CSE 4/546: Reinforcement Learning

Assignment 1 - Defining & Solving RL Environments

Checkpoint: February 20, Sun, 11:59pm

Part 1 [Total: 50 points] - Defining RL environments:

- For defining the RL environments which are for this case Deterministic and Stochastic environment that are based on Markov Decision process, I've used OpenAI Gym in Google colab.
- Both the environments are defined following the instructions in the problem statement. Both the environments are working with provided visualisation of the agent interacting with the environment.

Report Q/A for part 1:

1. Describe the deterministic and stochastic environments, which were defined (set of actions/states/rewards, main objective, etc).

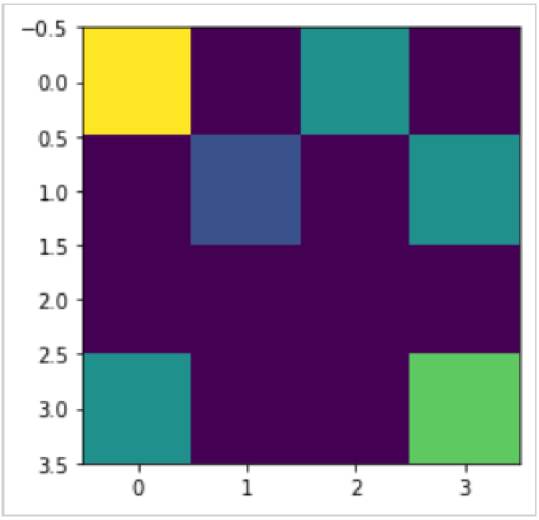
   Ans:
   - Set of actions: (Up, Down, right, left)
   - Set of states: There are 16 possible states, each state is a block in a 4x4 matrix blocks. There is the start state where the agent starts every time its reset. We have the end state 'Goal' where if the agent reaches the agent stops, this state is the main objective of the agent. There are 14 other states where we have random rewards for some states.
   - Set of rewards: (-2,0,+1,+5)
   - Main objective: This is the main objective of the agent to achieve and if the agent achieves this the agent stops, for our environment I've defined the main objective as 'Goal' state with a +5 reward.

   Deterministic environment: In this environment the transition probability is defined as P(s ', r|s, a) = {0, 1}, which means that if the agent is in a state and takes an action there is only one possible next state.

   Stochastic environment: In this environment the transition probability is defined as $\sum_{s/,r} \mathrm{P}(\mathrm{s}', \mathrm{r}|\mathrm{s}, \mathrm{a}) = 1$, which means that if the agent is in a state and takes an action there are more than one possible next state.

2. Provide visualizations of your environments.

   Ans:



   This is the visualization of the grid environment which is a 4x4 grid with 16 states in total. The agent always starts with the start position which is the top left block in the grid. The goal is at the bottom right position of the grid which is in light green colour block with a reward +5. There are 4 other states with possible rewards denoted with Teal and dark blue coloured blocks, the blocks with Teal colour have a reward +1 whereas the block with dark blue colour has a reward -2. All the other blocks with violet colour has a reward 0. The agent is denoted with yellow colour and every time the agent moves to another block or state the colour of the corresponding block changes to either yellow or a random colour.

3. How did you define the stochastic environment?

   Ans: For defining stochastic environment following steps were followed:
   - For action ‘Up’: Here when the agent calls for action ‘Up’ there’s a 95% chance that the agent takes the action ‘Up’ and a 5% chance that the agent takes the action ‘Down’.
   - For action ‘Down’: Here when the agent calls for action ‘Down’ there’s a 95% chance that the agent takes the action ‘Down’ and a 5% chance that the agent takes the action ‘Up’.
   - For action ‘Right’: Here when the agent calls for action ‘Right’ there’s a 95% chance that the agent takes the action ‘Right’ and a 5% chance that the agent takes the action ‘Left’.
   - For action ‘Left’: Here when the agent calls for action ‘Left’ there’s a 95% chance that the agent takes the action ‘Left’ and a 5% chance that the agent takes the action ‘Right’.

4. What is the difference between the deterministic and stochastic environments?

   Ans: The difference between deterministic and stochastic environments is that in deterministic environment if your agent selects an action it can only go to one possible next state but in the stochastic environment if the agent selects an action it can go to more than one possible next state.

5. Write a brief review explaining how you ensure the safety of your environments.

   Ans: We can take some of the few measures as follow:

   - We can ensure that the agent doesn’t go out of bounds of the environment by restricting the agents position within the boundaries.
   - While creating a stochastic environment we should take care of the transition probabilities as they play an important role.
   - We should specify the maximum time steps for which the agent will run and after reaching the maximum time steps or goal the agent should stop, or else the agent will just keep on moving without any goal or restrictions.
   - The environment should always have a start position and end goal position, so that every time the agent resets or reaches the goal, it should come back to the start position again.

