In The Name of God

Negin Esmaeilzade 97104034

Advanced Neuroscience HW6

Dr. Ali Ghazizade

Part 1,2)

For this part, the mapGenerator function is used. This function generates a map as requested and displays the inputs of cat location, rat location and the target location with filled squares. During the execution of the algorithm, this function is called for visualization. To implement the algorithm, the model-based method is used, which its relationships can be seen below:

$$\hat{V}(S_t) = r_t + \gamma \sum_{a} \sum_{S_{t+1}} (\pi(a, S_t) P(S_{t+1}|S_t, a) \hat{V}(S_{t+1}))$$

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

P(r|St) is considered to be zero and the discount factor parameter is considered to be 1. While each state value is updated via this formula using the valueUpdator function, the probabilities for all actions at each state is updated via these:

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

$$\pi(a_i, S_t) = \frac{\exp(\beta m_i)}{\sum_1^n \exp(\beta m_j)} \qquad m_i \rightarrow m_i - \epsilon\, P(a_i|S_t) E(\hat{V}(S_{t+1})|a_j) \qquad \text{If action 'a}_i\text{' is not chosen}$$

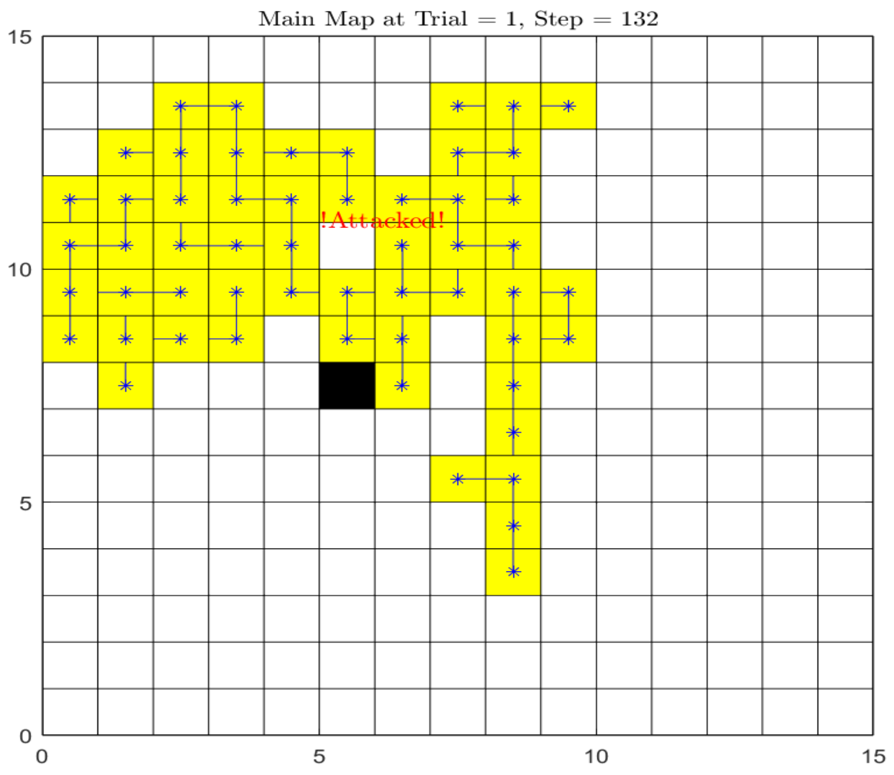
In the way that when ever we reached a state ‘S2’ from a previous state ‘S1’ the probability of taking the action of moving to S2 form S1 is updated in a map_width*map_height* number_of_actions matrix due to the value difference. In the above formulas β and ϵ are considered equal to 1.

In this part the learning process is recorded in 100 trials and can be seen in the ‘demo1-train’ file. Also, the trials 500 to 510 are recorded and can be seen in the ‘demo1-test’ file.

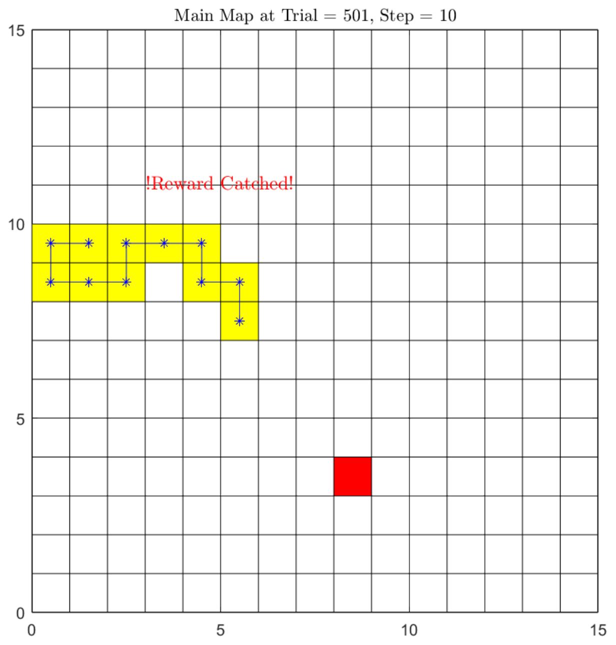
Note1: In all trials the cat position is fixed at [4.5,9.5] and the target position id fixed at [5.7,7.5].

