1 only.
We use the plot in question 11 to compute the value of λ for which accuracy is maximum. We find this value to be 0.18. As explained previously, the peak value is reached fairly early(i.e. for a small λ). This could possibly be implying that not much regularization is needed(as possibly not much overfitting to begin with).

Question 13: (15 points) For λ (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 λ (1) max. In this question, you should have 2 plots.

Solution:

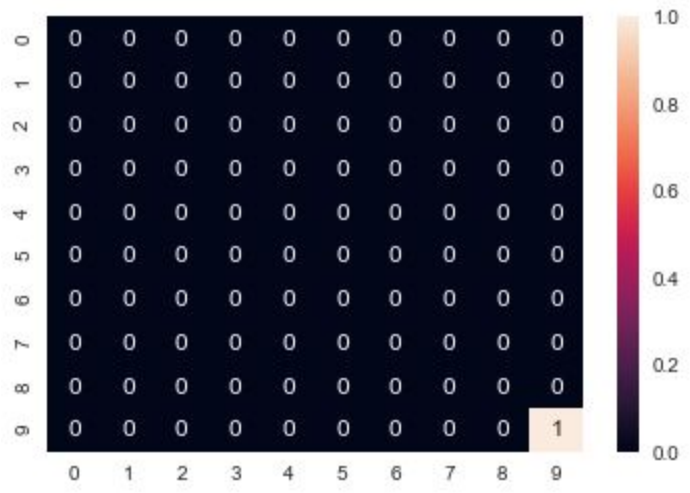


Fig: heat map of the ground truth reward

Case1 :If we go right in case of equal probabilities of right and down:

Fig: heat map of the extracted reward

Case2 :If we go down in case of equal probabilities of right and down:

Fig: heat map of the extracted reward

Observation:

We shall be providing explanations for Case 1 only.

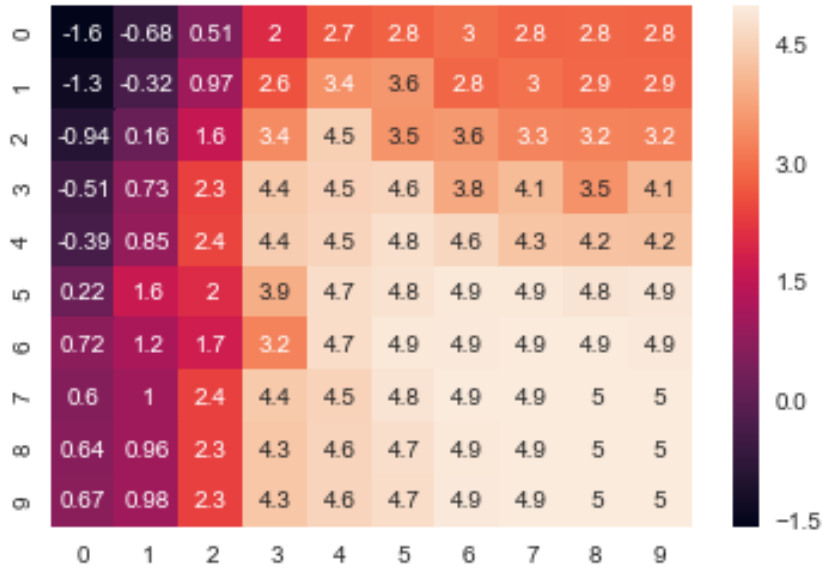
We generate heat maps of the ground truth reward and the extracted reward. The ground truth reward is the Reward function 1 and the extracted reward is computed by solving the linear program given by equation 2 with the λ parameter set to $\lambda_{max}^{(1)}$.

We observe that the rewards for the top-left corner are highly negative while that for bottom right are positive, which is expected as per the original reward function. Also, since we prefer to go right (instead of down in case of equal probability for both), we see that the left columns are highly penalized. However, we also notice certain anomalies where extracted reward differs much from the original. This makes sense because we just provide optimal policy to IRL, which is probably not enough information to extract the reward too closely.

In questions 14-17 we solve just using results of case 1

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.

