Large-Scale Data Mining: Models and Algorithms
ECE 232E Spring 2018

Project 3
Reinforcement learning and Inverse Reinforcement learning

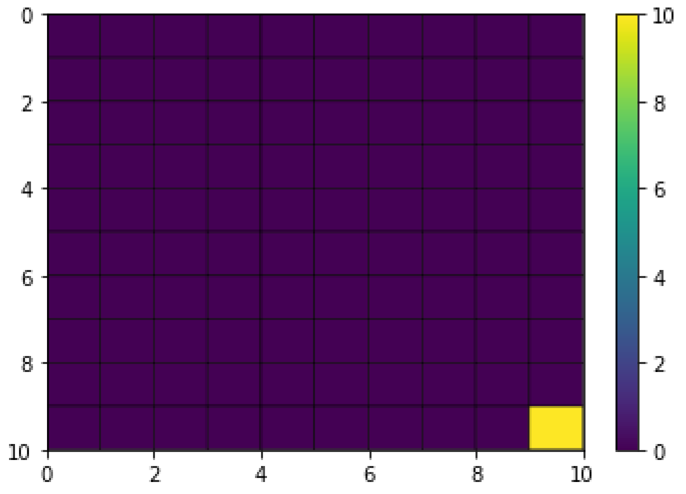
Qinyi Tang (204888348)
Shuo Bai (505032786)
Jinxi Zou (605036454)
Xuan Hu (505031796)
5/21/2018

1 Reinforcement learning

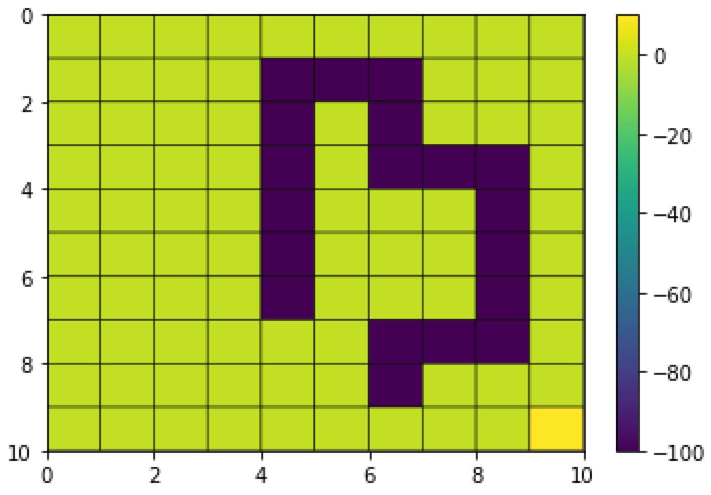
Answer for Q1:

We used the pyplot.pcolor function to generate the heat maps.

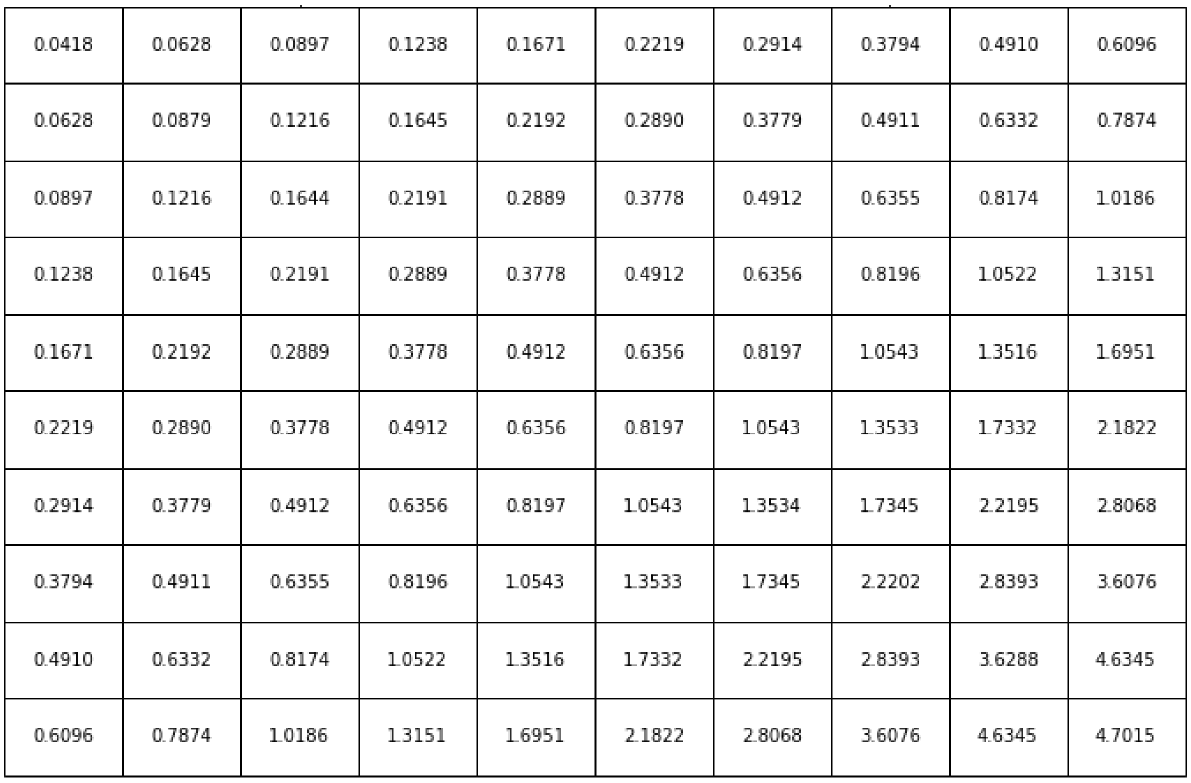
The heat map for the reward function 1 is as following:



The heat map for the reward function 2 is as following:

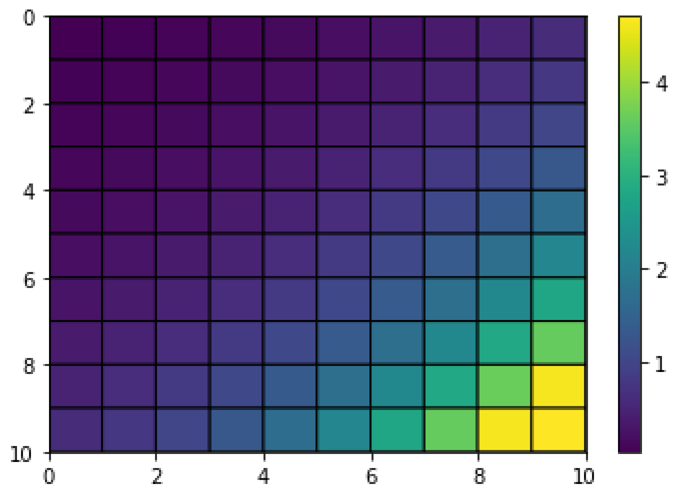


Answer for Q2: We created the environment for MDP by setting up the state-space, action set, transition probability, discount factor and reward function. The number of state was 100, number of actions was 4, the w was set to 0.1, the discount factor was set to 0.8 and used reward function1 as reward function. Then we implemented the value iteration algorithm and used environment we just set as input to find the optimal value of each state. The optimal values are shown in following plot.



0.0418	0.0628	0.0897	0.1238	0.1671	0.2219	0.2914	0.3794	0.4910	0.6096
0.0628	0.0879	0.1216	0.1645	0.2192	0.2890	0.3779	0.4911	0.6332	0.7874
0.0897	0.1216	0.1644	0.2191	0.2889	0.3778	0.4912	0.6355	0.8174	1.0186
0.1238	0.1645	0.2191	0.2889	0.3778	0.4912	0.6356	0.8196	1.0522	1.3151
0.1671	0.2192	0.2889	0.3778	0.4912	0.6356	0.8197	1.0543	1.3516	1.6951
0.2219	0.2890	0.3778	0.4912	0.6356	0.8197	1.0543	1.3533	1.7332	2.1822
0.2914	0.3779	0.4912	0.6356	0.8197	1.0543	1.3534	1.7345	2.2195	2.8068
0.3794	0.4911	0.6355	0.8196	1.0543	1.3533	1.7345	2.2202	2.8393	3.6076
0.4910	0.6332	0.8174	1.0522	1.3516	1.7332	2.2195	2.8393	3.6288	4.6345
0.6096	0.7874	1.0186	1.3151	1.6951	2.1822	2.8068	3.6076	4.6345	4.7015

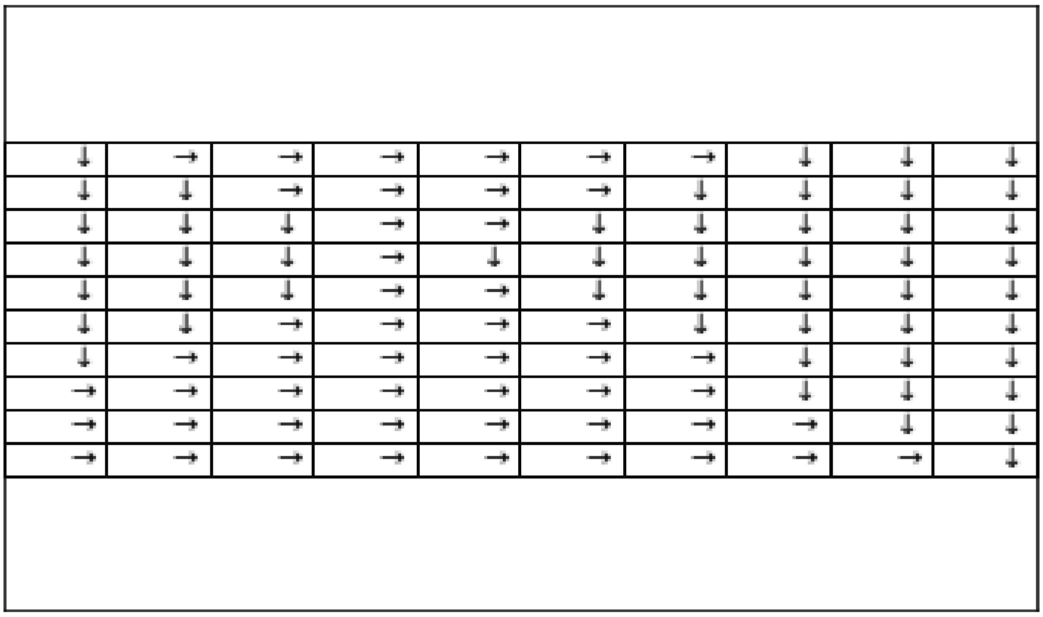
Answer for Q3:



Heat map of optimal state-value function

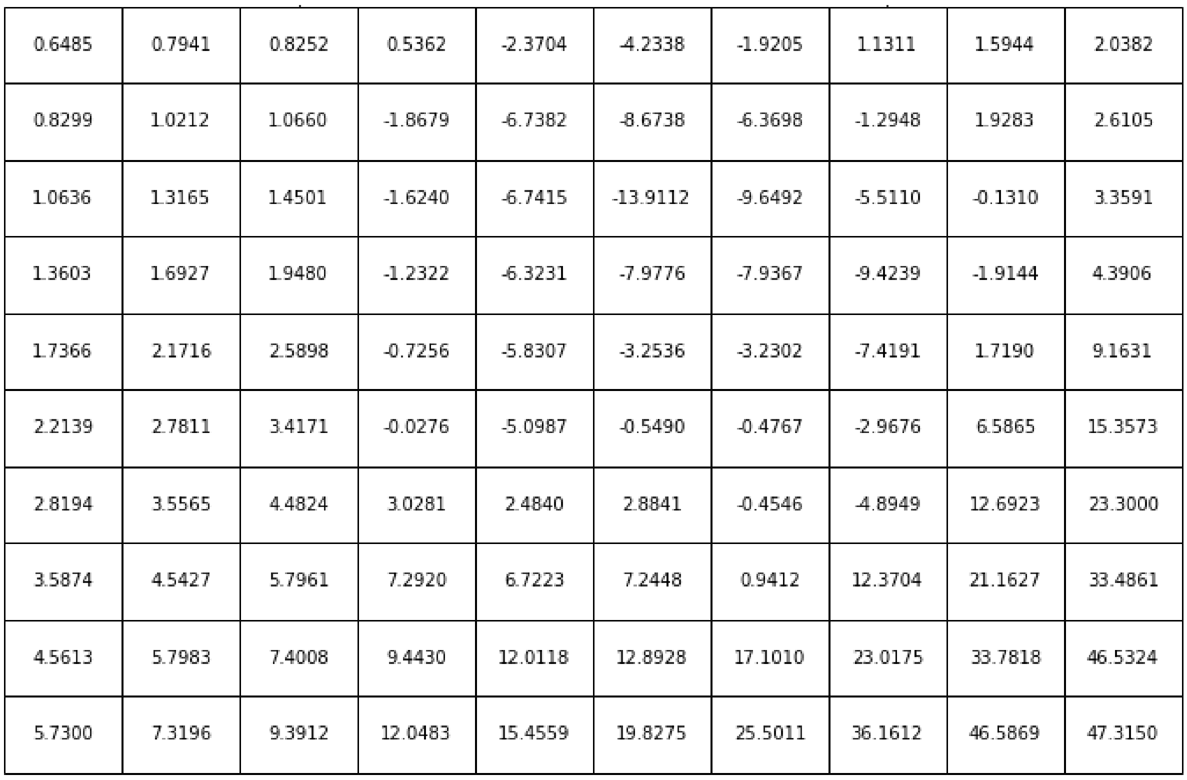
Answer for Q4: State-value function measures 'how good' it is for the agent to be in a given state. Optimal state-value is calculated by optimal policy, which means the best reward we could get after each step.Since the discount factor we choose is 0.8, it leads to far-sighted evaluation. From the heat map of reward function 1 we can see that only corner([10,10]) element gives a positive feedback 10 and all others' reward are zero. So as we move from old state to new state, the more close to state[10,10], the bigger optimal state value we can get. From the heat map of optimal state-value we could see this distribution obviously. The tendency of the distribution of optimal value is corresponding to the change of reward function. Since reward function is symmetrical along the diagonal, the optimal state-value is also symmetrical along this line.

Answer for Q5: The optimal policy of the agent matches our intuition. Ignore the element in diagonal, the actions of two sides are symmetrical, which is similar to reward function 1 and optimal state-value. The tendency of the arrows always point to element[10,10]. Notice that the element in diagonal can be both ↓ or → since the reward function is symmetrical. It is possible for the agent to computer the optimal action to take at each state by observing the optimal values of its neighboring state, that’s how value iteration algorithm comes from.



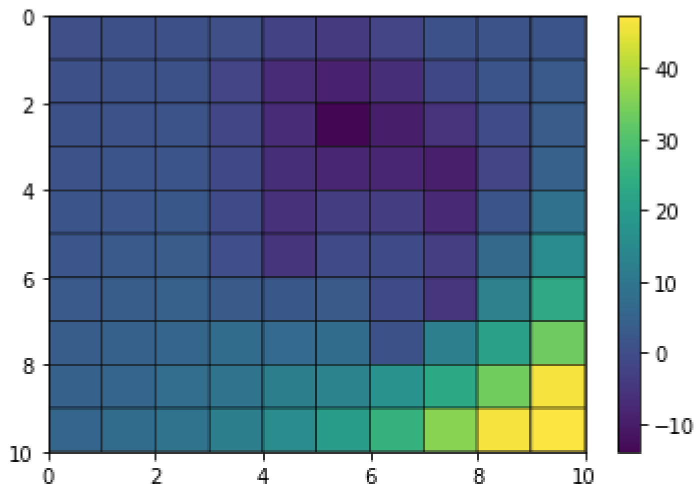
↓	→	→	→	→	→	→	↓	↓	↓
↓	↓	→	→	→	→	↓	↓	↓	↓
↓	↓	↓	→	→	↓	↓	↓	↓	↓
↓	↓	↓	→	↓	↓	↓	↓	↓	↓
↓	↓	↓	→	→	↓	↓	↓	↓	↓
↓	↓	→	→	→	→	↓	↓	↓	↓
↓	→	→	→	→	→	→	↓	↓	↓
→	→	→	→	→	→	→	↓	↓	↓
→	→	→	→	→	→	→	→	↓	↓
→	→	→	→	→	→	→	→	→	↓

