**ECE232E - Project 3**

**Reinforcement learning and**

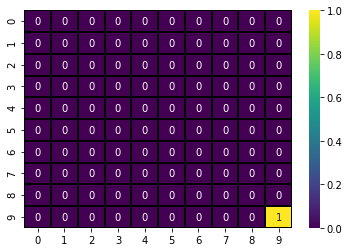
**Inverse Reinforcement learning**

*Zhechen Xu (805030074), Yufei Hu (404944367), Qi Zeng (204946904), Liangkun Zhao (204947003)*

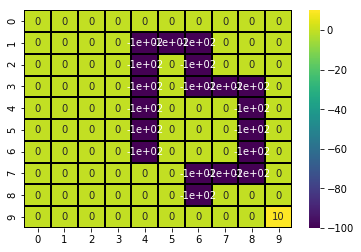
*May 20, 2018*

# **Part 2: Reinforcement learning (RL):**

**Q1:**



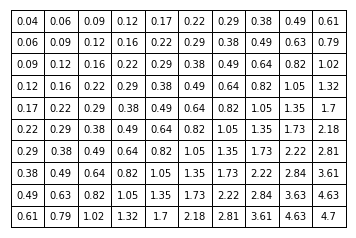
**Figure 1:** The heat map of Reward function 1



**Figure 2:** The heat map of Reward function 2

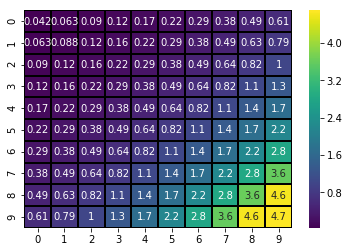
# **Part 3: Optimal policy learning using RL algorithms:**

**Q2:**



**Figure 3:** Optimal state value plot with reward function 1

**Q3:**



**Figure 4:** Heat map of the optimal state value with reward function 1

**Q4:**

As we can see in the heat map, the up left corner has the smaller values while the bottom right corner has the bigger values. At the same time, walking along the diagonal from up left to bottom right, the values become more and more big. The reason is that, according to the reward function 1, the only reward the agent can get is at the bottom right corner, which is 1, while the others are all 0. And we want to train an agent who can find the rewarded position (seek the greater value). So, the values are bigger when their positions are closer to the bottom right, which could lead the agent find the optimal position and get the final reward with minimum steps. Larger value indicates closer distance to the optimal state.

**Q5:**



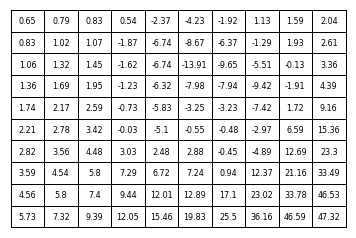
**Figure 5:** Optimal action plot with reward function 1

As we can see in the policy map, each arrow indicates the optimal action of the agent at different positions. Following the policy, our agent tends to use minimum steps to get to the final optimal position (bottom right).

For this particular task, the optimal action at most of the states can be determined by its neighbors except for the bottom right corner states ([9, 10], [10, 9], [10, 10]). Because, for this task, the reward function is mostly zero while the only positive value of it is at the bottom right corner, which makes the computation of the best value only depends on its neighbors  for:

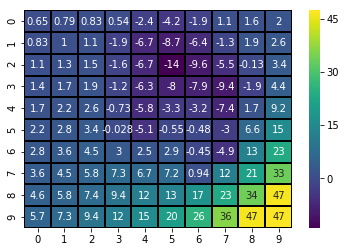


**Q6:**



**Figure 6:** Optimal state value plot with reward function 2

**Q7:**

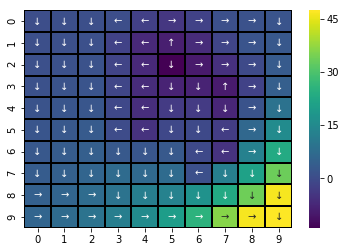


**Figure 7:** Heat map of the optimal state value with reward function 2

**Q8:**

Similar as the one with reward function 1. After the training process, we want the agent to find the final optimal position, which is, for reward function 2, the bottom right as well. Different from the reward function 1, reward function 2 has some negative rewards (penalty), which we don’t want our agent to get to those positions. So, the optimal values computed around those positions are negative. For the others, they have the same distribution as the previous one: walking along the diagonal from up left to bottom right, the values become more and more big. Larger value indicates the closer distance to the optimal state, while the negative values indicate the penalty.