CSE 4/546: Reinforcement Learning

Assignment 1 - Defining & Solving RL Environments

Due Date: March 6, Sun, 11:59pm

Part 2 [Total: 50 points] - Applying tabular methods:

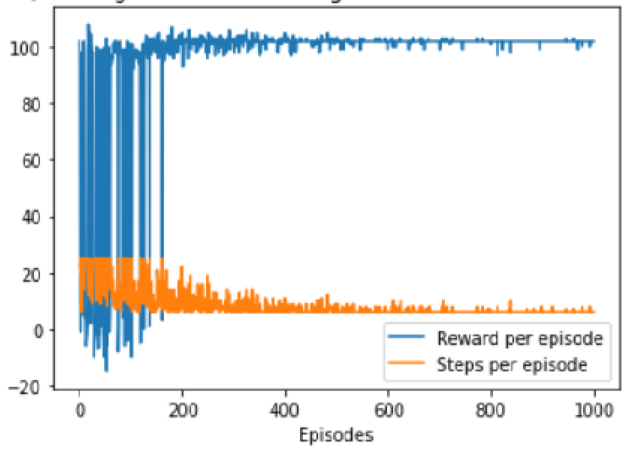
- Applying two tabular methods to solve both the deterministic and stochastic environments that were defined in Part 1.
- I've chosen Q learning and SARSA for my implementation. The following results for both the algorithms will be as follow:

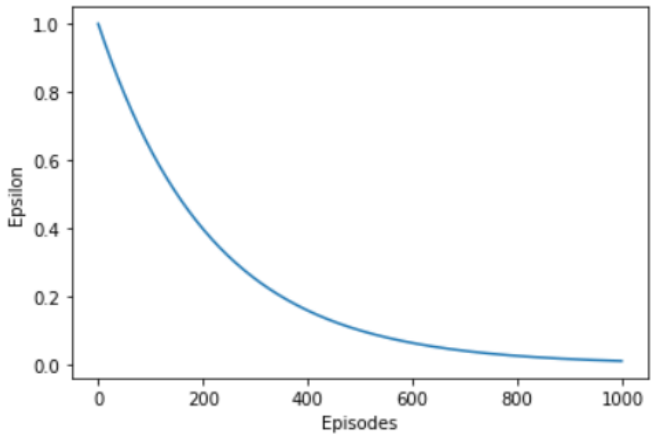
Results:

Q1:

- Applying Q-learning to solve the deterministic environment defined in Part 1.:

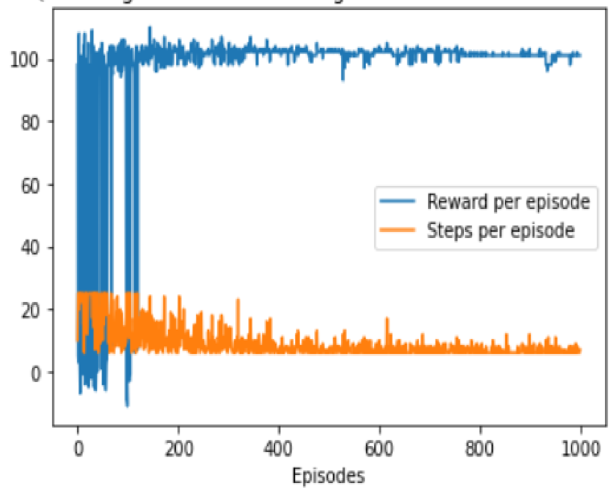
Q learning results after training for deterministic environment

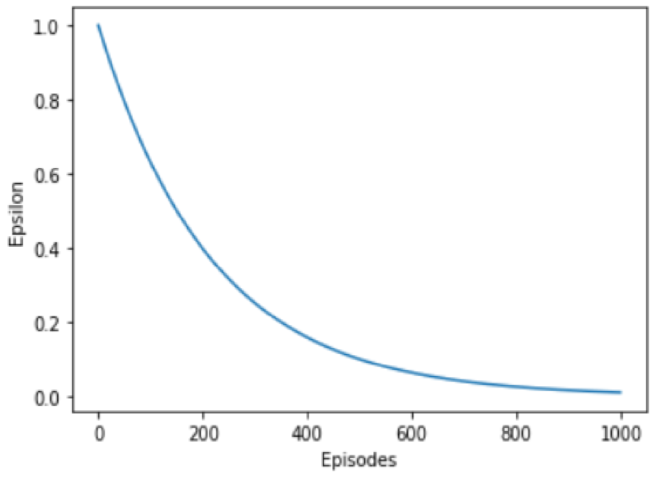




- Applying Q-learning to solve the stochastic environment defined in Part 1.:

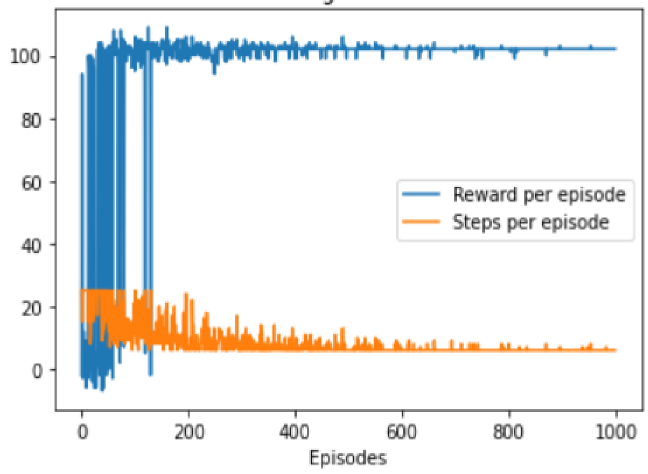
Q learning results after training for stochastic environment:

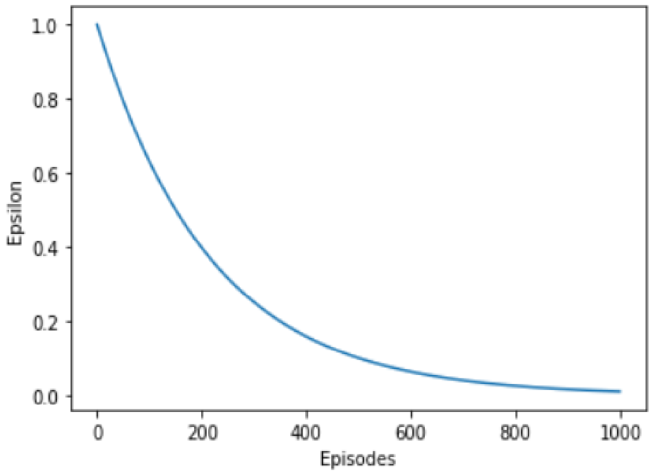




- Applying SARSA to solve the deterministic environment defined in Part 1:

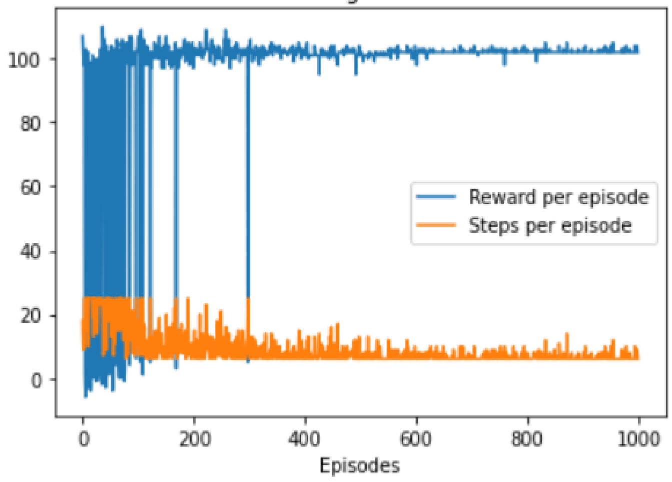
SARSA results after training for deterministic environment

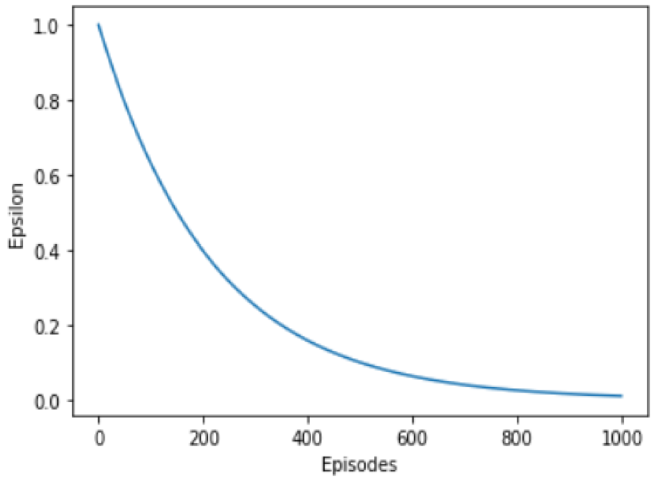




- Applying SARSA to solve the stochastic environment defined in Part 1.

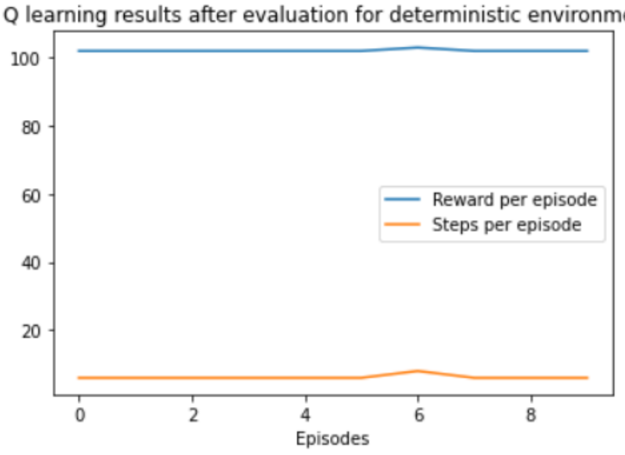
SARSA results after training for stochastic environment:



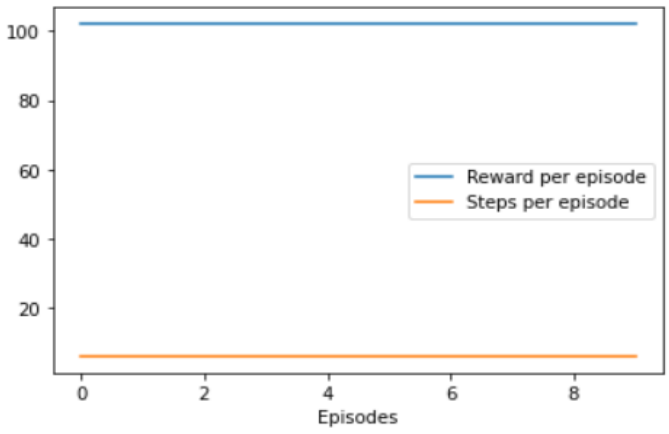


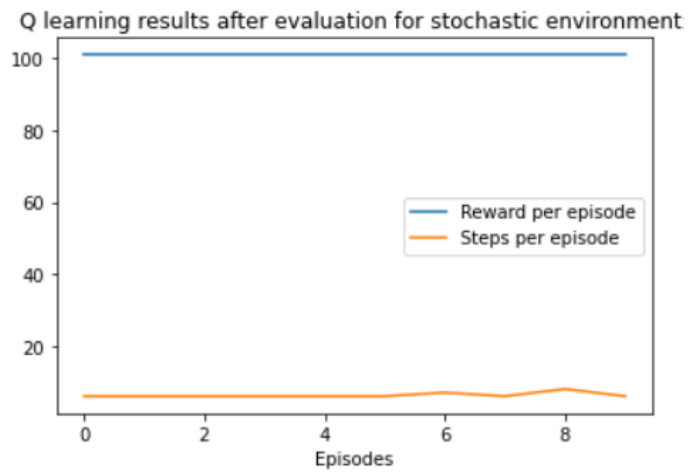
- The evaluation results. Running environment for 10 episodes, where the agent chooses only greedy actions from the learnt policy.