Answer for Q6: We modified the environment of the agent with the reward function 2. Then we used the functions we implemented in the Q2, and we get the following table of the optimal value



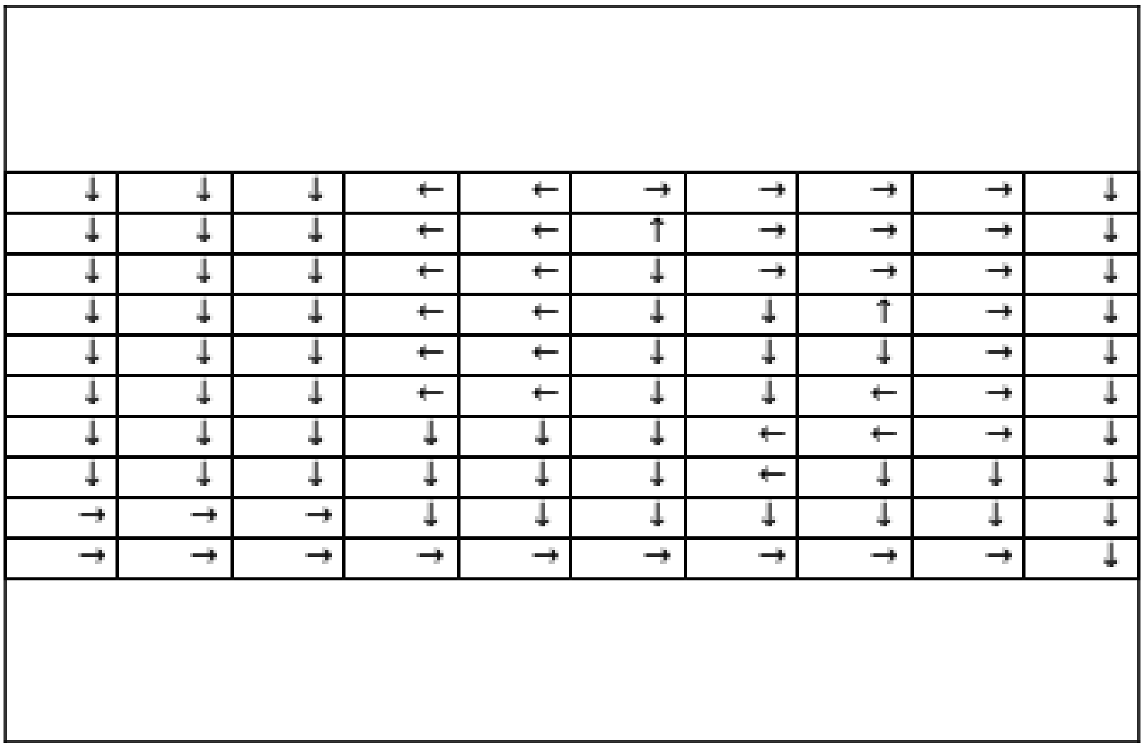
0.6485	0.7941	0.8252	0.5362	-2.3704	-4.2338	-1.9205	1.1311	1.5944	2.0382
0.8299	1.0212	1.0660	-1.8679	-6.7382	-8.6738	-6.3698	-1.2948	1.9283	2.6105
1.0636	1.3165	1.4501	-1.6240	-6.7415	-13.9112	-9.6492	-5.5110	-0.1310	3.3591
1.3603	1.6927	1.9480	-1.2322	-6.3231	-7.9776	-7.9367	-9.4239	-1.9144	4.3906
1.7366	2.1716	2.5898	-0.7256	-5.8307	-3.2536	-3.2302	-7.4191	1.7190	9.1631
2.2139	2.7811	3.4171	-0.0276	-5.0987	-0.5490	-0.4767	-2.9676	6.5865	15.3573
2.8194	3.5565	4.4824	3.0281	2.4840	2.8841	-0.4546	-4.8949	12.6923	23.3000
3.5874	4.5427	5.7961	7.2920	6.7223	7.2448	0.9412	12.3704	21.1627	33.4861
4.5613	5.7983	7.4008	9.4430	12.0118	12.8928	17.1010	23.0175	33.7818	46.5324
5.7300	7.3196	9.3912	12.0483	15.4559	19.8275	25.5011	36.1612	46.5869	47.3150

Answer for Q7:



Answer for Q8: The optimal state values are closely related to the reward function. From the figure above, we can see that the bottom right has the highest state values, because the reward value on the bottom right corner is the only positive reward (10) in the reward function 2. Also, in reward function, there are some states have the reward of the negative value (-100). Thus the states around them have the relatively low state value. Especially, the state in row 2 and column 5 has a smallest state value, because it has three neighbors with the negative reward. From the value iteration formula, we can see that the state values do depend on the reward of itself and its neighbors. Also, we can see that the states has more neighbors of the smaller reward, the smaller it will be. Also, if a state is both close to the negative rewards and positive rewards, the positive rewards will compensate the lower rewards, making it not that small. We can conclude that the states near the high reward have the larger state values and that the states near the low reward have the smaller state values.