**Q9:**

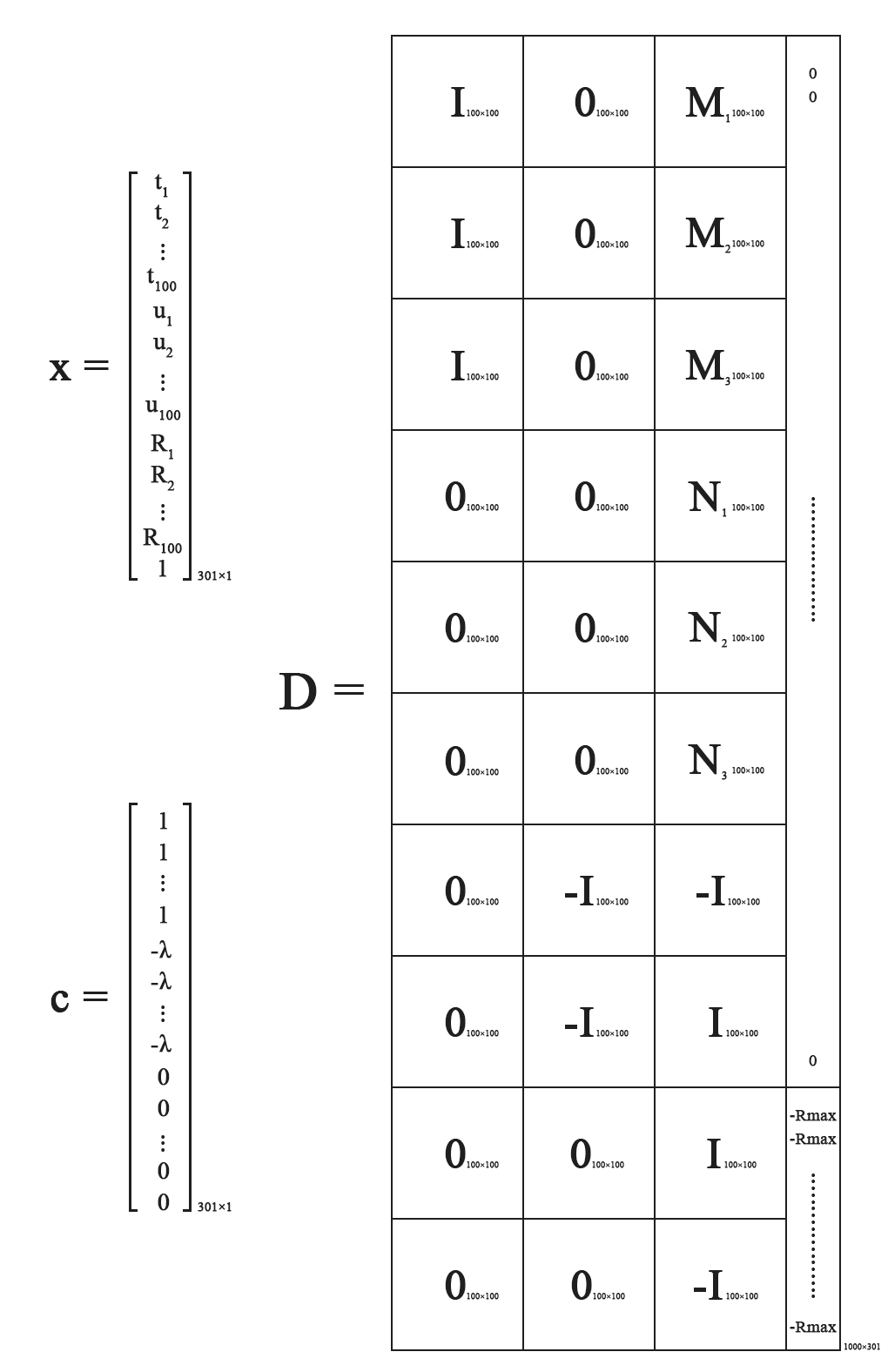


**Figure 8:** Optimal action plot with reward function 2

Yes, this policy matches my intuition. Following the policy, our agent tends to use the minimum steps to find the optimal state (bottom right). And it also could try to avoid those states with penalty. (If agent at the penalty state, the policy for that state will lend it jump out of the penalty area)

# **Part 4: Inverse Reinforcement learning (IRL):**

**Q10:**



**Figure 9:** Reformulation of the original LP formula

**x**, **c**, and **D** are clearly defined in the figure above. In c, there are 100 ones at the top, followed by 100 *–λ* and 101 zeros. In D, **I** refers to an identity matrix and **0** refers to an all-zero matrix with their dimensionalities denoted in the subscript. **Mi** is defined as follows:

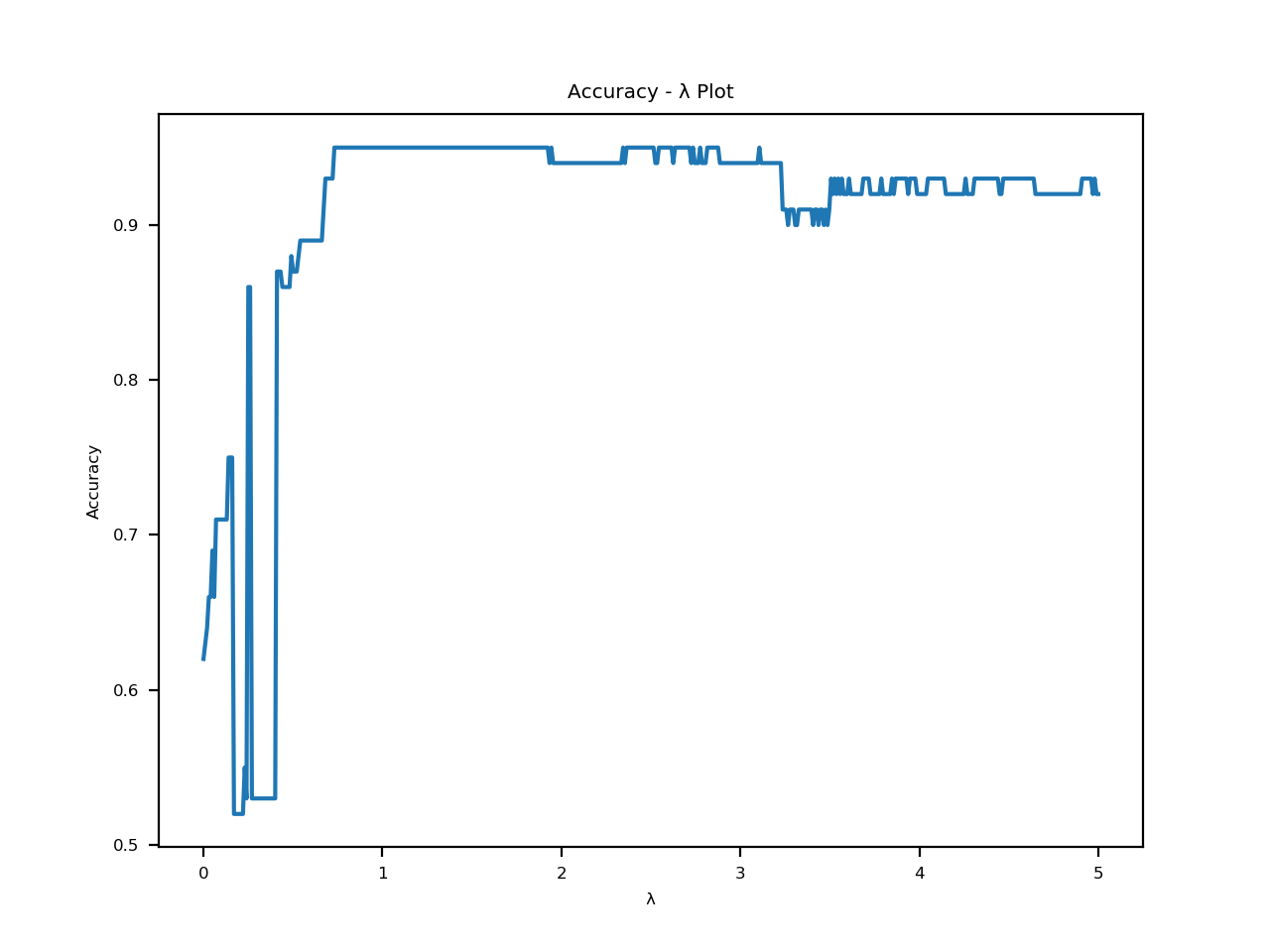


where j represents an entire row of matrix **M**.

Similarly, **Ni** is defined as:



**Q11:**

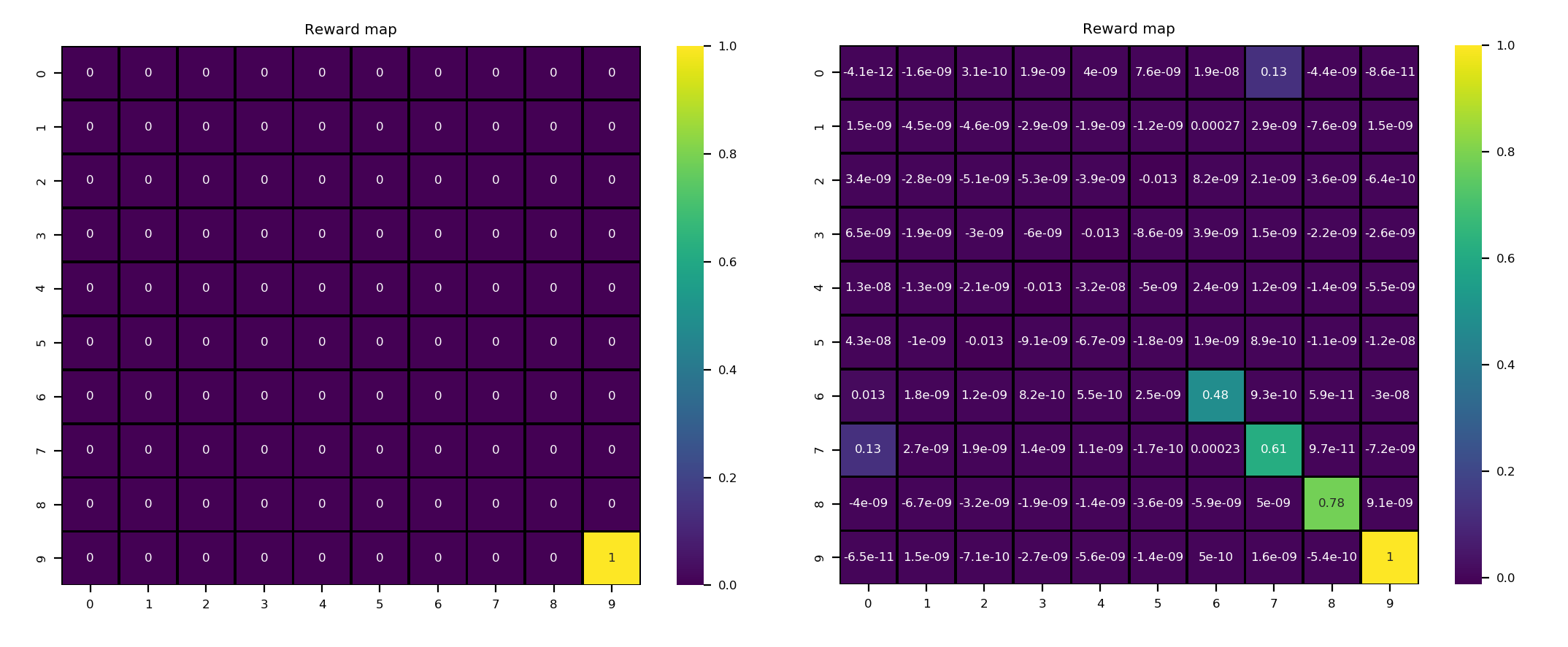


**Figure 10:** Accuracy along with different λ using reward function 1

**Q12:**

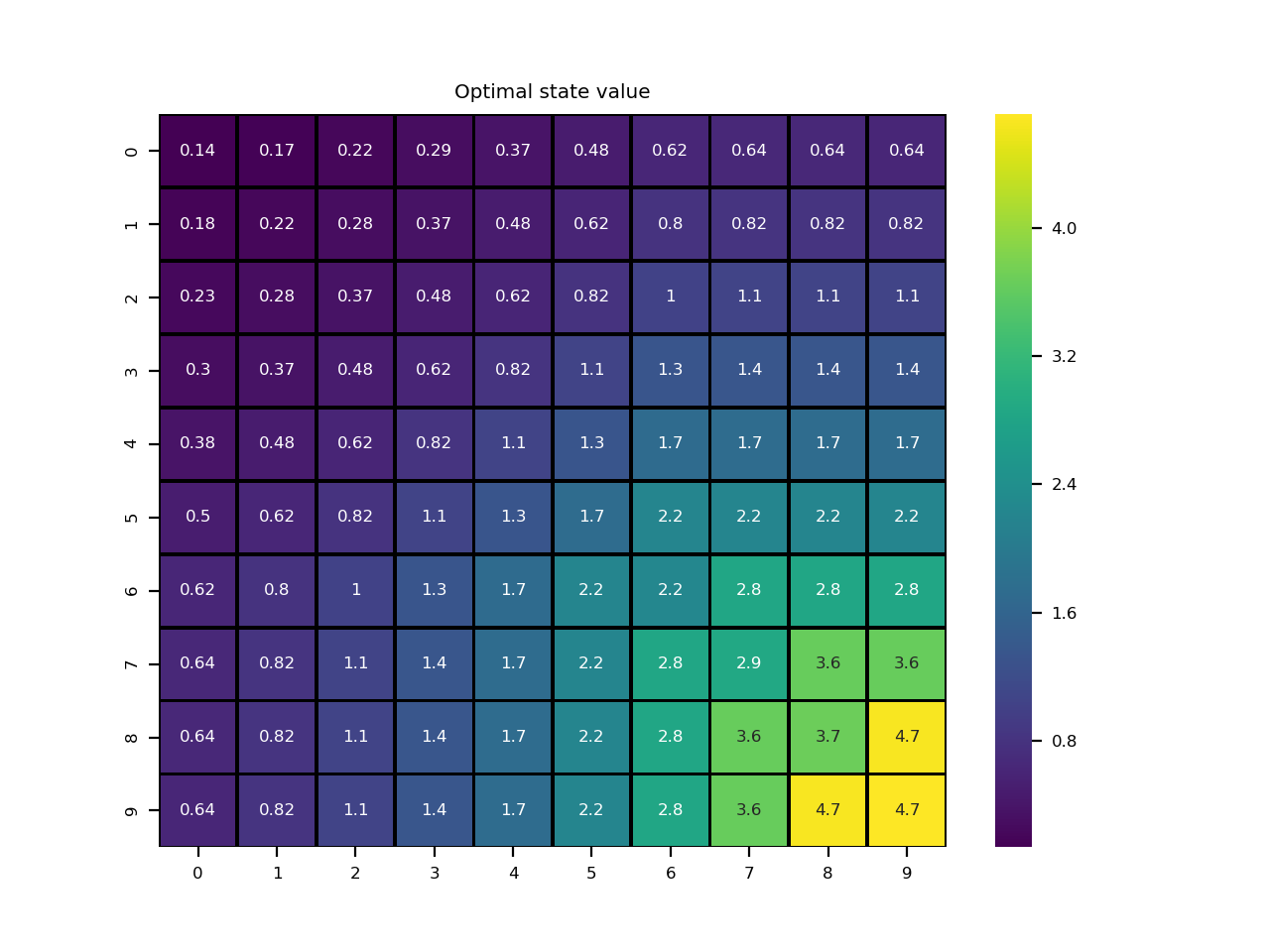


**Q13:**



**Figure 11:** The figure on the left is the heat map of the ground truth reward function 1; the figure on the right is the heat map of the extracted reward function 1.

**Q14:**



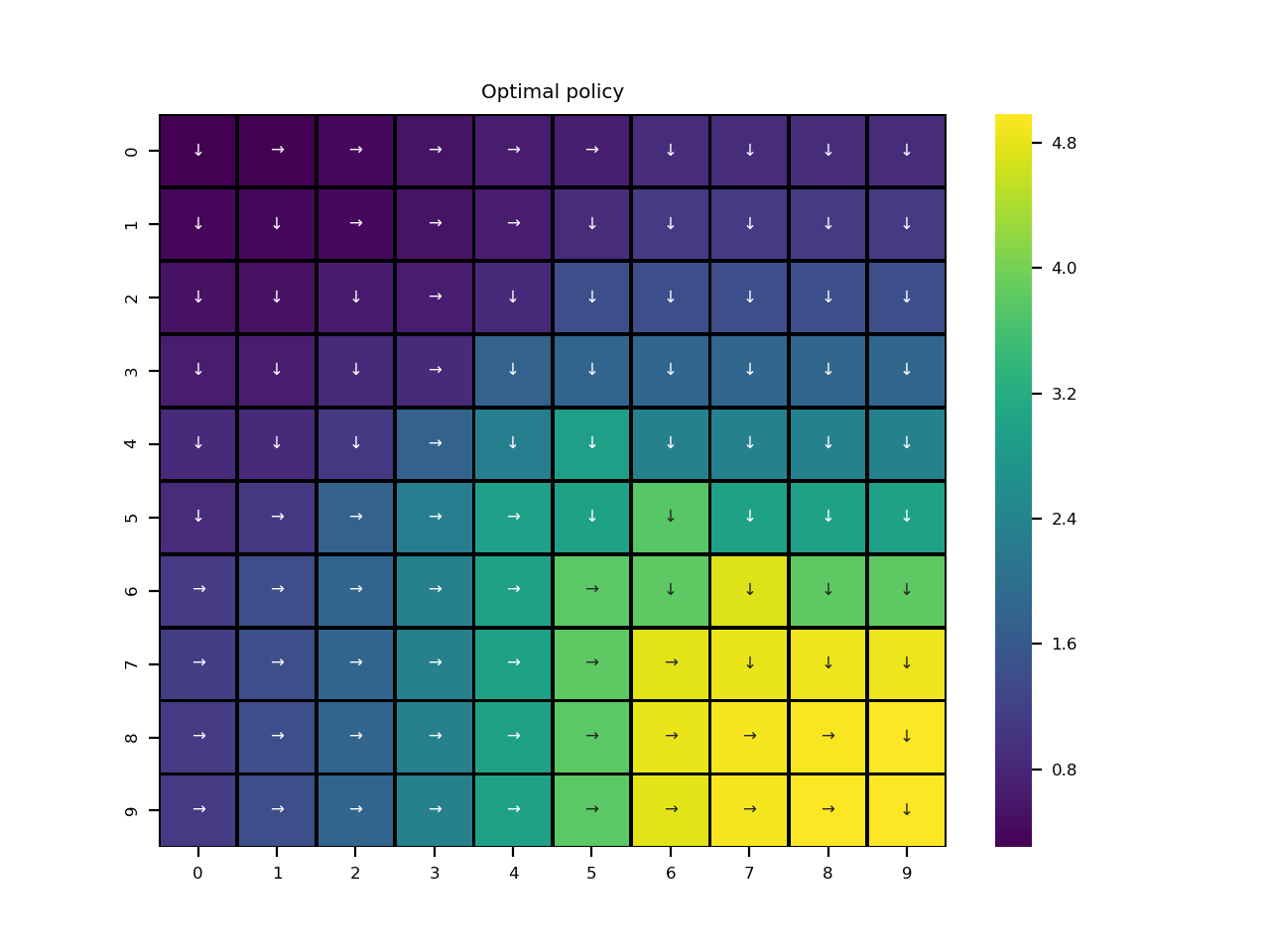
**Figure 12:** Heat map of the optimal state values based on extracted reward function 1

**Q15:**

The general trend of the optimal state value in the 2-D grid is the same. Both grids share the property that the closer to the lower-right corner the higher the optimal value of the state it is. They are same in magnitude.