Solution:



Observation:
We see that the values are similar from what you would expect from the extracted reward function. The bottom right have higher values and the top left have the least. A comparison has been made in the next question.

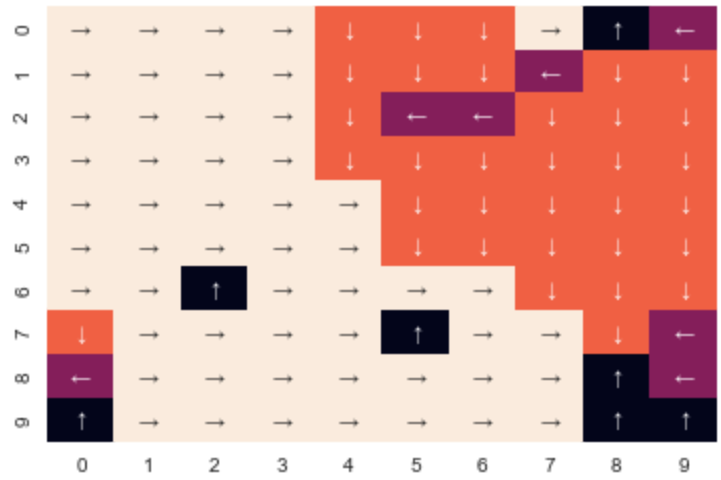
Question 15: (10 points) Compare the heat maps of Question 3 and Question 14 and provide a brief explanation on their similarities and differences.

Solution:
Since more than one reward function could map to the same optimal policy , we observe that while the extracted reward function is similar to the reward function, it is not exactly the same, which is why even the optimal state values computed on the extracted reward function is similar to the one we found for the reward function , but it is not the same.

Similarities : Both seem to be suggesting higher values in the bottom right and lower values in the top left. If we see along the top-left to bottom-right diagonal, we see that that the values increases, suggesting that the closer we are to the bottom-right end, the easier it is to achieve our goal state. However the graduality of that change differs in both. The reflection along the top-right to bottom -left values are only seen in the lower (towards the bottom right sub matrix elements of the policy of extracted reward function.

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.

Solution:



Observation:
The action policy seems to be in consistent with the reward function. We seem to be pointing towards the bottom right of the matrix,which is an expected result. Further comparisons are given in the next question.

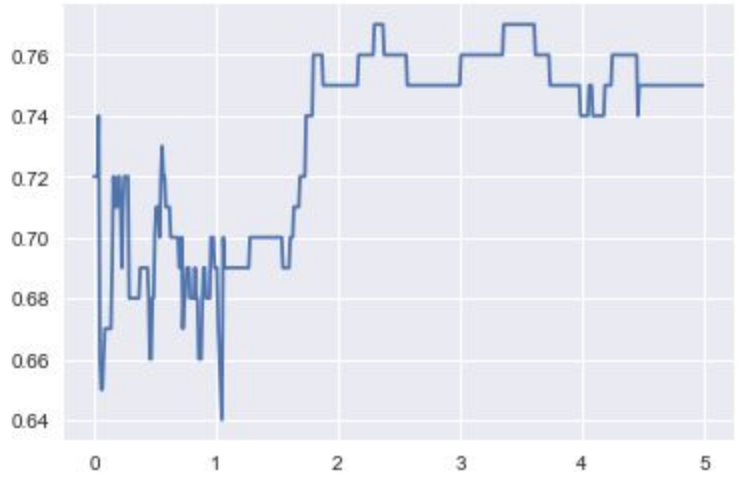
Question 17: (10 points) Compare the figures of Question 5 and Question 16 and provide a brief explanation on their similarities and differences.

Solution:
Both seem to be pointing towards the bottom right of the matrix which seems to be the right intuition. However we see that while the policy extracted from the actual reward function goes to the last cell of the last row, this doesn't happen in the policy of the extracted reward function. This again is because the extracted reward function wasn't exactly similar to the actual reward function. This isn't a flaw in the algorithm,it just happens that more than one reward function can lead to the same policy. However, this is is still good as it is similar in the general intuition.

Question 18: (30 points) Sweep λ from 0 to 5 to get 500 evenly spaced values for λ. For each value of λ compute OA(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 OE(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.

