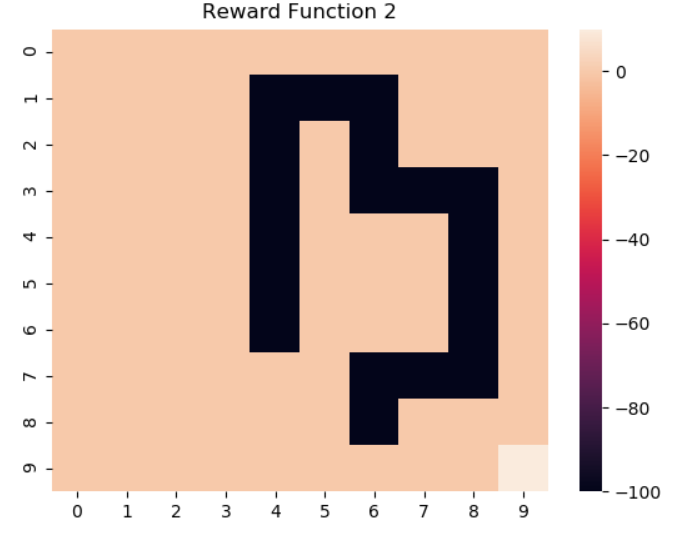
**ECE 232E Project 3**

Group members: Kushagra Rastogi (304640248)

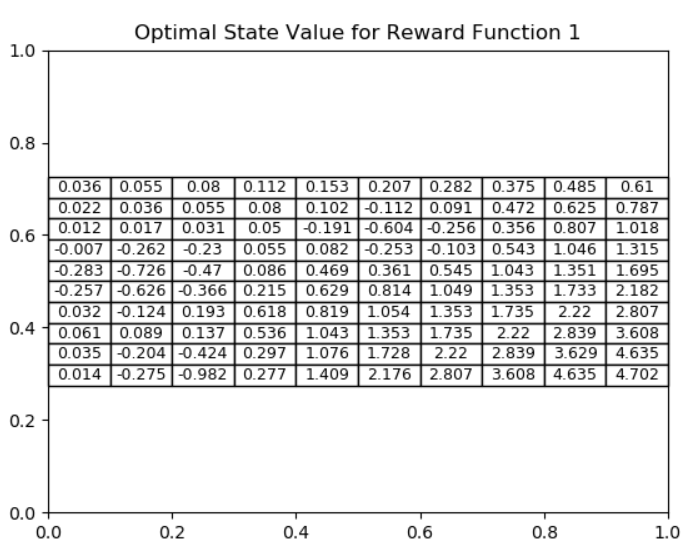
Jonathan Lee (104840173)

**Part I: Reinforcement Learning**

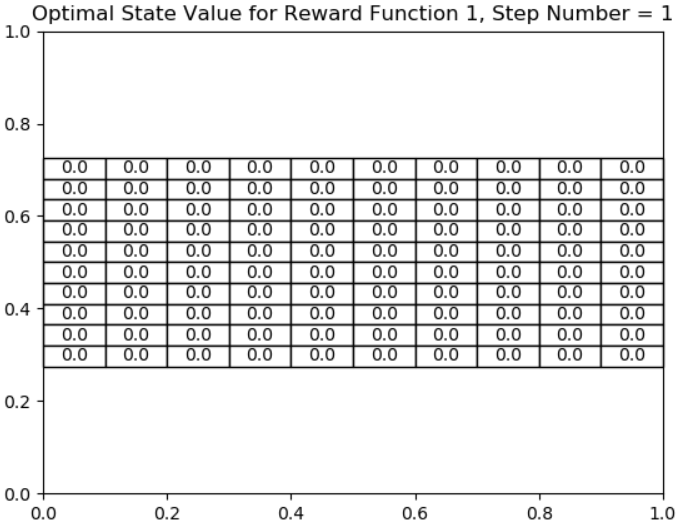
1. Heat maps are shown below.

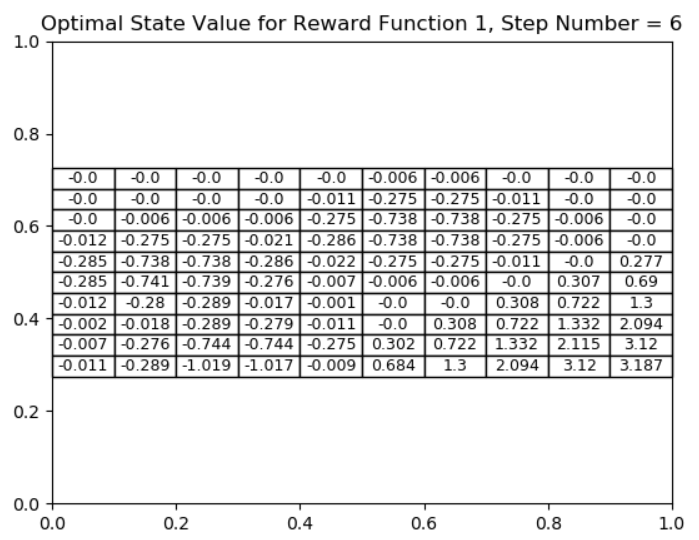


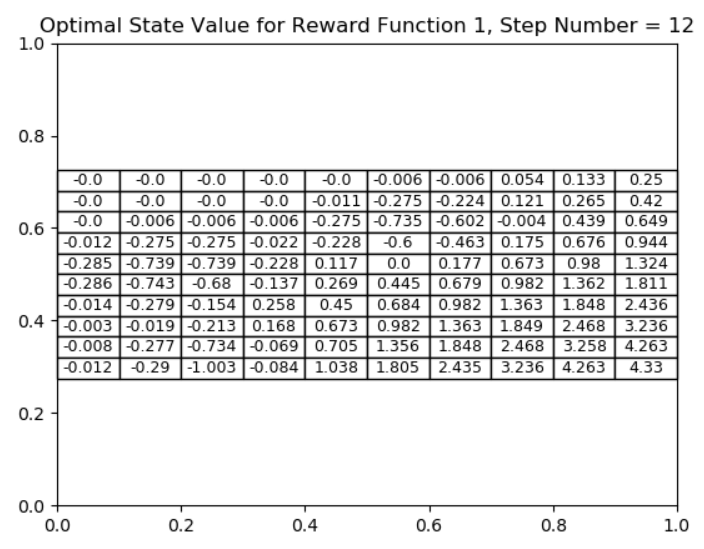
1. The plot is shown below.