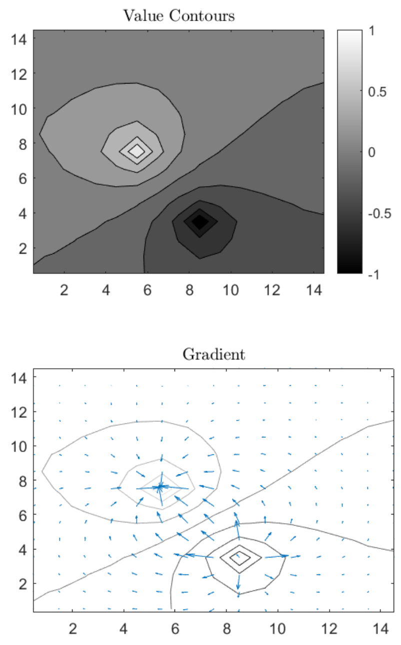
Note2: The target reward is considered equal to 1 and the cat punishment is considered equal to -1.

Here is a sample path the rat traverses before learning the map at the first trial:



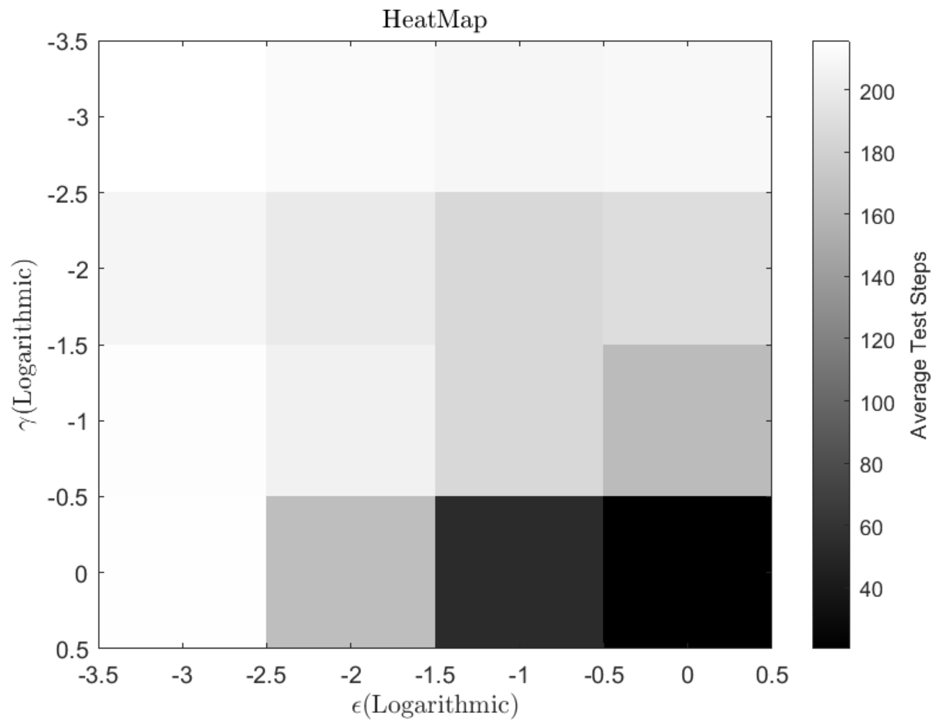
And here is a sample path the rat traverses after learning the map at the trial 501:





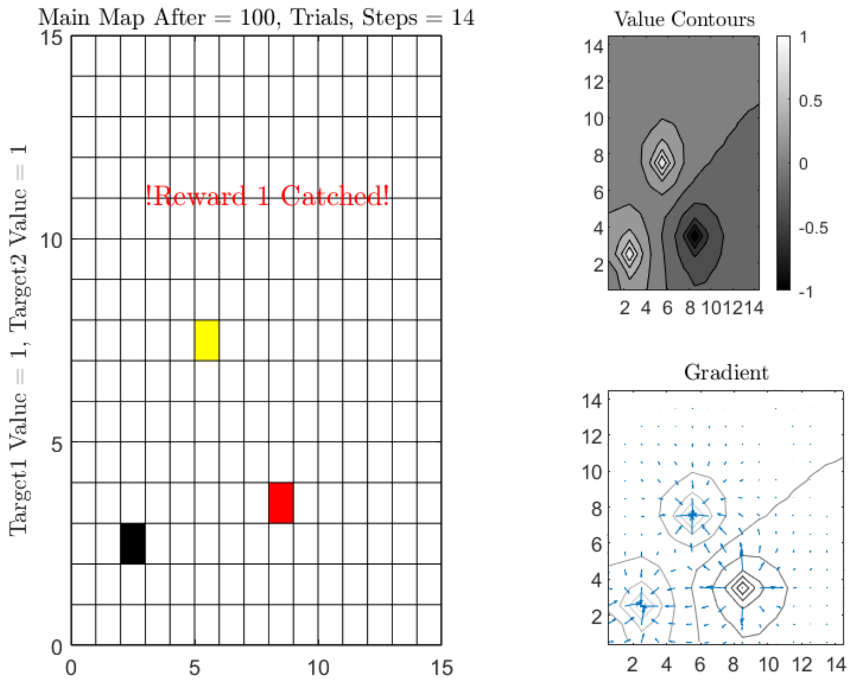
As we can see the rat learned the cells' values and picks less random moves.