Solution:
Case 1: If we go right in case of equal probabilities for down and right:

Case2:If we go down in case of equal probabilities for down and right:



Observation: We shall be providing explanations for Case 1 only.
We sweep λ from 0 to 5 to get 500 evenly spaced values for λ. For each value of λ, we compute OA(s) by following the process describe and plot λ (x-axis) against Accuracy (y-axis).

We observe the accuracy ranges from 0.64 to 0.79. This is higher than the accuracy range for reward function 1. This could be because reward function 1 is very simple, hence policy for it is simple and does not give as much information while reward function 2 and hence, the policy for it is more complex(tries to avoid obstacles as per the complex reward function) and gives relatively more information. Also, here the peak value is reached fairly late(i.e. for a relatively large λ). This could possibly be implying that regularization is needed(as possibly overfitting to begin with). This makes sense because complex functions tend to overfit and hence, need regularization, so as to prefer simpler ones.

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 λ (2) max. Please report λ (2) max

Solution:
Case 1: If we go right in case of equal probabilities for down and right:

max accuracy is:
0.79
best lambda is:
4.12

Case 2:If we go down in case of equal probabilities for down and right:

max accuracy is:
0.77
best lambda is:
2.3

Observation:
We shall be providing explanations for Case 1 only.
We use the plot in question 18 to compute the value of λ for which accuracy is maximum. We find this value to be 4.12. As explained previously, this is higher than the optimal λ for reward function 1. This could be because reward function 1 is very simple while reward function 2 is more complex. Hence, here the peak value is reached fairly late(i.e. for a relatively large λ). This makes sense because complex functions tend to overfit and hence, need regularization.

Question 20: (15 points) For λ (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 λ (2) max. In this question, you should have 2 plots.

Solution:

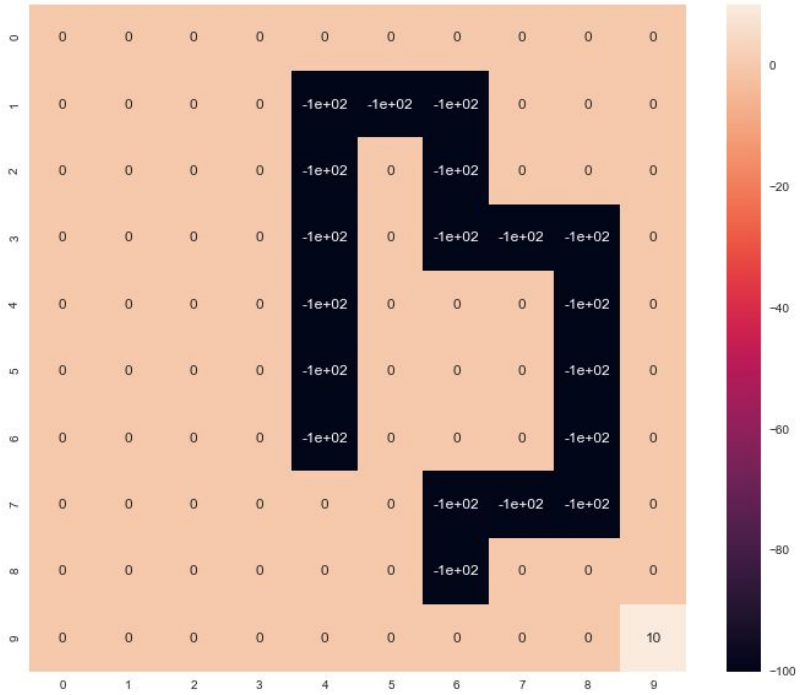


Fig: heat map of the ground truth reward

Case 1: If we go right in case of equal probabilities for down and right:

Fig: heat map of the extracted reward

Case 2:If we go down in case of equal probabilities for down and right:

Fig: heat map of the extracted reward

Observation:

We shall be providing explanations for Case 1 only.