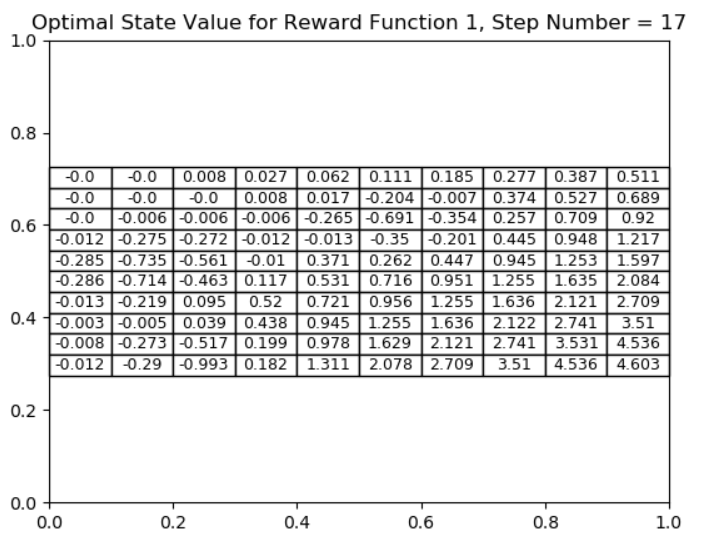


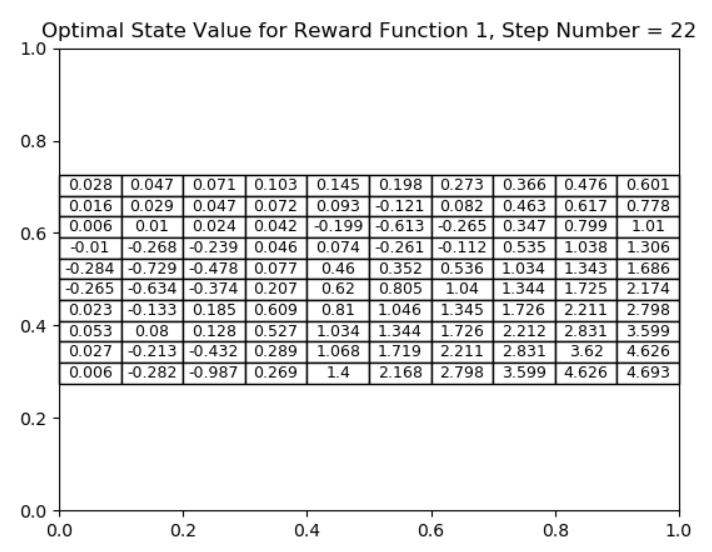
In our case, . The snapshots are shown below.





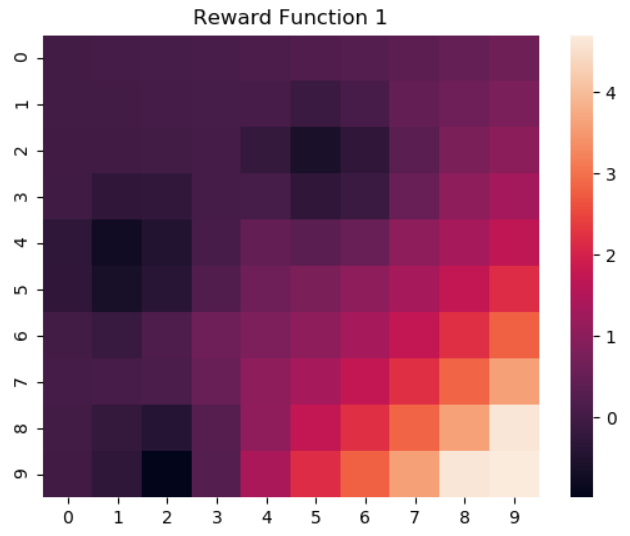




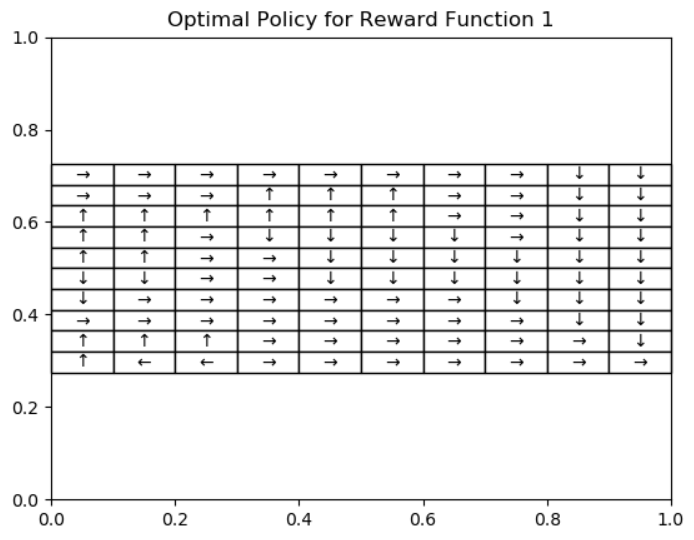


From the snapshots, we can see that in the early stages, the state values are sparse and most of them are 0. Also, the state values start to be changing from the bottom right corner which makes sense since state 99 has the highest reward. Moreover, the block of states with reward -10 have negative values. As the step numbers increase, the state values become less sparse and a clear pattern emerges depicting states that are favorable for the agent. These states have large positive values, unlike the blocks of low reward states that have negative values.

