ECE 232E Spring 2023

Reinforcement Learning and Inverse Reinforcement Learning

Reinforcement Learning

Question 1

In this question, the goal was to visualise the heatmaps of the ground truth reward functions given to us as part of the project. The visualisations for reward function 1 and 2 can be visualised in Fig 1 and Fig.2 respectively.

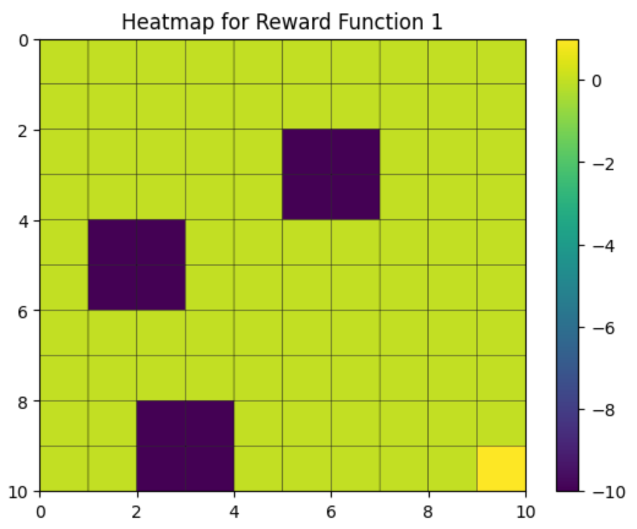


Fig 1. Heatmap for Reward Function 1

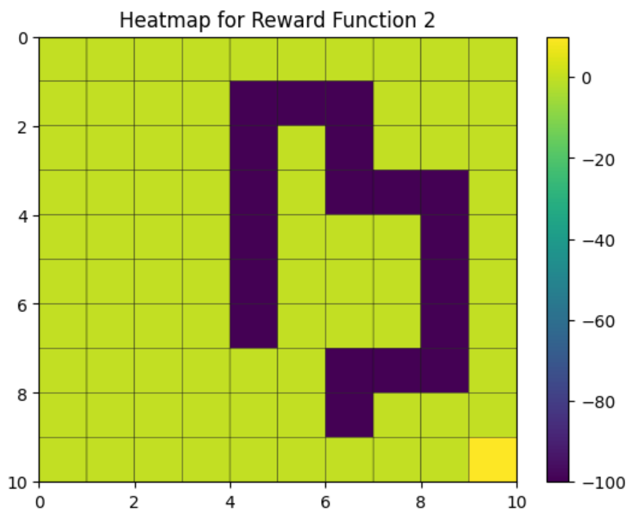


Fig 2. Heatmap for Reward Function 2

Question 2

In this question, the goal was to define a MDP by setting up the state-space, action set, transition probabilities, discount factor and reward function. In reinforcement learning, the agent interacts with the environment by observing it *(O_t)* and chooses an action *(A_t)* to maximise the reward *(R_t)* it receives from the environment. For this project, we assume that the environment is fully observable thus $O_t = S_t$ where the latter represents the state of the environment.

In this project, we assume that the environment can be modelled using a Markov Decision Process (MDP) which means that the future only depends on the current state S_t and action taken A_t. The mathematical definition of it is shown in equation (1).

$$P(S_{t+1}|A_t, S_t, A_{t-1}, S_{t-1}, \ldots.. A_0, S_0) \;=\; P(S_{t+1}|A_t, S_t) \tag{1}$$

An MDP is formally defined as tuple of the following $(S, A, P_{ss`}^{a}, R_{ss`}^{a}, \gamma)$. Where each symbol represents the following:

- *S*: *S* represents the set of states (since we are assuming that the environment is fully observable) which for this project is a total of 100 states represented in a 10x10 matrix, that holds discrete values.
- *A*: *A* represents the action space and for this project is discrete, since the agent at any point in time can only take 4 actions {l*eft, right, up, down*}.
- $P_{ss`}^{a}$: $P_{ss`}^{a}$ represents the probability transition matrix which in turn means that based on the current state and action, what is the probability of going into another state. For this project, the agent can be blown into a random neighbouring state with probability *w* and it will land in the desired state with probability *1-w.* For physical interpretation, *w* is considered the wind. The setup follows from equation (1) which means that the future is only dependent on the present and not the past and a few other rules set out in the project such as:
    - If state *S* and *S`* are not neighbours then the probability $P(S`|a,S) = 0$
    - Inside the grid, the probability of the agent going to one of its four neighbours is probability $1 - \frac{3}{4}w$ and the probability of being randomly blown into one of its four neighbours is with probability $\frac{1}{4}w$
    - If the agent is at corner (only 2 neighbours), it can intentionally get off the grid with probability$1 - \frac{1}{2}w$ and unintentionally fall of the grid with probability $\frac{1}{2}w$.
    - If the agent is at an edge(only 3 neighbours) it can intentionally get off the grid with probability $1 - \frac{3}{4}w$ and it can unintentionally get off the grid with $\frac{1}{4}w$
- $R_{ss`}^{a}$: $R_{ss`}^{a}$ is the reward matrix, it stores the information regarding the reward the agent should receive when moving from S_t to $S_t^{`}$ when using action A_t and for the scope of this project, it is given as part of the question
- *γ: γ* is the discount factor that can hold any value from 0 to 1. The closer it is to 0, the shorter sighted the agent is and focusses more on immediate reward than long term rewards

The value iteration algorithm focuses on finding the optimal value function for a fixed policy π(s) and is given by the equation (2) and (3) which can be solved iteratively using the Bellman-Ford equations.

$$V^{optimal}(s) \;=\; \max_{\pi} V^{\pi}(s) \tag{2}$$

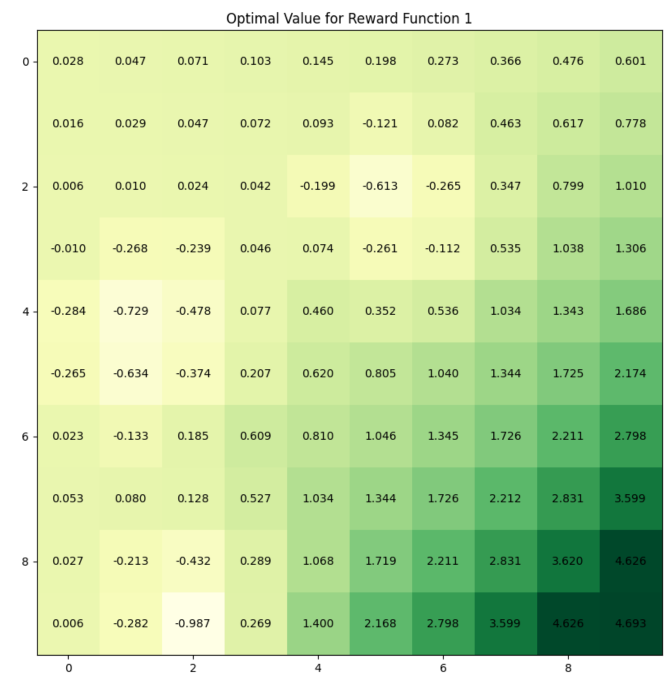
$$V^{optimal}(s) \;=\; \max_{a \in A} P_{s`,s}^{a}(R_{s`,s}^{a} + \gamma V^{optimal}(s`)) \tag{3}$$

Following this the initialisation and the estimation part of the algorithm were implemented which is given below:

- Initialise a set called *s* to zero whose dimensions match that of *S*

```
• Initialise $\Delta = \infty$
• While $\Delta > \epsilon$
    ○ $\Delta = 0$
    ○ Extract previous values of $V(s)$ to a new variable called $v$
    ○ Compute the new $V(s)$ using Bellman Ford's equation
    ○ Calculate maximum absolute deviation between $v$ and $V(s)$
    ○ $\Delta$ = maximum absolute deviation between $v$ and $V(s)$
```

For this question, the probability of w is set to 0.1, ϵ is set to 10^{-2}, γ is set to 0.8. The optimal value of each state can be seen in Fig.3 and the heatmap when regularly sampled can be seen in Fig 4-9.



0.028	0.047	0.071	0.103	0.145	0.198	0.273	0.366	0.476	0.601
0.016	0.029	0.047	0.072	0.093	-0.121	0.082	0.463	0.617	0.778
0.006	0.010	0.024	0.042	-0.199	-0.613	-0.265	0.347	0.799	1.010
-0.010	-0.268	-0.239	0.046	0.074	-0.261	-0.112	0.535	1.038	1.306
-0.284	-0.729	-0.478	0.077	0.460	0.352	0.536	1.034	1.343	1.686
-0.265	-0.634	-0.374	0.207	0.620	0.805	1.040	1.344	1.725	2.174
0.023	-0.133	0.185	0.609	0.810	1.046	1.345	1.726	2.211	2.798
0.053	0.080	0.128	0.527	1.034	1.344	1.726	2.212	2.831	3.599
0.027	-0.213	-0.432	0.289	1.068	1.719	2.211	2.831	3.620	4.626
0.006	-0.282	-0.987	0.269	1.400	2.168	2.798	3.599	4.626	4.693