However, the optimal state values in the lower-right corner are larger than those in question 3. In question 3 the optimal values of states in the same diagonal from the upper right to the lower left almost share the same value. While in question 14 the optimal values of states are larger if they are closer to index (9, 9) in the Euclidean distance. The reason for the difference is that in IRL algorithm the program learns from the optimal policy which tends to move to the lower-right corner.

**Q16:**

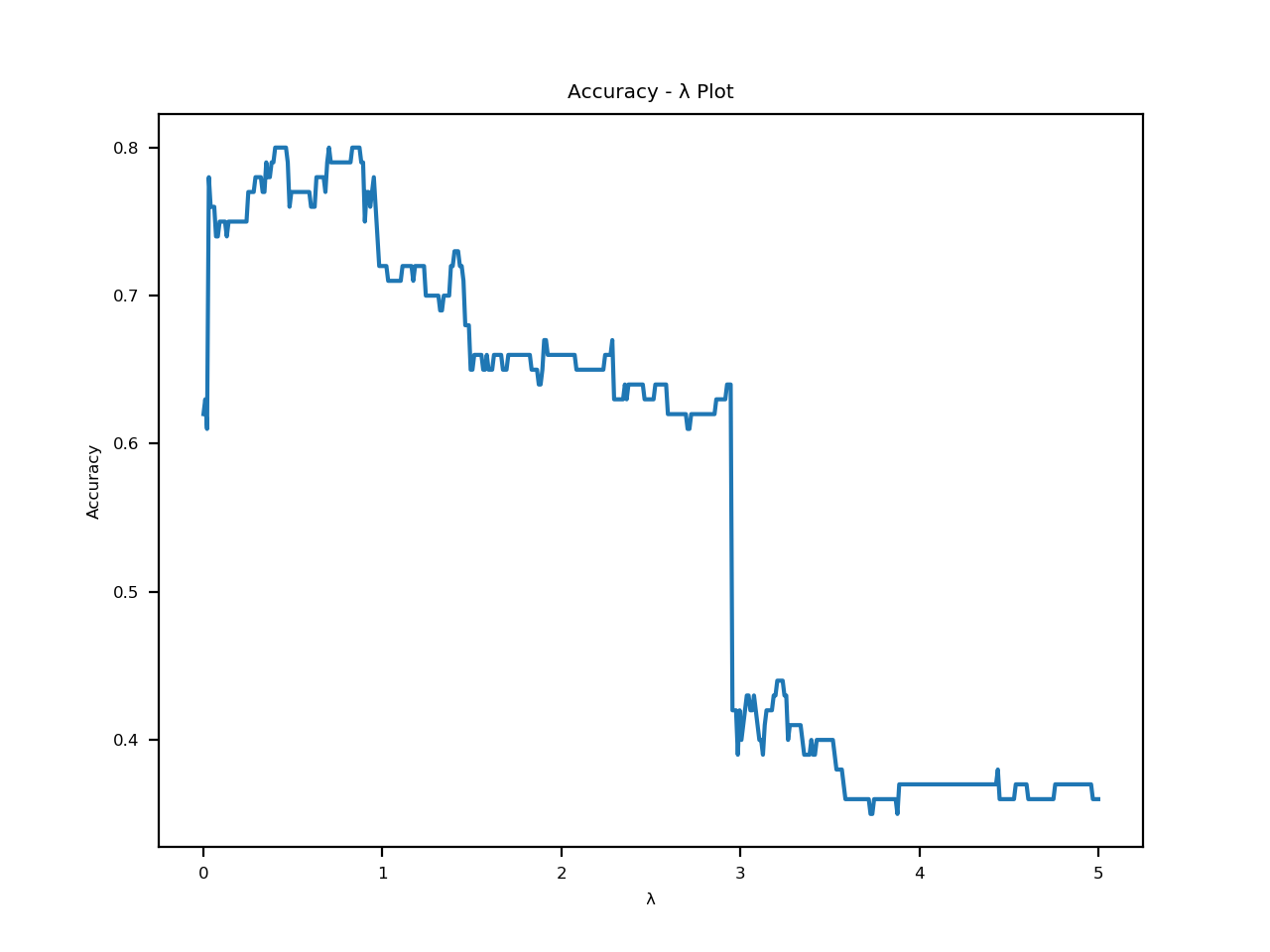


**Figure 13:** Extracted optimal policy based on extracted reward function 1

**Q17:**

According to the figure above, our extracted optimal policy is almost the same to the expert policy. This can be verified by the fact that the maximum accuracy for our extracted policy is 96%. Most of the actions point to the bottom-right grid which is exactly the most rewarding position. Their difference is too small to be observed.