Answer for Q9: As we could see from the heat map of reward function 2, the optimal policy is still corresponding to the distribution of reward function and optimal state-value. Optimal actions follow the tendency that they finally point to the element[10,10] which has the biggest reward. Besides that, optimal policy also avoid taking a step which may go to state that has a negative reward. In reward function, there are several states that have negative reward. So compare with the conclusion in question 5, in this question, optimal policy have some actions like ↑ or ←.



↓	↓	↓	←	←	→	→	→	→	↓
↓	↓	↓	←	←	↑	→	→	→	↓
↓	↓	↓	←	←	↓	→	→	→	↓
↓	↓	↓	←	←	↓	↓	↑	→	↓
↓	↓	↓	←	←	↓	↓	↓	→	↓
↓	↓	↓	←	←	↓	↓	←	→	↓
↓	↓	↓	↓	↓	↓	←	←	→	↓
↓	↓	↓	↓	↓	↓	←	↓	↓	↓
→	→	→	↓	↓	↓	↓	↓	↓	↓
→	→	→	→	→	→	→	→	→	↓

2 Inverse Reinforcement learning

Answer for Q10: The C, x, D can be expressed in the following format. In addition, we changed the zero matrix on the right of less or equal character to b.

$$C = \begin{bmatrix} 0 \\ I \\ -\lambda \end{bmatrix} \qquad 0 = \left.\begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}\right\} \text{100 row} \qquad -\lambda = \left.\begin{bmatrix} -\lambda \\ \vdots \\ -\lambda \end{bmatrix}\right\} \text{100 row}$$

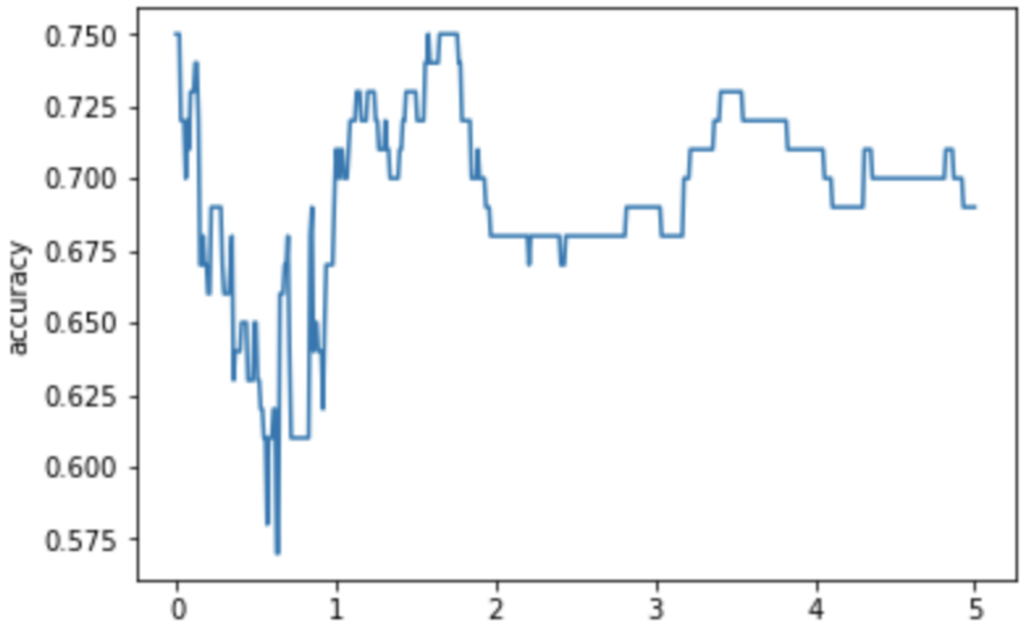
$$I = \left.\begin{bmatrix} \vdots \\ \vdots \end{bmatrix}\right\} \text{100 row}$$

$$X = \begin{bmatrix} R \\ t \\ V \end{bmatrix} \qquad R = \begin{bmatrix} R_1 \\ R_2 \\ \vdots \\ R_{100} \end{bmatrix} \qquad t = \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_{100} \end{bmatrix} \qquad V = \begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_{100} \end{bmatrix}$$

$$D = \begin{bmatrix} -(P_{a_1}(i) - P_a(i))(I - \gamma P_{a_1})^{-1} & I & 0 \\ -(P_{a_1}(i) - P_a(i))(I - \gamma P_{a_1})^{-1} & 0 & 0 \\ I & 0 & -I \\ -I & 0 & -I \\ I & 0 & 0 \\ -I & 0 & 0 \end{bmatrix}$$

$$b = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ R_{max} \\ R_{max} \end{bmatrix} \begin{matrix} \left.\begin{matrix} \\ \\ \\ \\ \\ \end{matrix}\right\} \text{800 rows} \\ \} \text{100 rows} \\ \} \text{100 rows} \end{matrix}$$

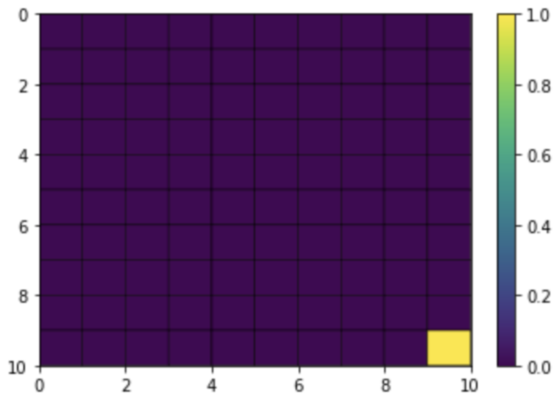
Answer for Q11: We calculated the estimated reward function from Linear Programming formulas in question 10. The we used the estimated reward function to find out the optimal actions of each state and compare them with true ground optimal actions. The accuracy of estimated optimal actions with penalty coefficient from 0 to 5 are shown in following plot.

