Programming Assignment 4
Manthan Thakar

1. Source Code

The structure of the source code is as follows:
You can execute the program by executing the following command:

```
python run.py [OPTIONS]
```

The reinforcement learning methods are implemented in `rl.py` and the analysis code pipeline is implemented in `analysis.py` (responsible for running different analysis configurations and plotting graphs).

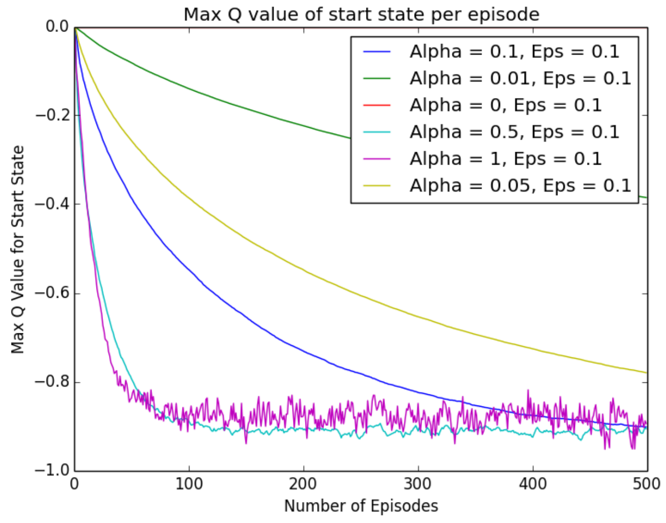
Although, the report contains limited graphs, other plots can be found inside `plots/` directory.

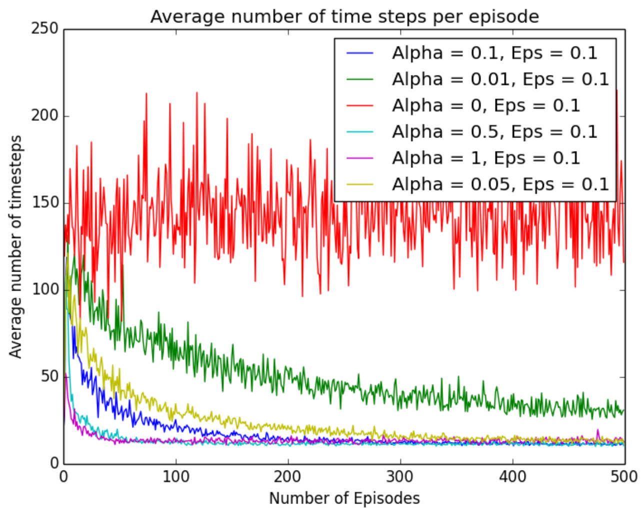
2. Effect of alpha on Q-Learning and Sarsa Lambda

In order to measure the effect of Alpha on Q-learning and Sarsa Lambda, 500 experiments of both algorithms are run with 500 episodes and varying values of alpha. The average maximum value of Q for the start state and average number of time steps per episode are plotted for each episode and are averaged across experiments. For this particular experiment, the value of epsilon is kept at its default value of 0.1.

Q-Learning

In the following section, we discuss the effect of alpha on Q-Learning algorithm





- **`Alpha = 0`**
  For `alpha = 0,` we observe that the maximum value of Q stays at 0 due to the way we make updates to Q.

$$Q(s_t,a_t) \leftarrow (1-\alpha)\cdot \underbrace{Q(s_t,a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\left(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1},a)}_{\text{estimate of optimal future value}} \right)}^{\text{learned value}}$$

  From the update rule (shown above) of Q-learning, we can show that when alpha = 0 the algorithm only updates the Q value to the old estimate (since the part multiplied by alpha becomes 0), which is 0 in the beginning. This can be verified in the figure above where the max Q value for the start state remains 0 (blue line, figure on the left).

