In the name of God

Arsalan Firoozi – 97102225

Homework 6 – Advanced Neuroscience

In this homework, I've considered a 15x15x4 m matrix and a 15x15 values matrix. I've trained the model with coefficients of Learning rate equal to 1, Discounting factor equal to 1, and Beta value equal to 0.5 for 100 trials. Updating rules are listed below:

$$\delta_t = P(r|S_t) + \gamma \sum_{S_{t+1}} P(S_{t+1}|S_t) V(S_{t+1}) - V(S_t)$$

$$V(S_t)_{new} = V(S_t)_{old} + \eta \cdot \delta_t$$

$$m_i \rightarrow m_i + \epsilon \left(1 - P(a_i|S_t)\right) E(\hat{V}(S_{t+1})|a_i)$$ If action 'a_i' is chosen

$$m_i \rightarrow m_i - \epsilon \, P(a_i|S_t) E(\hat{V}(S_{t+1})|a_j)$$ If action 'a_i' is not chosen

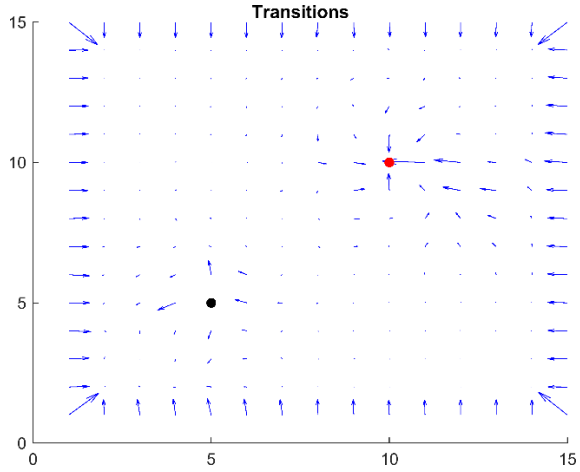
$$\pi(a_i, S_t) = \frac{\exp(\beta m_i)}{\sum_1^n \exp(\beta m_j)}$$

It is worth mentioning that I applied the above rules right after all steps of a trial were passed. This approach results in a much better performance considering the gradient of transitions in comparison with the approach of applying update rules after each step (By this approach, the result shows random arrows with low probabilities).

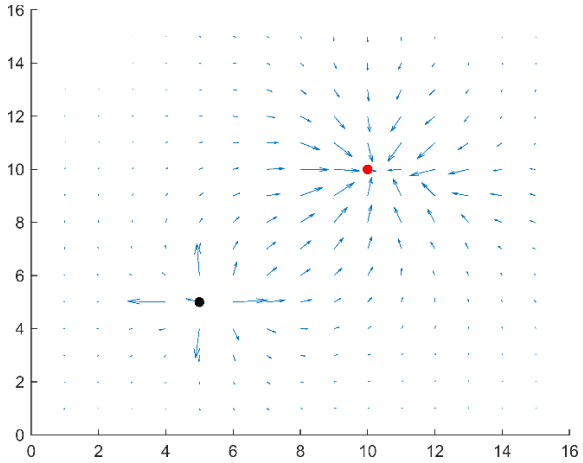
The term $P(r|S_t)$ for updating delta in my implementation was the ratio of the value of state and absolute sum of all values of states.