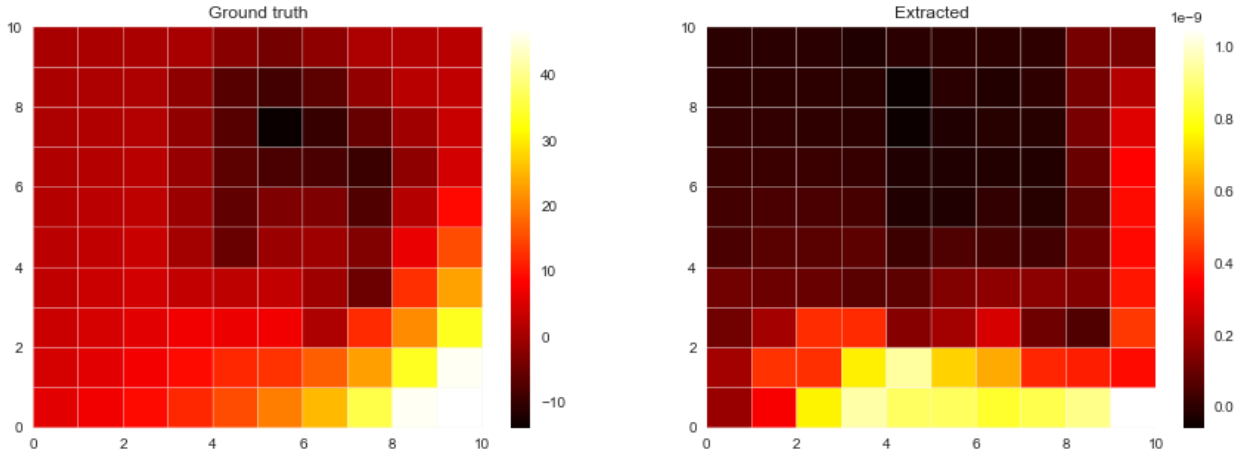
We generate heat maps of the ground truth reward and the extracted reward. The ground truth reward is the Reward function 2 and the extracted reward is computed by solving the linear program given by equation 2 with the λ parameter set to $\lambda_{max}^{(2)}$.

We observe that the extracted reward looks similar to the ground truth one in that we are able to see similar blackish(low reward i.e. penalty) regions in the extracted reward, which is expected as per the original reward function. This extracted reward seems a bit closer to the ground truth as compared to the reward1 extracted in Q13. This makes sense as the policy in this case seems more informative. However, we also notice certain anomalies like the (8,2) cell where the reward value is the highest which is unexpected. Also, (8,3) also seems abnormally high. This makes sense because we just provide optimal policy to IRL, which is probably not enough information to extract the reward too closely.

In all further questions, we solve just using results of case1

Question 21: (10 points) 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.

We used the extracted reward function in this question. It was calculated by solving the linear program. The optimal state values are in a 2D grid. In order to calculate the optimal state values, we used the the optimal state value function that was already implemented in the first part, reinforcement learning section.



The above 2 heat maps can be seen as similar. Below we have explained on the similarity and analysis further.

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

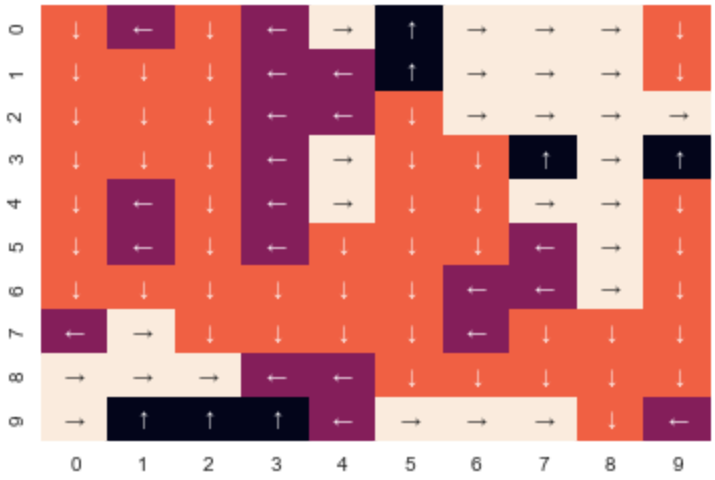
The 2 plots above shows the heat maps of the optimal values obtained. We can see the optimal state values of the extracted reward function is very similar to the ground truth reward function. The ground truth state values shows a gradual increase in the optimal state value as can be seen in the bottom right of the grid. This is observed due to the presence of the value “10” in the bottom right corner of the second reward function. We can see a similar observation in the optimal state values as it also has the same region in the bottom right corner. Also, we see the top half of the grid in ground truth has lower optimal values and the extracted reward has a similar darker portion in the top half of the grid.