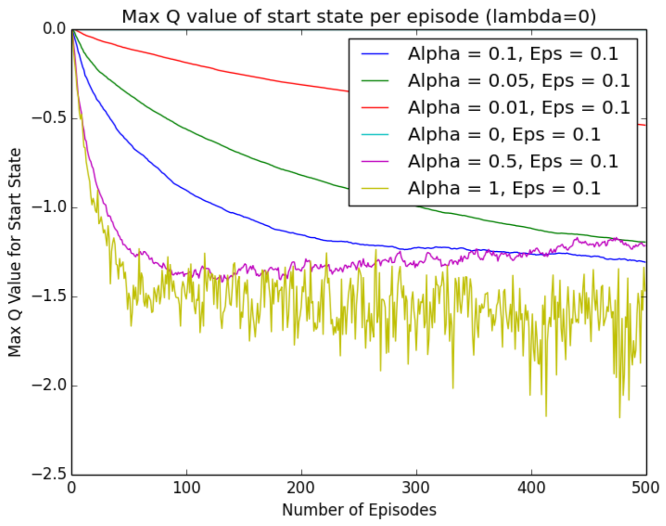
  The average number of time steps are significantly more when then alpha = 0, which is justified since the updates only depend on the estimated old value of Q and hence take longer to converge.

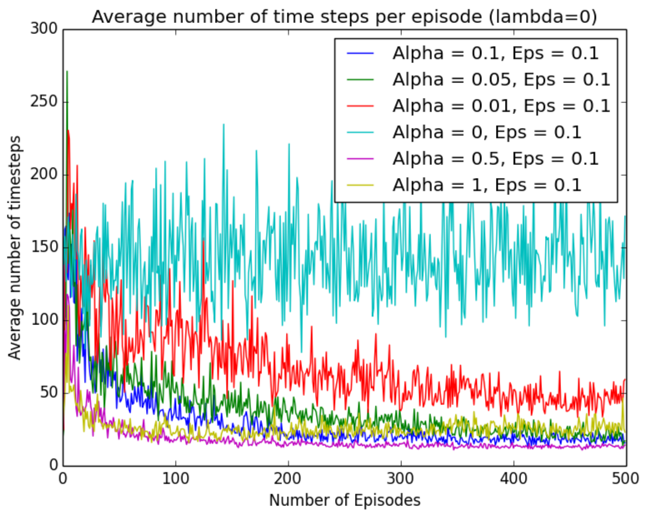
- **`Alpha > 0`**
  For other values of Alpha, we can see that the maximum value of Q for start state decreases as we increase alpha. Therefore, it can be deduced that the Maximum value of Q is inversely proportional to the learning rate for alpha > 0. It can be shown from the update rule (shown above) that as alpha approaches value of 1, the update rule gives more important to the old value of Q(St, at) and assigns more weight to the learned value (in the picture), this leads to smaller values of Q for bigger values of alpha. Interestingly, for 0.1, 0.5 and 1 values of alpha, the maximum Q value converges to the same value after 500 steps.

  Consequently, it can be deducted from the time steps graph that the time steps remain mostly in the range of 50 to 100 for alpha > 0. Which means on an average each episode takes lesser time steps to converge than when alpha = 0. Interestingly, for 0.1, 0.5 and 1 values of alpha, the average time steps also converge to the same value after 500 steps.

Sarsa Lambda

In the following section, we discuss the effect of alpha on Sarsa Lambda algorithm.





As discussed for Q-Learning, the max Q value for start state remains 0 for Sarsa Lambda as well. For Sarsa Lambda, we can see that the Max Q values decrease more rapidly than Q-Learning, but the Max Q value is still inversely proportional to learning rate. Moreover, the smallest value for Max Q is smaller in Sarsa than in Q-learning.

The average number of time steps taken per episode for Sarsa Lambda follows similar trend as Q-Learning. But here, we can see that the average number of time steps are greater than Q-Learning. Moreover, the number fluctuates quite a bit for alpha > 0, whereas for Q-Learning, there's not much fluctuation for alpha > 0.