Fig. 3 – Optimal Value for Reward Function 1

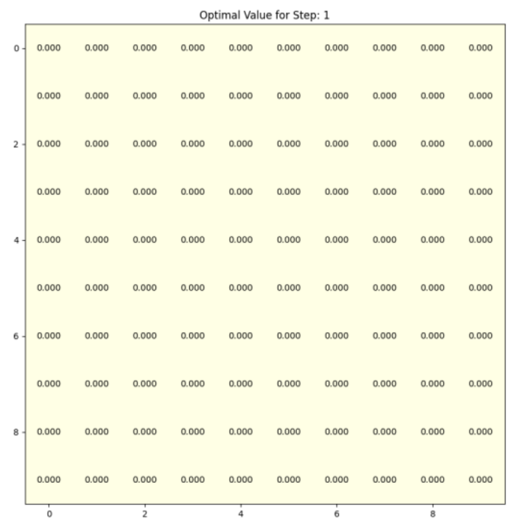


Fig. 4 – Optimal Value for Reward Function 1 at timestep $t = 1$

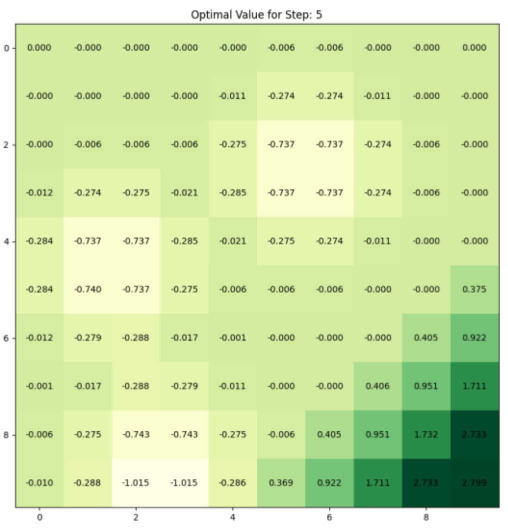


Fig. 5 – Optimal Value for Reward Function 1 at timestep $t = 5$

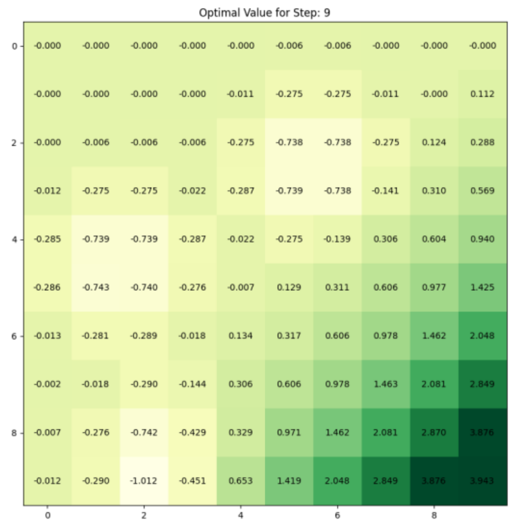


Fig. 6 – Optimal Value for Reward Function 1 at timestep $t = 9$

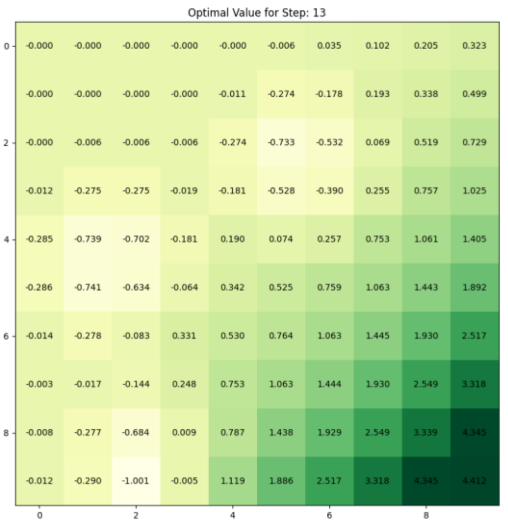


Fig. 7 – Optimal Value for Reward Function 1 at timestep $t = 13$

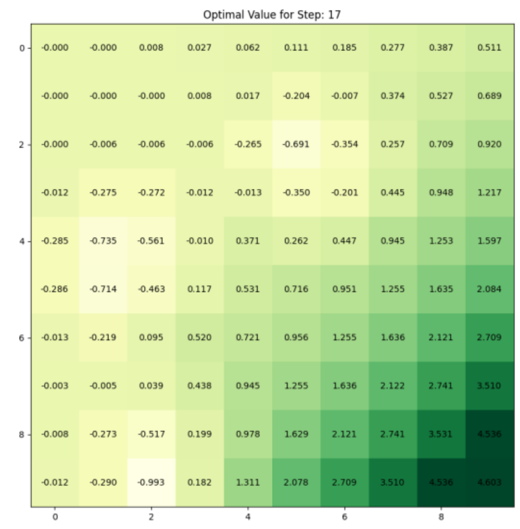


Fig. 8 – Optimal Value for Reward Function 1 at timestep $t = 17$

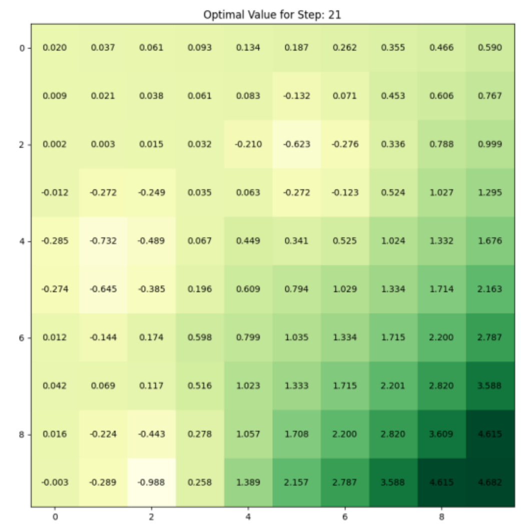


Fig. 9 – Optimal Value for Reward Function 1 at timestep $t = 21$

The agent took a total of 22 steps to converge to the optimal policy, hence N=22. The following observations can be made as well from the plots.

1. From Fig 3 and 5 -9, we can see that state 99 has the highest value and the further the agent is from this state, the lower the value of that state is and it can even be negative highlighting that the policy of the agent is now oriented towards travelling to state 99.
2. The negative patches present in the state where the agent gets a negative reward can be seen in the plots as well, which highlights that the agent learns to stay away from these blocks since it's total value decreases if it does visit one of these patches which can be seen by the negative value assigned to these patches.
3. The value iteration algorithm logarithmically converges on the optimal state which can be seen from how in the first few steps, the algorithm starts converging on the optimal value of the state and this highlights that the bellman ford algorithm is ϵ-suboptimal at any point.

Question 3

In this question, the goal was to generate a heatmap of the optimal value function which was obtained from the previous question. The heatmap can be visualised in Fig.

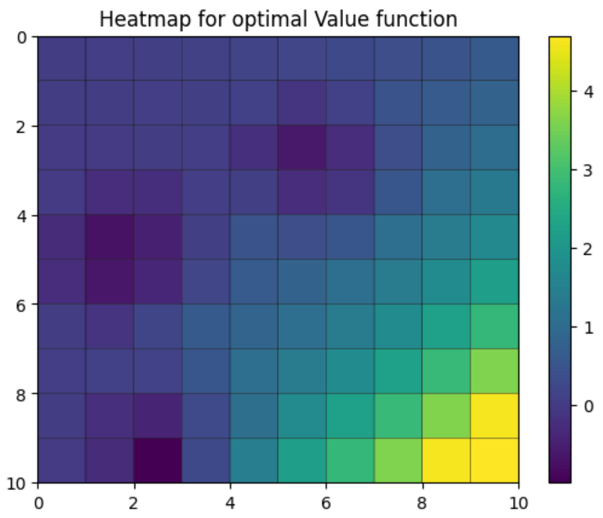


Fig.10 Heatmap visualisation of the Optimal Value Function obtained for Reward 1

Question 4

The following inferences can be made from Fig. 3 and are listed below

1. State 99 and its surrounding states have the highest values across the entire grid, which is a reflection of the reward function distribution since State 99 is the only state with a positive reward. Rest of the states either hold no reward or a negative reward. The states surrounding State 99 also hold similarly high values because of the discounting process brough out by γ.
2. The further the state is from state 99, the lower the value assigned to that state and this does make intuitive sense since the agent has to travel more or carry out more actions to reach the states with higher rewards. Furthermore, states closer to the optimal state have a higher value, and states closer to sub-optimal states have a lower value.
3. The patches in the grid where the agent can get negative rewards can be seen in Fig.3 as well, since the states associated with these negative rewards also have a negative value attached to it and the surrounding states also have a negative value attached to it, this highlights that the agent should avoid these states and the surrounding states as well since it can reduce the total value the agent can attain.

Question 5

In this question, the goal is to visualise the optimal policy of the agent and the results for this can be seen in Fig. 11

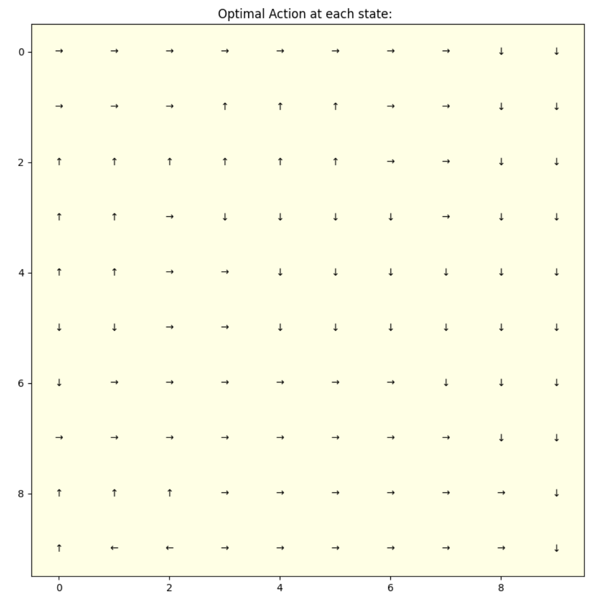


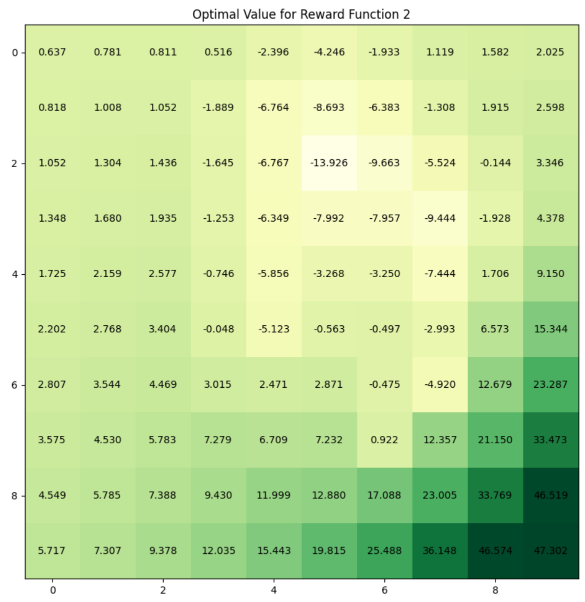
Fig.11 Optimal Policy for the Agent for Reward Function 1

After comparing the heatmap in Fig. 10, the optimal values in Fig.3 and the optimal policy in Fig.11 the following inferences can be made. At each state, the arrow points towards one of the four neighbouring states whose value is higher than the current value, that is the policy searches for the locally optimal value and its possible to interpret the actions as the greedy algorithm that greedily runs over the value finding the highest value. Therefore this does confirm the intuition that has been corroborated with the optimal value graph in Fig.3 since the optimal policy of the agent is to find the shortest route towards state 99.

Furthermore it is possible for the agent to compute the optimal action by observing the optimal values of its neighbours, as we've highlighted before since the action at each state is based on moving towards a more optimal state, the action taken at any state depends on the optimal values of the neighbours, which can be considered an implication of equation (3) where the action is taken based on the reward and surrounding value functions. Therefore it is possible for the agent to learn an optimal policy by observing the value of its neighbours.

Question 6

In this question, the goal is to recompute the optimal value of the grid using reward function 2. The results of this can be seen in Fig. 12.