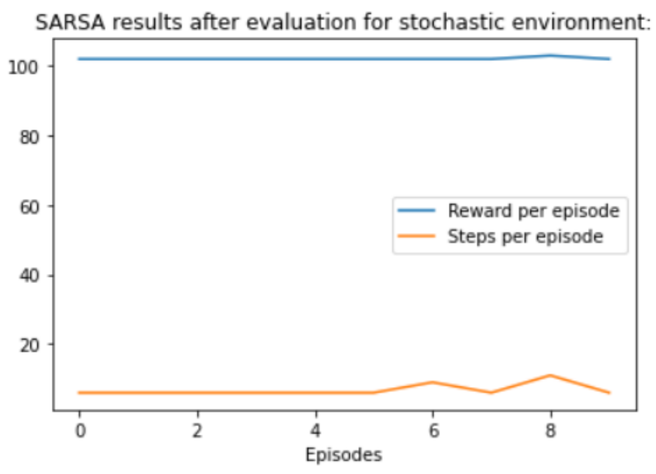
Q learning results after evaluation for deterministic environment



SARSA results after evaluation for deterministic environment:

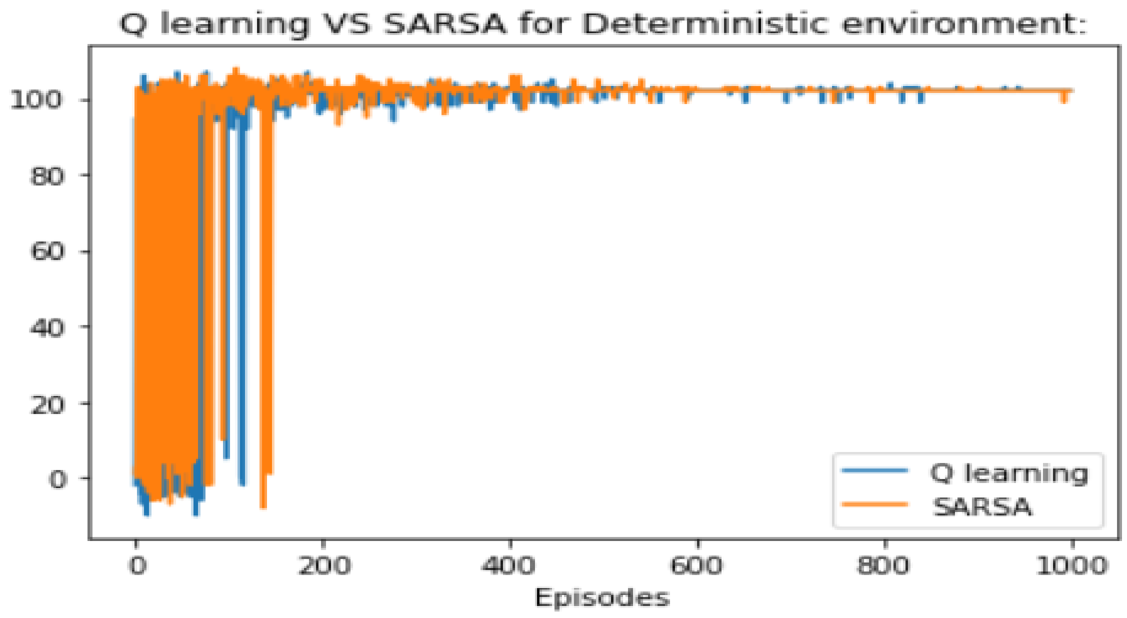






Q2:

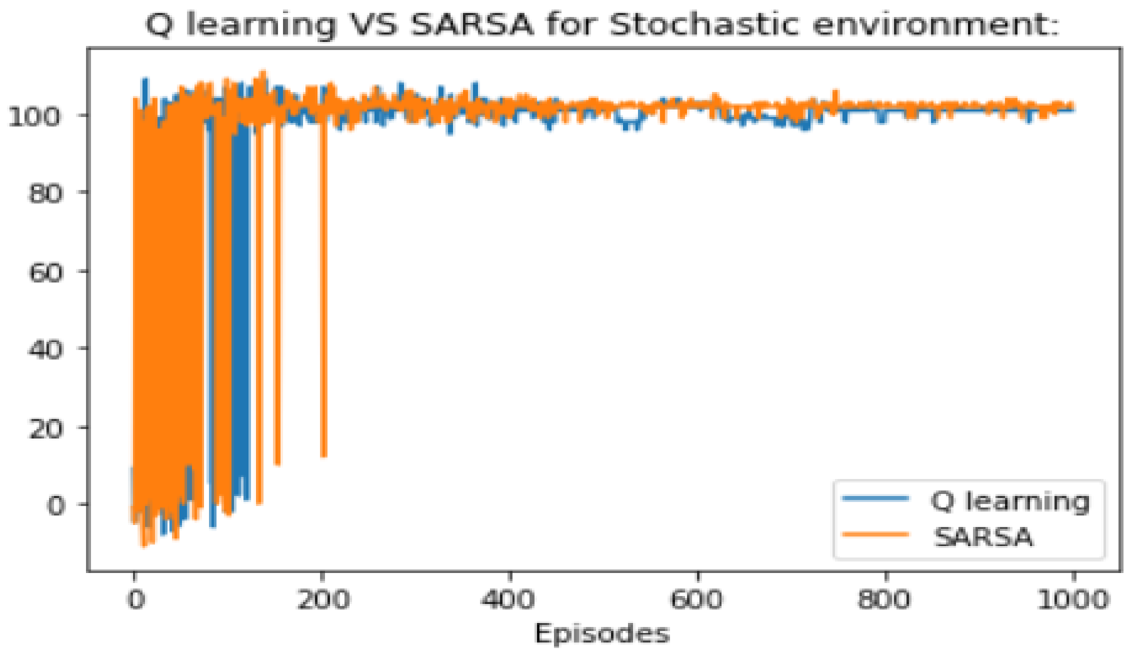
Comparing the performance of both algorithms on the same deterministic environment (by plotting rewards per episode):