3. Effect of epsilon on Q-Learning and Sarsa Lamda

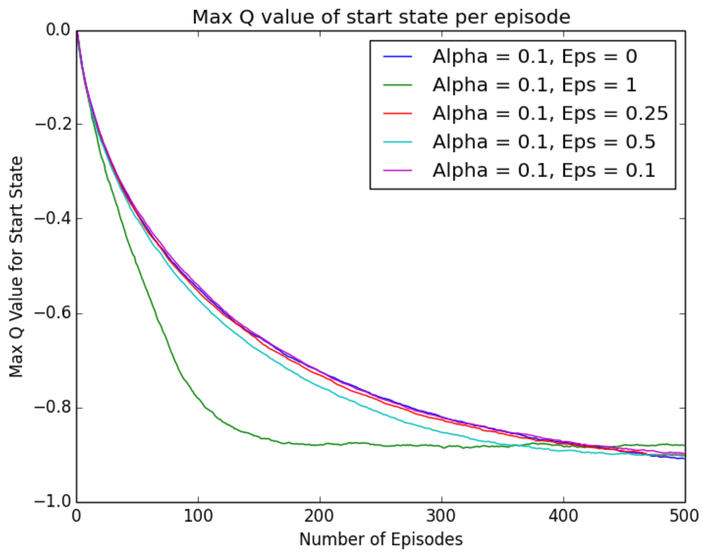
In order to measure the effect of Alpha on Q-learning and Sarsa Lambda, 500 experiments of both algorithms are run with 500 episodes and varying values of alpha. The average maximum value of Q for the start state and average number of time steps per episode are plotted for each episode and are averaged across experiments. For this experiment, however, the value of epsilon is changed.

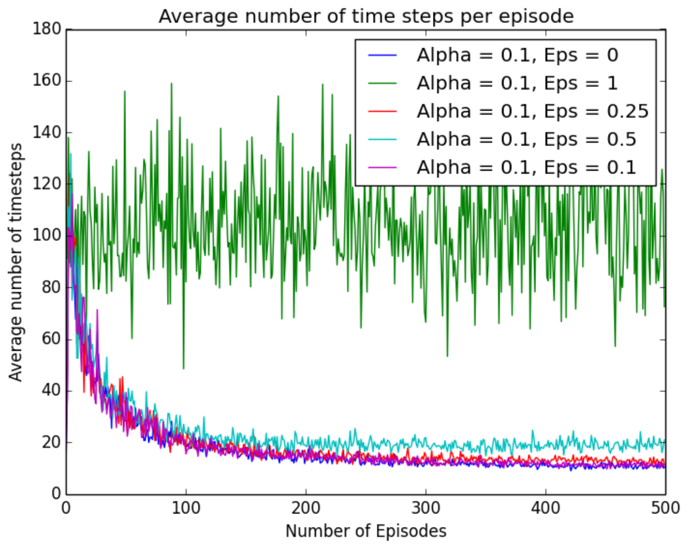
Q-Learning

From the previous section, we can see that alpha 0.1 and 0.5 give a good model, since the values are not close to zero for Maximum Q values and neither are they too small (like in the case of alpha = 1). Moreover, for these two values the number of steps per episode are also reasonable. Therefore, we show a few plots for alpha = 0.1, alpha=0.05 with different values of epsilon.

Alpha = 0.1:

For alpha = 0.1, we plot (below) maximum value for Q (left) and average times steps per episode (right).