1. The heat map is shown below.



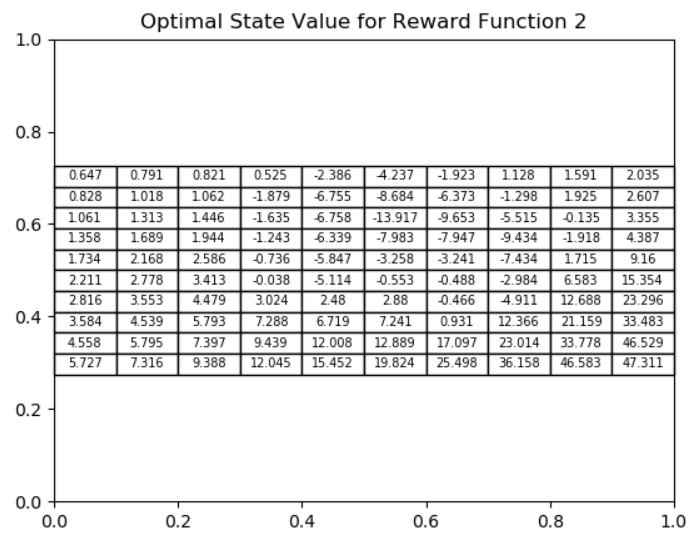
1. As the reward increases, the color gets lighter. For instance, state 99 has the highest reward of 1 and hence the states surrounding state 99 have a light color. On the other hand, the states which had reward of -10 are shaded with a very dark color. Therefore, the color gets darker as the agent moves closer to low reward states and the color gets lighter as the agent moves closer to high reward states.
2. The optimal policy is shown below.



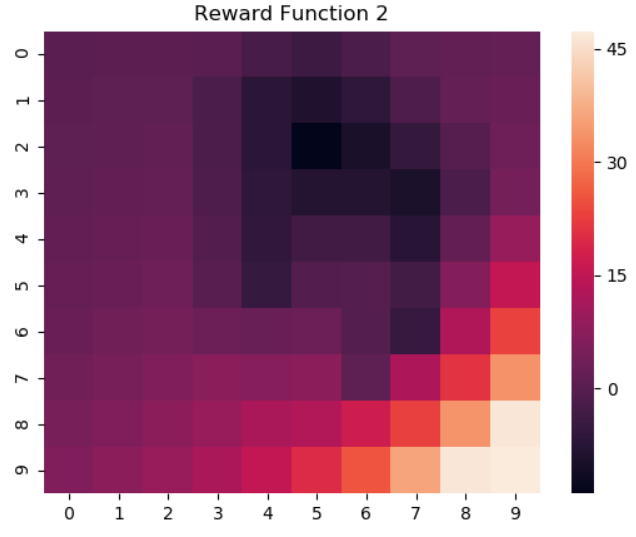
The intuition would be to take actions that maximize the total cumulative reward. The agent should try to move towards the highest reward states. Hence, the optimal policy of the agent matches our intuition since the agent tries to move away from the block of states with reward of -10 and moves closer to state 99 which has the highest reward. Moreover, the general trend of the actions is to move right and down or vice versa. In this way, the agent has the highest probability of reaching state 99.

The optimal policy depends on the transition matrix, discount factor and optimal values of neighboring states. In the value iteration algorithm, we take the action that maximize the reward in all directions by considering the optimal value of the neighboring states. Hence, it is possible for the agent to compute the optimal actions by observing the optimal value of the neighboring states.

1. The plot is shown below.

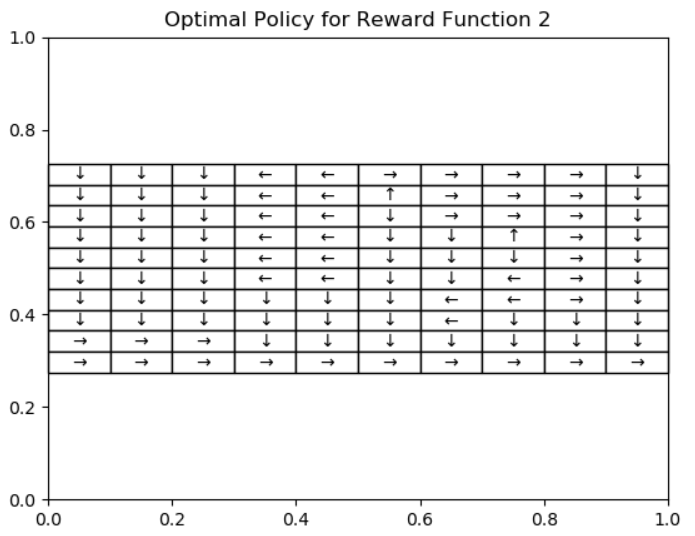


1. The heat map is shown below.



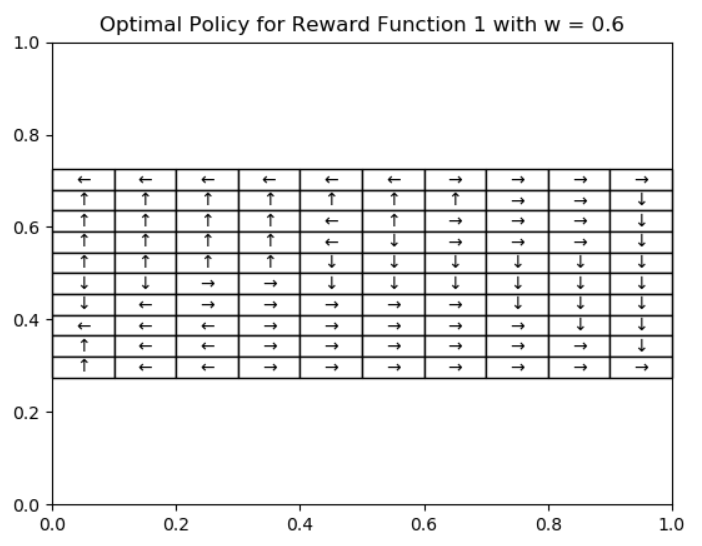
We notice the same trend here as the one in Question 4. State 99 has the highest value and state 52 has the lowest value. Thus, the color becomes lighter as the value of the state increases. Likewise, the color becomes darker as the value of the state decreases. Evidently, there is a clear relationship between the value and reward of a state. The higher the reward of a state, the more valuable it is. Therefore, the ring of low reward states is dark while state 99 has the lightest color.

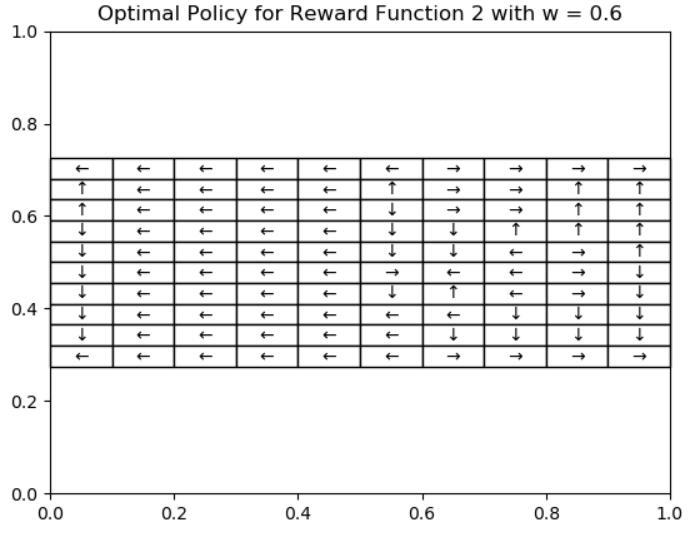
1. The optimal policy is shown below.