We can make another observation that the bottom left corner and the right edge of the grid have neither dark or light values for the region. In the extracted reward we see slightly higher values for the same. This does not mimic the ground truth but it doesn't make a lot of difference as it represents the safe regions on the grid that direct towards the bottom right corner without any blockages in the path. These are the regions which show the agent a path towards bottom right corner safely.

The visible difference between the two is that the top left corner in extracted reward has very low values as compared to the ground truth values. But it isn't of significant difference and doesn't harm the performance much.

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.

Just like Q16, we have shown our optimal policy for the second reward function. For visualization purpose, we have generated a grid with the optimal action.



Overall, we observe that major policies of our extracted function matches with ground truth values. Also, compared to reward function the percent match is significantly more. We have explained in detail in the following question.

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

The above plot show us the the optimal policy of the extracted reward function and we are comparing with the optimal policy of the expert. These have many similarities among each other. The count and proportion of the number of arrows (down, right, left, up - in decreasing order of the count) are maintained by both the figures, where down arrows are downward facing arrows are maximum and upward facing arrows are minimum.

Except the boundary edges, the inner portion of the grids have similarity as they took a path which avoided the center portion of the grid showing the agent learned not to choose that path with lowest rewards.

The left edge is similar in both the grids where they start with the down arrows and on reaching bottom they turn to right arrows indicating the path towards the bottom right corner. The top & right edges also show this similarity by first going with right arrows and then tuning to down arrows. They are doing the same thing of reaching the bottom right corner for the highest reward function by circumventing the negative rewards in the mid half of the grid.

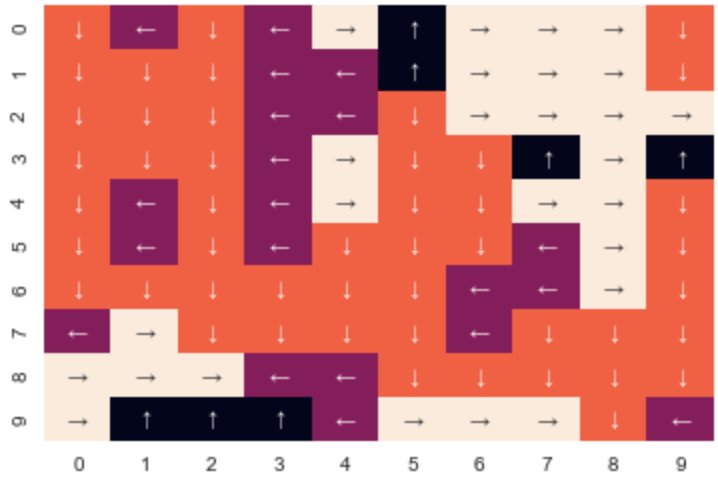
Both are very similar to each other. Both may take different paths but reaching the common objective is key. For eg, in the 4th column, the ground truth path takes a left and goes down which reaches the bottom right corner but the extracted reward goes down and takes right to reach the corner. Both are moving towards the common objective of reaching the corner. One major difference is the presence of cycles/deadlocks in the extracted reward which may not lead to the bottom right corner as the agent can get lost while traversing through the same cycle again and again.

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, compute 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.

From figure 23 we see that the optimal policy has some discrepancies. One of the major discrepancies of the optimal policy of the agent is the deadlock situation that inadvertently arises. We reach a situation in the optimal policy where two adjacent squares show opposite directions, pointing towards each other.