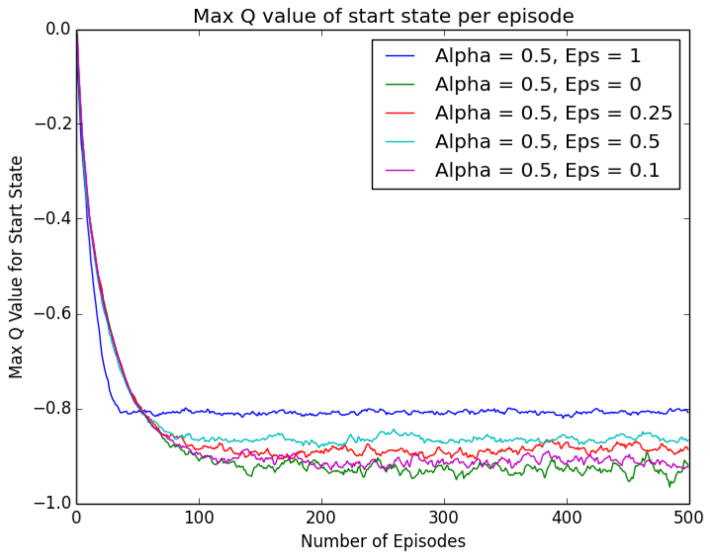


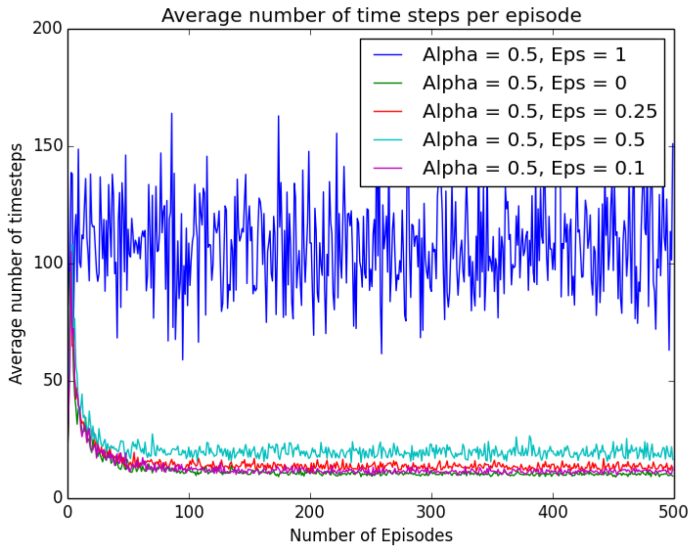


In the picture on left, we can see that all the Max Q values almost converge to the same value around -0.9 after 500 episodes, but for epsilon = 1, the number of time steps are far more than other values. This is because, when epsilon = 1, the epsilon greedy policy will choose random states which leads to more transitions to get to the terminal state. For epsilon value of 0, 0.25 and 0.1 we get far lesser time steps than other values.

Alpha = 0.5:

For alpha = 0.5, we plot (below) maximum value for Q (left) and average times steps per episode (right).





For alpha = 0.5, we see similar patterns for Max Q value. Here, we see that, the maximum value across different configurations is given by alpha = 0.5; eps = 1, which is similar to previous observation for alpha = 0.1. Moreover, eps = 1 takes more time steps to get to terminal state due to the randomization in e-greedy policy.

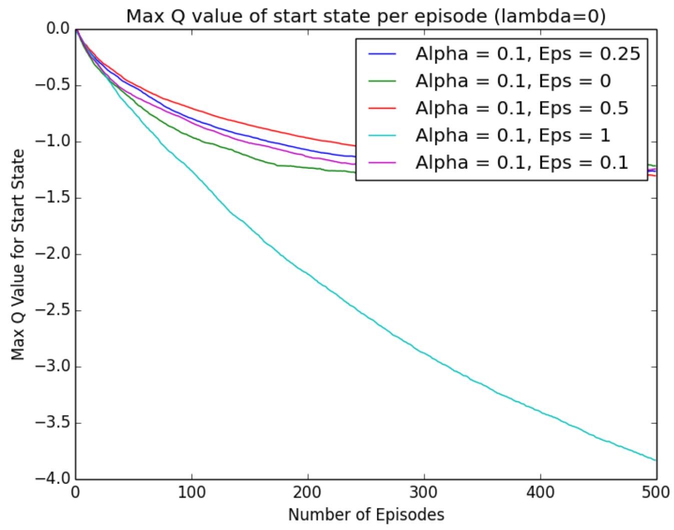
For Alpha = 0.5, we can see that epsilon values of 0.5 and 0.25 are good since it takes less time steps to get to terminal state and yet they are closer to the maximum Q value.