The intuition would be to move closer to state 99 and move away from the ring of low reward states. The agent would have to do this by going around the ring of low reward states. Hence, the optimal policy matches our intuition since the agent takes actions that maximizes its chances of reaching state 99. The overall trend is that if the agent is at the top and away from the ring of low reward states, it can move down and right. If the agent is near the ring of low reward states, then it needs to go down/left to get out and then go down and right to reach state 99.

1. The optimal policy is shown below for both reward functions.

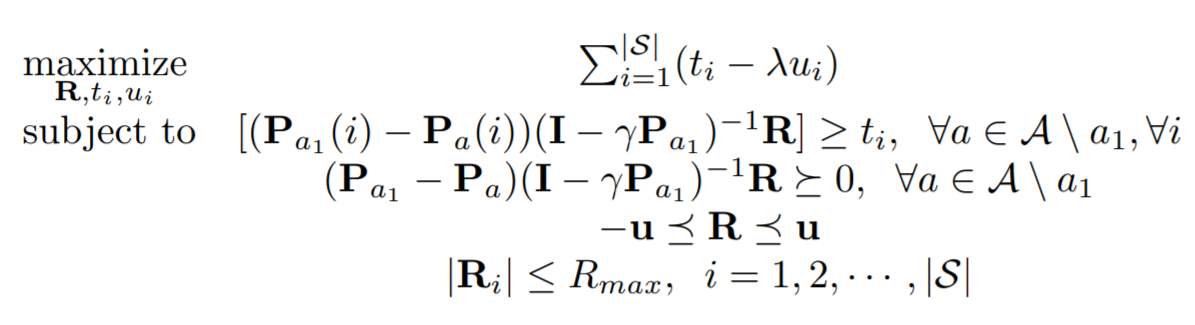




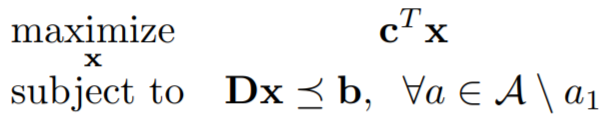
The value controls the amount of exploration that the agent does. The higher the , the more the agent will explore and take random actions. As a result, the values of the states are more spread out meaning that more states take on a whole spectrum of values instead of just the high and low values. Hence, the agent has more than one state that it would like to land on. For instance, according to the policy maps, state 0 and state 99 appear to be high value states for the agent. However, we know that state 99 has the highest reward and value and a higher value of will deter the agent from reaching state 99 since the agent will explore more actions. Therefore, gives the better optimal policy since it allows the agent to do some exploration while helping it to reach state 99 which is the highest value state.

**Part II: Inverse Reinforcement Learning**

1. For this portion of the assignment, we explore IRL algorithms for extracting reward functions and hence optimal policies for finite state spaces. One such algorithm we delve into is the linear programming (LP) formulation. The problem statement of the IRL is shown below in LP formulation:



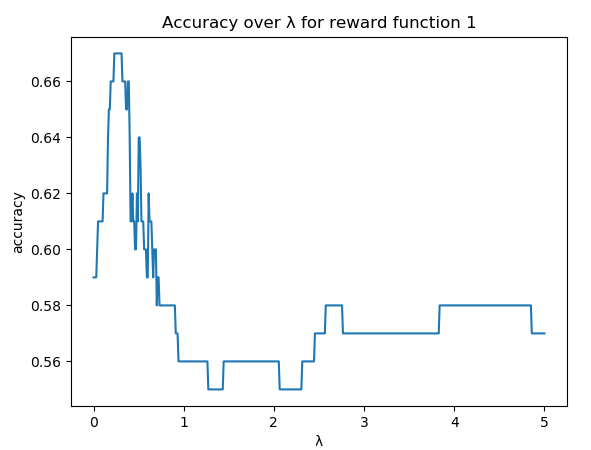
where Pa is the transition probability matrix of taking action a, |S| is the total number of states, and λ is the penalty coefficient. The variables in this problem are the extra optimization variables, t and u, along with the reward vector, R. The variable of interest in this problem is R, as we will be using the solution to compute a policy from methods mentioned in the previous part. In order to tackle this LP, there is a simpler formulation that we will work with:



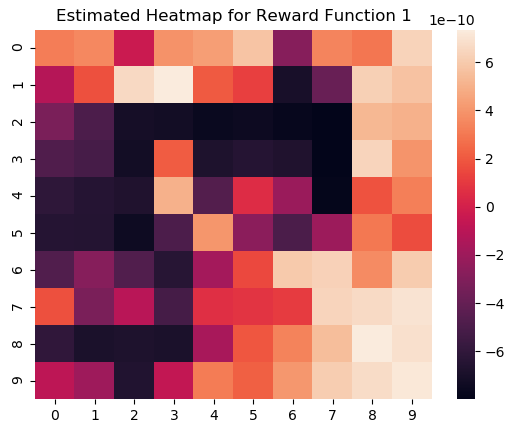
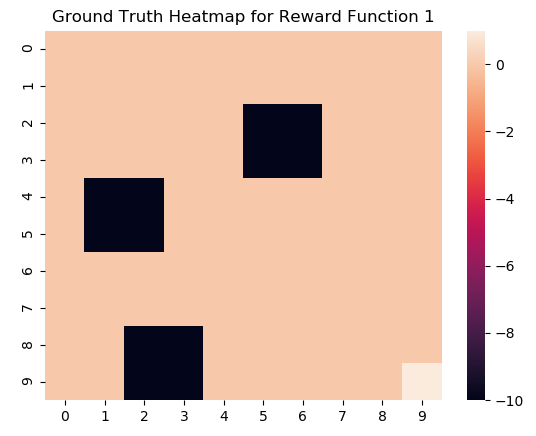
where