- For both the algorithms as the training progresses the rewards per episode increases and tries to achieve the maximum reward by following the optimal policy.
- At the end of the training for both algorithms the reward per episode is almost the same, therefore we can say that both the algorithm are working well and are converging almost at the same point.

Q3:

Comparing the performance of both algorithms on the same stochastic environment (by plotting rewards per episode):



- The performance of both algorithm in the stochastic environment is similar to that of the deterministic, in here also the rewards per episode are increasing as the training progresses.
- Here also at the end of the training the rewards per episode at end of training is almost same for both the algorithm and are converging at almost the same point, both the algorithm seems to work well and are trying the maximise the reward per episode by following optimal policy.

Q4:

<u>Q Learning:</u>

- In the Q learning algorithm the agent tries to maximize the rewards as the training progresses and tries to obtain the optimal policy whereby achieving maximum reward it can achieve by taking minimum steps in an episode.

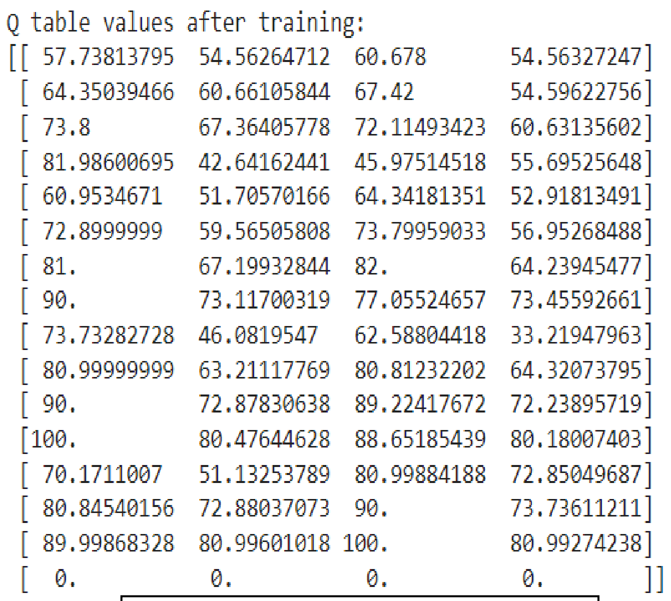
- The Q table values are updated by the following formula:

$$Q(S,A) \leftarrow Q(S,A) + \alpha\underbrace{[\overset{\text{Target}}{\boxed{\underset{\text{Immediate Reward}}{R} + \gamma \max_a Q(S',a)}} - \overset{\text{Prediction}}{\boxed{Q(S,A)}}]}_{\text{loss}}$$

The current state q value is updated by considering the reward for the action taken and by the q value of the next state (the max q value of the next state is selected). The action in a state is chosen by the following the epsilon greedy policy.

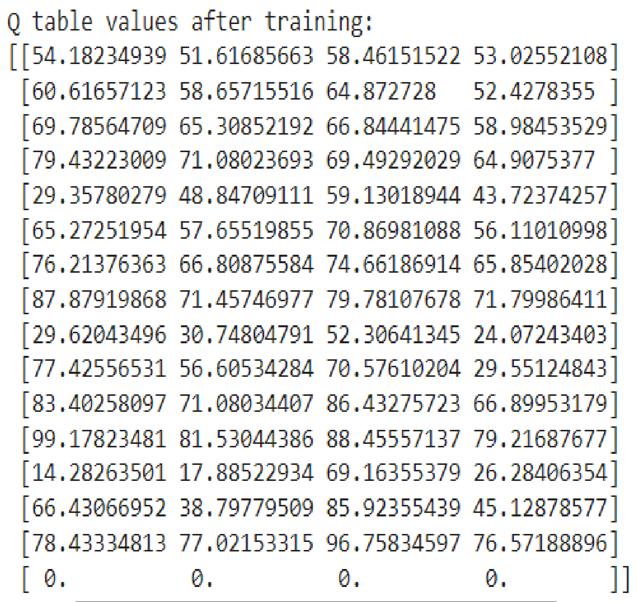
- A key feature of the algorithm is that the Q values for the terminal state is always zero for all the actions.