1. There is 2 files "Demo_afterTraining.avi" and "Demo_beforeTraining.avi" which shows how the model is trained.

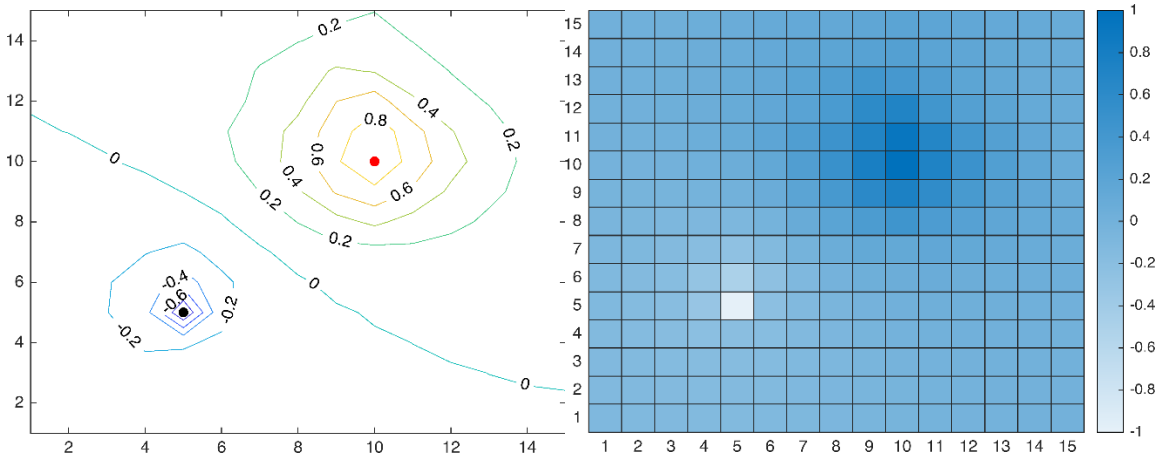
2. I’ve used the probability of transitions in each state and derived a vector indicating the overall tendency of the agent to move in that state. Longer arrows show higher probabilities:



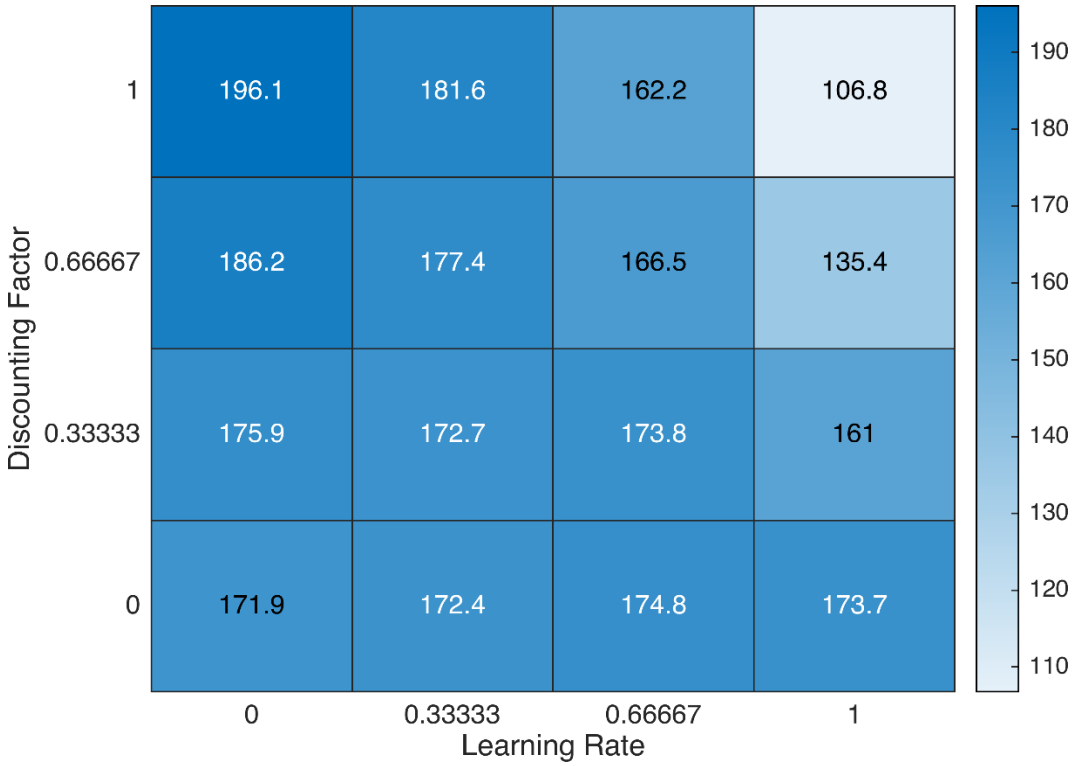
The plot below is the gradient of states’ values:



The heatmap of states’ values and the contour plot:

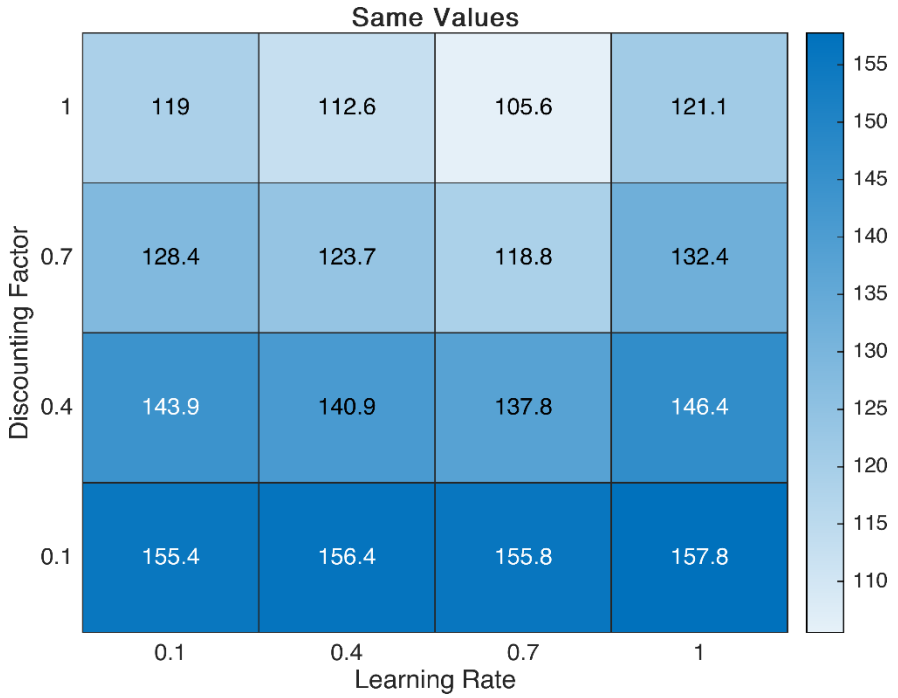


3. For 5 values of discounting factor and 5 values of the learning rate, I have trained the model 100 times as the number of trials. Also, I've repeated training 100 times to cancel the noise by mean across repetitions:



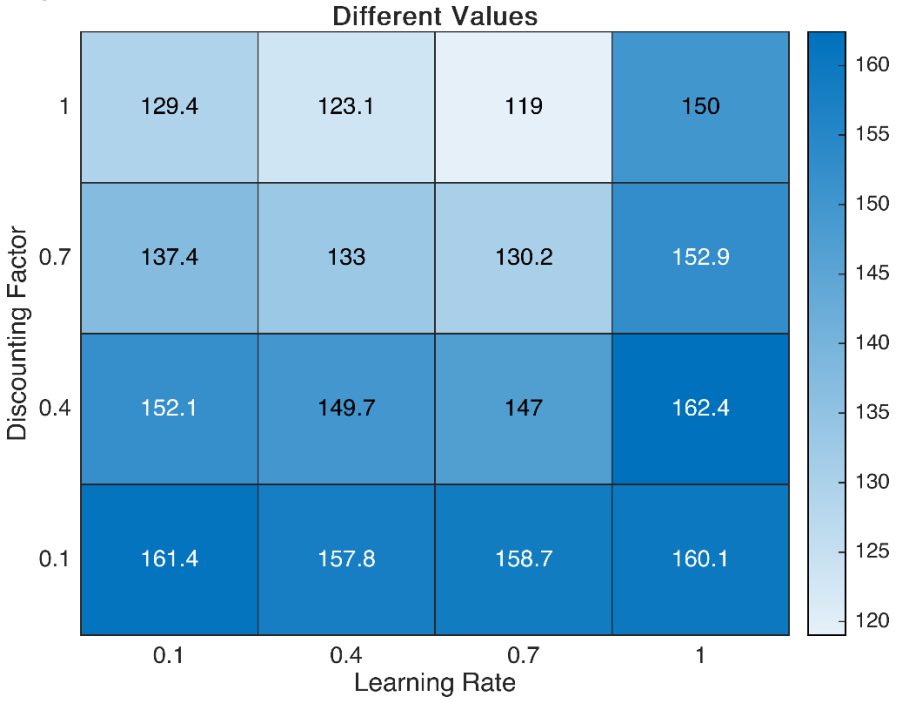
By decreasing the learning rate, the model needs more iterations to train and obtain the optimal transition probabilities. So it's expected to have a higher number of steps by decreasing the learning rate.
Also by decreasing the discounting factor, since the effect of right and wrong decisions on the states far from the goals become smaller, the model needs more iterations to train and obtain the optimal transition probabilities. So it's expected to have a higher number of steps by decreasing the discounting factor.
The result shows the same effect as discussed above.