0.637	0.781	0.811	0.516	-2.396	-4.246	-1.933	1.119	1.582	2.025
0.818	1.008	1.052	-1.889	-6.764	-8.693	-6.383	-1.308	1.915	2.598
1.052	1.304	1.436	-1.645	-6.767	-13.926	-9.663	-5.524	-0.144	3.346
1.348	1.680	1.935	-1.253	-6.349	-7.992	-7.957	-9.444	-1.928	4.378
1.725	2.159	2.577	-0.746	-5.856	-3.268	-3.250	-7.444	1.706	9.150
2.202	2.768	3.404	-0.048	-5.123	-0.563	-0.497	-2.993	6.573	15.344
2.807	3.544	4.469	3.015	2.471	2.871	-0.475	-4.920	12.679	23.287
3.575	4.530	5.783	7.279	6.709	7.232	0.922	12.357	21.150	33.473
4.549	5.785	7.388	9.430	11.999	12.880	17.088	23.005	33.769	46.519
5.717	7.307	9.378	12.035	15.443	19.815	25.488	36.148	46.574	47.302

Fig.11 Optimal Value funciton for Reward Function 2

Question 7

In this question, the goal was to visualise the optimal policy for reward function 2 using a heatmap and explain the distribution of the optimal state values for reward function 2. The results from the visualisation can be seen in Fig. 12

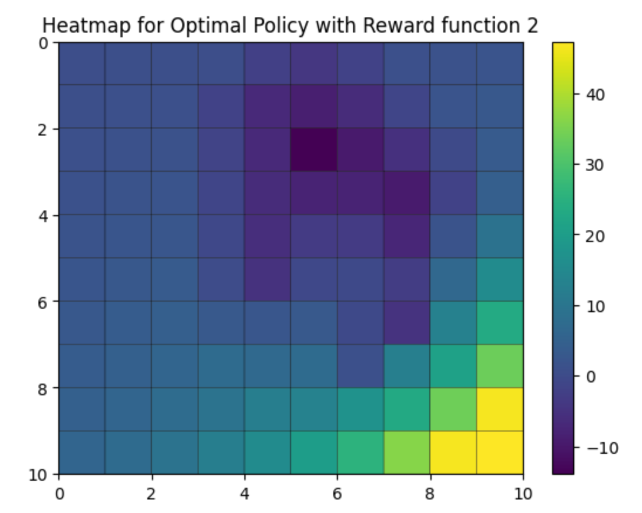


Fig.12 Heatmap for Optimal Policy for Reward Function 2

The following inferences can be made from Fig.11 and 12.

1. Similar to Reward Function 1, state 99 has the highest value attached to it that can be seen in the heatmap as well, since it has the highest reward value of +100, attached to it. The neighbouring states are comparatively high as well, which can be attributed to their proximity to the optimal state which is state 99.
2. State 52, has the lowest value of all states since it is surrounded by negative rewards in all directions but one, making it the least optimal state to be in since the agent is quite likely going to land in a state with a negative reward making it highly undesirable.
3. Similar to the results from reward 1, states closer to more desirable states have a higher value attached to it and states closer to undesirable states are tagged with a lower value
4. The patch in the grid which has negative values have the lowest values of all and this does make intuitive sense since the agent will lose points if it traverses in that patch, however an interesting point to raise here is that in the patch, the patch closer to state 99 have comparatively higher values attached to it even though the agent will get a negative reward when it visits this state. This highlights that states proximal to optimal states have a higher value attached to it, however the surrounding states are still higher than it even though they are slightly further away from state 99 and this is because these states have a reward of 0 and not -100 which highlights that the reward does come into importance when calculating the value.

Question 8

The optimal policy and the optimal action derived from the optimal policy can be seen in Fig. 13 and Fig. 14 respectively.

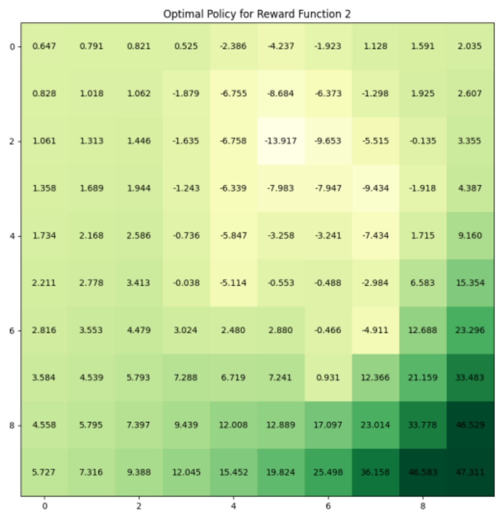


Fig.13 Optimal Policy for Reward Function 2

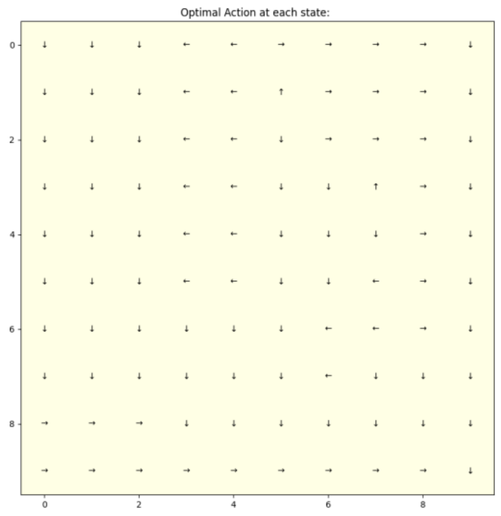


Fig.14 Optimal Action for Reward Function 2

From Fig.12 -14, the following inferences can be made which are in line with the results generated with reward function 1. All paths lead to state 99 since it is the most optimal state of all states. Interestingly, the optimal policy does indicate that the agent is trying to avoid the negative patch of rewards altogether by circumventing it, therefore highlighting that while the agent does want to reach the state 99 in the shortest time, it does want to maximise its reward and if a longer path will maximise its cumulative reward the agent will take this path since it guarantees a higher chance of receiving a larger cumulative reward. The optimal action at each state is based on the values of the neighbours and the action taken then is to traverse the more optimal state from the current state, while avoiding the negative reward patch present in the grid. Therefore, the optimal policy does agree with the intuition developed and it is possible for an agent to develop an optimal policy by observing the values of its neighbours.

Question 9

In this question, the goal is to recompute the optimal policy for reward function 1 and 2 after changing *w=0.6*, and comparing the results between *w=0.1* and *w=0.6*. The optimal action and policy for reward function 1 are visualised in Fig.15 and Fig.16 respectively and for reward function 2 in Fig.17 and Fig. 18 respectively.

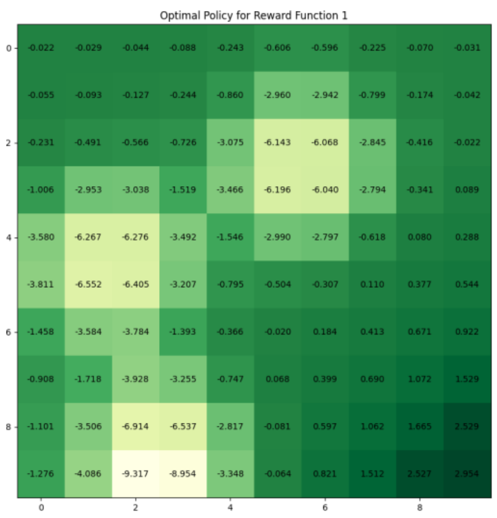


Fig.15 Optimal Policy for Reward Function 1, with *w=0.6*

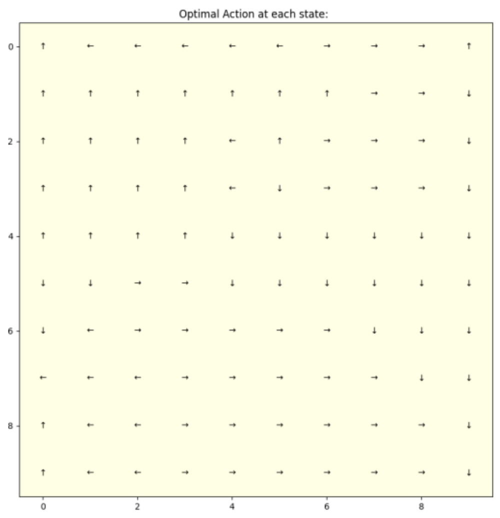


Fig.16 Optimal Action for Reward Function 1 for *w=0.6*

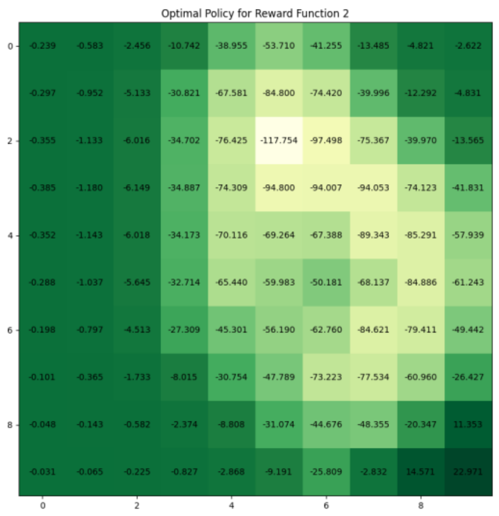


Fig.17 Optimal Policy for Reward Function 2 for *w=0.6*

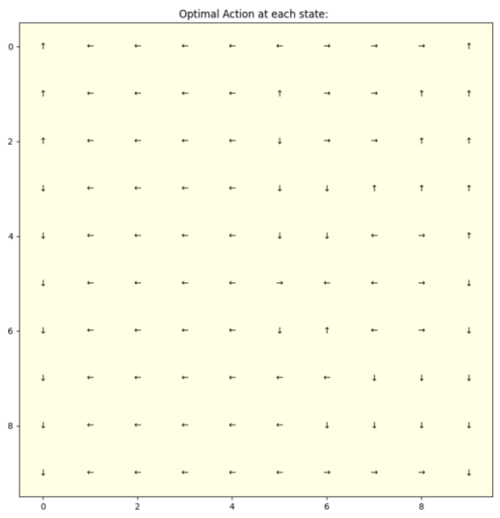


Fig.18 Optimal Action for Reward Function 2 for *w=0.6*

The following inferences can be made from comparing the results generated for *w=0.1* and *w=0.6*. Interestingly when we set *w=0.6*, it means that the agent is more likely to end up in a random neighbour than it will by intention and this can be seen in the optimal policy heatmap since most states except the negative reward patches have been are considered suboptimal.

The values for the negative patches have been amplified and are even more negative, and most patches besides state 99 are mostly homogenous in their value and this does make intuitive sense since the agent can randomly end up in a neighbouring state more so than by control, since the agent has lost more control over its navigation, it cannot control the cumulative reward more accurately, since the reward function is more stochastic now than it was deterministic before, thus all states are of similar importance.

In reward function 2, when inspecting Fig. 18, it can be seen that there is no visible path to the state 99 for most states and this is because of the proximity of the negative patch to the optimal state 99, and because of the lack of control over its navigation. The optimal policy at this point is to avoid that section of the grid altogether, because while the goal of the agent is to maximise its reward, the chance of ending up in the negative patch outweighs the benefits the agent will get by visiting state 99, thus the optimal action at most states is to get away from the negative patch in reward function 2.

Interestingly in both reward functions, the optimal action for most states is to get to a corner since the corners in both reward functions are the furthest from the negative patches, thus prevent it from getting a negative reward.

A good analogy of the effect of *w* is the exploration-vs exploitation dilemma in Reinforcement learning, since in exploitation the agent makes the best use of the knowledge present to it to make a decision, while in exploration the agent tend to spend more time gaining knowledge of the environment which can cause it have suboptimal performance since it is trying to figure out the environment, When there is no exploration-exploitation balance, the agent can either not achieve the highest cumulative reward since the agent has limited knowledge of the environment and on the other side the agent might be more invested in exploring the environment than it is in maximising the reward thus still leading to suboptimal performance.