Part 3) Four different values of [0.001,0.01,0.1,1] are considered for the learning rate (ϵ) and also for the discount factor (γ). For each combination the rat is learned through 470 trials and tested after, in 30 trials. The average steps needed for the rat to reach the goal at the test phase is measured for each cell of the heat map. However, in order to reduce the random noise, this procedure is run for 10 times an the average heatmap resulted is shown below:

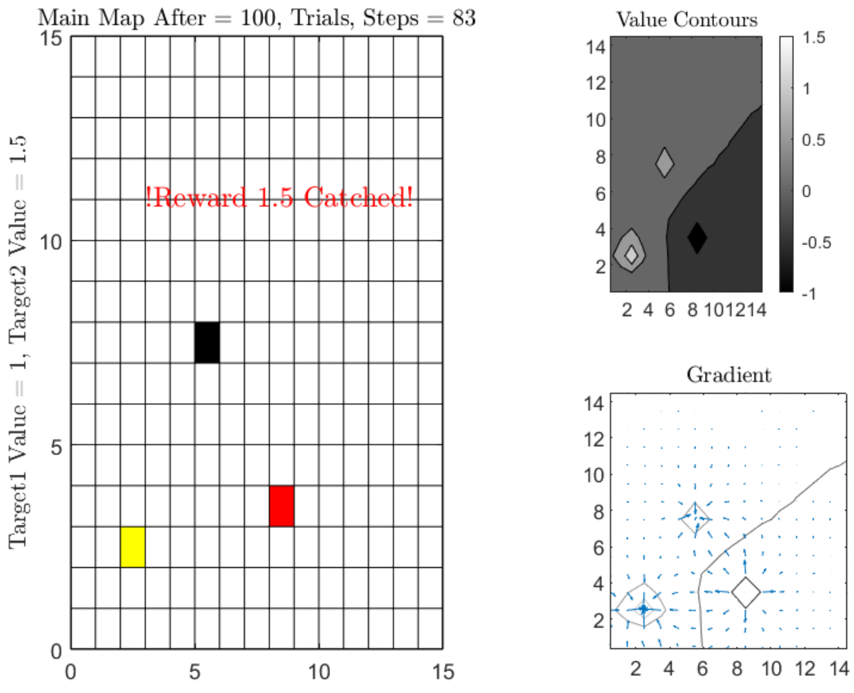


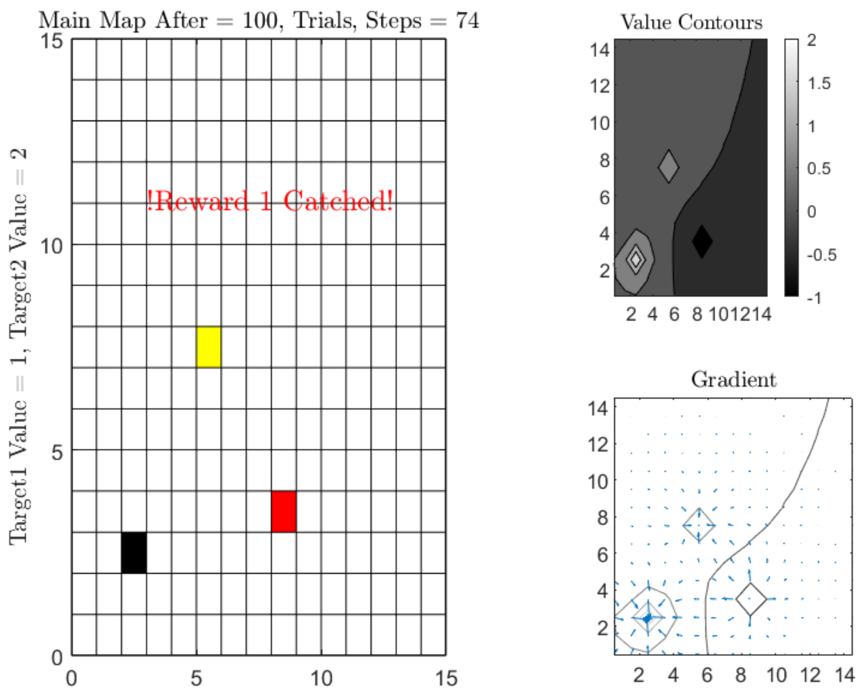
As we predicted the learning rate and the discount factor have direct relations with the average steps. Which means that when the learning rate is higher (bounded to 1) the algorithm can converge to the optimum target in less trials.