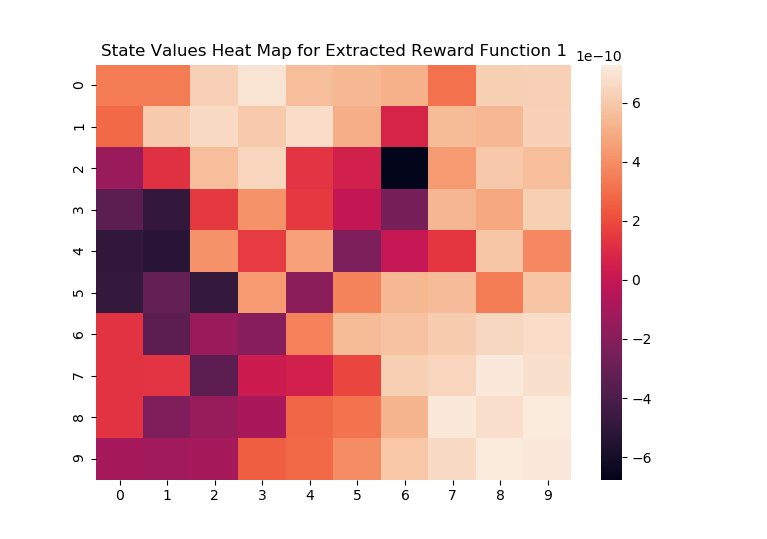
In the next sections, we will observe how well the extracted reward function resembles the optimal reward function. Consequently, we measure how accurate is the generated policy of the extracted reward function, Oa, from the optimal policy, Oe. In other words, we examine how closely the “agent” behaves with respect to the “expert.” We do this by testing 500 evenly spaced values from 0 to 5 of λ. The next question will demonstrate the accuracy of these values.

1. The accuracies for evenly spaced lambdas from [1, 5] are shown below for reward function 1:



1. The maximum accuracy over all lambdas for reward function 1 is 67% occurring at
2. The heat maps for the ground truth (left) and the estimated truth (right) of reward function 1 are shown below:

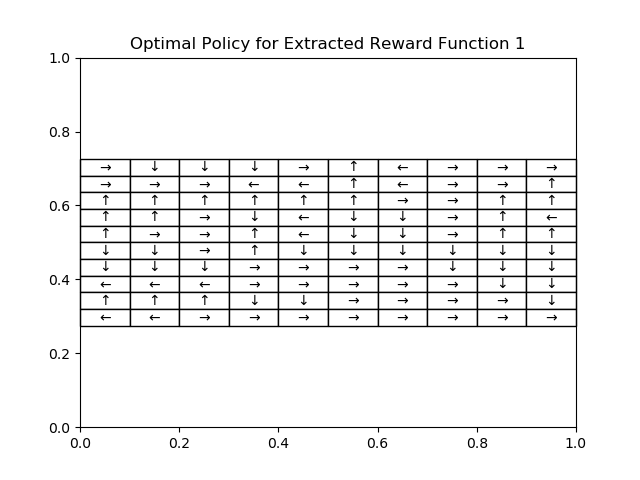




1. Similarities:
   1. The highest valued states are around state 99 which makes sense since state 99 has the highest reward in the original reward function.
   2. As the value of a state decreases, the shade of the state also decreases. There are 3 well-defined low reward regions in the original reward function. The extracted reward function also displays this feature, albeit to a lesser extent. There is a distinctive region of low reward states on the right side and there are two regions on the left side that are slightly merged but are distinctive enough to be considered independent.

Differences:

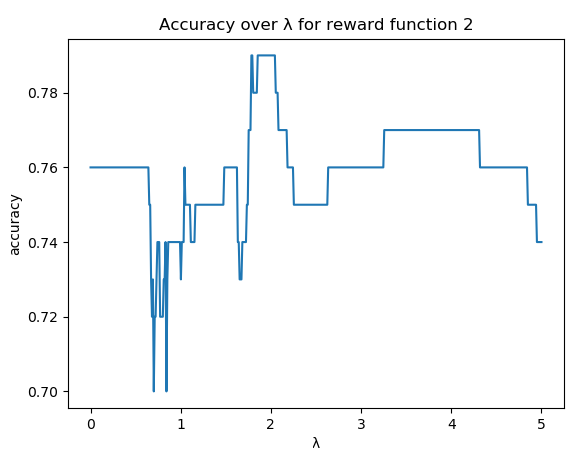
1. The scale of the rewards is much different. In the original reward function, the highest and lowest rewards were 1 and -10 respectively. In the extracted reward, the rewards are on the magnitude of 1e-10. Thus, the extracted rewards are on a much smaller scale.
2. The original reward function had only one high-valued region in the bottom right corner. In the extracted reward function, there are two high-valued regions: one in the bottom right corner and one in the top left half of the grid.



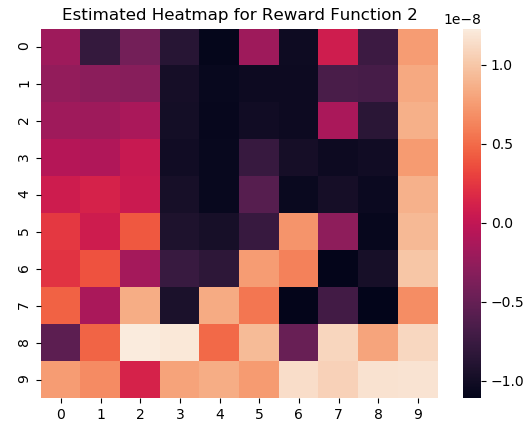
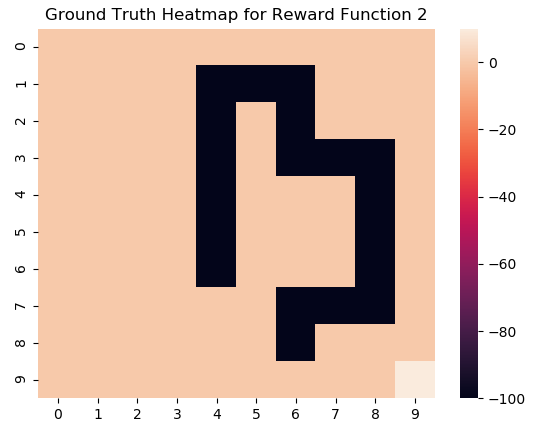
1. Similarities:
   1. For both the reward functions, the agent tries to go to state 99 since it is the highest valued state. In both cases, the agent tries to do this by taking right and down actions for the majority of its moves.