For further experiments in this project, the *w=0.1* was used since it allowed the agent to reach the optimal state in fewer steps and the value function of each state more accurately reflected the importance of that state, and the policy was inclined towards reaching the optimal state (state 99) unlike the other policy when *w=0.6* was used where the optimal policy was for the agent to reach as far as away from the negative patches as possible.

Inverse Reinforcement Learning

Question 10.

c: The cost or penalty associated with taking an action in a particular state. It can be expressed as a function of various factors, including the reward, action probabilities, and scaling factor.

$$c = \begin{bmatrix} \mathbf{1}_{|\mathcal{S}|\times 1} \\ -\boldsymbol{\lambda}_{|\mathcal{S}|\times 1} \\ \mathbf{0}_{|\mathcal{S}|\times 1} \end{bmatrix}$$

x: The state features or state representation. It depends on the specific problem and how the states are encoded. In general, it can be denoted as a vector:
x = [t,u,R]

D: The dataset of demonstrations or expert trajectories. It contains observed state-action pairs (s, a) provided by the expert. It can be represented as a matrix:

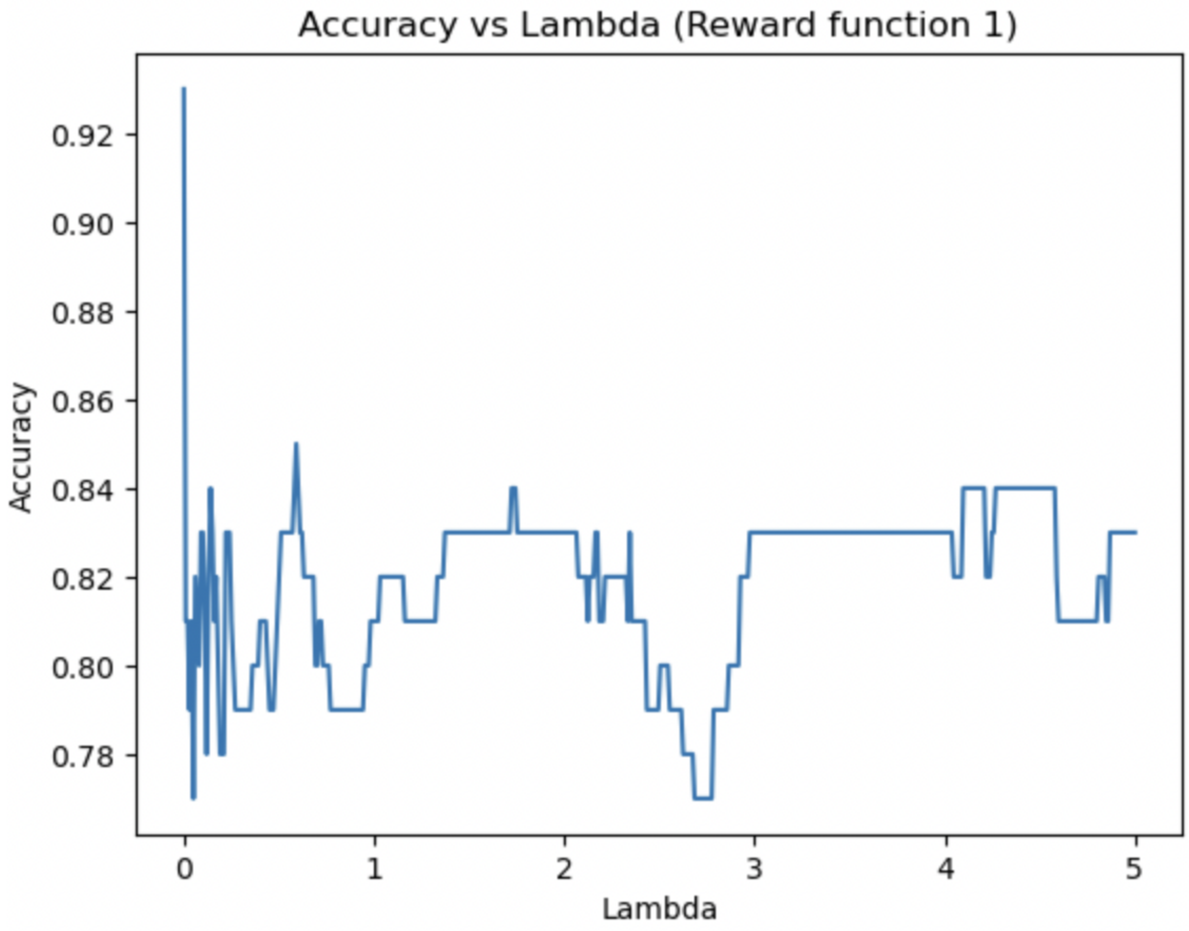
$$D = \begin{bmatrix} \boldsymbol{I}_{|\mathcal{S}|\times|\mathcal{S}|} & \mathbf{0} & (\boldsymbol{P}_a - \boldsymbol{P}_{a_1})(\boldsymbol{I} - \gamma \boldsymbol{P}_{a_1})^{-1} \\ \mathbf{0} & \mathbf{0} & (\boldsymbol{P}_a - \boldsymbol{P}_{a_1})(\boldsymbol{I} - \gamma \boldsymbol{P}_{a_1})^{-1} \\ \mathbf{0} & -\boldsymbol{I}_{|\mathcal{S}|\times|\mathcal{S}|} & \boldsymbol{I}_{|\mathcal{S}|\times|\mathcal{S}|} \\ \mathbf{0} & -\boldsymbol{I}_{|\mathcal{S}|\times|\mathcal{S}|} & -\boldsymbol{I}_{|\mathcal{S}|\times|\mathcal{S}|} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{I}_{|\mathcal{S}|\times|\mathcal{S}|} \\ \mathbf{0} & \mathbf{0} & -\boldsymbol{I}_{|\mathcal{S}|\times|\mathcal{S}|} \end{bmatrix}$$

b: The policy or behavior model. It represents the probability of taking an action a given the current state s, according to the learned behavior. It can be denoted as:

$$b = \begin{bmatrix} \mathbf{0}_{|\mathcal{S}|\times 1} \\ \mathbf{0}_{|\mathcal{S}|\times 1} \\ \mathbf{0}_{|\mathcal{S}|\times 1} \\ \mathbf{0}_{|\mathcal{S}|\times 1} \\ (\boldsymbol{R}_{max})_{|\mathcal{S}|\times 1} \\ (\boldsymbol{R}_{max})_{|\mathcal{S}|\times 1} \end{bmatrix}$$

Question 11.

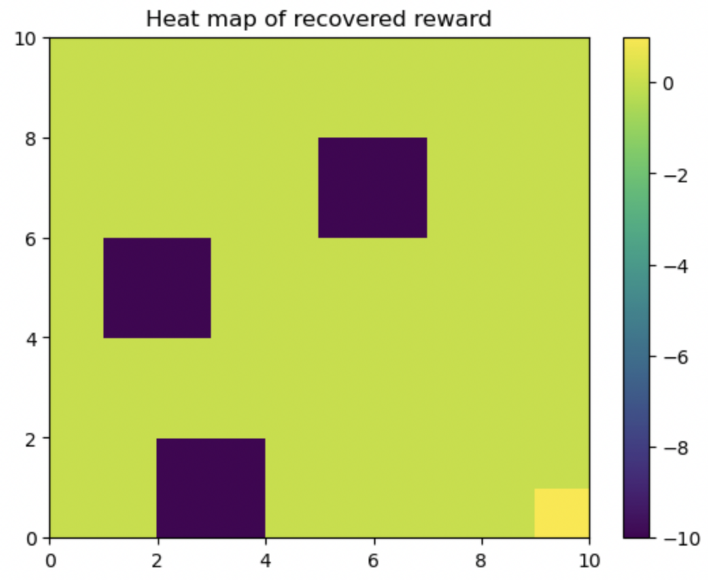
We observe that the accuracy initially rises with an increase in until it reaches a globally optimal policy and then decreases. The accuracy then somewhat improves but becomes trapped in a local optimum. As a regularizer (L1 or Lasso normalisation), it encourages reward vectors that are simpler, therefore this is to be expected.

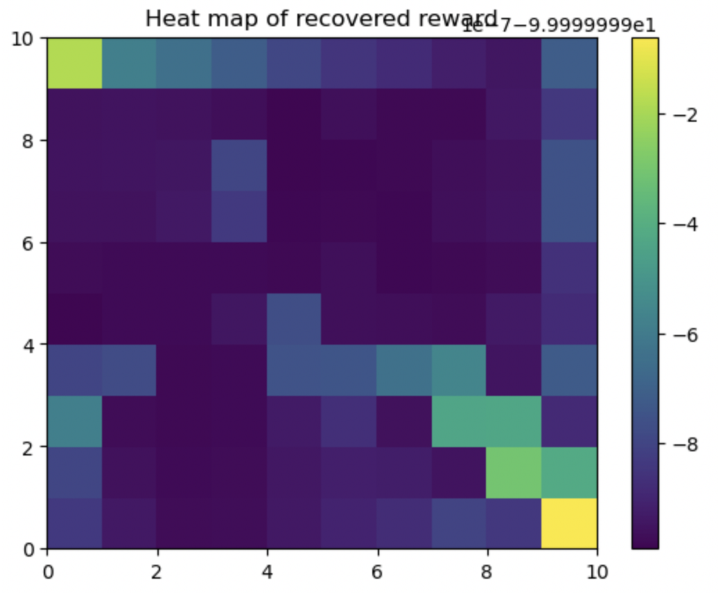


Question 12.
Value of lambda for best accuracy 0.0
Best value of accuracy 0.93

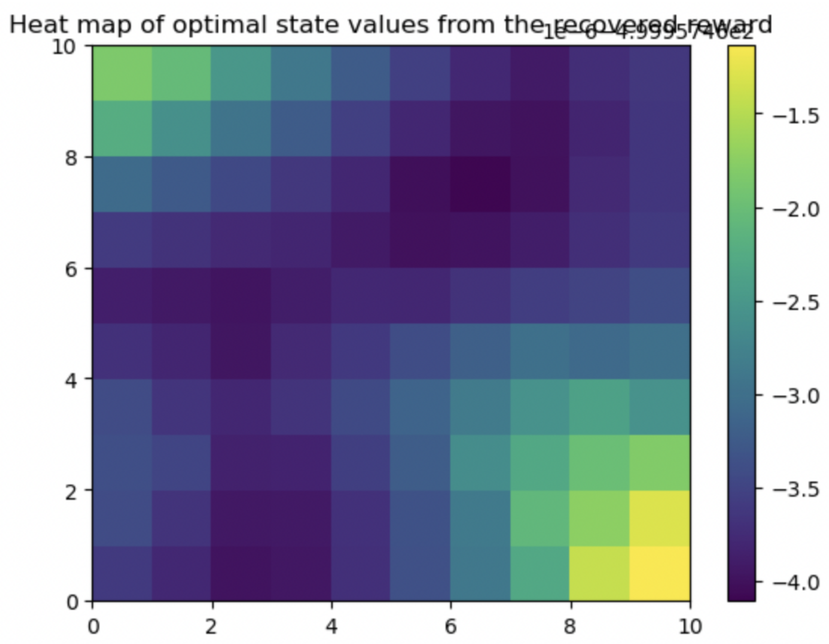
Question 13.