**Q18:**

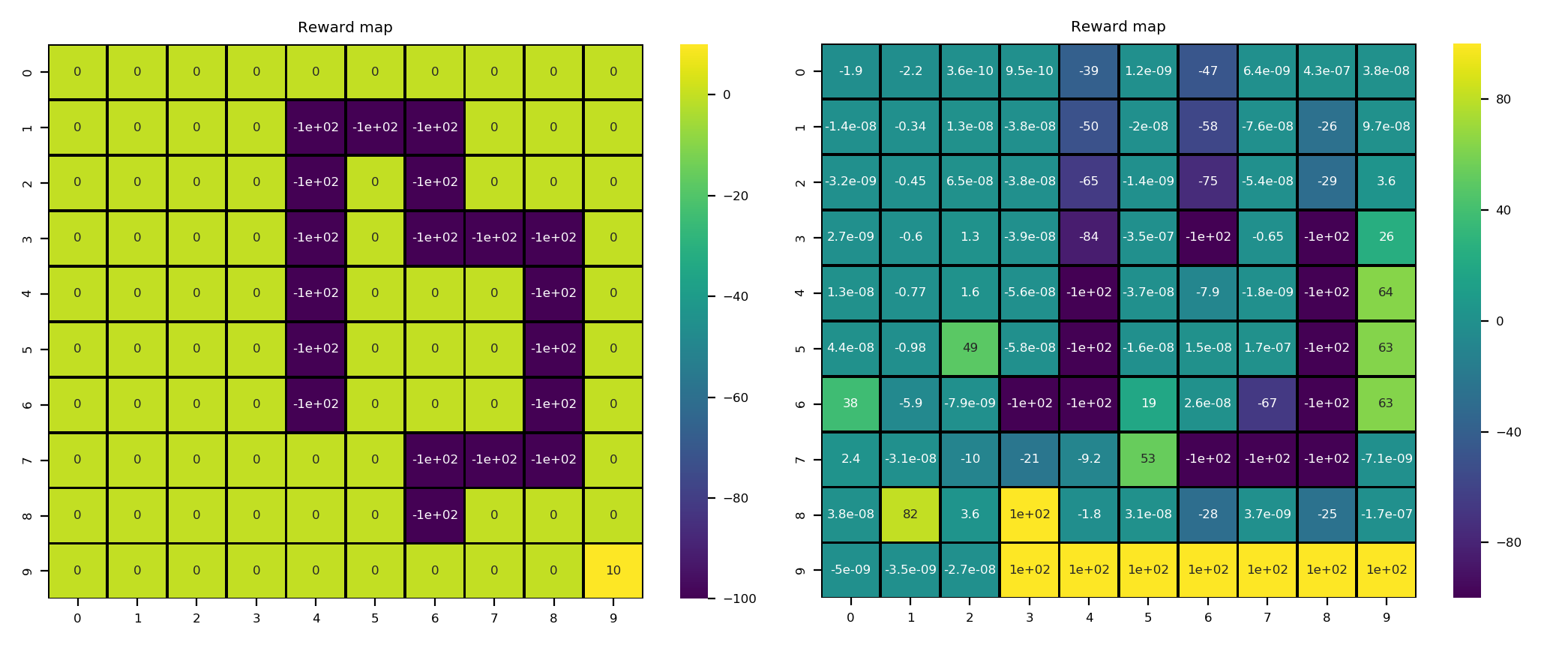


**Figure 14:** Accuracy along with different λ using reward function 1

**Q19:**

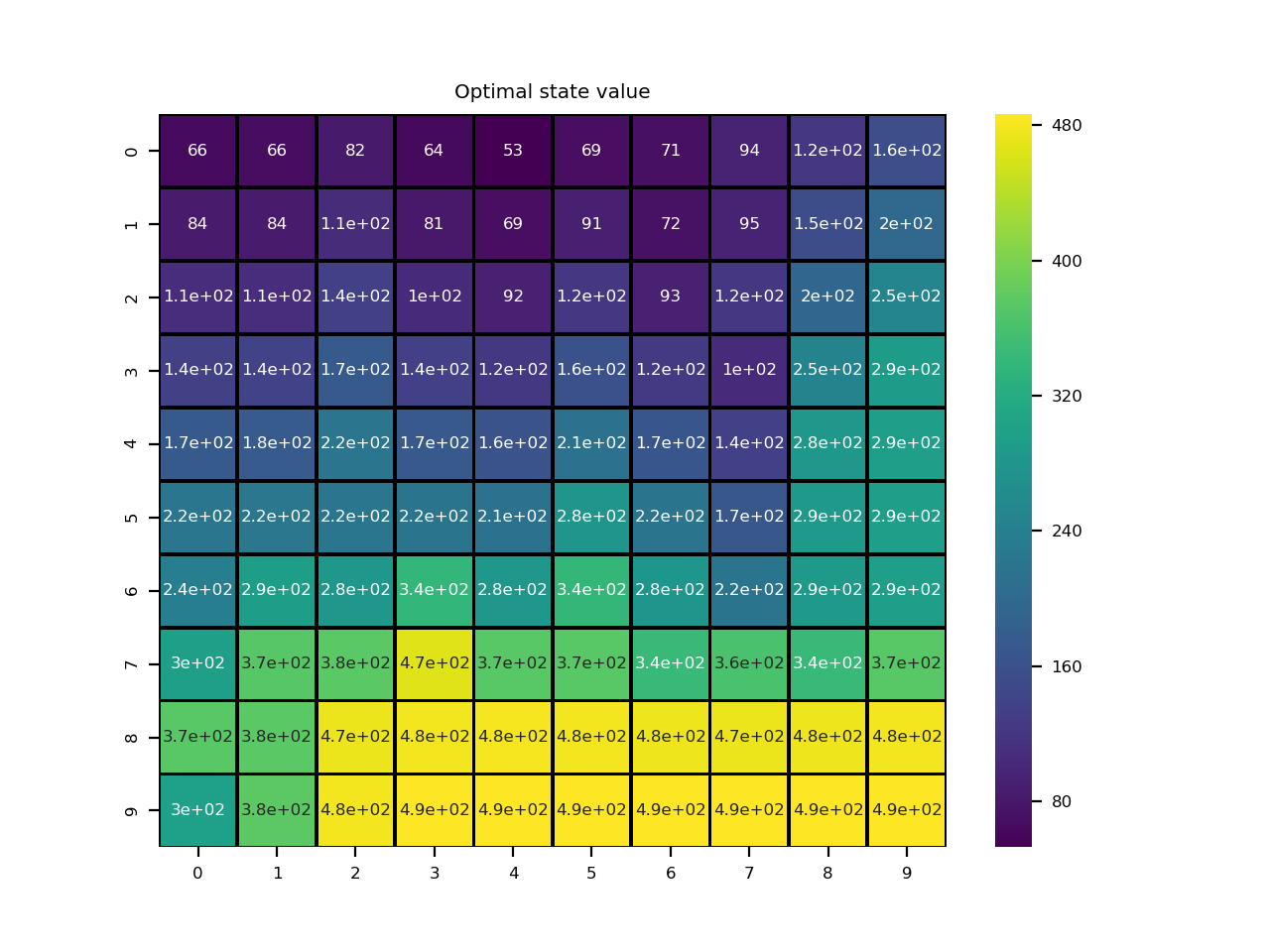


**Q20:**



**Figure 15:** The figure on the left is the heat map of the ground truth reward function 2; the figure on the right is the heat map of the extracted reward function 2.

**Q21:**



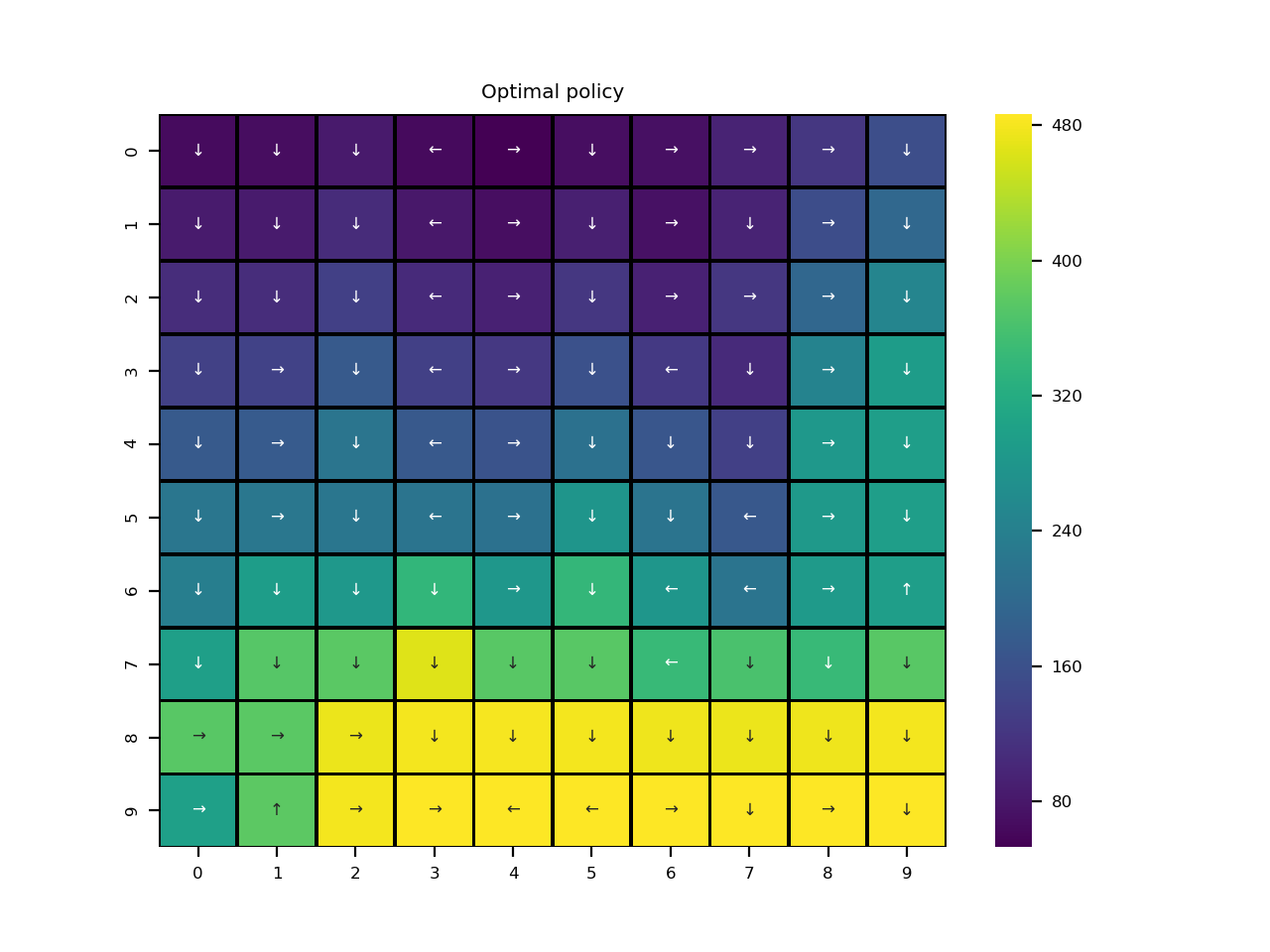
**Figure 16:** Heat map of the optimal state values based on extracted reward function 2

**Q22:**

The general trend of the optimal state value in the 2-D grid is the same. Both grids share the property that the closer to the bottom and the lower-right corner, the higher the optimal value of the state it is, which is consistent to our intuition. We could consider states with -100 reward values as obstacles which we want to go around. Thus, actions tend to go along the bottom to reach best rewarding position.

However, the optimal state values at the bottom and in the lower-right corner are larger than those in question 7. The reason why that happened because we extract the reward function only based on expert optimal policy. From optimal policy plot, we can observe that the major of policy of expert is made from downward and rightward action. Thus, it is reasonable that in the area close to bottom right corner, state values are larger.

**Q23:**



**Figure 17:** Extracted optimal policy based on extracted reward function 2

**Q24:**

The optimal actions of agent are same as the optimal actions of expert for most states. The major differences occur at the states which adjacent to obstacle. The reason why such problems happen is because the agent does not know enough information of the future states, so it could make decision only based on the state value of the state next to current state.