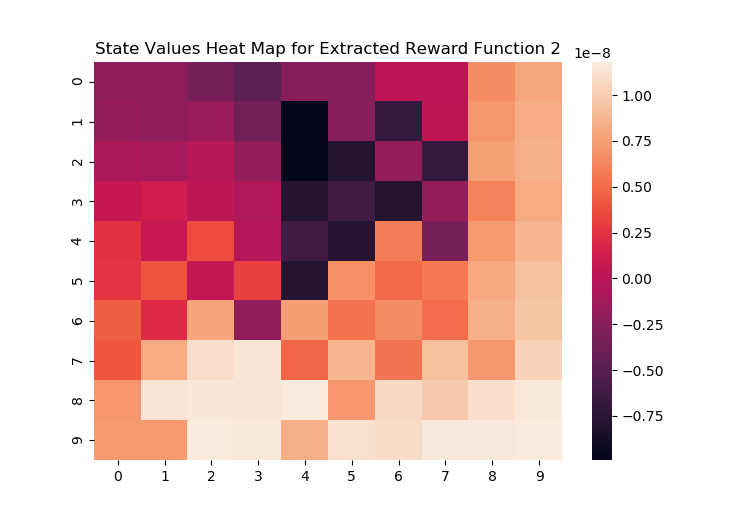
Differences:

1. In the extracted reward function, there are a few states where the optimal action makes the agent move out of the grid. For instance, in state 9, the optimal action is left but this would make the agent jump out of the grid.
2. In the extracted reward function, there are a few states which can be described as local optima and the agent cannot escape from this local optima. For instance, states 21 and 31 have arrows pointing towards each other. This means that the agent will toggle between these two states forever.
3. The accuracies for evenly spaced lambdas from [1, 5] are shown below for reward function 2:



1. The maximum accuracy over all lambdas for reward function 2 is 79% occurring at
2. The heat maps for the ground truth (left) and the estimated truth (right) of reward function 2 are shown below:

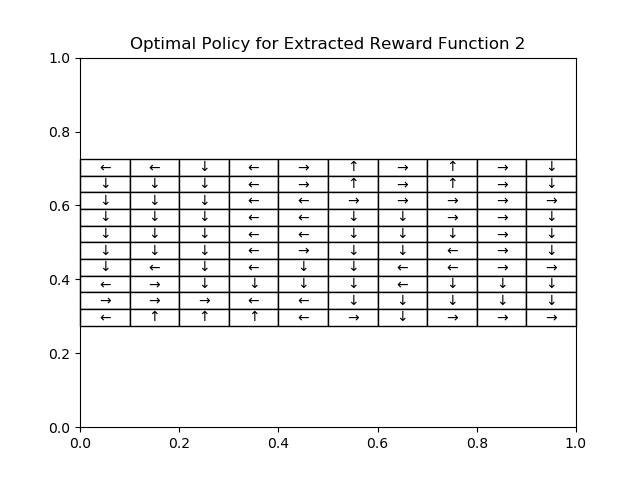




1. Similarities:
   1. State 99 continues to be a high value state in both reward functions. In both reward functions, the shade of the state becomes lighter as the value of the state increases.
   2. The position of the low value region is roughly the same in both reward functions.

Difference:

1. Unlike the original reward function, the extracted reward function has two regions of high values states: one in the bottom right corner and one in the bottom left half.
2. The reward magnitude is quite different. In the extracted reward function, the values are on the magnitude of 1e-8 whereas the original reward function was on the magnitude of 1e2.

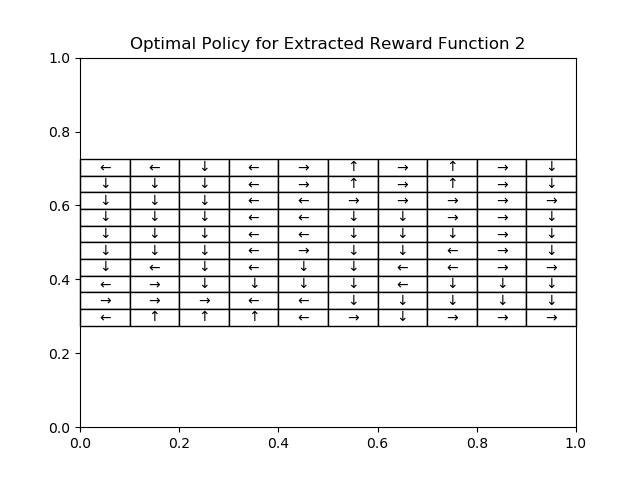


1. Similarities:
   1. In both reward functions, the agent is trying to move to high value states like state 99. It does this by primarily taking down, left and right actions. In both reward functions, the agent uses all four directions to reach the optimal state.
   2. There are a few states where two states are pointed away from each other. For instance, states 76 and 86 are pointed away from each other. This could mean that the two states are not compatible with each other.

Differences:

1. For the extracted reward function, there are a few states where the optimal action makes the agent move out of the grid. For instance, the state 7 has a left action which would make the agent move out of the grid. This phenomenon does not occur in the original reward function.
2. For the extracted reward function, there are a couple of states that are pointed towards each other. For instance, states 28 and 38 have arrows pointing towards each other. This signifies a local optima and the agent gets stuck here since it will just be oscillating between the two states. This is a shortcoming of solving an optimization problem using iterative methods.
3. From the figure given in question 23, it is observed that there are 2 major discrepancies that are limiting the performance of the extracted reward functions:
4. The estimated optimal policy for several boundary states perform moves that bring them off the grid (an “off-grid” move)
5. Depending on where the agent starts on the grid, it may fall into an inescapable local optimum.