The IRL algorithm accurately identified the regions surrounding state 99 as having high reward, whilst the regions surrounding the negative-reward blocks have been allocated low immediate rewards. The extracted reward decreases as one gets away from state 99, which has the largest reward.





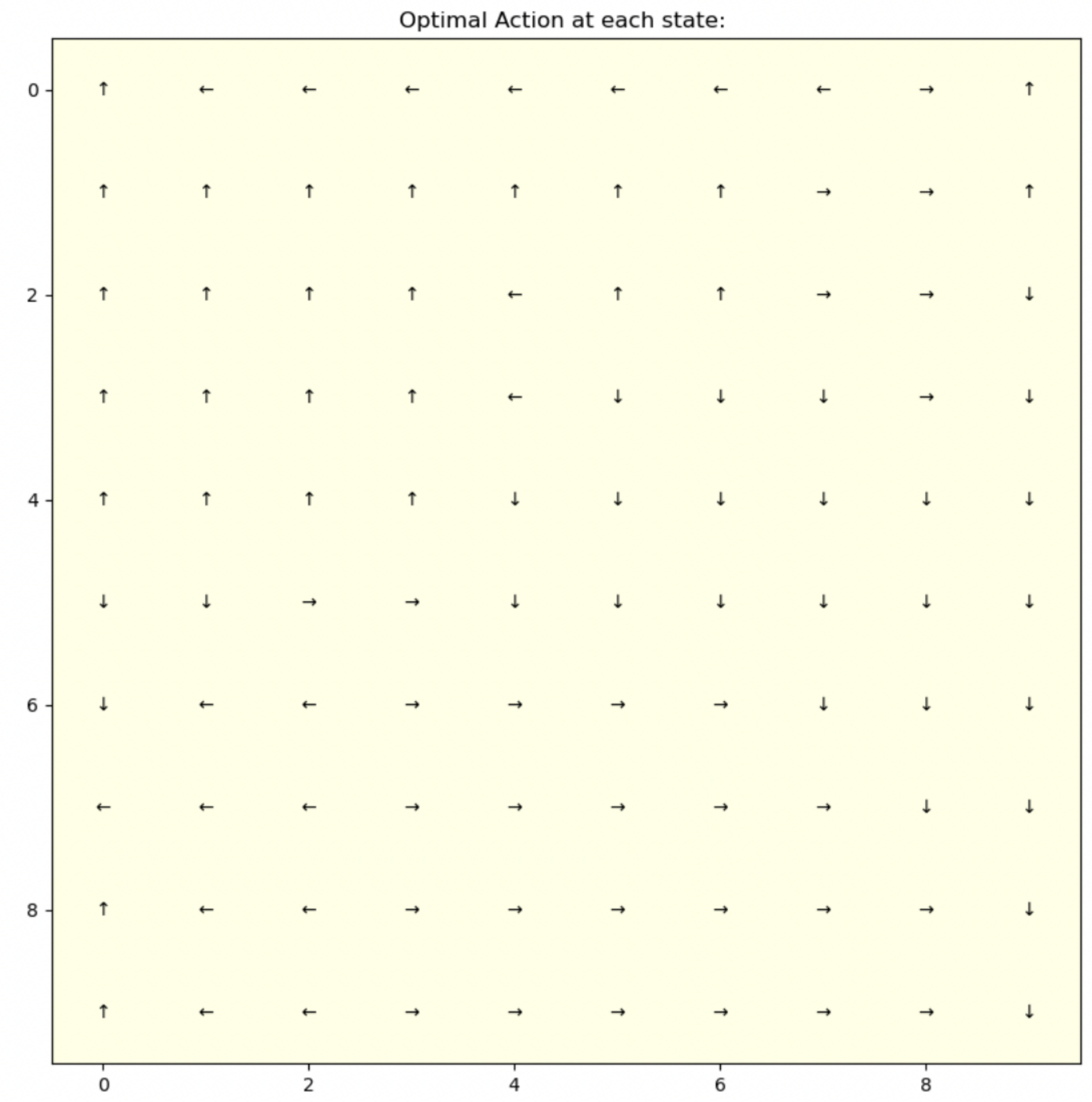
Question 14.



Question 15.

On comparing Q3 and Q14, we can observe that there are a few distinct similarities and differences. Both models return 99 as the state with highest optimal value. Further, the agent's reward is related to the optimal value of neighbor states i.e., low reward corresponds to lower optimal value of neighbor states and vice versa. However, there are differences as well. One major difference is that the agent's reward is less minimal in Q14 when the state is less optimal. Also, the scale of rewards is also less in Q14.

Question 16.



Question 17.
Comparing Q5 and Q16 we see that most frequently occurring actions are up and down. However, in Q16, some actions point to each other and some actions move the agent off-grid. These are two major disadvantages that can be observed. Further, Q5 is observed to always lead the agent to state 99 whereas the same cannot be said for the agent in Q16.

Question 18.

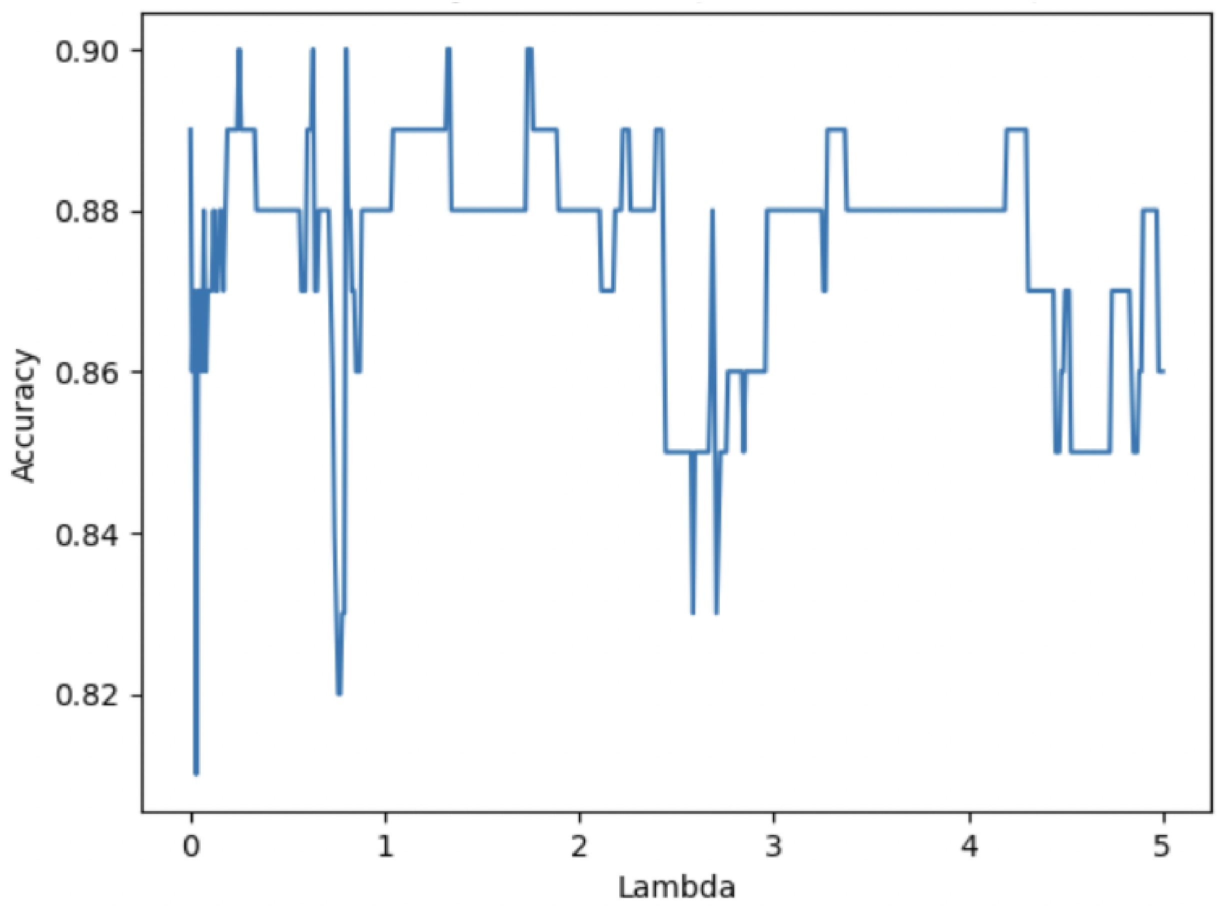
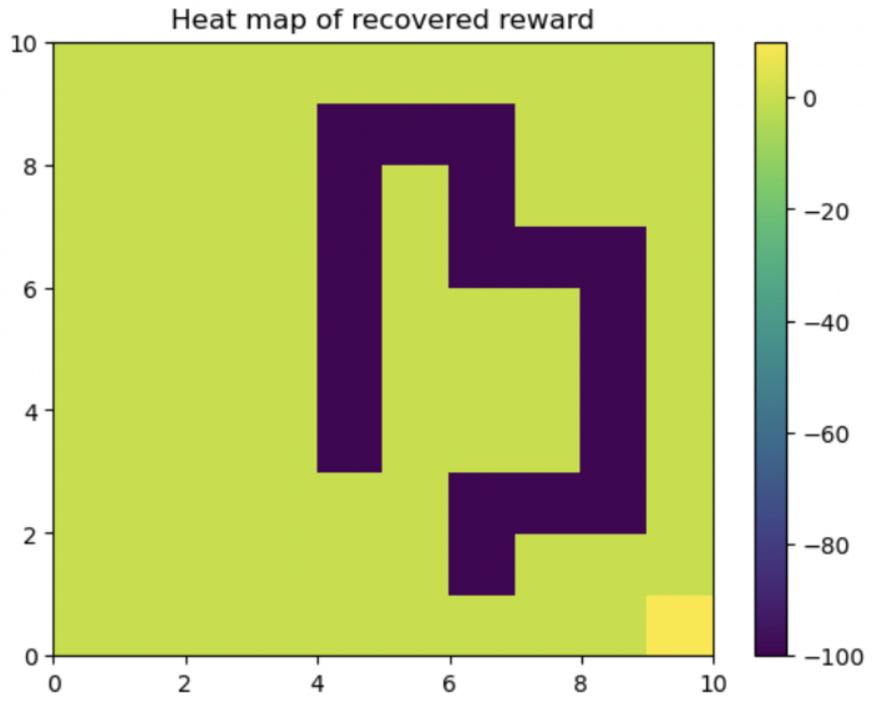