We observed was that there were some **deadlocks** in the grid which lead the agent to repeat them again and again if it entered such a deadlock. This could either be 2 arrows facing each other for eg (left arrow and right arrow together in 2 columns) or down arrow and up arrow consecutively in 2 rows. If an agent enters such a deadlock it would not be able to escape it and would keep on going.

This can be seen in the below grid.

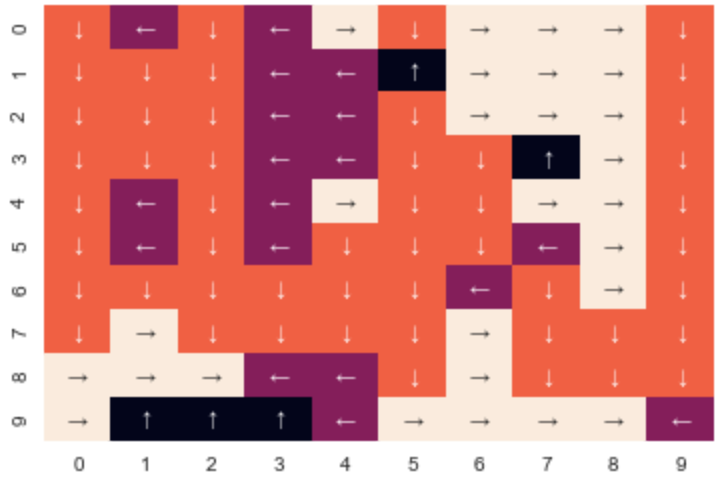


Here there is a deadlock in (8,2) and (8,3). This is because both the values have a very high reward value. As a result, the agent wants to stay at that place where the reward is high and hence prefers it which results in an infinite loop. This can be extended to cycles where if an agent enters will be stuck in a cycle.

The other discrepancy we observed was the existence of arrows on the edges of the grid which lead the agent to go **off the grid**. For eg: on the left edge if there are arrows facing left, the agent would be mislead into going off the grid.

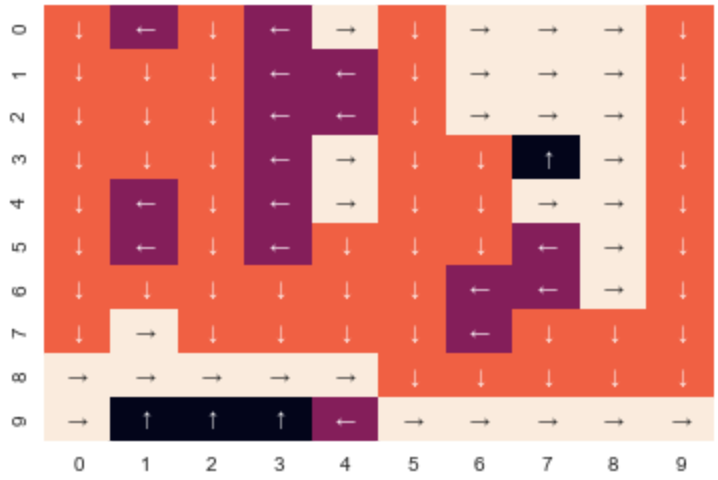
On solving the discrepancy which lead the agent to go off the grid we achieved the following optimal policy with an accuracy of **0.83**. Here as you can see the arrows which lead of off grid are not present. We resolved this by scanning the optimal policy and if such a state is encountered which is on the edge and goes outside the grid, it would be replaced by the next best action for that state. We kept a track of the next best state for each state in the grid and if such a state was encountered, it was replaced making sure that there is no state which goes out of the grid now.

Highest accuracy = 0.83
Best Lambda: 4.03



Now after getting this grid we have an optimal policy without any states leading off the grid. But it has 2 location which are leading to a **deadlock** namely (0,5) and (1,5). And (8,2) and (8,3) is also leading to a deadlock. We implemented a deadlock/cycle detection algorithm which detected such cycles which will be replaced by the **next best action** to be taken which breaks such a deadlock. We achieve the following grid which breaks such a deadlock.