4. By having more than one goal on the map, an agent has a higher probability to get to one of the goals. So it's expected to have a lower number of steps in all cells of the heatmap.
Considering the same values for 2 states:

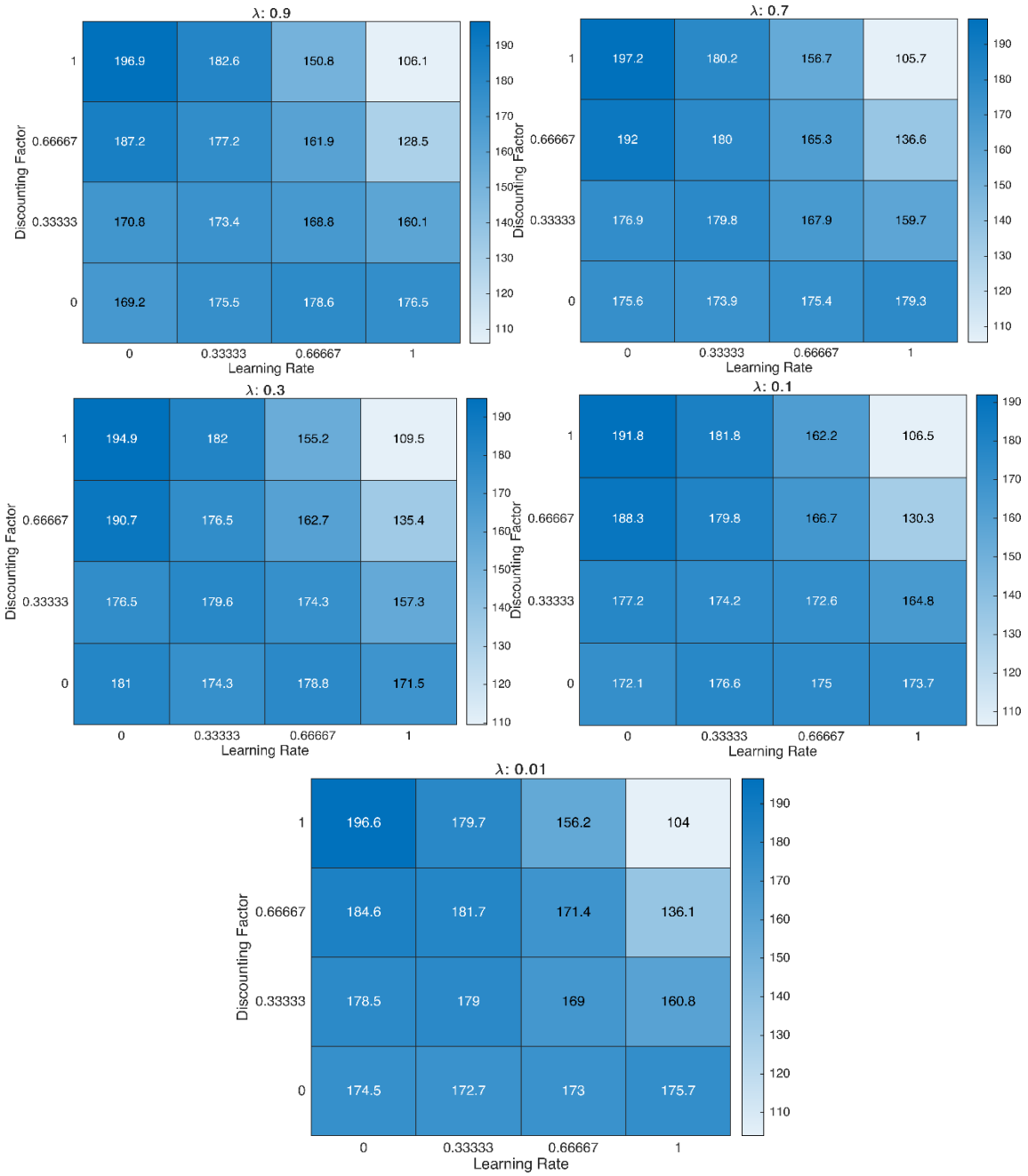


The result obviously shows a decrease in most of the states. Learning rate and discounting factor show effects like Q3.