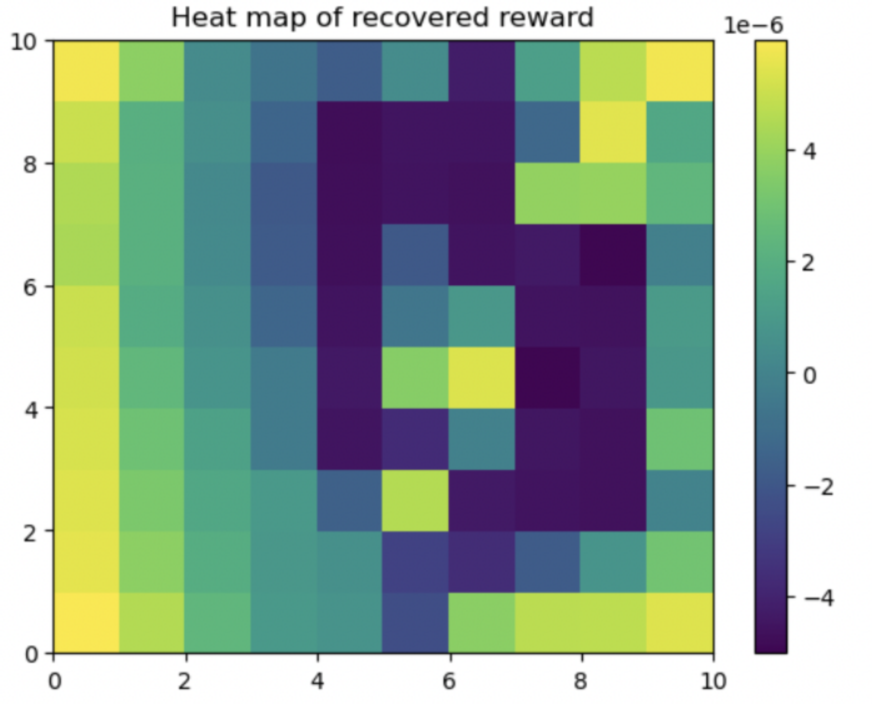
Fig Question 18. Accuracy vs Lambda (Reward function 2)

Question 19.
Value of lambda for best accuracy 0.250501002004008
Best value of accuracy 0.9

Question 20.

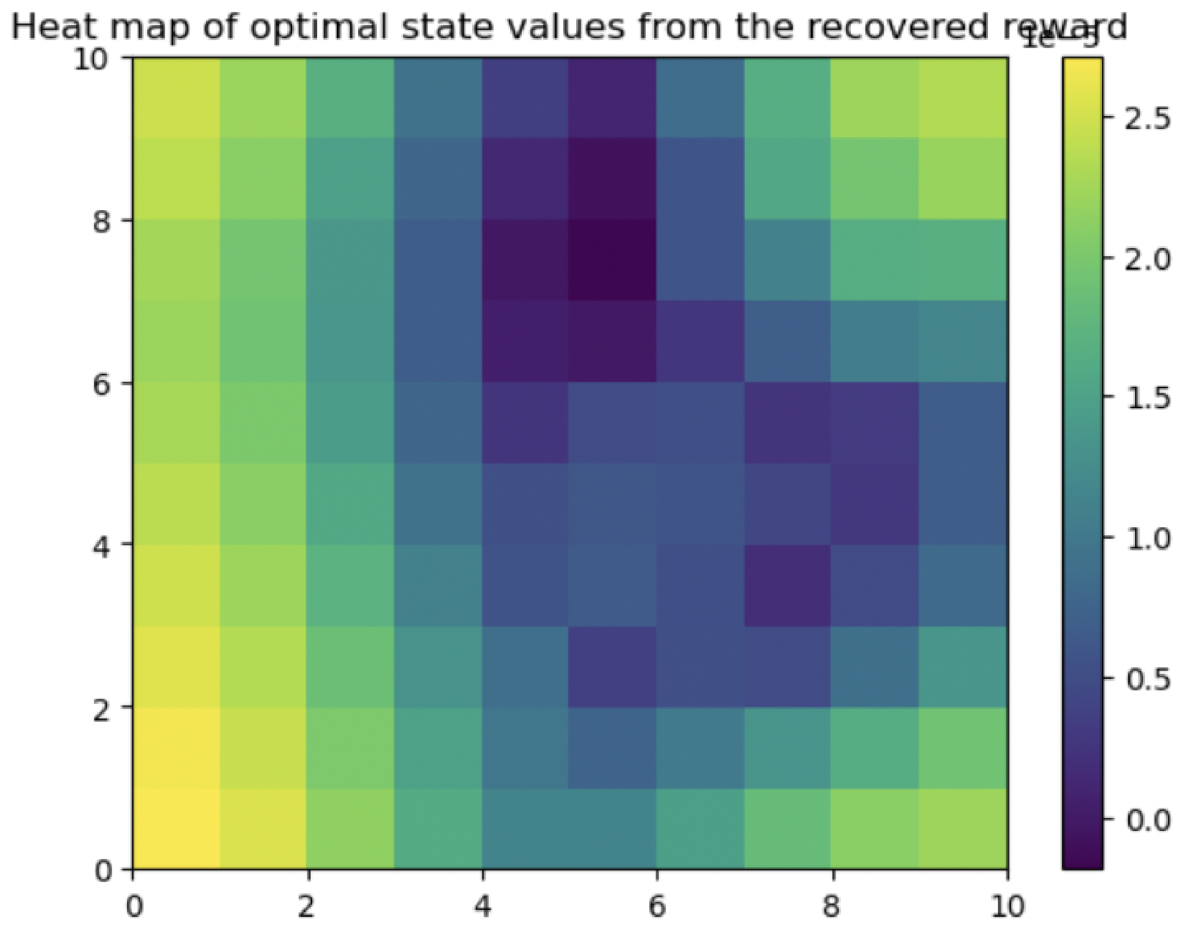


Heat map of recovered reward
10
8
6
4
2
0
0
2
4
6
8
10
0
−20
−40
−60
−80
−100



Heat map of recovered reward
1e−6
10
8
6
4
2
0
0
2
4
6
8
10
4
2
0
−2
−4

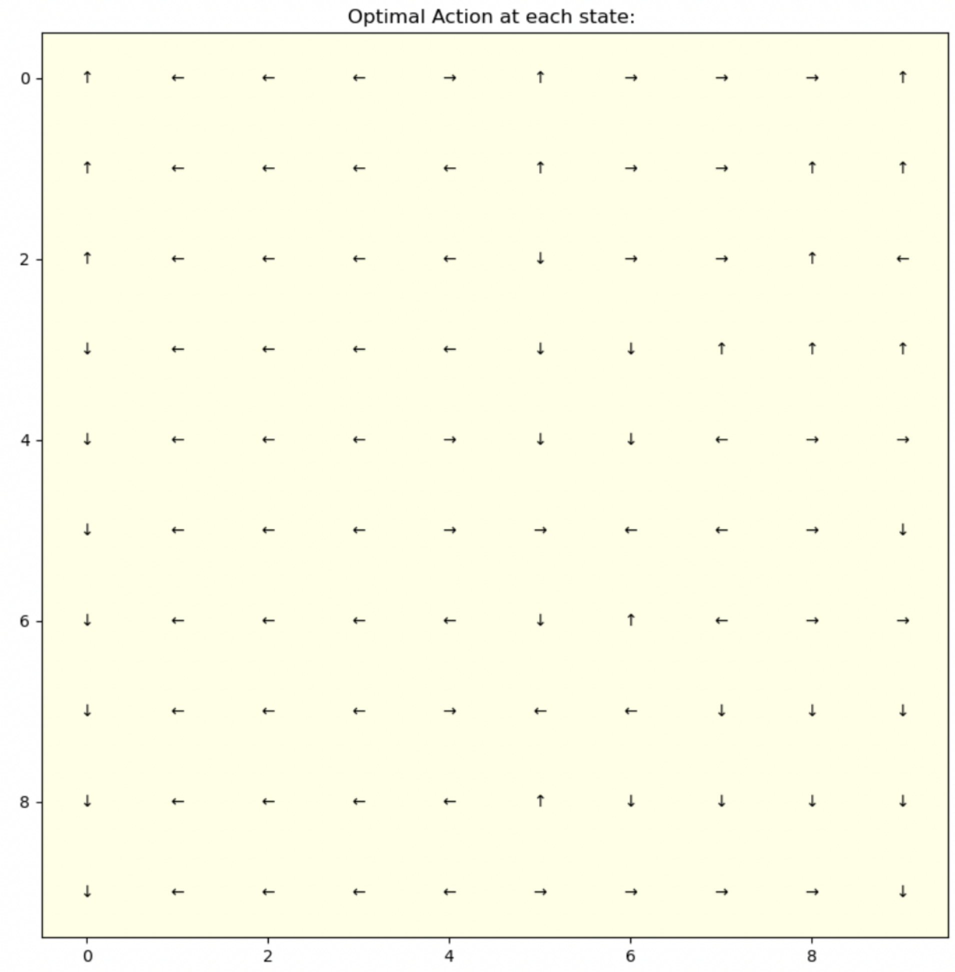
Question 21.



Question 22.

Comparing Q7 and Q21, we can see that there are few similarities. The first is that state 99 has high value in both models. The second is that the agent's reward is related to the optimal value of neighbor states i.e., low reward corresponds to lower optimal value of neighbor states and vice versa. Moving on to the differences, the regions in Q7 are better defined and the scaling is larger.

Question 23.



Question 24.

Comparing the 2 questions Q9 and Q23, we can see that in both cases, the actions are evenly distributed although there are a few arrows diverging away from each other. In q23, it is possible for the agent to move off-grid. In Q9, we see that the agent will definitely reach the optimal state from any given state.

Question 25.