Both discrepancies are illustrated below for reward function 2 (Q23):



Off-grid moves

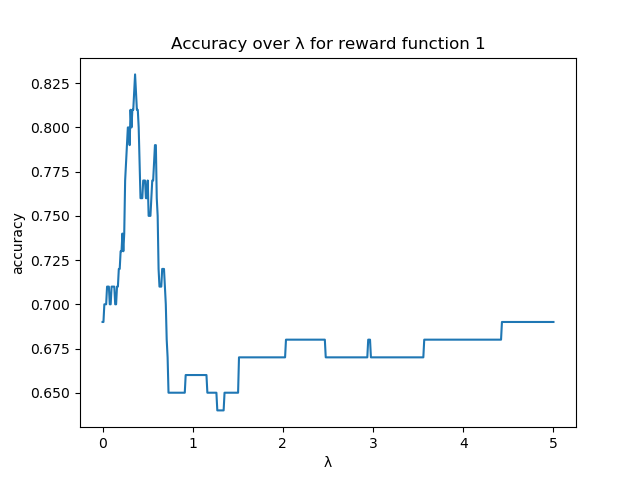
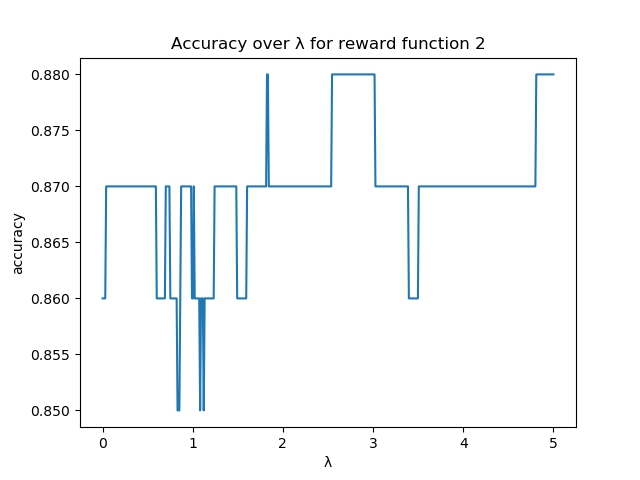
Local optima

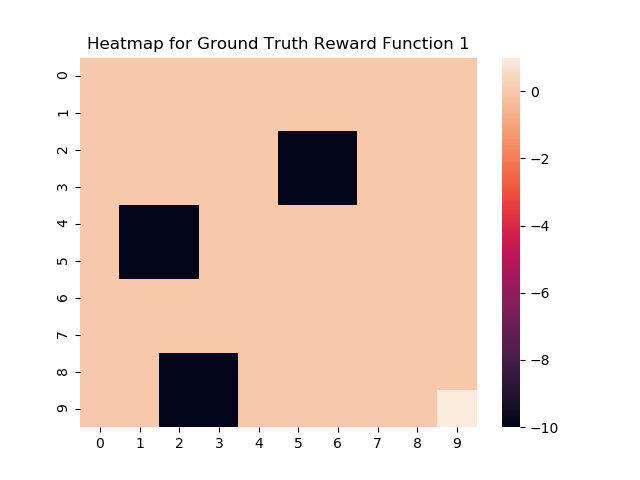
The former is a simple fix. The latter, however, requires a more complex approach/algorithm than the current IRL algorithm. There were two noticeable improvements we made to the algorithm.

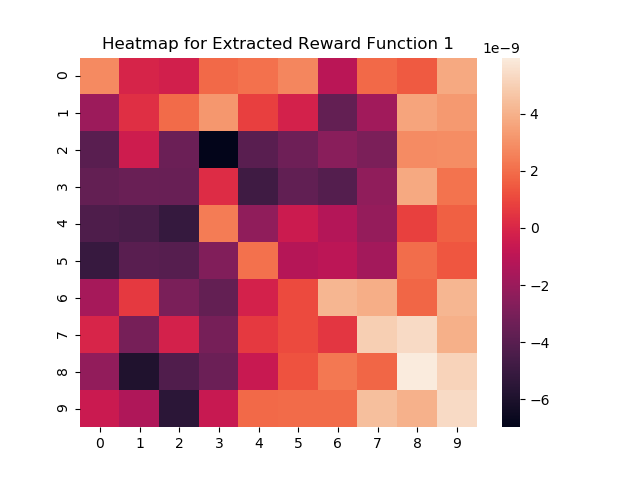
First, we tackled the problem of removing “off-grid” moves by setting the states values of boundary/edge states to be -∞. As a result, the optimal policy method will have to choose the “best in-grid” move to ensure that each edge state will remain inbounds when performing the estimated optimal policy.

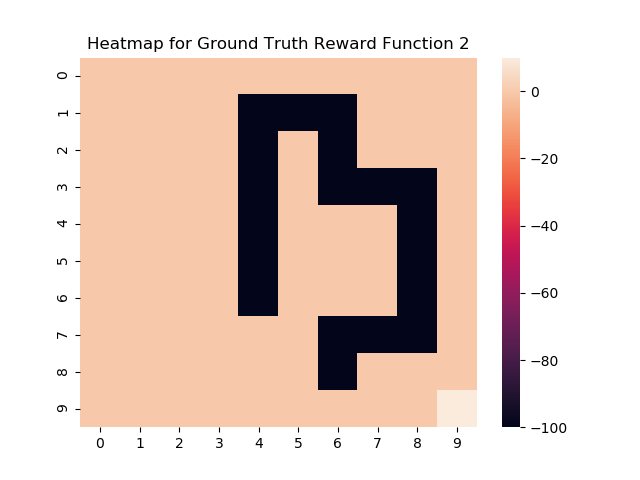
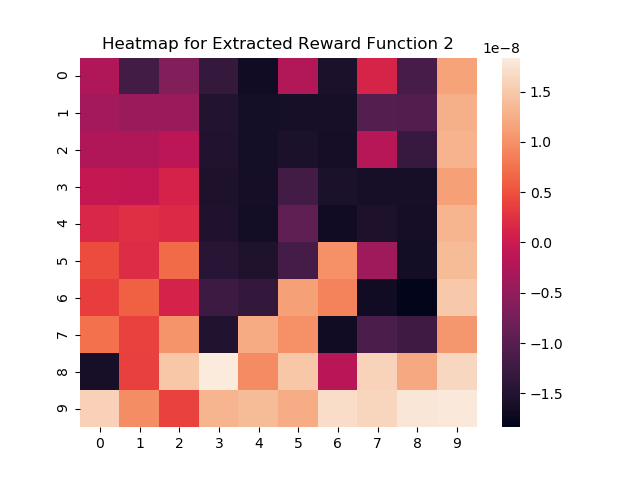
The second improvement we made was decreasing the error tolerance level, ε, by several orders of magnitude. The original assignment was to set ε to 0.01. For the purposes of question 25, we decreased ε to 10-12. This, in turn, causes the Value Iteration algorithm presented in the report to perform more iterations for a more accurate optimal policy.