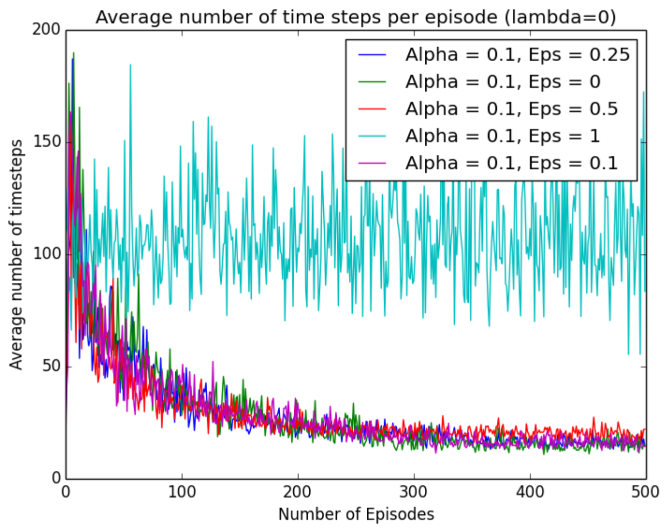
Sarsa Lambda

From the previous section, we can see that alpha 0.1 and 0.5 give a good model for sarsa lambda, since the values are not close to zero for Maximum Q values and neither are they too small (like in the case of alpha = 1). Moreover, for these two values the number of steps per episode are also reasonably small. Therefore, we show a few plots for alpha = 0.1, alpha=0.05 with different values of epsilon.

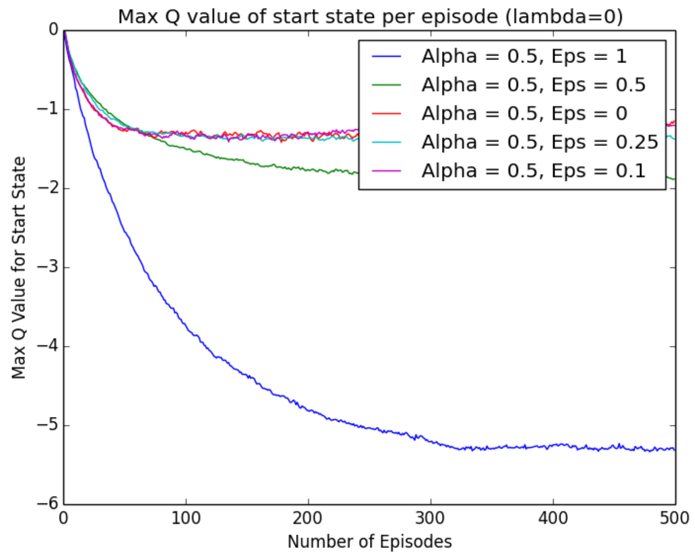
Alpha = 0.1:

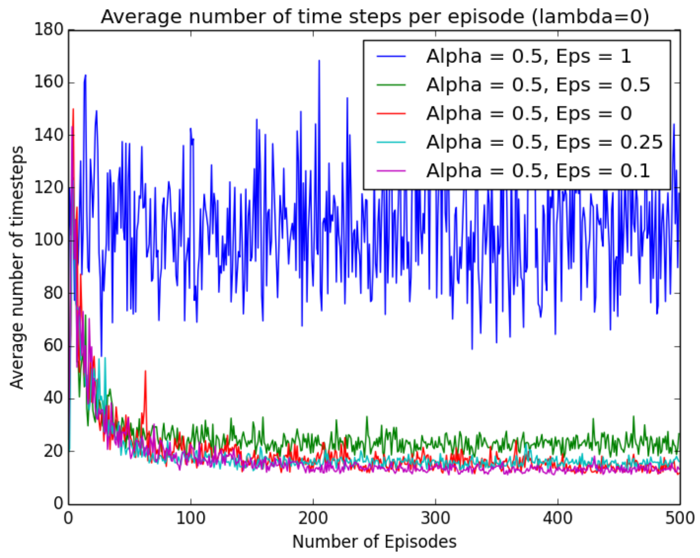
For alpha = 0.1, we plot (below) maximum value for Q (left) and average times steps per episode (right).





Alpha = 0.5:





For sarsa lambda (alpha = 0.1 and 0.5 both), we see that the smallest value in Max Q plot is given when epsilon = 1. Which is different than Q-Learning. For alpha = 0.1, we see that all epsilon values except for epsilon = 1, converge to the same Max Q value, but that is not true for alpha = 0.5.