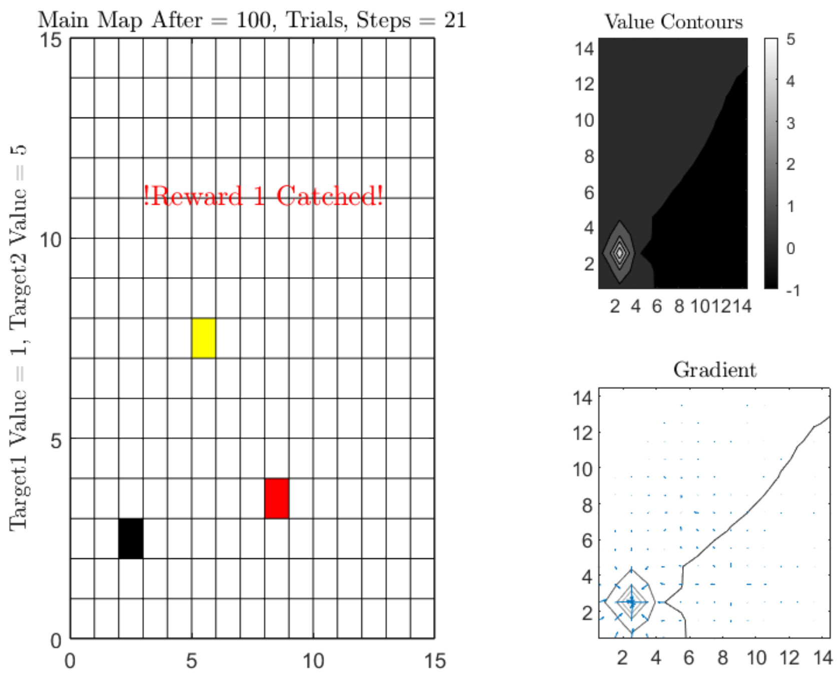
Part 4) In this part a second target is added at the fixed position of [2.5,2.5]. 5 different values of [1,1.5,2,5,10] are considered for this target and the results after 100 learning trials are reported below:

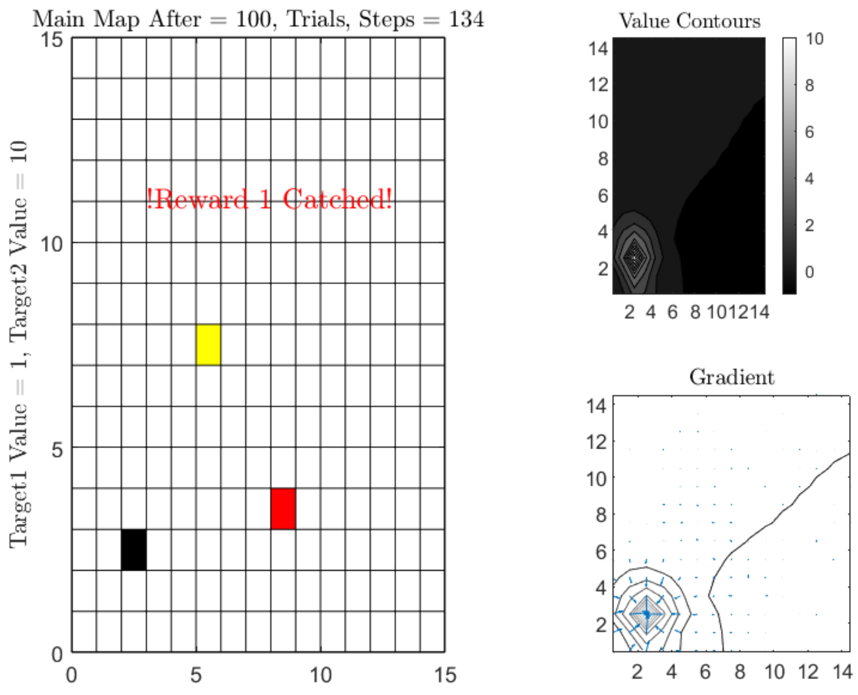


As we can see when both targets have the same value, all conditions an probabilities to reach for both are almost the same.



