Advanced Topics in Neuroscience HW6

Aliakbar Mahmoodzadeh 98106904

1. Plot the paths before and after training Plot the Gradients and contour plot of learned value:

in this problem, to learn the map and each value of the state, we use Model Free and Epsilon_Greedy method and in the Epsilon-Greedy method, we consider two things; first, we use constant Epsilon across all trials, and second, we use dynamic Epsilon-Greedy and it means the value of Epsilon doesn't constant and it changes during the trials.

So first of all, we learned the map with Model-free and it works in this way: two state have meaningful reward and punishment and it means when an agent goes to this state, the trials are dine and we want to maximize the reward and it means we want to find nearest and most valuable state.

So in this method, the agent in each current state finds the most valuable state around itself and goes to it and after that the previous state's value is updated and put into the Value Matrix that saves the each state's value for each trial

also agents have 4 movements (Up, Down, Left, Right), but in some state, its movement is restricted to (Right, down) and …

Agent updates the State's value by this formula:

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

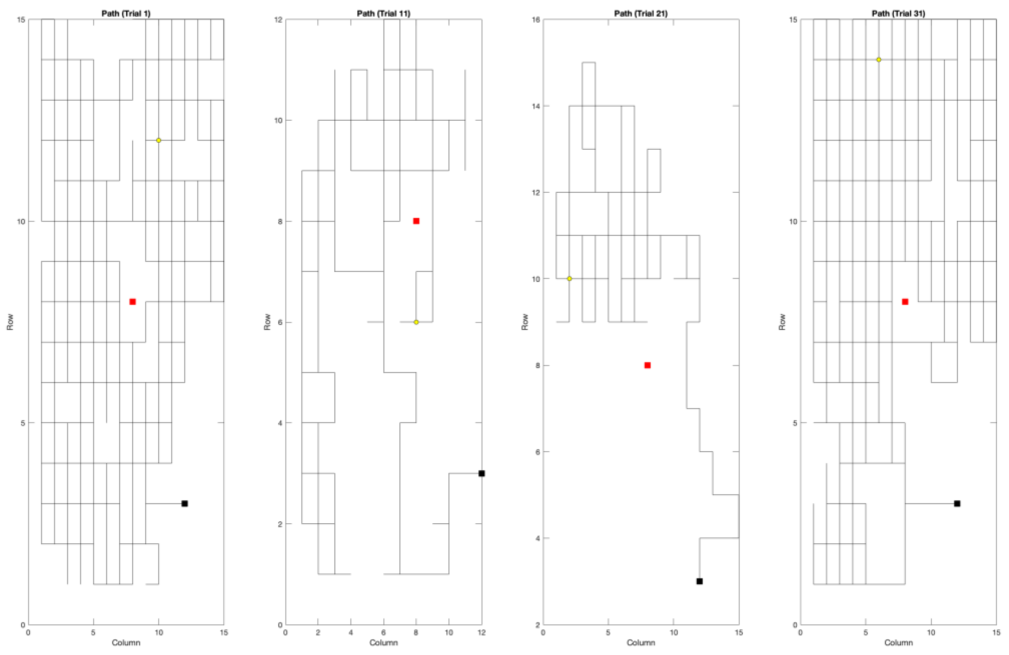
$$\delta_t = r_t + \gamma V(S_{t+1}) - V(S_t)$$

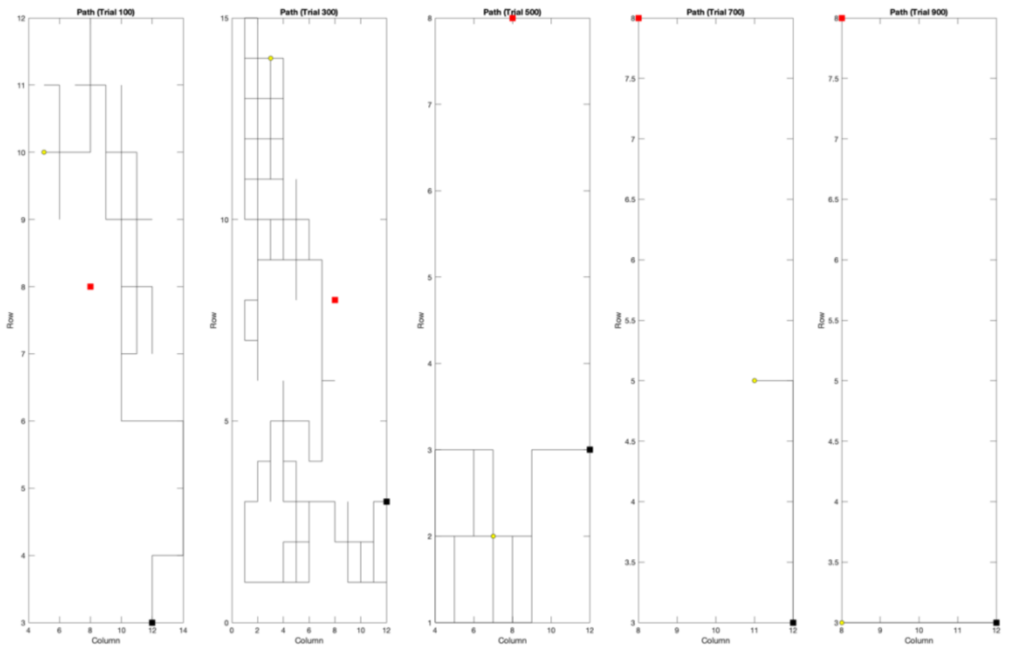
Etha and Gamma are the constant values, Etha is the learning rate and Gamma is the Discount factor.

This formula tells that the agent want to maximize its state values so it goes to the state the has the higher value than its current value and if the highest value for next transition is the same for some state for instance the values of next state are (1.0, 2.1, 2.1, 2.1) the agent randomly(Uniform) goes to one of the highest state. And this is the agent's policy, and this is the result for this method:

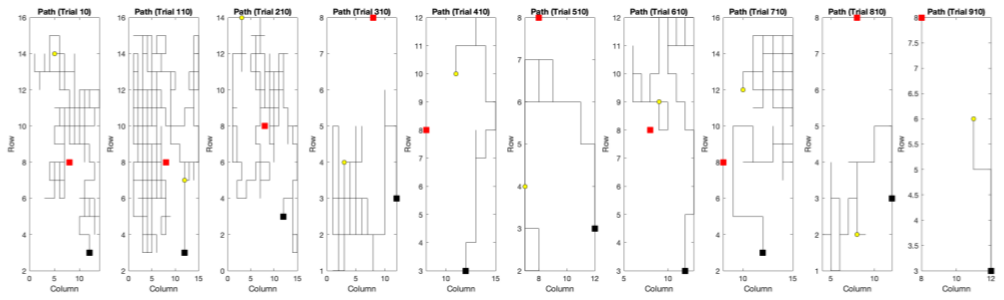
First: Model Free

1.1) Plot the Path:





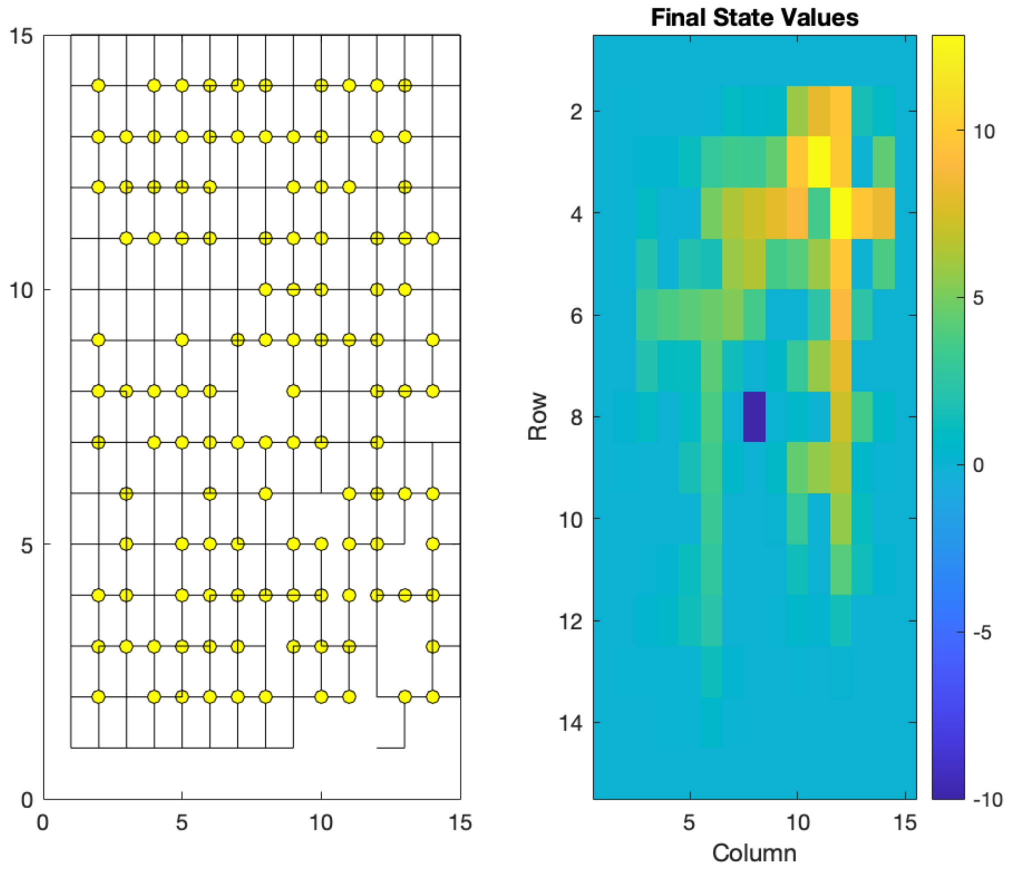
Trial 10 to 910:



The red Point is the Punishment state and the Black point is reward State and the agent randomly is placed with Yellow point(Please zoom in)

The location of each important state is:
target = [3, 12]; % Target location (black)
cat = [8, 8]; % Cat location (red)

and this is the heatmap that shows the final value of each state in this method:

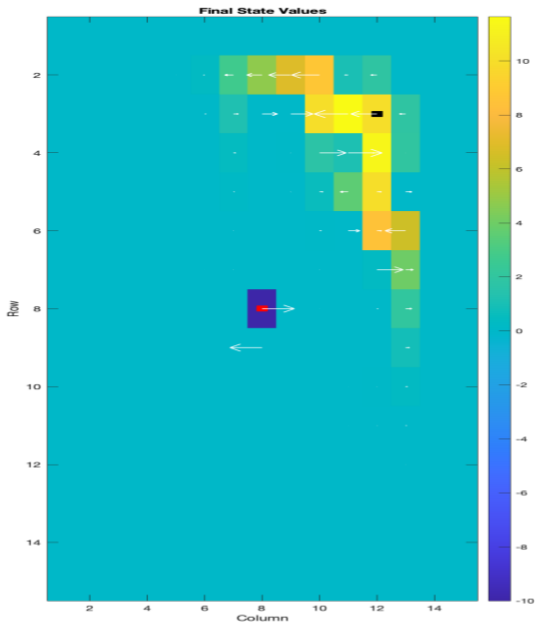


the Right plot shows the final value of each state and the state with punishment shows as Dark blue and state with reward shows as light yellow at (row =3 and column = 12)

the left plot shows the initial starting point(it is random) and each line is created in trials and shows the agent's path across all trials.

!! The Video is in Gif folder !!

And this is the gradients:



This plot tells us the agent how moves in general because the gradients mean in each state, the agent went to what state more than another state

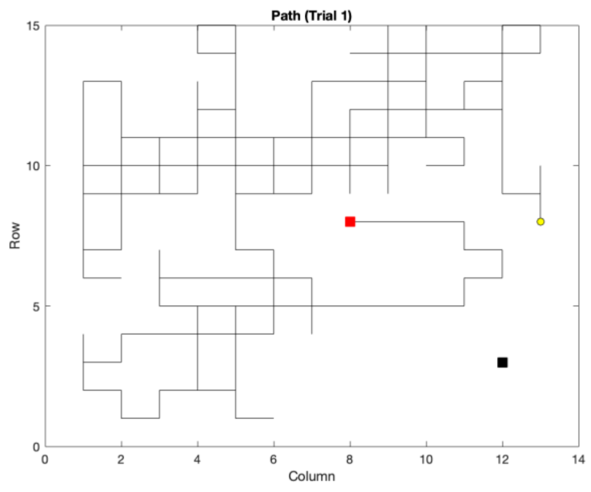
And as you see, the gradients show that the agent escape from punishment state to the reward state