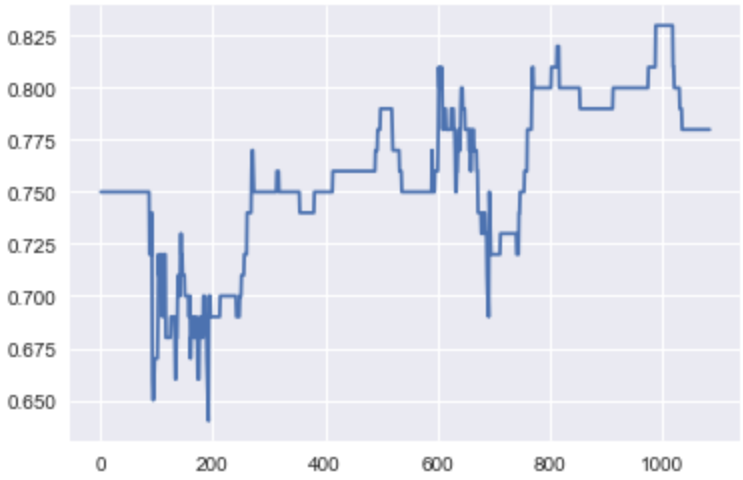
We have been able to resolve deadlocks in this reward function and can be done to an extent. Deadlock would also be caused if there was a pair of U-D neighbor states in the agent's policy. We fixed this discrepancy by changing all R-L deadlock states to R policies and all U-D deadlock states to D policies as shown.

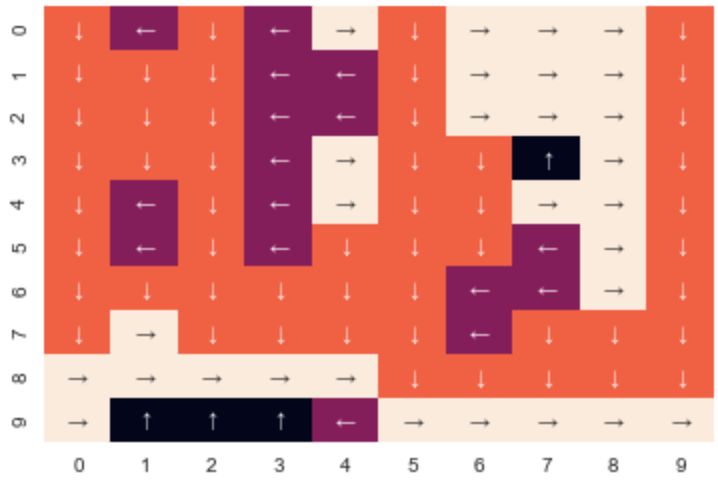


However this solution of deadlock isn't enough and is much tougher. In our example, the problem was solved after changing the directions of 4 cells to their second best directions. We also improved the **accuracy from 79 to 83**. But this can lead to a situation where a deadlock solution can propagate a new deadlock in an infinite loop.

One solution that we can think of is the introduction of a **time penalty**. In the reinforcement learning algorithm, the solution doesn't think of the length of the path taken or the time taken for the path. It does not try to find the highest reward in the fastest time. But if we introduce a penalty that penalizes with time we will make the algorithm reject those paths in which it'll be stuck for a long period of time. This will stop from any loop which is causing the highest reward to be achieved and the agent will get it. It will penalize the agent from taking a longer period of time and will avoid the path.

The **other discrepancy** that we noticed was in the corner state(bottom right) which was the best state for our reward function. The extracted policy causes the agent to move back in the grid when encountering such a state. But upon reaching the best state the agent should just stay there or move off the grid. It shouldn't move back into the grid of spaces. The optimal policy should be such that it takes you to a state of highest reward. This issue can be resolved by replacing the direction with the second best action at any of the edge states which point out of grid.





By comparing the above 2 figures we can see that the deadlock state in (8,2) and (0,5) has been successfully resolved and the best state (9,9) now has a policy moving off the grid.

One other issue we observed was that the extracted optimal policy was not able to completely avoid the trap region of -100 rewards. In an ideal case we would want the policy to avoid it completely by the use of arrows which indicate a direction opposite from it but the extracted optimal policy is not able to achieve this. However the optimal policy points in a direction in some regions such that the agent continues to move into a region of negative reward.