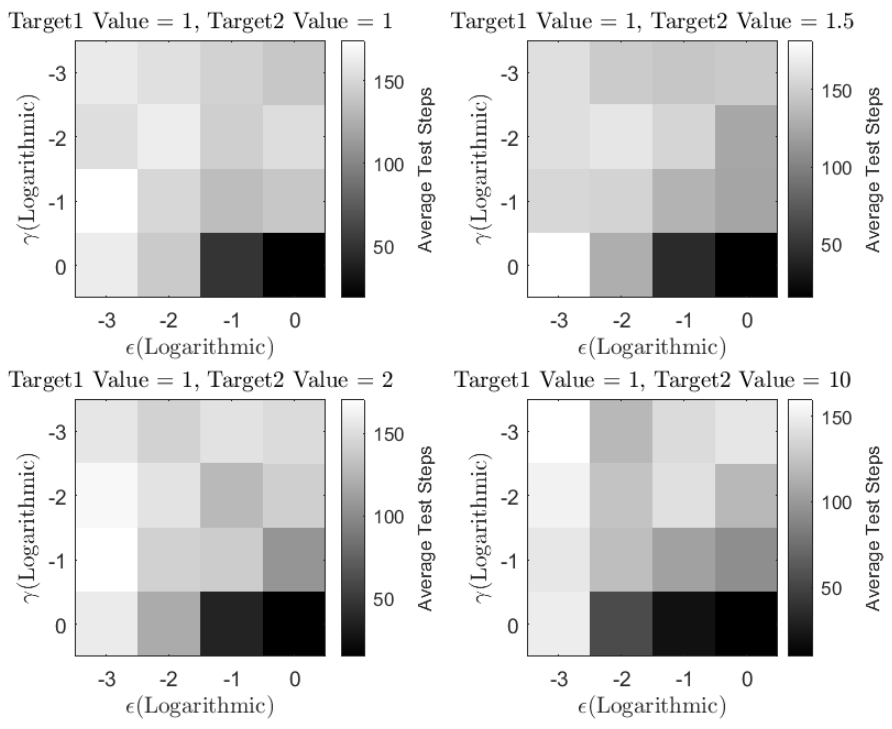

Main Map After = 100, Trials, Steps = 74
Target1 Value = 1, Target2 Value = 2
!Reward 1 Catched!
Value Contours
Gradient


Main Map After = 100, Trials, Steps = 21
Target1 Value = 1, Target2 Value = 5
!Reward 1 Catched!
Value Contours
Gradient



As we can see when a target's value is dominant, the rat learns to reach each that more than the other target.

Here are the heat maps, considering the effect of learning rate and discount factor for each target value.



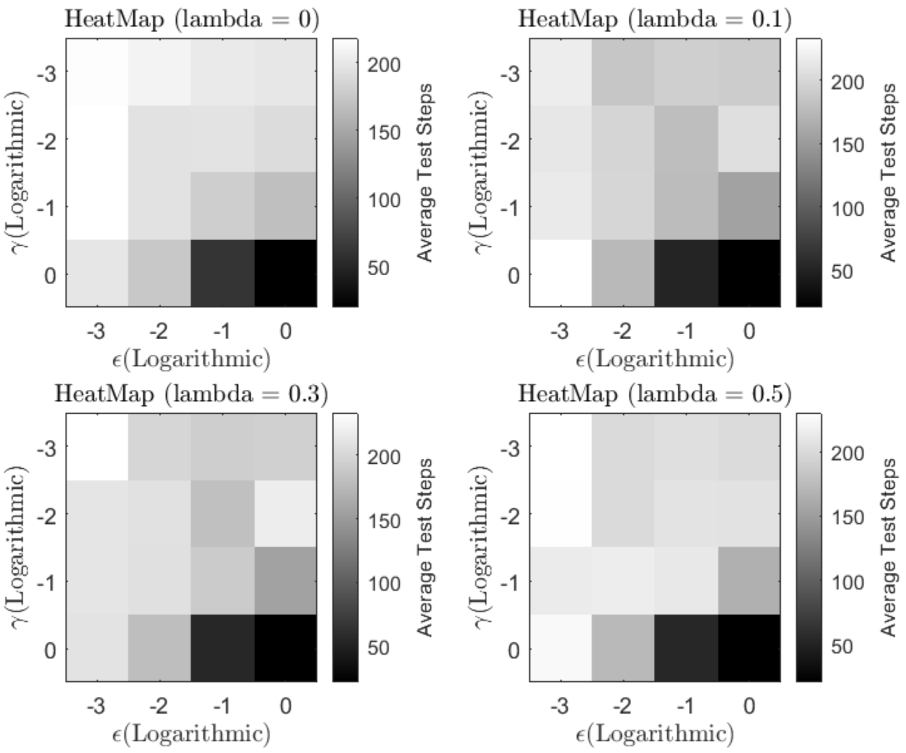
In comparison to the previous part, having only 1 target, the average steps get less, means that when we have two targets, the probably of reaching a target gets higher and the rat can learn this in less trials.