- Below are my Q values table for Q learning for both the environment:



```
Q table values after training:
[[ 57.73813795  54.56264712  60.678        54.56327247]
 [ 64.35039466  60.66105844  67.42         54.59622756]
 [ 73.8         67.36405778  72.11493423  60.63135602]
 [ 81.98600695  42.64162441  45.97514518  55.69525648]
 [ 60.9534671   51.70570166  64.34181351  52.91813491]
 [ 72.8999999   59.56505808  73.79959033  56.95268488]
 [ 81.          67.19932844  82.          64.23945477]
 [ 90.          73.11700319  77.05524657  73.45592661]
 [ 73.73282728  46.0819547   62.58804418  33.21947963]
 [ 80.99999999  63.21117769  80.81232202  64.32073795]
 [ 90.          72.87830638  89.22417672  72.23895719]
 [100.          80.47644628  88.65185439  80.18007403]
 [ 70.1711007   51.13253789  80.99884188  72.85049687]
 [ 80.84540156  72.88037073  90.          73.73611211]
 [ 89.99868328  80.99601018 100.          80.99274238]
 [  0.           0.           0.           0.        ]]
```

Deterministic environment



```
Q table values after training:
[[54.18234939 51.61685663 58.46151522 53.02552108]
 [60.61657123 58.65715516 64.872728   52.4278355 ]
 [69.78564709 65.30852192 66.84441475 58.98453529]
 [79.43223009 71.08023693 69.49292029 64.9075377 ]
 [29.35780279 48.84709111 59.13018944 43.72374257]
 [65.27251954 57.65519855 70.86981088 56.11010998]
 [76.21376363 66.80875584 74.66186914 65.85402028]
 [87.87919868 71.45746977 79.78107678 71.79986411]
 [29.62043496 30.74804791 52.30641345 24.07243403]
 [77.42556531 56.60534284 70.57610204 29.55124843]
 [83.40258097 71.08034407 86.43275723 66.89953179]
 [99.17823481 81.53044386 88.45557137 79.21687677]
 [14.28263501 17.88522934 69.16355379 26.28406354]
 [66.43066952 38.79779509 85.92355439 45.12878577]
 [78.43334813 77.02153315 96.75834597 76.57188896]
 [ 0.          0.          0.          0.        ]]
```

Stochastic environment

SARSA:

- The SARSA algorithm works in a similar way as Q learning wherein the agent tries to maximize the rewards as the training progresses and tries to obtain the optimal policy whereby achieving maximum reward it can achieve by taking minimum steps in an episode.

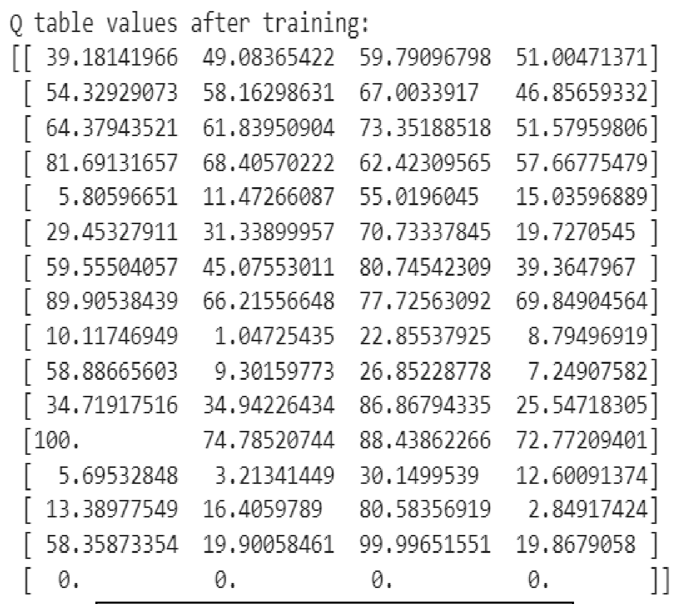
- The Q table values are updated by the following formula:

$$Q(S, A) \leftarrow Q(S, A) + \alpha\left[R + \gamma Q(S', A') - Q(S, A)\right]$$

The current state q value is updated by considering the reward for the action taken and by the q value of the next state (The next state q value is chosen by selecting the action the agent will take in the next state). The action in a state is chosen by the epsilon greedy policy

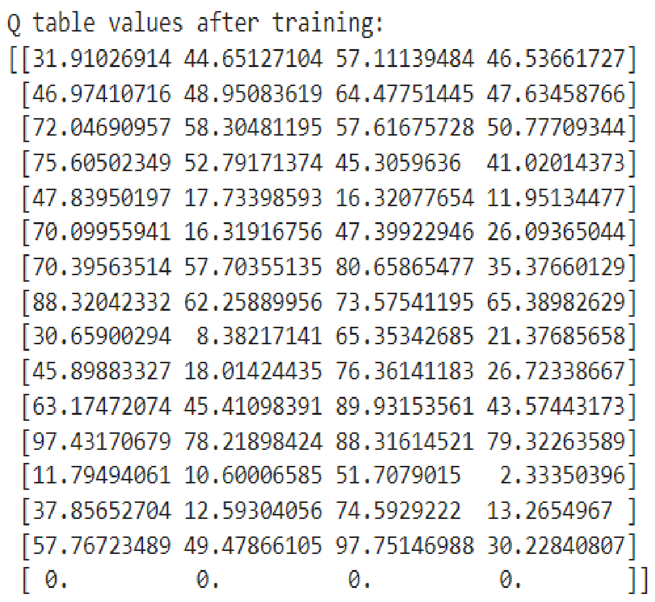
- Even in SARSA the q values for the terminal state is zero.