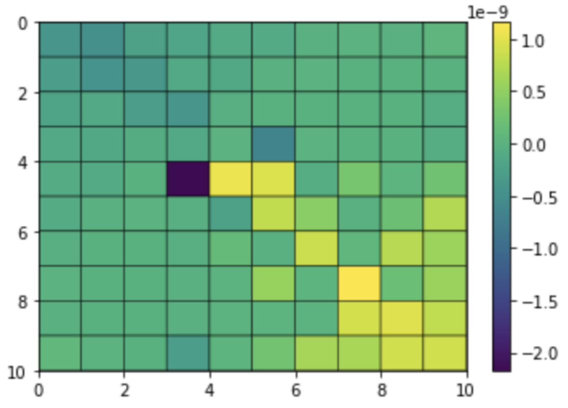


Answer for Q12: From the result we plot above, We get the max accuracy at around 0.75, we have the following position to get such a high accuracy which is at [0, 1, 2, 158, 165, 166, 167, 168, 169, 170, 171, 172, 173,174, 175, 176], we drop the first 3 values, for the lmada is very low. The result may get overfitting. We choose some medium value in the left value which is 0.75.

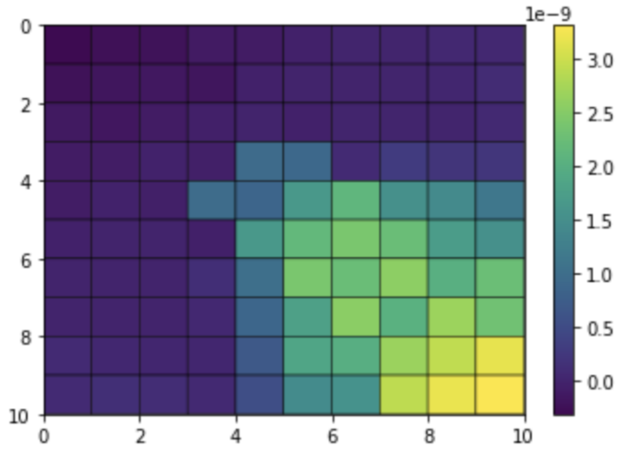
$$\lambda max = 0.75 \blacksquare$$

Answer for Q13:



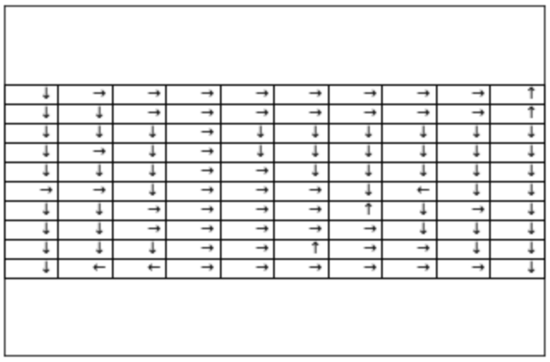


Answer for Q14:

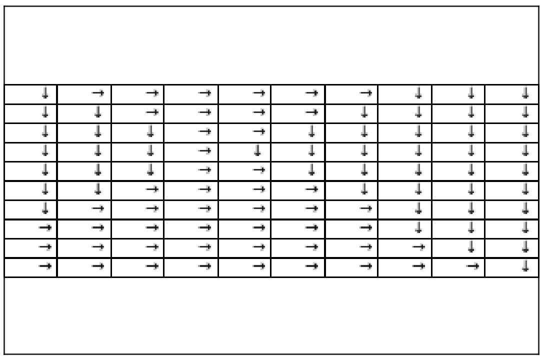


Answer for Q15: Similarities and differences: The total tendency is similar. In the previous part's reward function, all position is zero except the most right down position is one. the reward function in Q13 looks similar, but they are not strictly same. The trend is that as the position goes to right left. The value gets larger. Therefore, the final optimization state has larger value at the right down position. However, the LP solution gives a continuous result, the reward function shows a continuous change from left up to right down, not the discrete change. The result will have more larger-than-zero value from center to right down.

Answer for Q16:



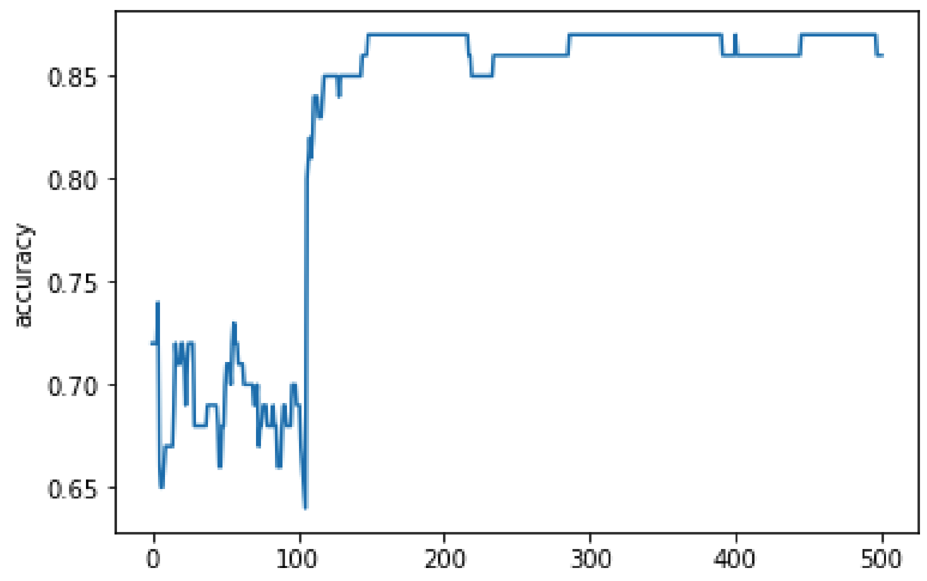
↓	→	→	→	→	→	→	→	→	↑
↓	↓	→	→	→	→	→	→	→	↑
↓	↓	↓	→	↓	↓	↓	↓	↓	↓
↓	→	↓	→	↓	↓	↓	↓	↓	↓
↓	↓	↓	→	→	↓	↓	↓	↓	↓
→	→	↓	→	→	→	↓	←	↓	↓
↓	↓	→	→	→	→	↑	↓	→	↓
↓	↓	→	→	→	→	→	↓	↓	↓
↓	↓	↓	→	→	↑	→	→	↓	↓
↓	←	←	→	→	→	→	→	→	↓