Part 5) For this part, a list of visited cells in made considering the visiting order. Then using the function 'TDLambdaUPDator' the value of all previous states in a single trial is updated proportional to the amount of the current state value update, using this formula:

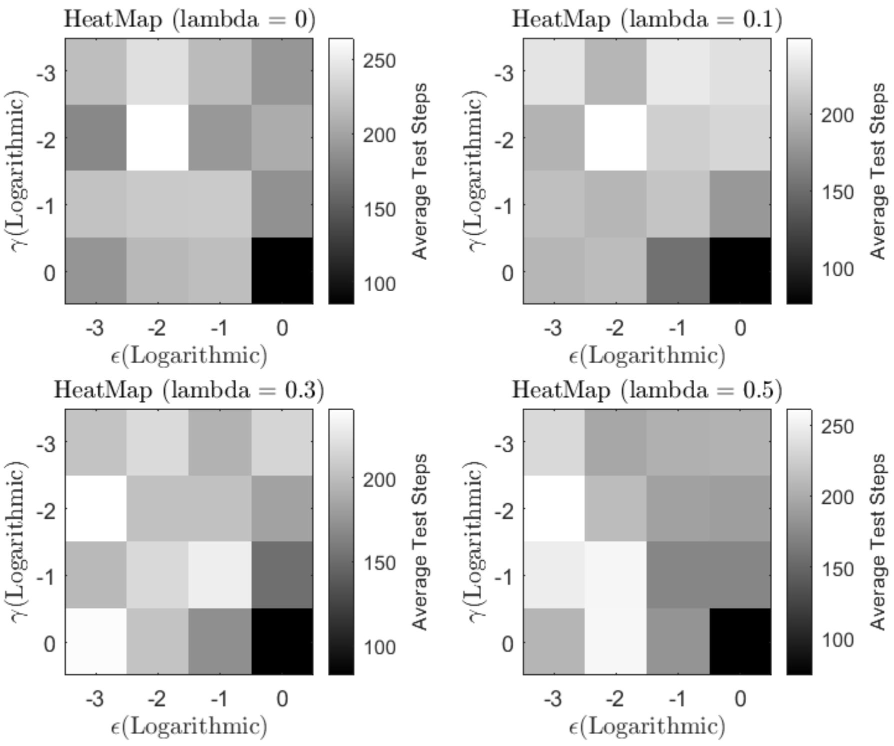
$$\Diamond \quad V_{t+1}(s(t-t')) \quad := \quad V_t(s(t-t')) + \lambda^{t'}(V_{t+1}(s(t)) - V_t(s(t))) \qquad \forall t' = 1, 2, \ldots .$$

4 different values of [0,0.1,0.3,0.5] is considered for the λ parameter. Where $\lambda = 0$ represents no difference with the previous section algorithm.

Here is the resulted heat map for each λ amount, considering 470 training trials.



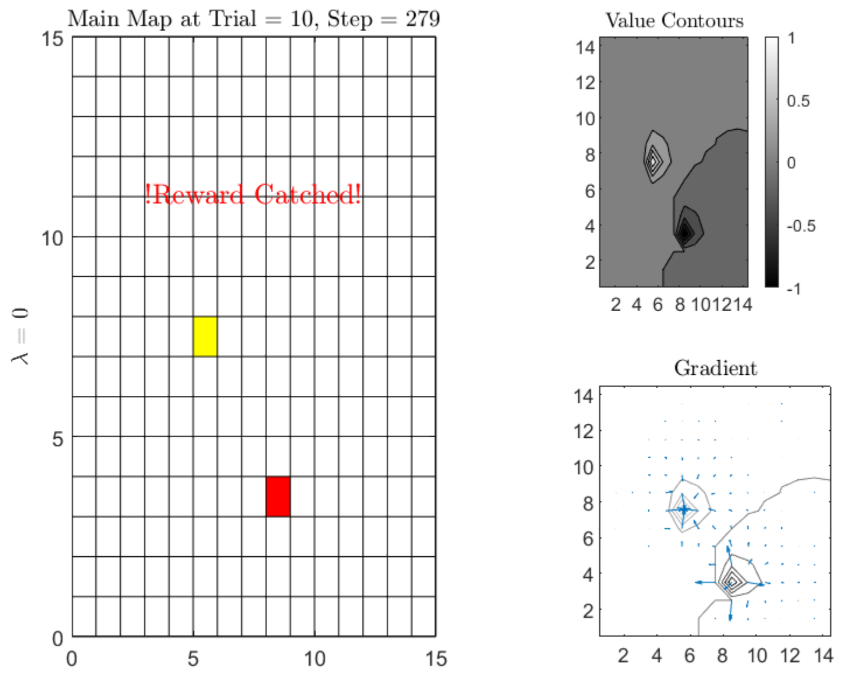
Here is the resulted heat map for each λ amount, considering 40 training trials.