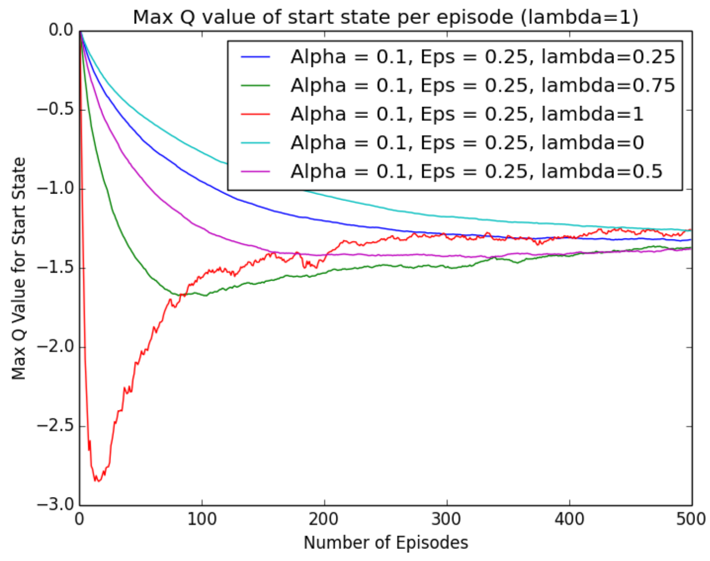
When epsilon = 1, for both alphas, the number of time steps is the highest, which is not surprising as discussed above. Here, also we see that 0.25 epsilon is a good value for the parameter, like Q-Learning because it goes to the converged value in lesser time steps.

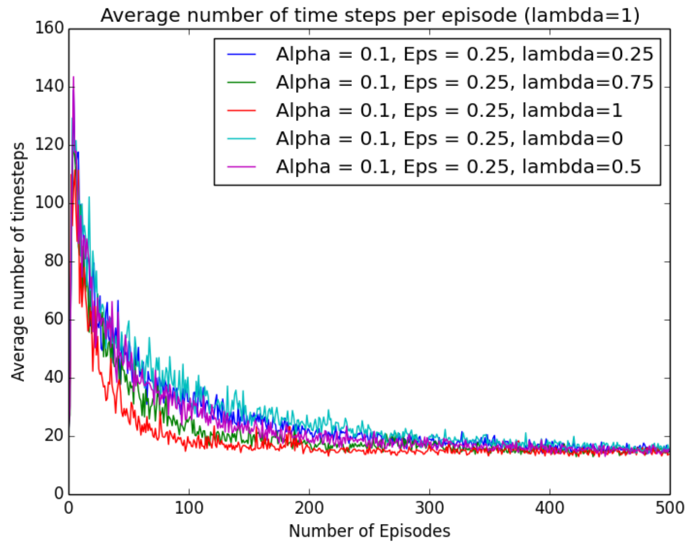
4. Effect of alpha, epsilon and lambda on Sarsa Lambda

As discussed in the previous section, we see that epsilon=0.25 along with alpha=0.1 makes up for a good model. Alpha = 0.5 also shares some of the characteristics, but the max Q values are too scattered for that configuration. Therefore, in this section, we discuss the effects of lambda on Sarsa Lambda when alpha = 0.1 and epsilon=0.25. We plot the graphs for Max Q values and average time steps, for this configuration below.

Interesting patterns emerge as we change the value of lambda. The Max Q value for start state changes dramatically when lambda = 1. It almost reaches the smallest Max Q value in the first few episodes and recovers back to get to the converging value. Similarly, for lambda = 0.75, the Max Q value is at its smallest in first few episodes and bounces back. This can be attributed to Sarsa Lambda's approach of updating all the affecting states, and since when lambda = 1, all the previous states are updated equally, we see fluctuations in the Q value.

On the other hand, when lambda = 0, the algorithm only updates the value for previous affecting state, therefore the values don't fluctuate much, which can be seen when lambda = 0.