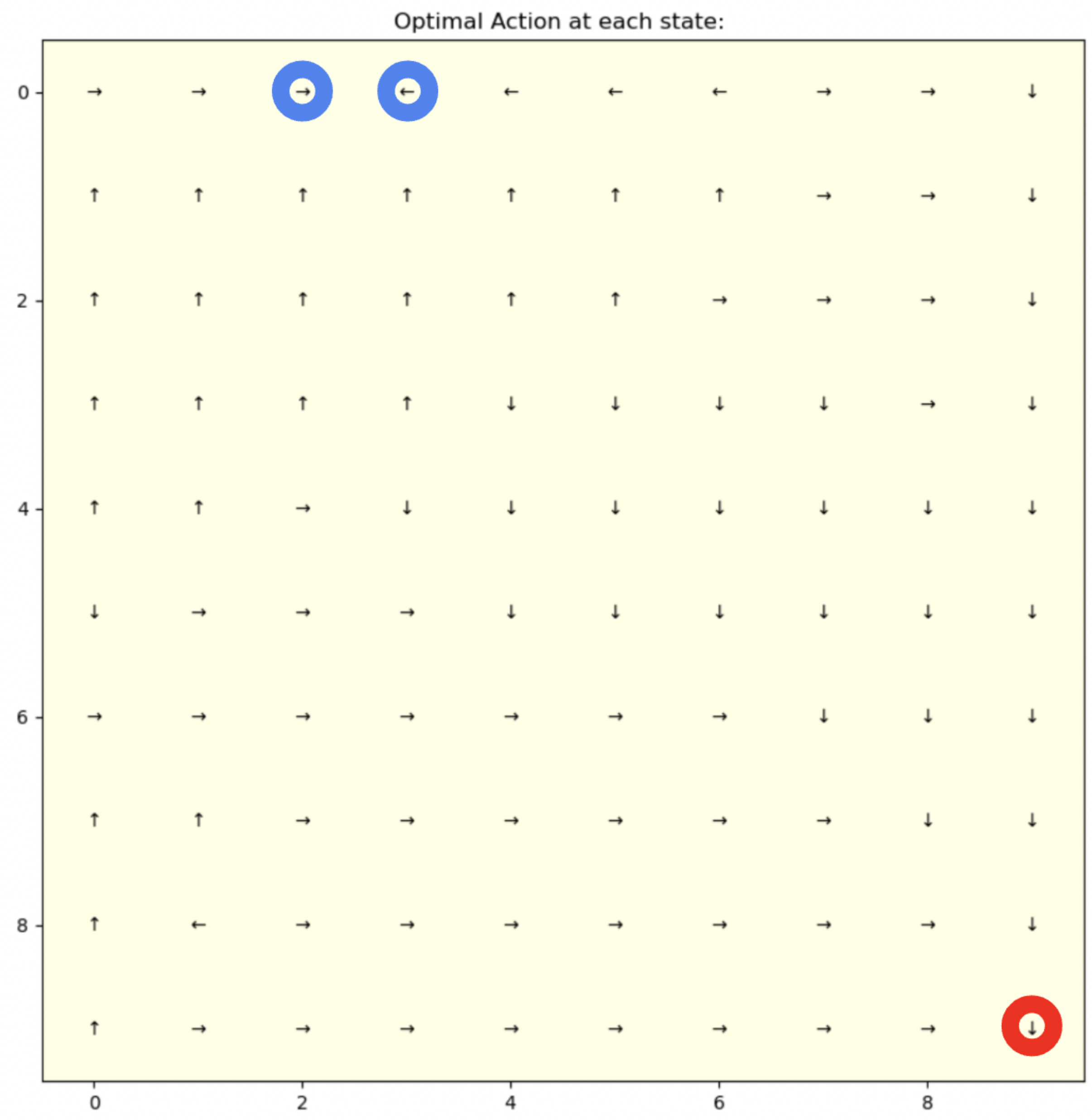


Fig 25 1. Showcasing possible discrepancies in the action decision direction. Observing this through marking on the arrows. Red marking is for discrepancy-1 and Blue markings are for discrepancy-2 defined below.

On observing the action directions (arrows) we can see a few cases where there are major inconsistencies.

Discrepancy 1 of being blown off the grid

Example for this is shown by the red-marker. In this case the wind can blow the agent off the grid. This shouldn't be a valid operation and possible action. Such an action is possible majorly because in these cases the reward function accumulates in a way that moving out of the grid gets a much higher score. This can happen early on in the learning process, as compared to the original policy. But as we notice there is not a lot of states for going off the grid which can be observed in the high accuracy score.

Discrepancy 2 of being stuck in a infinite loop of conflicting actions
Example of this is shown by the blue marker. In this case the action is stuck between states that are essentially a loop. So the actions will keep-going back and forth potentially making the agent be stuck at a single location for a longer period of time.

To directly fix the first problem we hard code the agent to give a really high negative score (maximum negative number in python) when the action blows the agent off the grid. This results in a more guaranteed training where the actions are more optimal towards the edges of the grid. Fixing the case of being stuck in an infinite loop is much harder to fix overall.

Subsection 25 - 1: For reward function 1

For this case (reward 1)
Value of lambda for best accuracy 0.0
Best value of accuracy 0.81

All the plots below are for the function after the fix. In this case we see that discrepancy-1 which was already very low in the base algorithm remains nearly the same. Whereas discrepancy-2 need not necessarily changes.

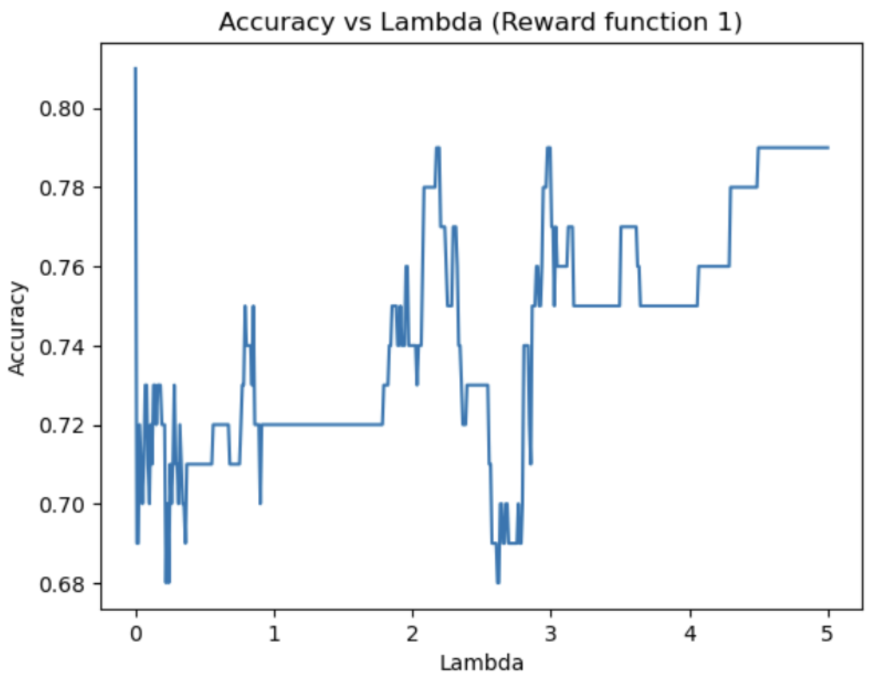
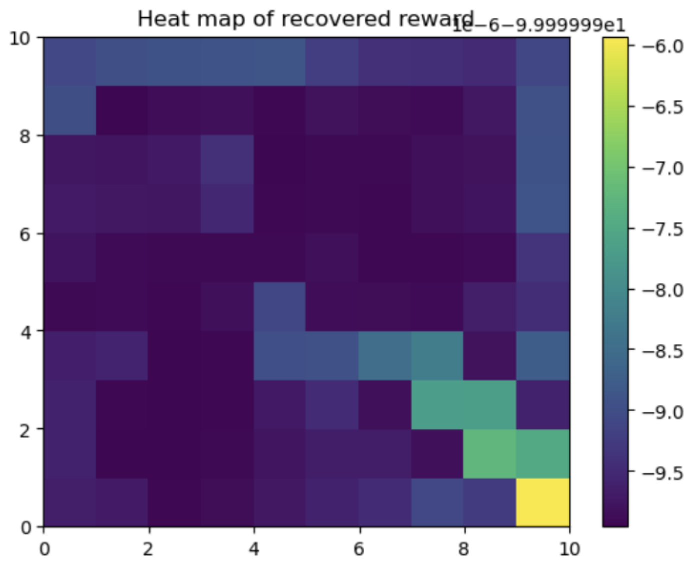


Fig Q25 2. Lambda vs accuracy plot for the reward function 1. This is done to pick the best lambda value



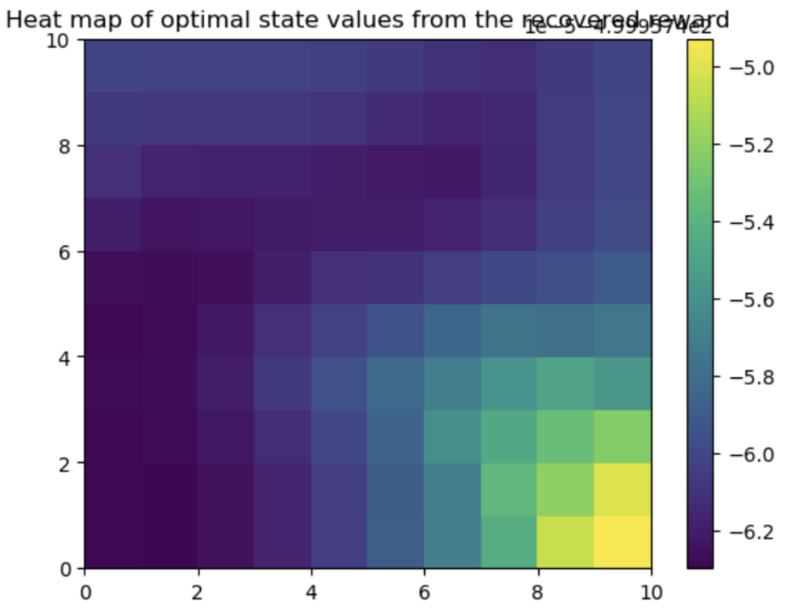


Fig 25 3 Comparing the heat map of recovered reward (top) and heatmap of optimal state values (below)

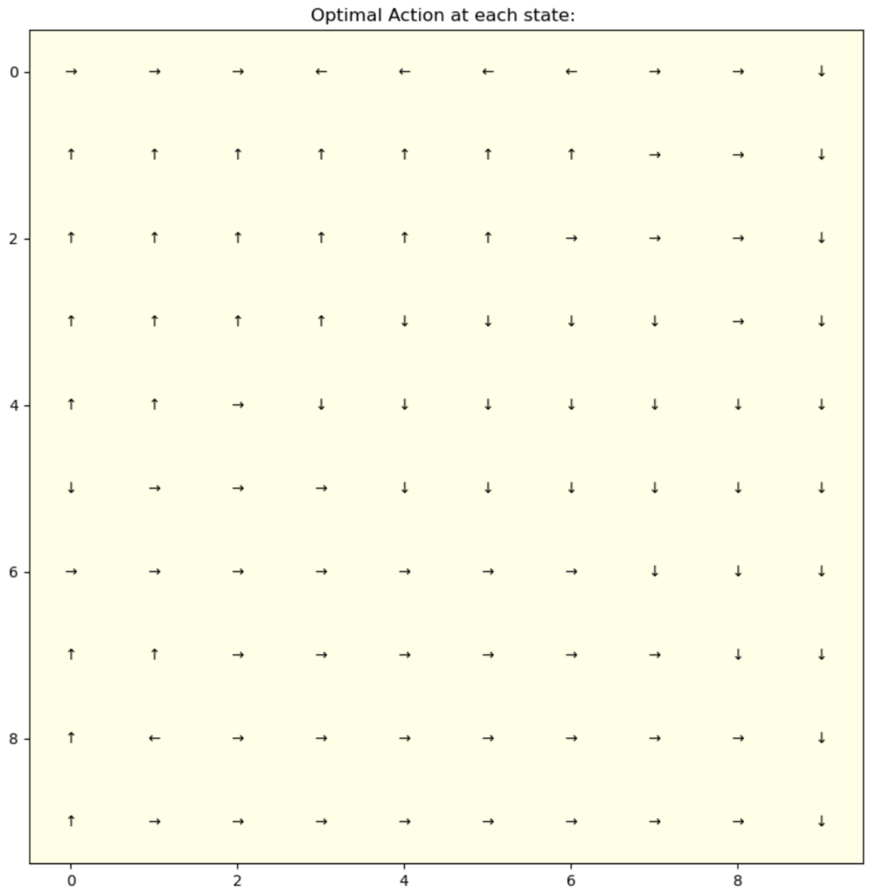


Fig 25 4. Direction of action for each state recovered for reward -1

Subsection 25 - 1: For reward function 2

Value of lambda for best accuracy 0.5410821643286573
Best value of accuracy 0.8

Similar to the case with reward function-1 we see that for reward function-2 discrepancy-1 need not necessarily improve and as this was never meant to fix discrepancy-2 there is not much of improvement. Below are the plots after the fix in the algorithm.