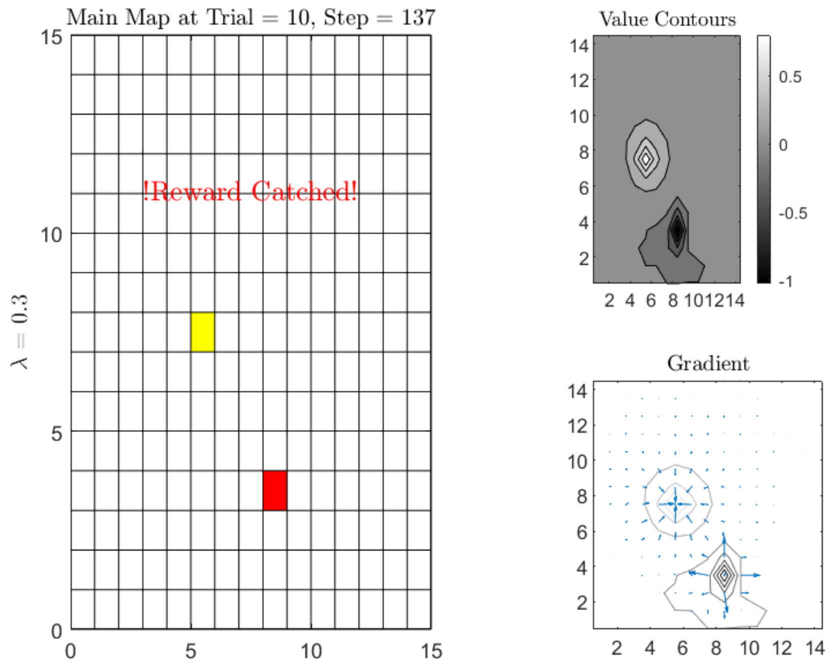


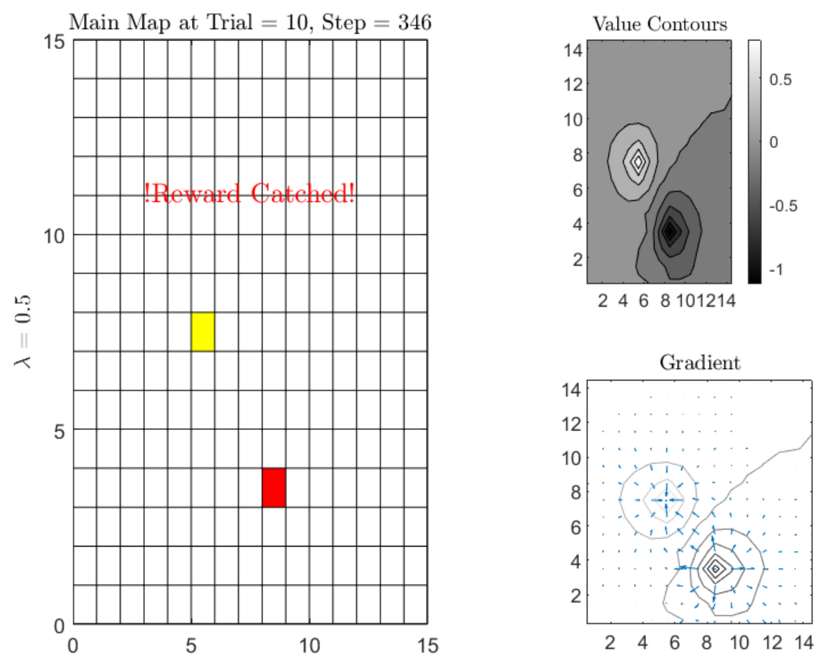
As we can see this method speeds up the algorithm for learning map values in the way that the heat map after 470 trials has almost no difference with the heat map after 40 trials, means that the rat could be trained well after only 40 trials.

The difference of the learning speeds can be detected comparing the two demo files ‘demo1-train’ and ‘demo2’.

We can see the map and the contour plots for each λ amounts after 10 training trials below:

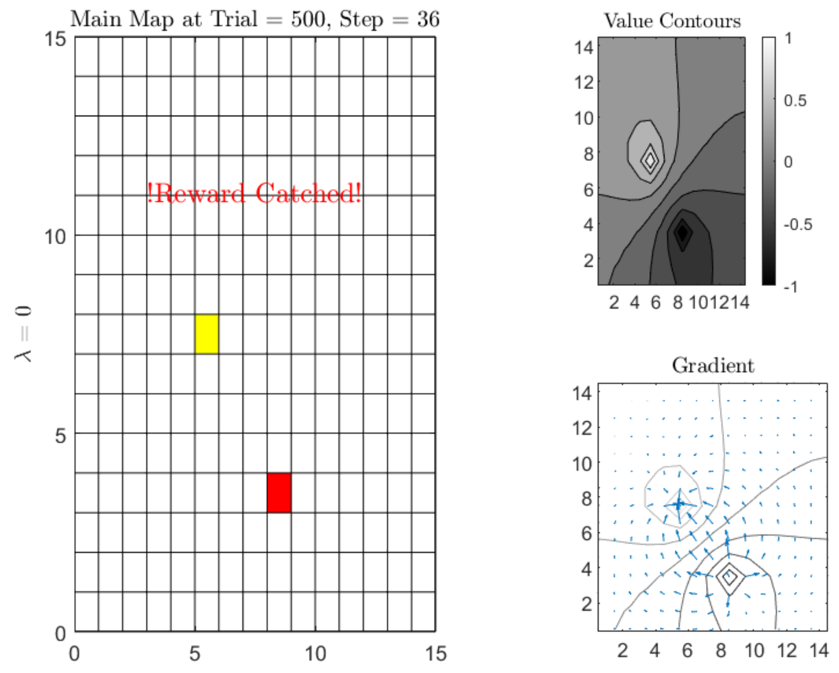


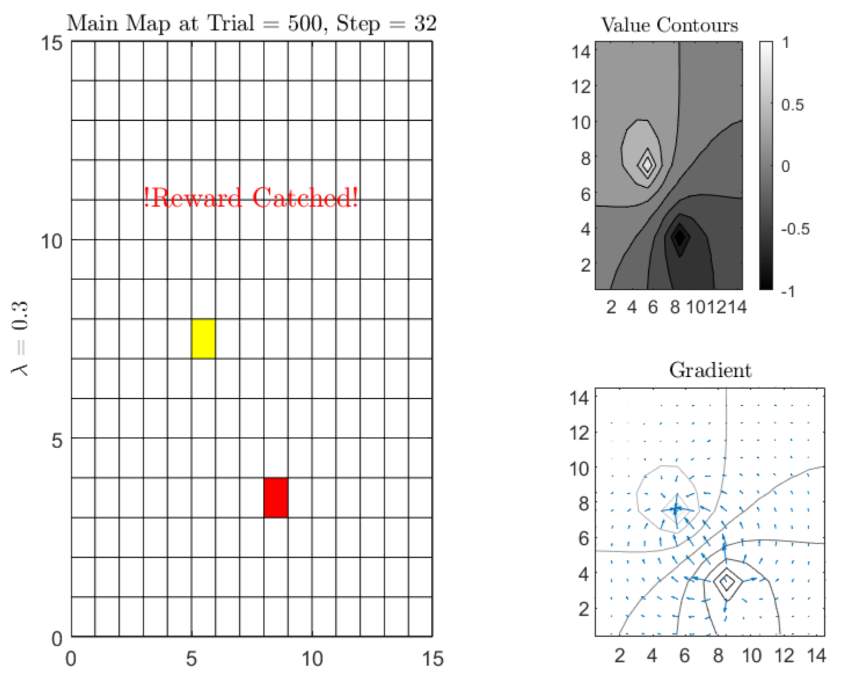


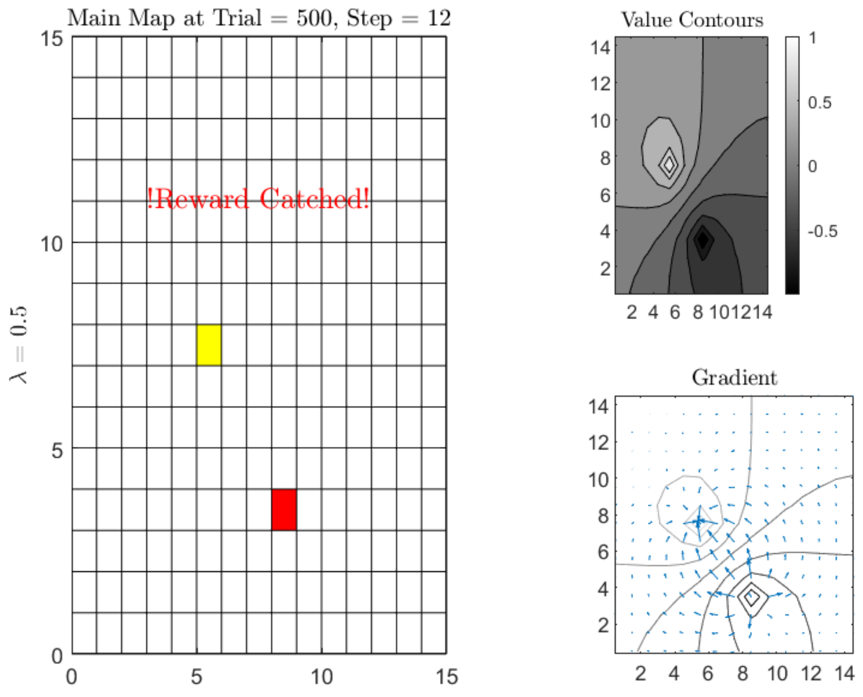


The gradient in each plot well shows the difference in map exploring speed of the two algorithms.

We can see the map and the contour plots for each λ amounts after 500 training trials below for comparison:




Main Map at Trial = 500, Step = 32
!Reward Catched!
λ = 0.3
Value Contours
Gradient


Main Map at Trial = 500, Step = 12
!Reward Catched!
λ = 0.5
Value Contours
Gradient