Considering same values for 2 states (0.5, 1):

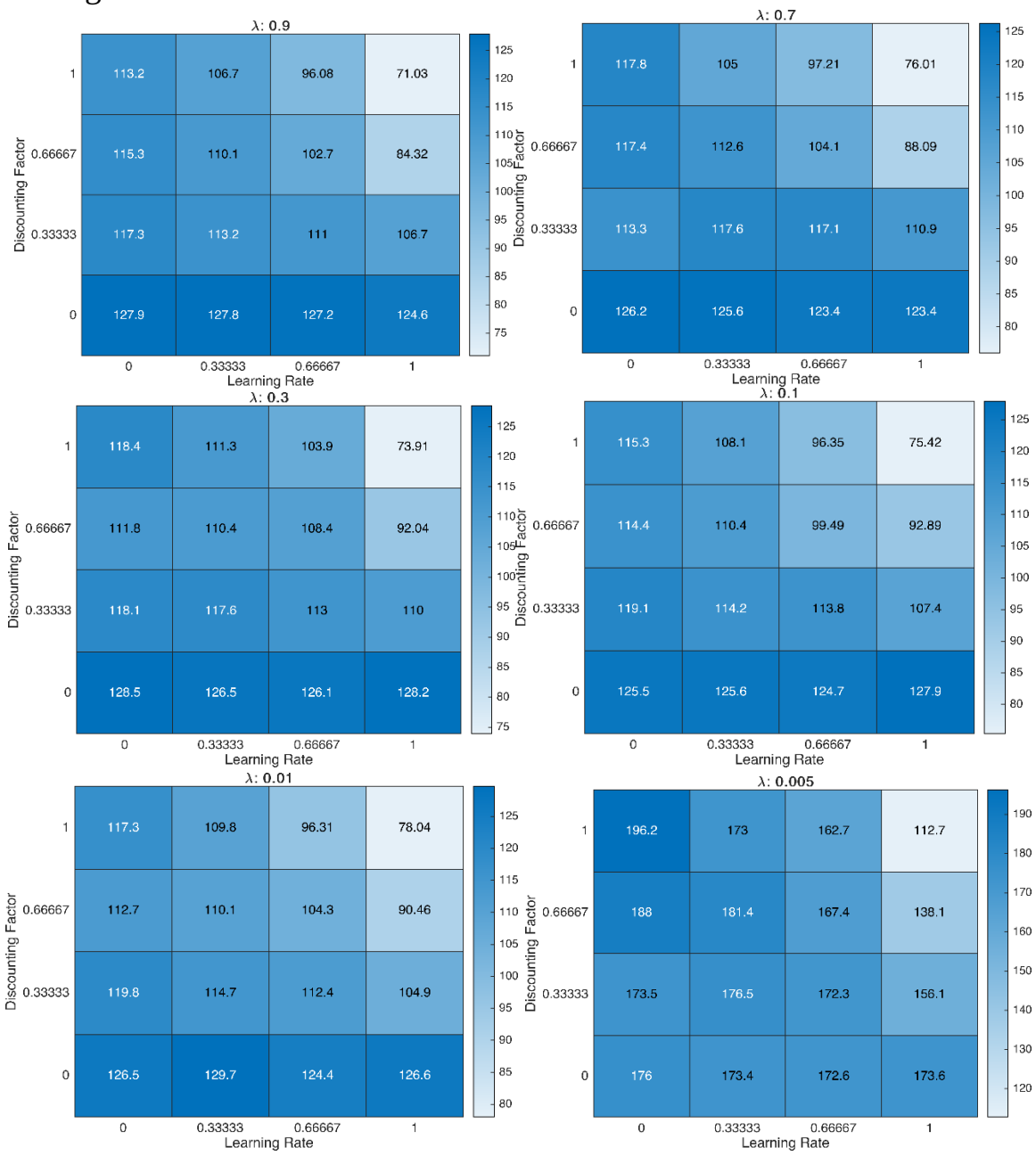


5. Implementing TD(λ) for 1 goal in the map:



Since the rate of converging to optimal probability by TD Lambda is higher, I expect to obtain lower steps by increasing lambda from zero to 1. Comparing lambda 0.01 and 0 shows that small values of lambda lead to better convergence. But bigger lambda does not help the model.

For 2 goals with same values:



For 2 goals, TD Lambda seems to have a strong effect since there is a change from lambda near 0 to lambda near 0.01. By increasing lambda, we have the impact but compared to lambda 0.01, higher lambda yields a weaker result!