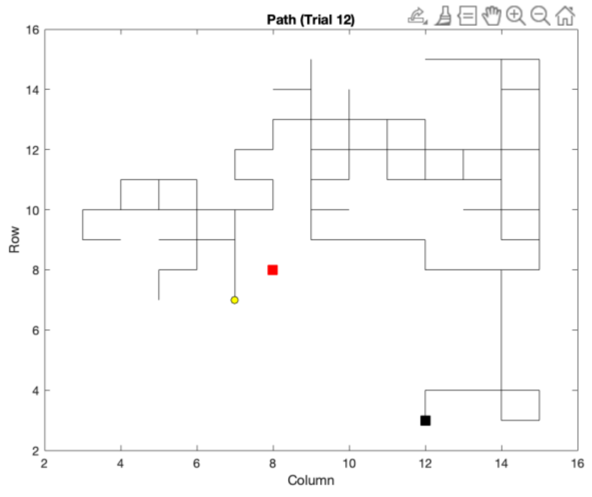
2. Seccond: Epsilon_Greedy

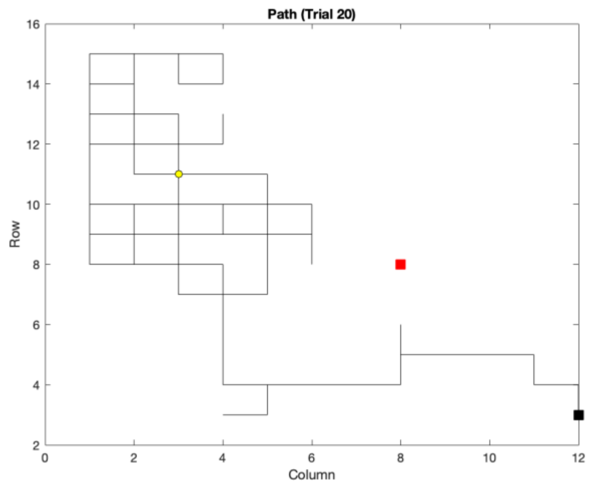
There is a trade-off between exploration and exploitation, and it means if agent always goes to the state with the highest value, it cannot see another state and is it possible that there is a better way and it missed.

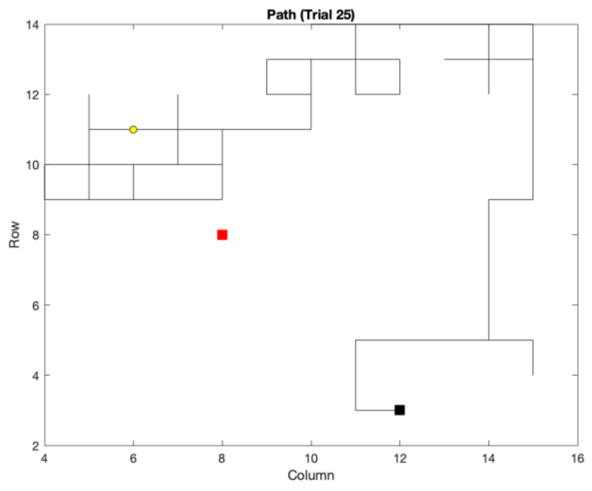
So to solve this issue, we use the Epsilon-Greedy method and which means when an agent wants to decide to go, it goes to the highest possible state with ε Probability, and with 1- ε probability, it goes to another state.

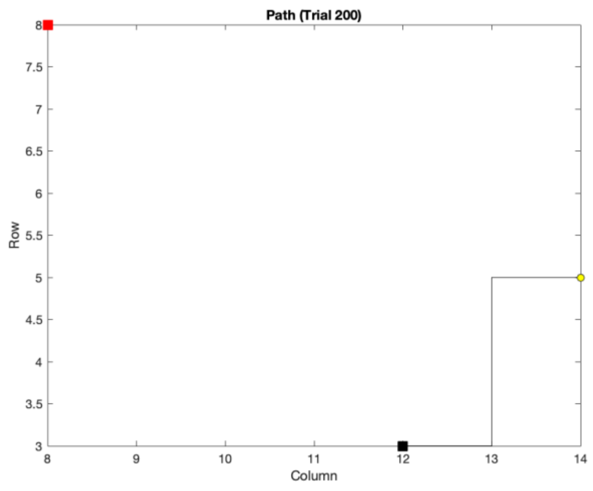
1.2) Plot the Path:

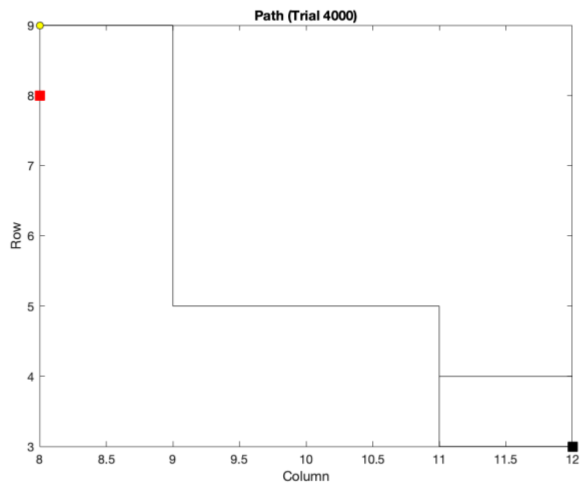




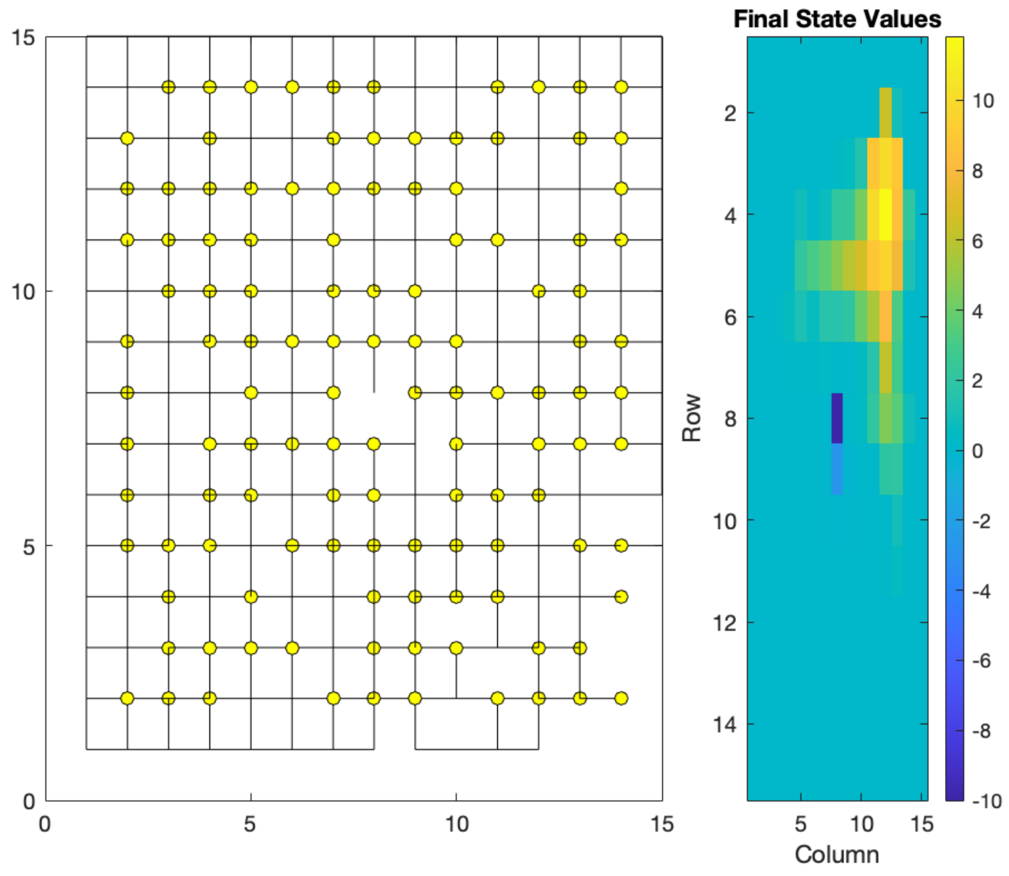






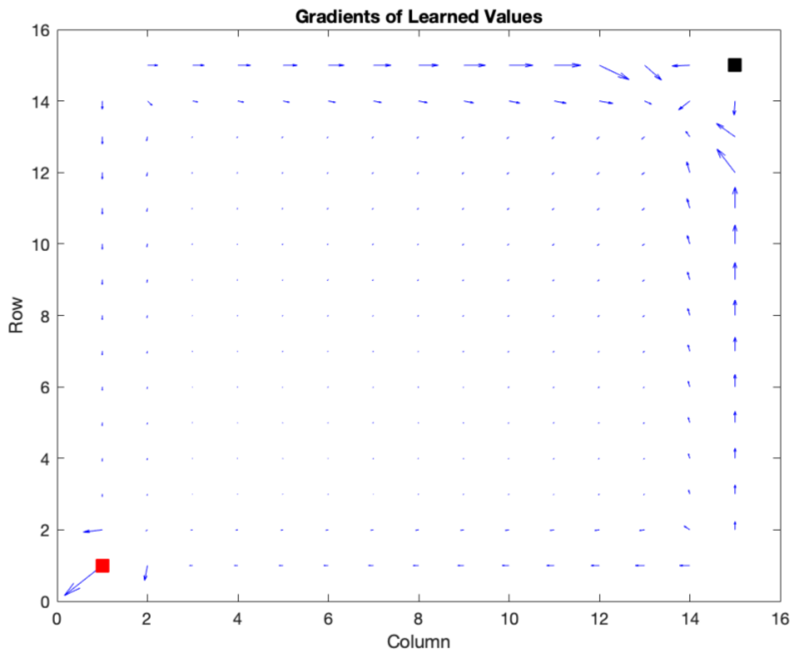


and this is the heatmap that shows the final value of each state in this method:

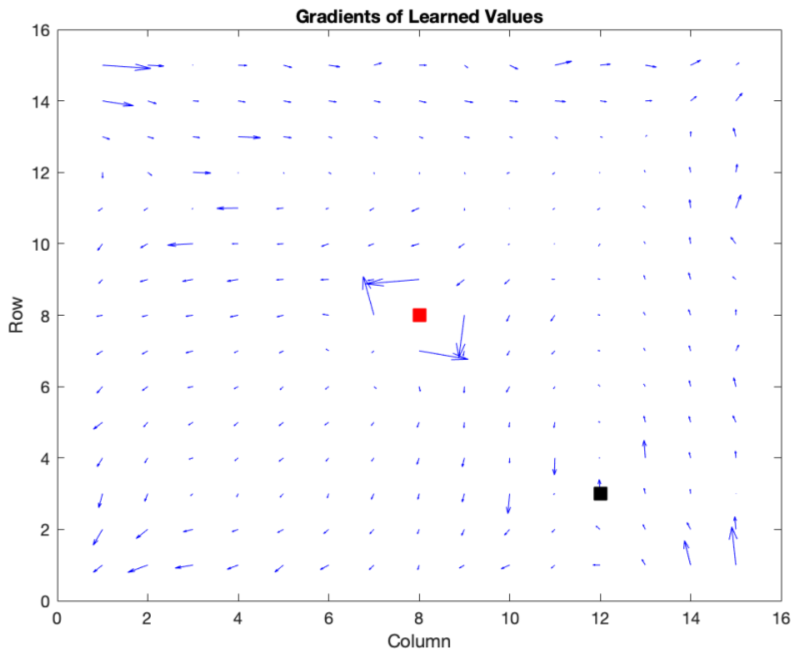


and for two location I plotted the gradients, this is result:

Location 1:



Location2:

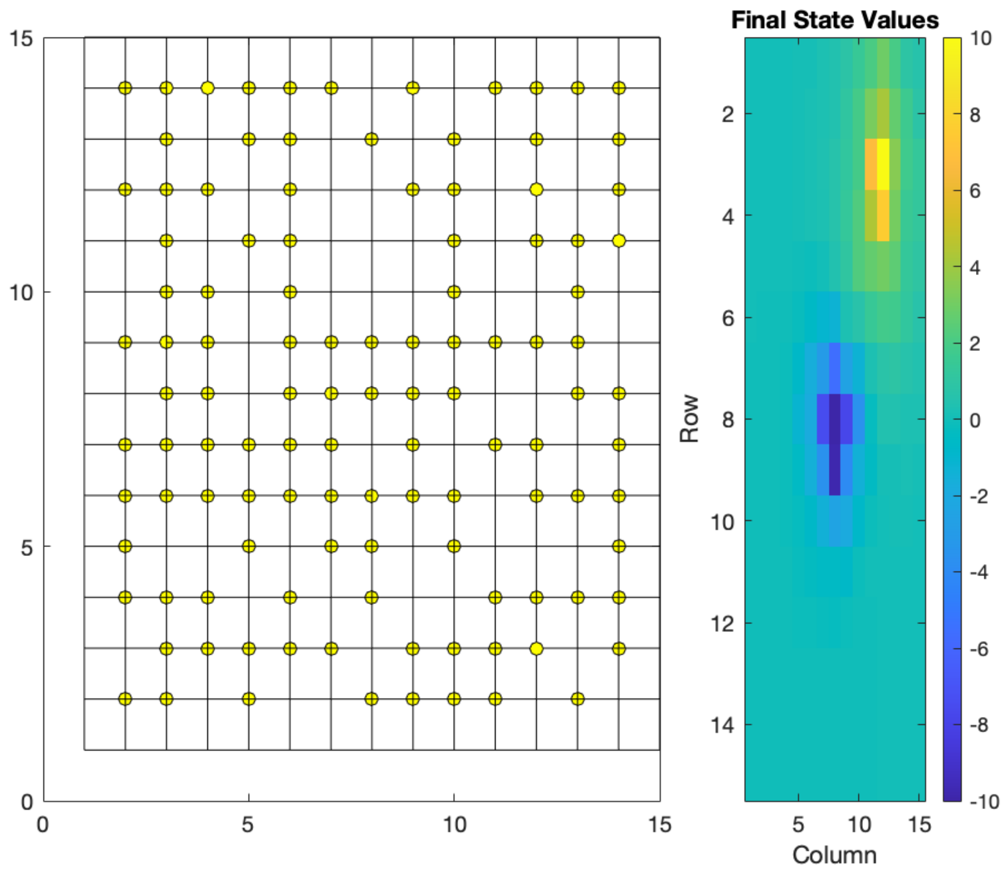


Explain the plot: And as you see by this method we went to more states than previous method, and the gradients are more smooth, so by this method, we have a chance to find a better way and maybe don’t fall into local optimum; but it can increase our steps because goes to the useless state with lower value.

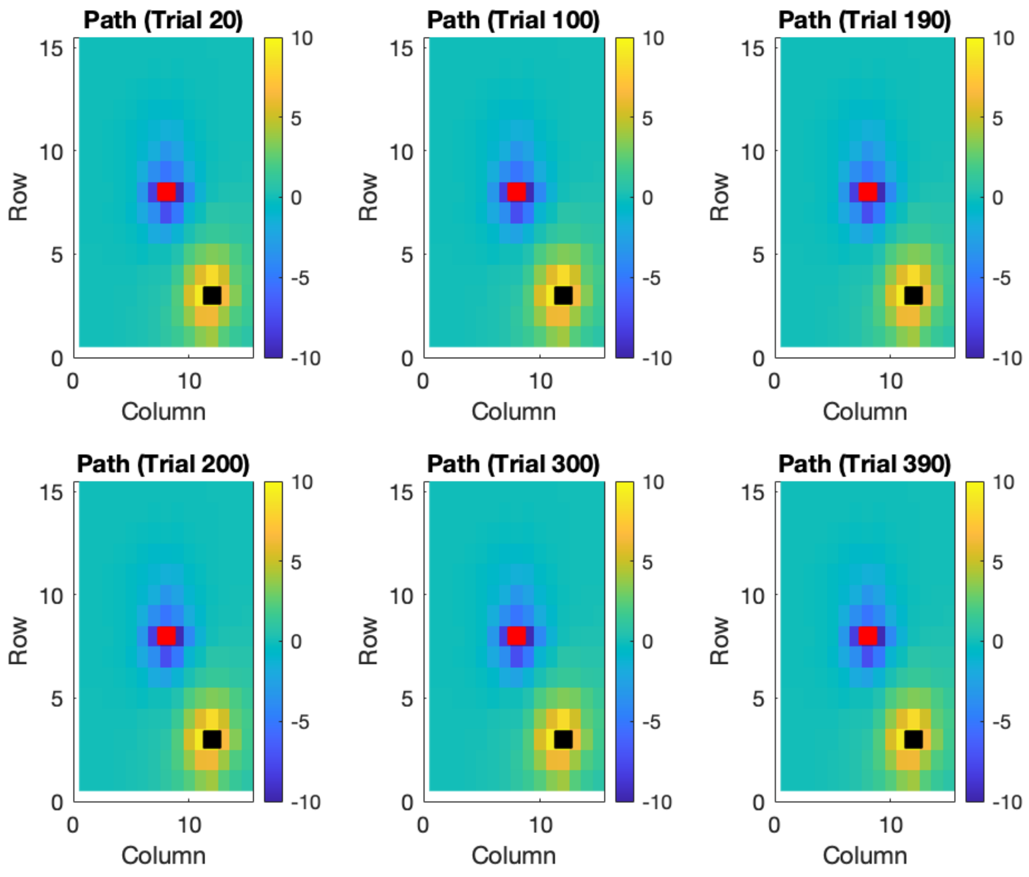
3. Seccond: Epsilon_Greedy

Now we can improve the algorithm with this trick, we can use the Epsilon-Greedy method but the Epsilon is not a constant value and it changes in each trial and the logic behind this is that after some exploration we are going to be familiar with the maze so after that, we can only find the higher value in state and transition to them
So if we set Epsilon as a parameter that increases in each trial, it means we do more exploitation more than exploration.

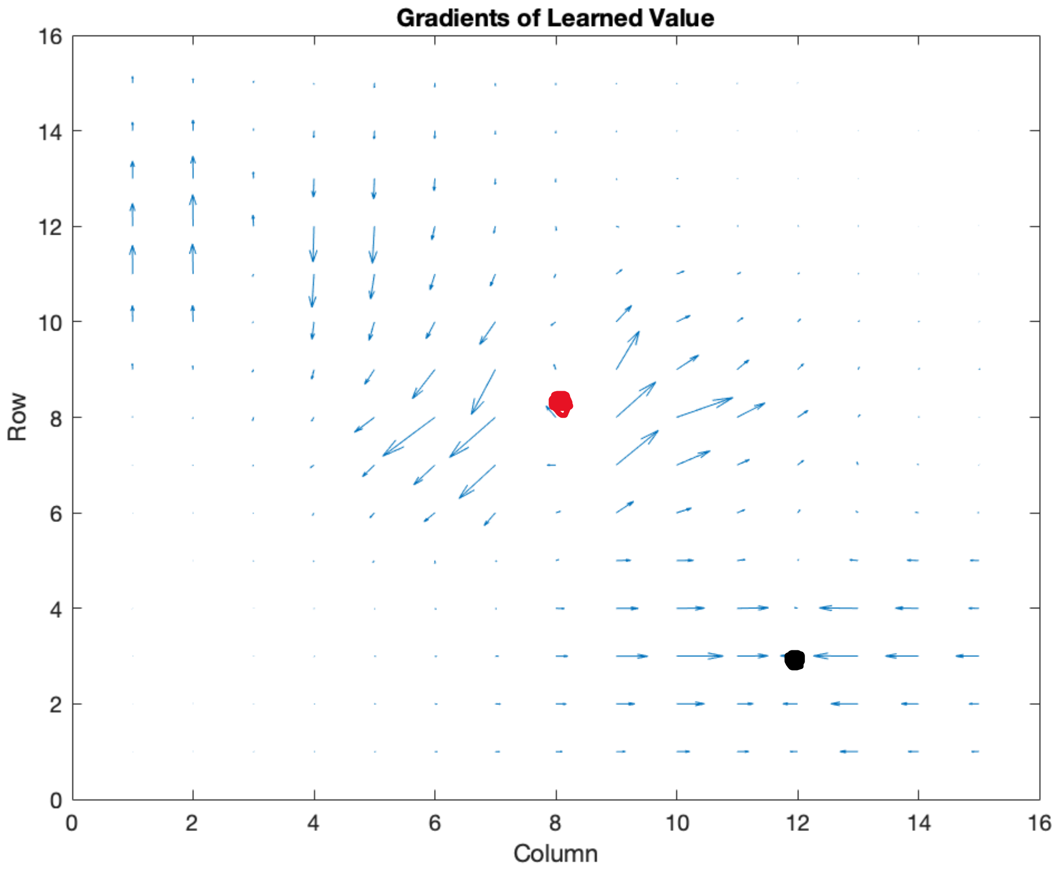
1.3) and this is the heatmap that shows the final value of each state in this method:



as you can see, this plot has a very different pattern compared to the previous plot, the most significant advantage of this method is the agent as you see went to all points of the table (with the same trail) and discover a any transition for giving the reward and it means it is more faster than previous method.



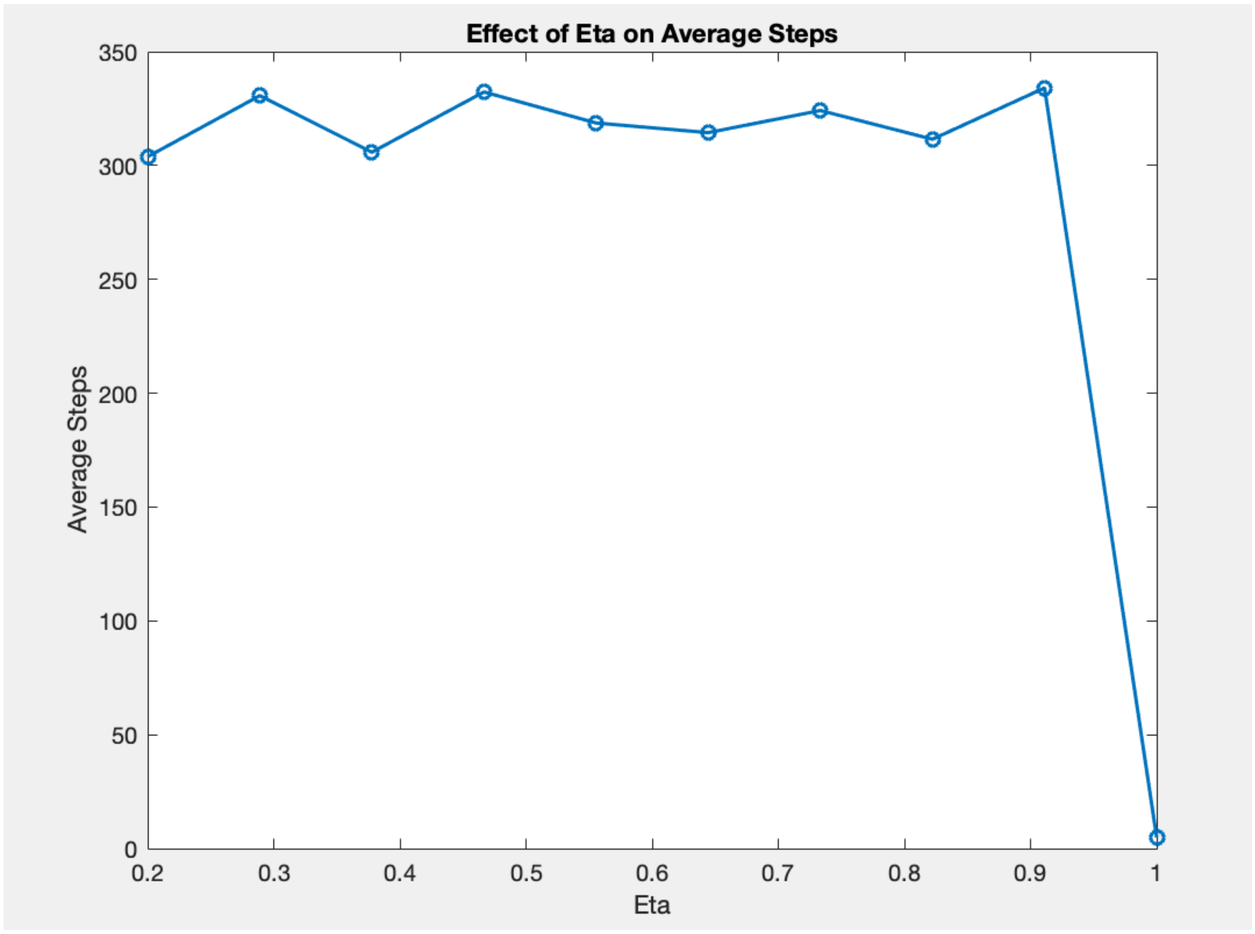
And this is the gradients:



The punishment is in (8,8) and the reward is in (3, 12) and as you see it is more smooth than previous gradients and it is very convincing because the agent wants to escape from red point to black point and this is the exact thing that this plot say

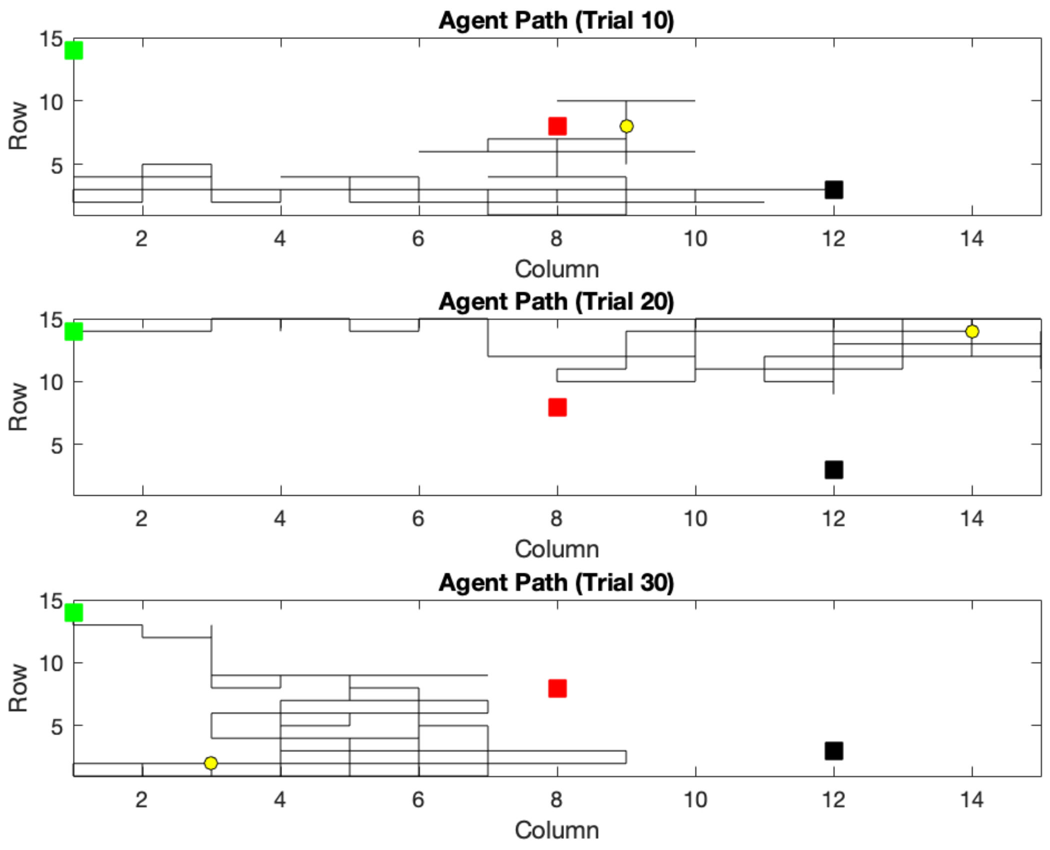
2. What is the effect of the learning rate (ε) and discount factor (γ) in this problem?