5. Best Parameters for Q-Learning

As we've discussed in previous sections (section 3 and 4), the best parameters for Q-Learning are alpha = 0.1, epsilon = 0.25. Using these parameters, we get the optimal policy when we run Q-Learning on 10 x 10 grid for **700** episodes.

Here's the path chosen by the policy obtained using Q-Learning on 10 x 10 grid:

```
[['L' 'R' 'R' 'R' 'R' 'R' 'R' 'R' 'R' 'U']
 ['U' 'D' 'R' 'R' 'R' 'R' 'U' 'R' 'R' 'U']
 ['L' 'R' 'R' 'R' 'R' 'R' 'R' 'R' 'U' 'U']
 ['U' 'U' 'R' 'R' 'R' 'R' 'U' 'R' 'R' 'U']
 ['D' 'U' 'R' 'R' 'U' 'U' 'R' 'U' 'R' 'U']
 ['U' 'U' 'R' 'U' 'U' 'U' 'U' 'R' 'R' 'U']
 ['U' 'U' 'U' 'U' 'U' 'U' 'R' 'U' 'U' 'U']
 ['R' 'U' 'R' 'U' 'R' 'R' 'R' 'U' 'R' 'U']
 ['U' 'R' 'R' 'U' 'L' 'D' 'R' 'D' 'L' 'U']
 ['U' 'R' 'R' 'R' 'U' 'D' 'R' 'R' 'R' 'R']]
```

The values in the grid depict the direction to be taken from that state (cell).

6. Best Parameters for Sarsa Lambda

As we've discussed in previous sections (section 3 and 4), the best parameters for Q-Learning are alpha = 0.1, epsilon = 0.25 and lambda = 0.25. Using these parameters, we get the optimal policy when we run Q-Learning on 10 x 10 grid for **900** episodes.

Here's the path chosen by the policy obtained using Sarsa-Lambda on 10 x 10 grid:

```
[['R' 'R' 'R' 'R' 'R' 'R' 'R' 'R' 'R' 'U']
 ['U' 'L' 'R' 'R' 'R' 'R' 'R' 'U' 'U' 'U']
 ['U' 'U' 'D' 'U' 'L' 'U' 'U' 'R' 'R' 'U']
 ['U' 'R' 'U' 'L' 'R' 'U' 'R' 'U' 'R' 'U']
 ['U' 'L' 'U' 'D' 'U' 'D' 'U' 'R' 'U' 'U']
 ['L' 'U' 'R' 'L' 'D' 'R' 'R' 'R' 'U' 'U']
 ['L' 'D' 'L' 'D' 'R' 'U' 'R' 'R' 'R' 'U']
 ['U' 'L' 'R' 'U' 'L' 'R' 'U' 'R' 'R' 'U']
 ['U' 'D' 'U' 'R' 'R' 'R' 'R' 'R' 'U' 'U']
 ['R' 'R' 'R' 'R' 'R' 'R' 'R' 'R' 'R' 'U']]
```

7. How would one in general choose alpha, lambda and epsilon? How does they affect Q-Learning and Sarsa Lambda?

Alpha is the learning rate which affects the updates made to the Q values (i.e. state action values). If learning rate is too small, the updates will be very slow and more time steps will be required. On the other hand, if learning rate is too high, when the agent is close the solution, it might miss the solution and jump around.

Epsilon is the parameter that affects how the next action is picked in epsilon-greedy strategy. If the value of epsilon is too high, the next action picked in Q-Learning and Sarsa Lambda, may be exploratory and will not use the Q values that the algorithm has learned so far. If epsilon is too low, the algorithm will choose the next state based on Q values learned so far. Therefore, this parameter affects how much does the agent explore as compared to how much it exploits.

Lambda affects the Sarsa Lambda algorithm while updating the Q values for all states and actions. If lambda is close to 1, Sarsa lambda only credits the previous state and when it is close to 0 it gives more and more credit to all the previously visited states.

One way of choosing alpha, lambda and epsilon would be to compare the average reward obtained per episode using different combinations of these parameters. We can then choose the best model, based on the maximum rewards obtained.

In general, the parameters that help us maximize the reward per episode and decrease the number of steps to terminal state are good parameters.