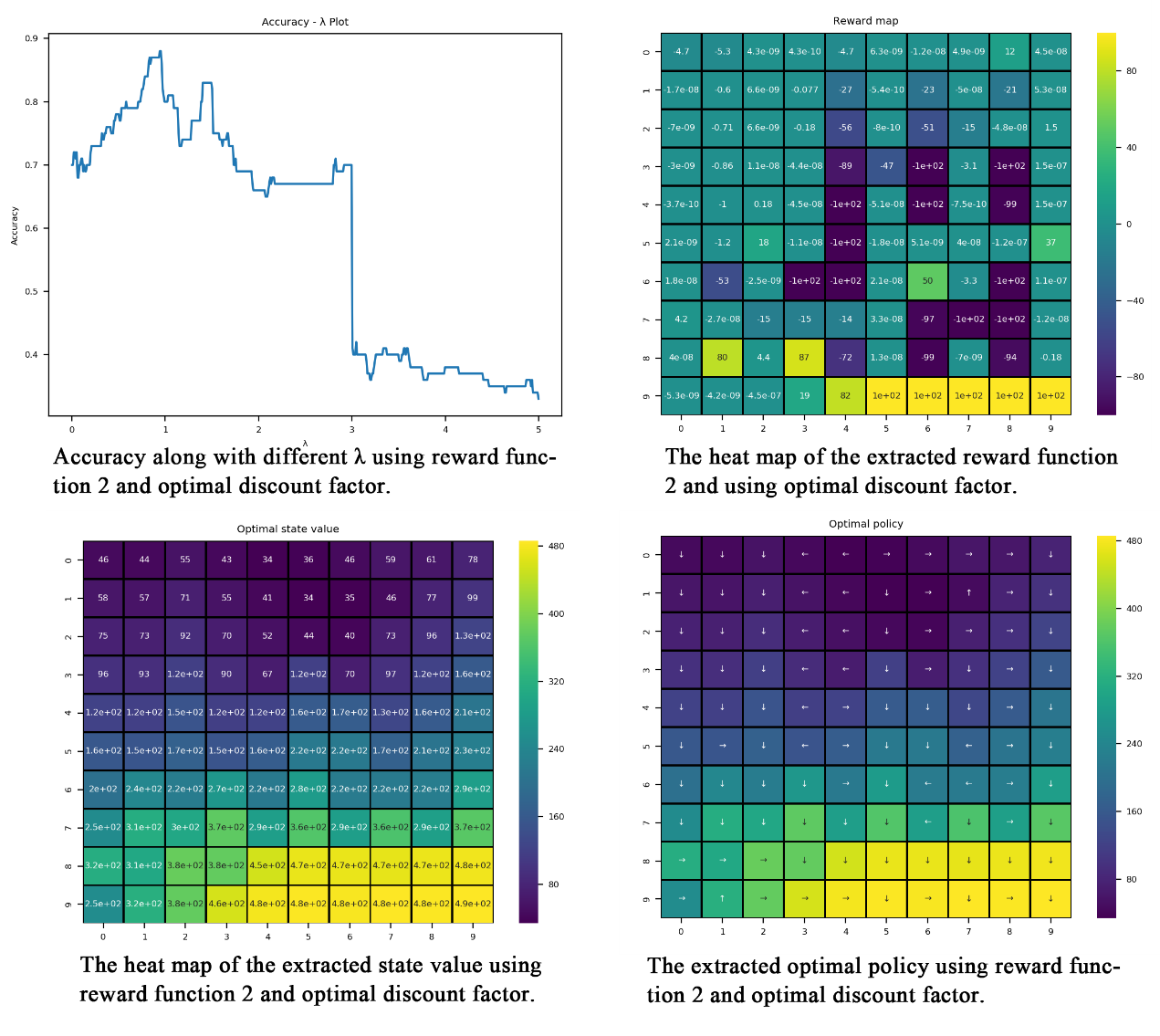
**Q25:**

The major discrepancies we have observed by comparing Figure 8 and Figure 17 is that many arrows in our extracted optimal policy point to an opposite direction compared to those in the original optimal policy at the same grids. Let traps refer to those grids in the ground truth reward map with rewards of -100. To be more specific, there are two types of such discrepancy. One type is that some arrows in grids close to the traps, for example [1, 4] to [5, 4], point to completely opposite directions. Another type is some arrows in grids close to the bottom of the reward map, for example [9, 4] and [9, 5], point to opposite directions.

For these two types of discrepancy, we believe the underlying cause is that many grids tend to focus only on a narrower range of their neighboring grids in the value iteration algorithm. This is caused by a small discount factor, implying that the agents have shorter memory of accumulated rewards in the upcoming future, thus lowering the importance of the rewards brought by farther grids. However, farther grids could sometimes be very critical in the case when there is a second trap sitting right next to a first trap. The best strategy of handling this kind of trap setting is to choose a direction escaping from both traps instead of running into the middle of these two traps. The second case which makes the farther grids important when deciding the optimal policy is when the rewarding grids on the left of the agent are fewer than those on the right of the agent. The optimal solution should be going to the right instead of left.

Based on the observations above, we think that the discount factor should be adjusted bigger to overcome the problem as the agent would take farther grids into account when deciding the optimal policy. Then we conduct a linear search of an optimal discount factor. The optimal discount factor we have found is 0.886842 after searching it from 0.8 to 0.95.

Using this new discount factor, the accuracy, extracted reward function, extracted optimal state value, and extracted optimal policy are shown in the figure below:



**Figure 18:** The accuracy, extracted reward function, extracted optimal state value, and extracted optimal policy using our new discount factor.

As shown in the figure above, our maximum accuracy is improved by as much as 9% comparing to that of using a discount factor of 0.8. The best news is we have successfully fixed the two discrepancies described above. Now the arrows in grids close to the traps, for example [1, 4] to [5, 4], point to same directions of the expert. Besides, the arrows in grids close to the bottom of the reward map, for example [9, 4] and [9, 5], also no longer point to opposite directions.