- Below are my Q values table for SARSA for both the environment:



```
Q table values after training:
[[ 39.18141966  49.08365422  59.79096798  51.00471371]
 [ 54.32929073  58.16298631  67.0033917   46.85659332]
 [ 64.37943521  61.83950904  73.35188518  51.57959806]
 [ 81.69131657  68.40570222  62.42309565  57.66775479]
 [  5.80596651  11.47266087  55.0196045   15.03596889]
 [ 29.45327911  31.33899957  70.73337845  19.7270545 ]
 [ 59.55504057  45.07553011  80.74542309  39.3647967 ]
 [ 89.90538439  66.21556648  77.72563092  69.84904564]
 [ 10.11746949   1.04725435  22.85537925   8.79496919]
 [ 58.88665603   9.30159773  26.85228778   7.24907582]
 [ 34.71917516  34.94226434  86.86794335  25.54718305]
 [100.          74.78520744  88.43862266  72.77209401]
 [  5.69532848   3.21341449  30.1499539   12.60091374]
 [ 13.38977549  16.4059789   80.58356919   2.84917424]
 [ 58.35873354  19.90058461  99.99651551  19.8679058 ]
 [  0.           0.           0.           0.        ]]
```

Deterministic environment



```
Q table values after training:
[[31.91026914 44.65127104 57.11139484 46.53661727]
 [46.97410716 48.95083619 64.47751445 47.63458766]
 [72.04690957 58.30481195 57.61675728 50.77709344]
 [75.60502349 52.79171374 45.3059636  41.02014373]
 [47.83950197 17.73398593 16.32077654 11.95134477]
 [70.09955941 16.31916756 47.39922946 26.09365044]
 [70.39563514 57.70355135 80.65865477 35.37660129]
 [88.32042332 62.25889956 73.57541195 65.38982629]
 [30.65900294  8.38217141 65.35342685 21.37685658]
 [45.89883327 18.01424435 76.36141183 26.72338667]
 [63.17472074 45.41098391 89.93153561 43.57443173]
 [97.43170679 78.21898424 88.31614521 79.32263589]
 [11.79494061 10.60006585 51.7079015   2.33350396]
 [37.85652704 12.59304056 74.5929222  13.2654967 ]
 [57.76723489 49.47866105 97.75146988 30.22840807]
 [ 0.          0.          0.          0.        ]]
```

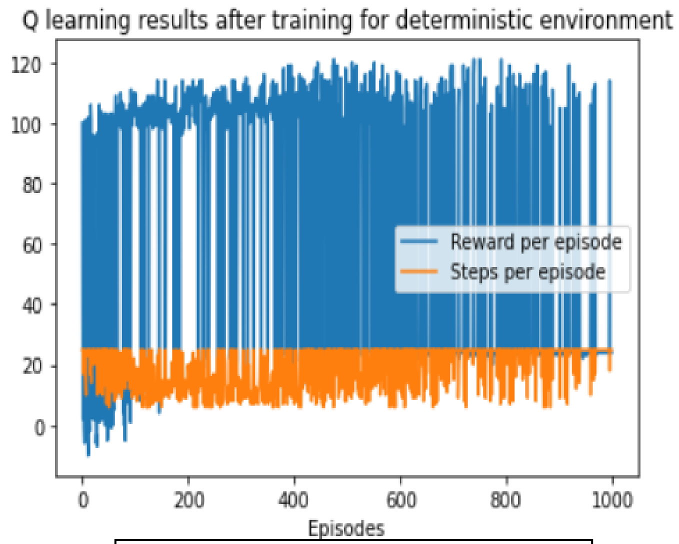
Stochastic environment

Extra Points:

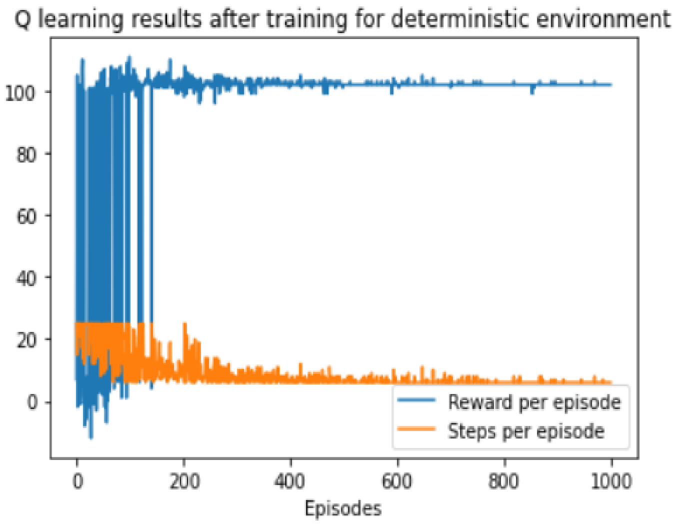
Hyperparameter Tuning:

Discount factor (γ):