For small learning rate, the learning is slow and becomes faster as the learning rate increases. The same procedure has been done to discount factor gamma. The larger the discounting factor, the faster the training.

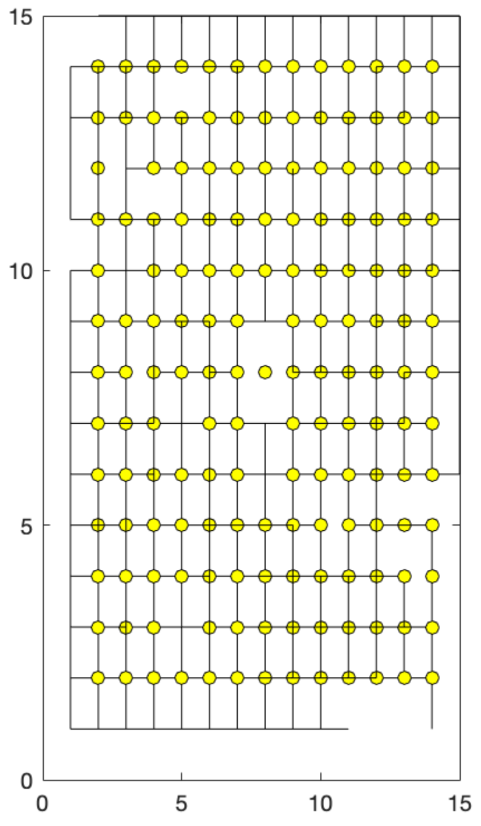


3. Two target squares with different reward

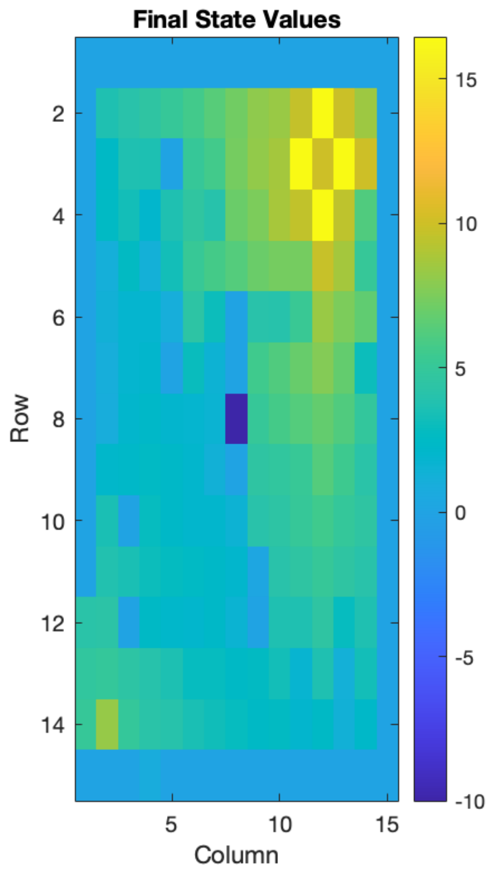
In this section, we put two reward state in the table, and the reward value of these states are not equal and reward_1 = 10 and
reward_2 = 10 and this is the result:



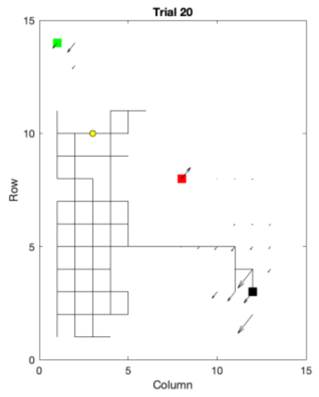
GRADIENTS:



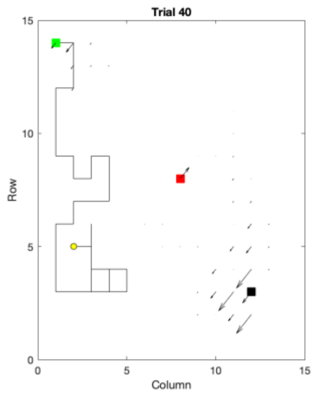
15
10
5
0
0
5
10
15



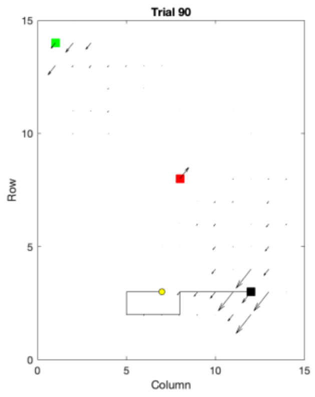
Final State Values
Row
Column



Trial 20
Row
Column

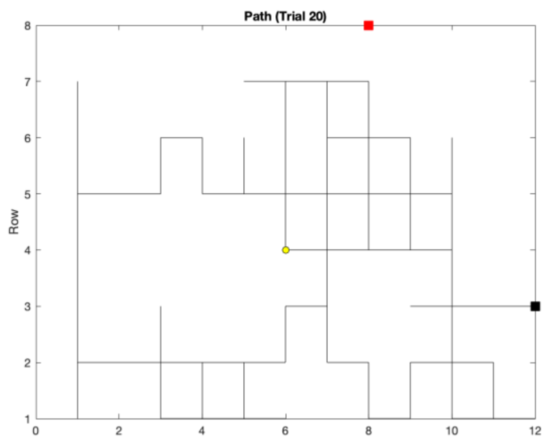


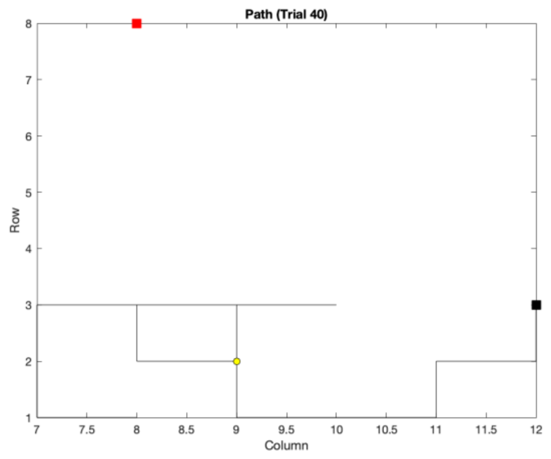
Trial 40
Row
Column

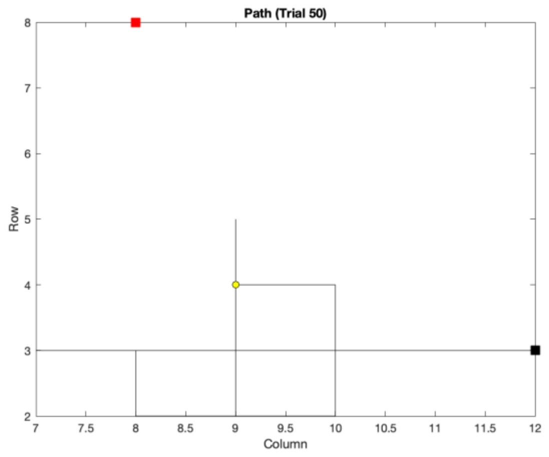


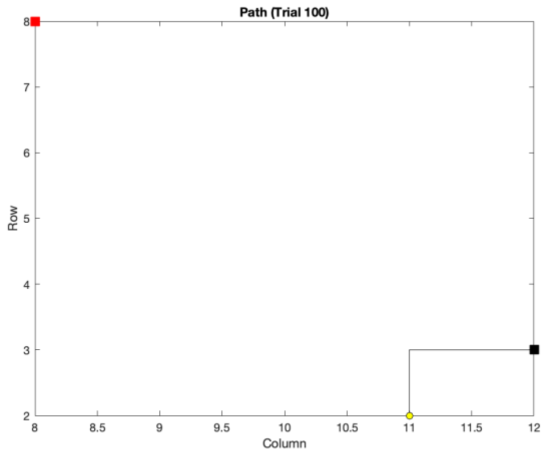
Trial 90
Row
Column

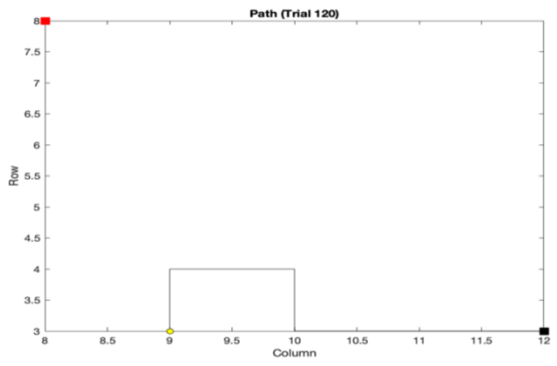
4. Implement the TD(λ) algorithm:



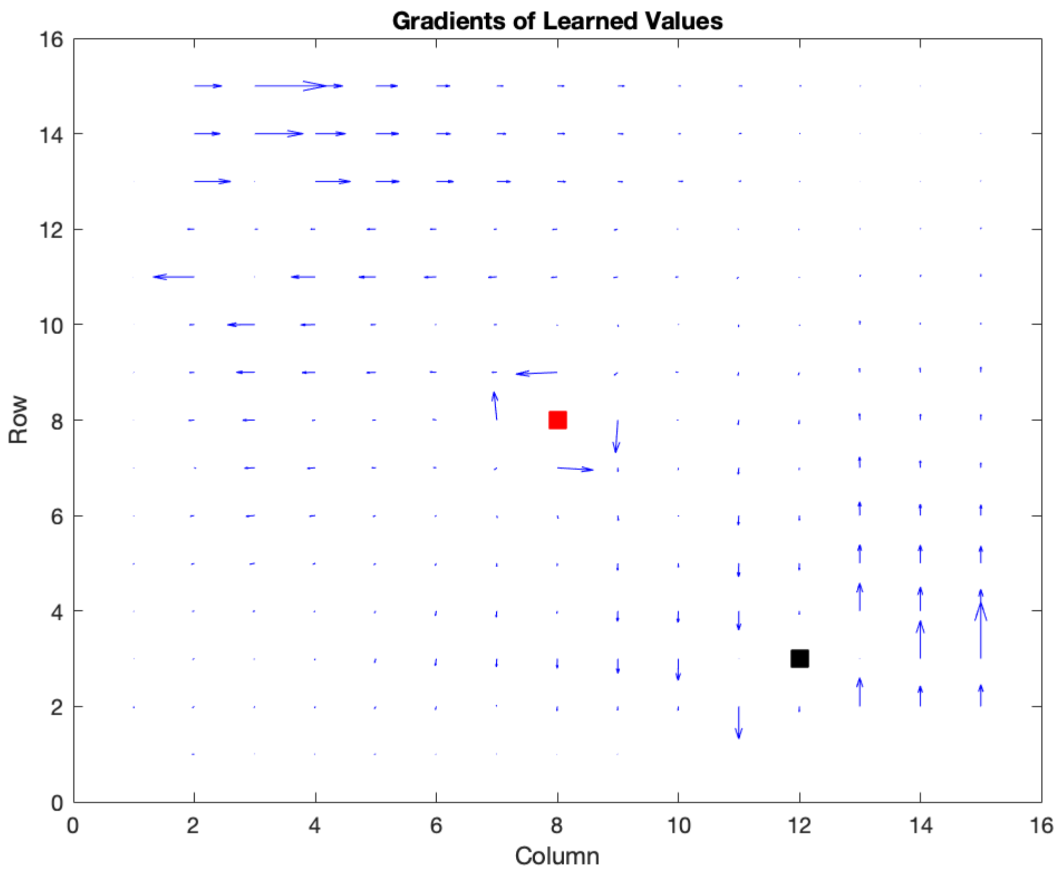








This is the gradients:


Gradients of Learned Values
Row
Column