↓	→	→	→	→	→	→	↓	↓	↓
↓	↓	→	→	→	→	↓	↓	↓	↓
↓	↓	↓	→	→	↓	↓	↓	↓	↓
↓	↓	↓	→	↓	↓	↓	↓	↓	↓
↓	↓	↓	→	→	↓	↓	↓	↓	↓
↓	↓	→	→	→	→	↓	↓	↓	↓
↓	→	→	→	→	→	→	↓	↓	↓
→	→	→	→	→	→	→	↓	↓	↓
→	→	→	→	→	→	→	→	↓	↓
→	→	→	→	→	→	→	→	→	↓

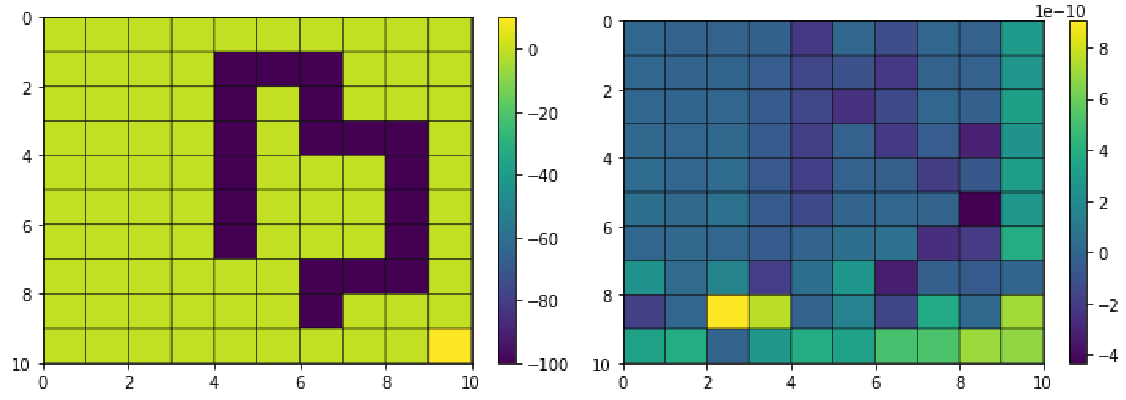
Answer for Q17: The left is the plot in question 16 and right is the question 5. From the comparison we can observe some differences and similarities. In total, the symmertric property is fulfilled because of the similarity between reward function. However, there are some slight difference for some of the arrow is reverser. It is caused by the bias between the estimated reward function and original reward function.

Answer for Q18: We calculated the estimated reward function from Linear Programming formulas in question 10. The we used the estimated reward function to find out the optimal actions of each state and compare them with true ground optimal actions. The accuracy of estimated optimal actions with penalty coefficient from 0 to 5 are shown in following plot.

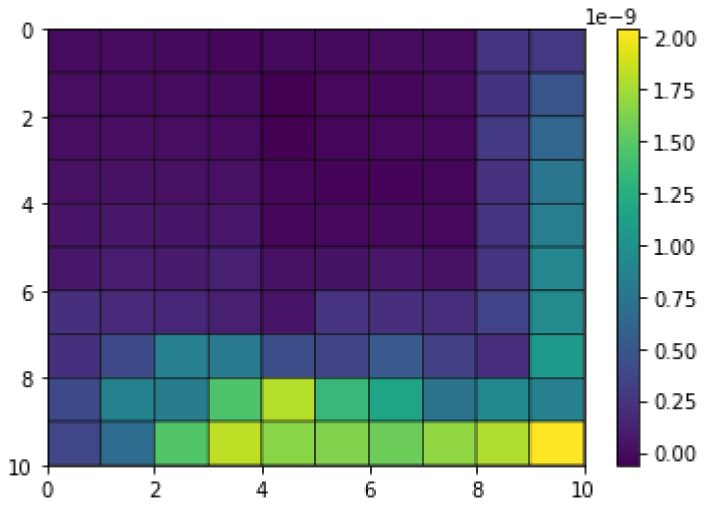


Answer for Q19: From the plot above we can get the conclusion that there are more than one λ that could reach the max accuracy. The maximum accuracy is 0.87. Here we take λmax as 2.0.