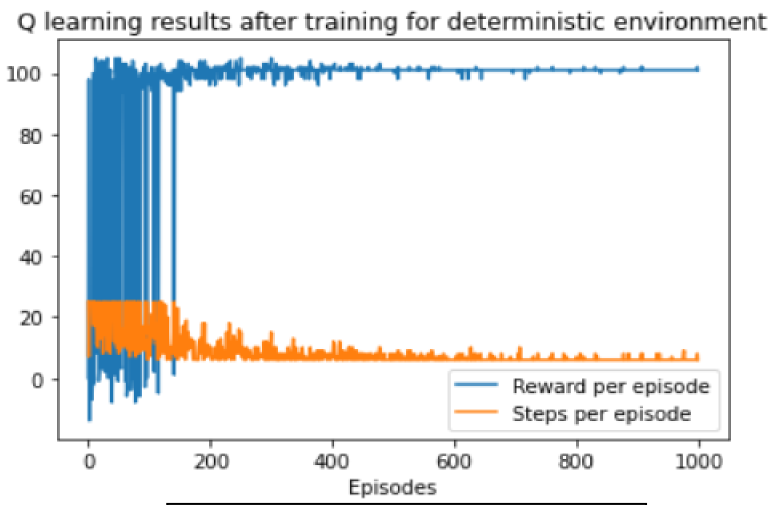
- For discount factor I took three different values and ran it on for Q learning on deterministic environment, below are the three different results:



Gamma=0.1



Gamma=0.25



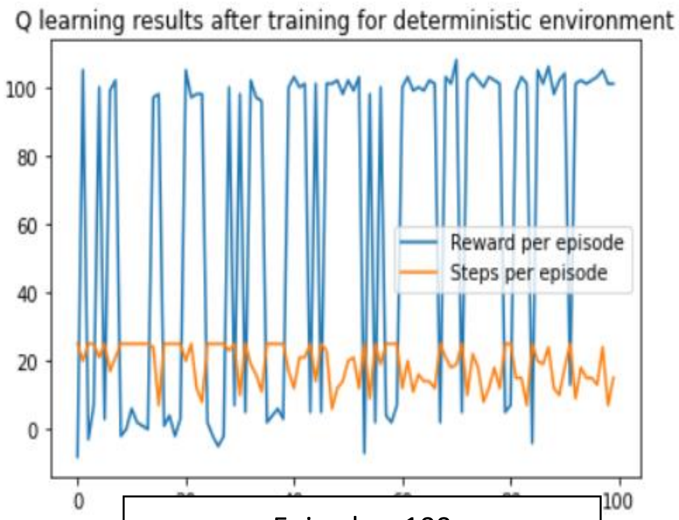
Gamma=0.9

- As depicted from the above three figures, we can see as we increase the gamma values the agent tries to maximize its reward as keeping the gamma value high motivates the agent to maximize the reward as the future discount aren't discounted that much.

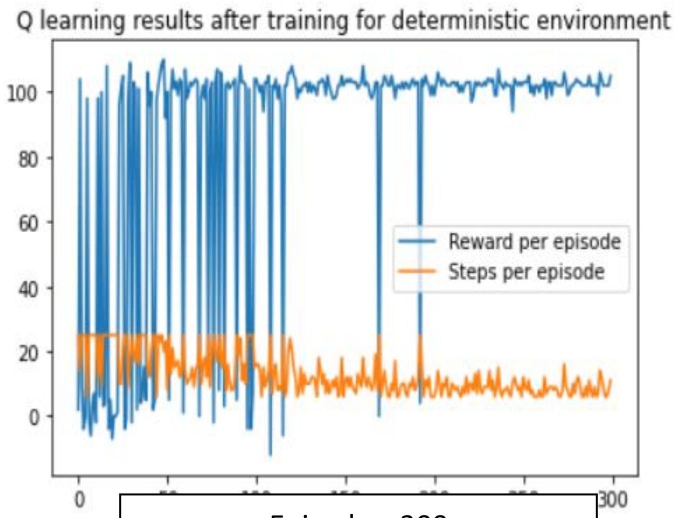
- The most efficient Gamma value will be to keep it as high as possible around 0.9 to 0.99 as Gamma<1.

Number of episodes:

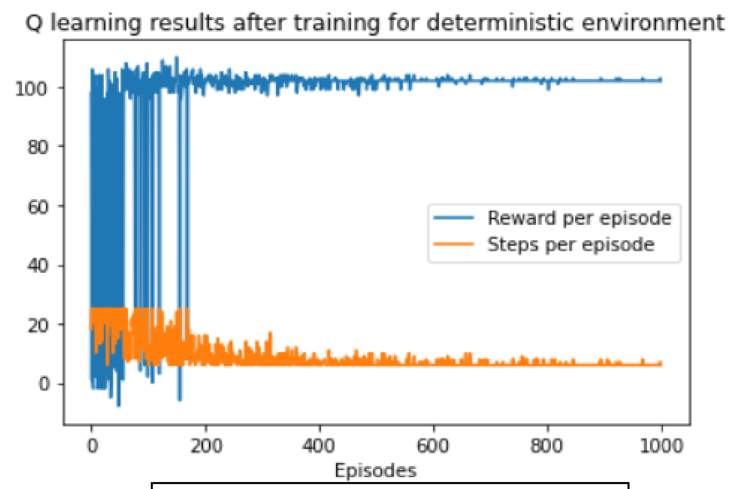
- For number of episodes I took three different values and ran it on for Q learning on deterministic environment, below are the three different results:



Episodes=100



Episodes=300



Episodes=1000

- As depicted in the figures as we increase the no of episodes the agent converges to the maximise the reward per episode; this is because if we keep the no of episodes low the agent has less experience and isn't trained to the full extent. So to have the agent trained well for the algorithm we should keep the no of episodes to a high value.

- The most efficient value for no of episodes for training will be to keep it at a high value so the agent can be trained well and maximise its reward. From the above three values no of episode=1000 is the best.