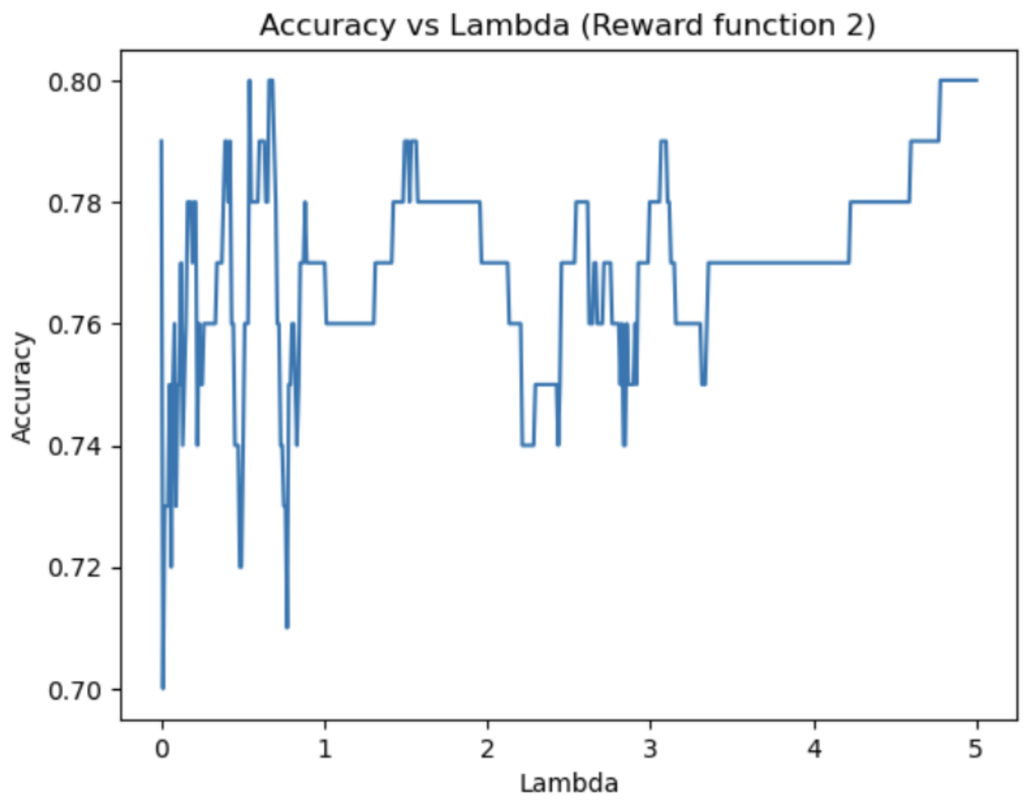
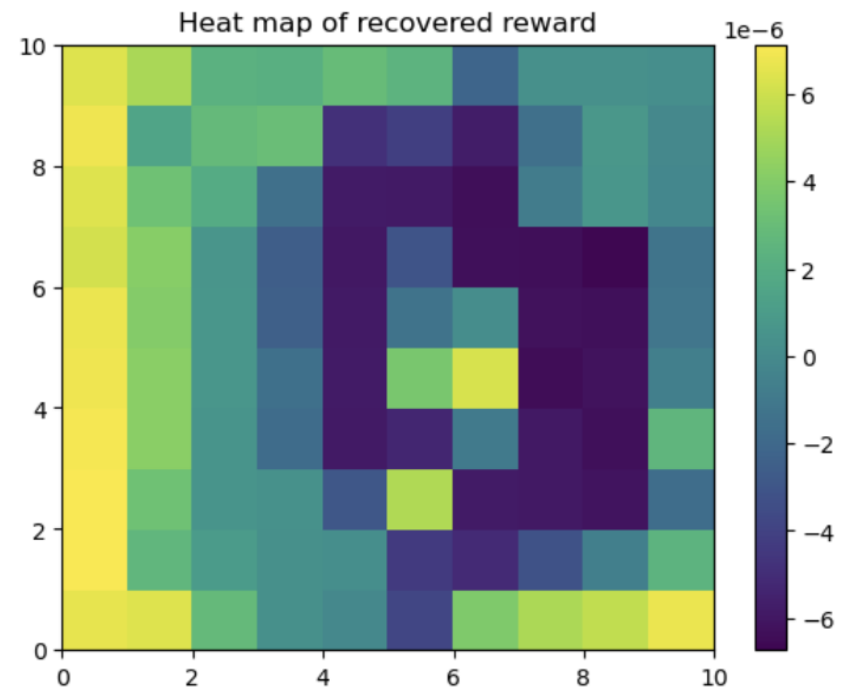


Fig Q25 5. Lambda vs accuracy plot for the reward function 2. This is done to pick the best lambda value.



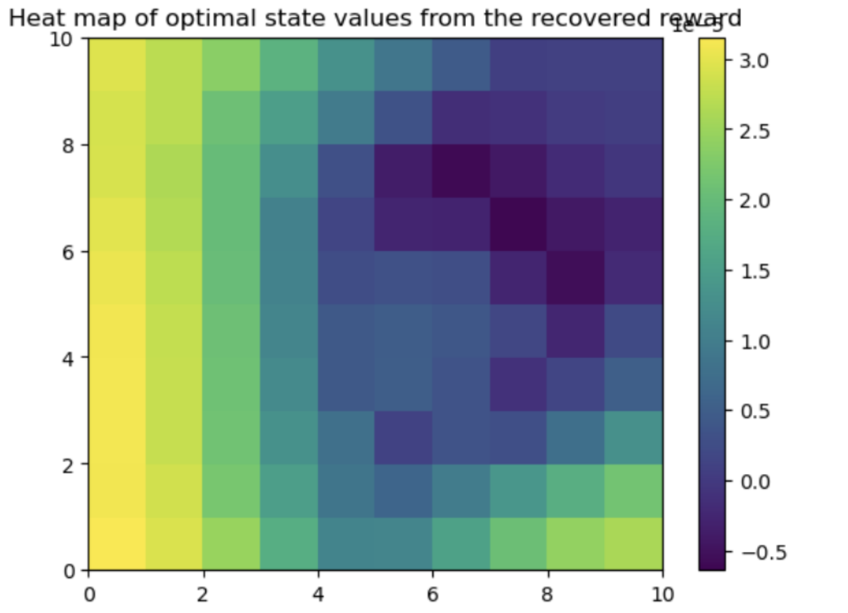


Fig 25 6. Comparing the heat map of recovered reward (top) and heatmap of optimal state values (below)

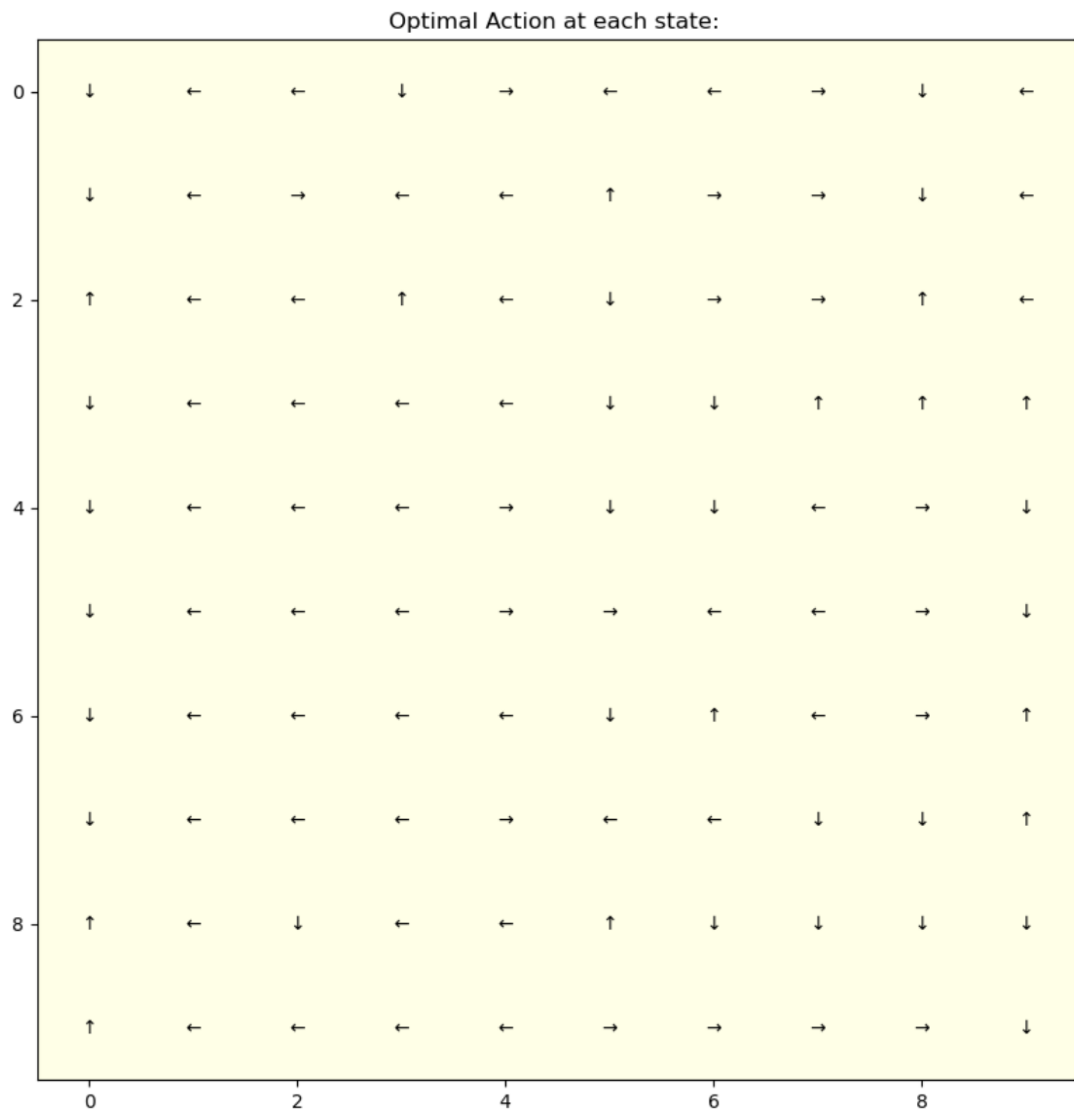


Fig 25 6. Direction of action for each state recovered for reward 2