Answer for Q20:

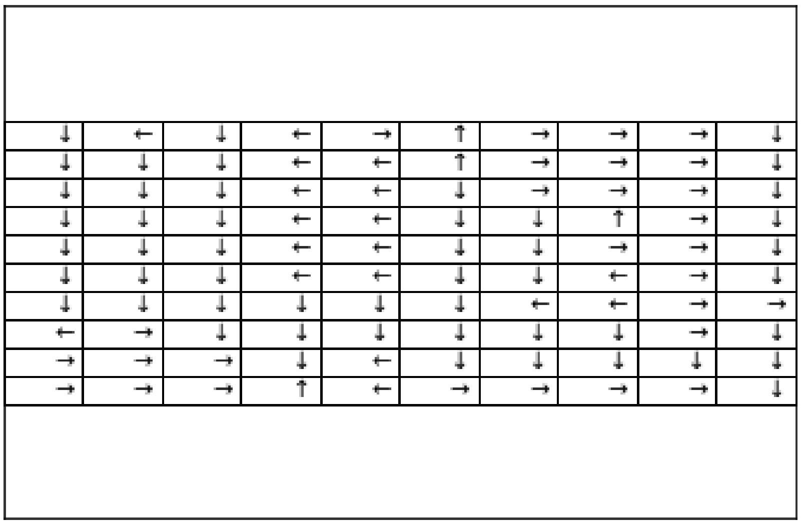


Answer for Q21:



Answer for Q22: Compare with the plot of optimal state value in question 7, they both show a distribution similar to reward function2. The optimal state-value become higher as the state close to element[10,10], and they become lower when the step close to states which value are negative. However, the heat map in question 21 obtain high value around state[5,9] and [4,10] ([column, row]), while heat map in question 7 is more close to the distribution of reward function2 and there is no outliers such as we mentioned in question 21. That's due to the extract reward function we get isn't as same as the original one. From question 21, we notice that the max reward isn't converge to the state[10,10], but in other abnormal state. That's why optimal state-value of extract reward function obtain big value in local area.

Answer for Q23:



↓	←	↓	←	→	↑	→	→	→	↓
↓	↓	↓	←	←	↑	→	→	→	↓
↓	↓	↓	←	←	↓	→	→	→	↓
↓	↓	↓	←	←	↓	↓	↑	→	↓
↓	↓	↓	←	←	↓	↓	→	→	↓
↓	↓	↓	←	←	↓	↓	←	→	↓
↓	↓	↓	↓	↓	↓	←	←	→	→
←	→	↓	↓	↓	↓	↓	↓	→	↓
→	→	→	↓	←	↓	↓	↓	↓	↓
→	→	→	↑	←	→	→	→	→	↓

Answer for Q24: Compare with the optimal policy in question 9, they both show a distribution that point to the state which we could get the best reward([10,10]). Their optimal policies both walk away from the negative state and move to positive state. However, the optimal action in question 9 has only one target, that's [10,10]. When we leave from the any starting point, whether [1,1], [10,1] or any point else, their final goal is [10,10], they converge to a fixed state. For optimal policy in question 23, we get a different conclusion. If we start from different states, they may converge to different place like[10,10] or [4,9] and [4,10]. That's due to the reward function we extract converge to local state[3,9].

Answer for Q25:

We identify the two major discrepancies between the policy derived from Q23 and the policy derived from Q9 and their causes as following:

1. There are one more converge center in our extracted rewards at the location of [3,9]. Therefore, the states near it has the tendency to take the action towards it. For example, the state [4, 10] and the state [5,9] take the action towards the reward peak nearby. The reward peak near the state [3,9] is extracted because our policy which the optimal action of the expert is taking the action "↓" as optimal in the first three

columns and the first 8 rows, and it is taking the action “→” as optimal in the first 3 columns and last two rows. This policy is derived because of the tendency towards the positive reward in the bottom right corner. However, when we doing the inverse reinforcement learning, the agent regards reward on [3,9] as the local optimum instead of the knowing the distant optimum. This is hard to avoid. It comes from arbitrary symmetry-breaking in the chosen policy and are caused by the limitation of the reinforcement learning of not knowing the distant reward. This can be partially alleviated by reducing the discount value or using the ε-greedy algorithm.

2. The second discrepancy we identified is that there are three states on the edges take the action out of the border. ([1,8],[6,1],[10,7]) The reason for it is that there are comparatively small reward values near these three states. Instead of risking move to these comparatively small reward, the optimal state tends to stay in there original state. And from our transition probability definition in project statement, the states on the edge have a higher probability to stay at their original states when they are going to move outside the edge. This is also partially caused by the limitation of the reinforcement learning of not knowing the distant reward and tending to stay at the local optimum.

First, we try to solve the second discrepancy by modifying the optimal policy function. When deciding the optimal action, we add a function to decide whether it is going to be out of the border. If it does, we try to choose the next optimal action avoid it moving out of the border. (In the question25_1.ipynb file) We re-run the optimal policy algorithms, the new accuracy is 0.88.

Next, we have a try to fix the first discrepancy by using the ε-greedy algorithm. ε-greedy algorithm is a trade-off between the exploitation and exploration. Exploitation vs Exploration is often as the dilemma in the reinforcement learning. “Exploitation” means making the best decision given current information, whereas “Exploration” means gather more information by randomly make the decision. The ε-greedy algorithm selects a random action with probability ε, and selects the optimum with the probability $1 - \varepsilon$. We choose ε as 0.1 and correspondingly modify the optimal policy function. We re-run the optimal policy algorithms with the modification for the edge judgement for the first discrepancy (In the question25_2.ipynb file). The new accuracy is 0.88.