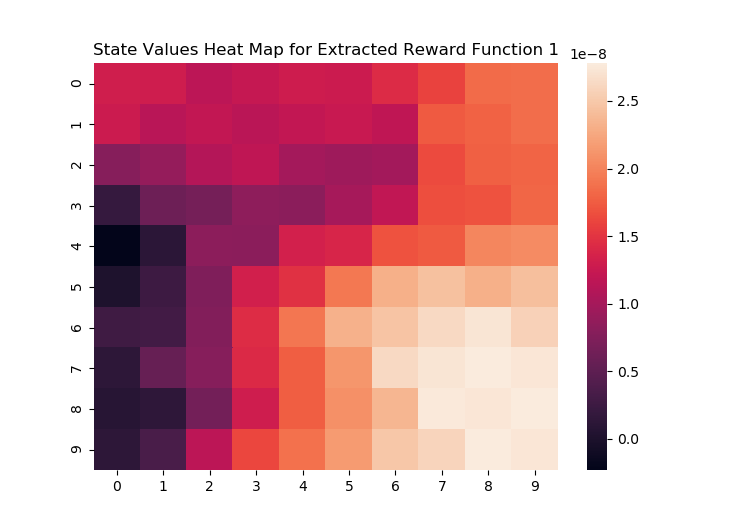
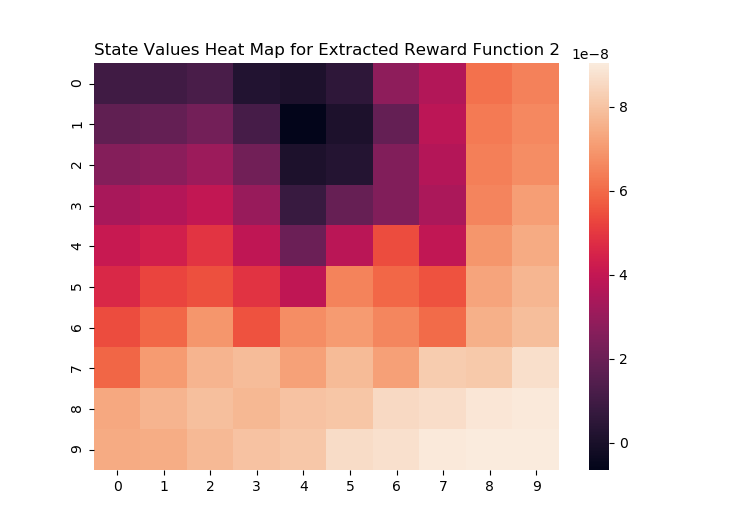
Incorporating these two improvements and rerunning the IRL algorithm led to the results shown below.



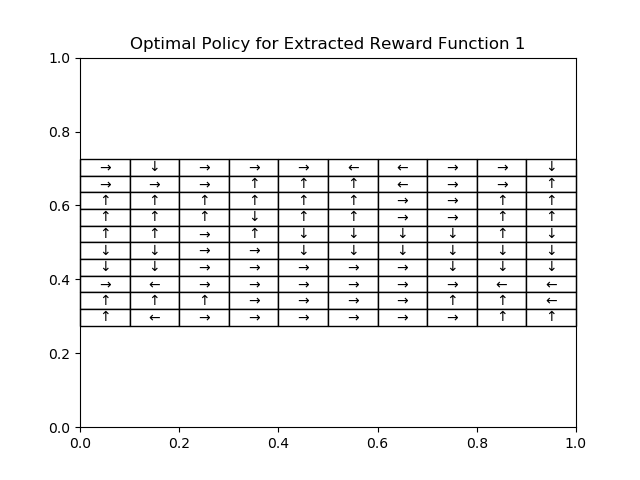


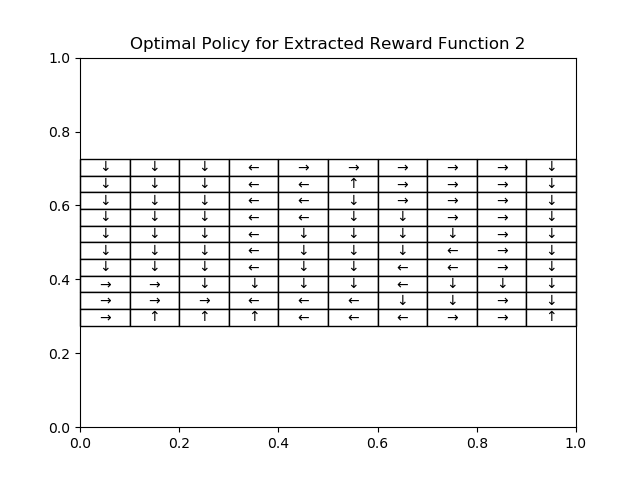






As it can be seen, the figures illustrated above imply a significant increase in performance compared to the original IRL algorithm. For instance, the range of accuracies improved from a range of [55%, 67%] to [64%, 83%] for reward function 1 and a range of [70%, 79%] to [85%, 88%] for reward function 2. In addition, the original reward functions seem to stabilize at approximately 57% and 76%, respectively.





With the improvements implemented, the reward functions now stabilize at around 70% and 87%, respectively, resulting in a 13% increase for reward function 1 and an 11% increase for reward function 2.

When comparing the heatmaps for the ground truth of the reward functions to the “improved” extracted reward functions, they are more noticeably similar in resemblance than the comparison between the ground truth and the “original” extracted reward functions. For example, for the improved reward function 1, it is now easier to distinguish between the states that have negative immediate rewards to the ones that don’t by observing the darker/black grids. For the original reward function 1, the dark grids are more smeared/spread out, making it harder to determine the optimal policy given this function.

As for the optimal policies generated from the improved IRL algorithm, it is clear that the first discrepancy of “off-grid” moves has been eradicated, i.e. all of the state moves from the boundaries now take the states to a future state that is within the grid.

In conclusion, although the first discrepancy is easily avoided through the alteration of the Value Iteration algorithm, the discrepancy of getting stuck at local optima is a much harder task to mitigate and is outside the scope of this assignment. This simple IRL algorithm will not be able to account for escaping local optima, but perhaps a change in the formulation of the linear program may help in this regard.