ROBOT GRID WORLD

Main Objective

The environment used is a grid world in which the Robot must explore and reach the bag of gold while collecting intermediate gold rewards. While exploring for the gold in the world the robot must avoid the monster and avoid falling into the pit.

States

The environment is a grid of 5x5 matrix. All the states can be represented using X and Y co-ordinates

S = { [4,0], [4,1], [4,2], [4,3], [4,4],

[3,0], [3,1], [3,2], [3,3], [3,4],

[2,0], [2,1], [2,2], [2,3], [2,4],

[1,0], [1,1], [1,2], [1,3], [1,4],

[0,0], [0,1], [0,2], [0,3], [0,4],

}

Actions

There are four possible actions that the robot can take.

A = {Up, Down, Left, Right}

Rewards

There are 4 positive rewards that the robot can receive. 3 are intermediate rewards (+2, +2, +5) and 1 is the reward on reaching the goal (+10) i.e. collecting the bag of coins. If the robot falls in the pit (-5) or gets eaten by the monster (-10), then the robot receives a negative reward. Additionally, for each action step taken by the robot it receives a negative reward (-0.5) and if the robot tries to go outside the grid then it receives a negative reward (-0.1)

R = {-10, -5, -0.5, -0.1, +2, +5, +10}

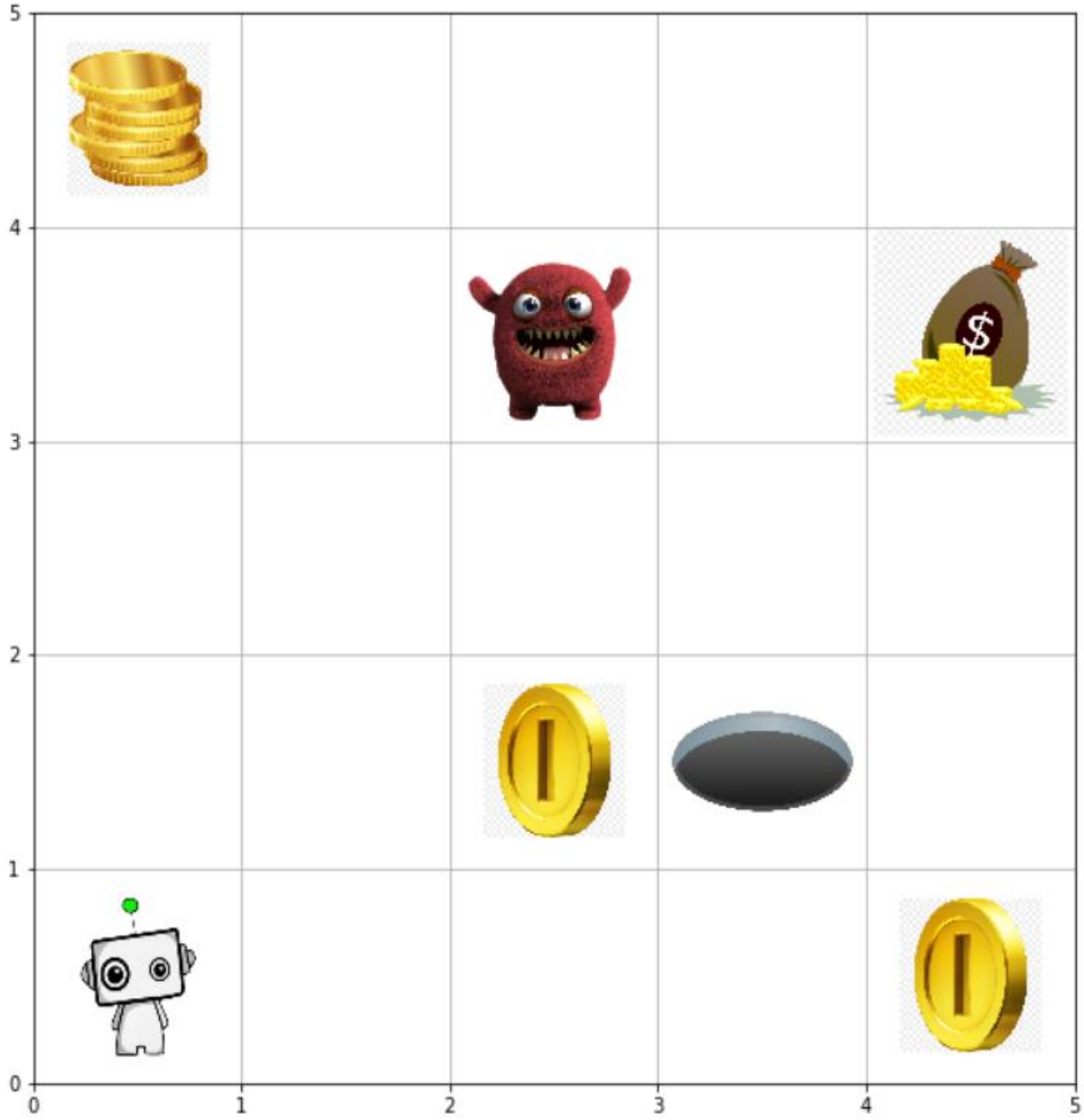


Fig 1: Visual Representation of the environment

- Agent starts at s = [0,0]
- Final goal position (bags of coins +10) is at s = [4,3]
- Intermediate Rewards 1 and 3 (+2) are at s = {[2,1], [4,0]}
- Intermediate Reward 2 (+5) is at s = [0,4]
- Pit (-5) is present at state s = [3,1]
- Monster (-10) is present at state s = [2,3]

Types of Environments

Deterministic Environment: Deterministic environment is the environment in which there is no uncertainty. The environment is fixed and set of same actions executed will always have same results.

Stochastic Environment: Stochastic environment is the environment in which there is uncertainty. The environment changes or the actions performed by the agent in the environment changes. In this type of environment set of same actions executed will not always have same results.

The Robot Grid World Environment can be either Deterministic or Stochastic.

Deterministic Environment:

In deterministic environment for each action taken by the robot the environment will perform the same action i.e. the states transition probability distribution can be defined as $P(s', r \mid s, a) = \{0,1\}$

Example: If the robot decides to go Left then the probability of the robot going Left is 1

Stochastic Environment:

In stochastic environment for each action taken by the robot the environment may decide whether to perform the same action or some other action. This is defined by the probability associated with the action in the environment. So, based on the probability the environment decides which action is to be taken. The probabilities are defines in such a way that the $\sum_{s',r} P(s', r \mid s, a) = 1$
In the code numpy.random.choice is used to achieve this. The probability of the action happening is 0.8 while the probability of opposite action happening is 0.08 and the probability of one of other two actions happening is 0.06 each.

Example: If the robot decides to go Left then the probability of agent going left is 0.8 and the probability of agent going to Right is 0.08 and probability of going to Up is 0.06 and probability of going Down is 0.06

In the given code, to run the agent in the deterministic environment or stochastic environment, a string variable type is passed while creating the object of the environment. It can take two values i.e. 'deterministic' or 'stochastic'. The value passed in this variable while creating the object of the environment class determines what kind of environment it'll be.

Safety in AI

Since the environment is a gird world of 5x5 matrix the robot cannot go out of the world. To ensure the safety of the robot a check is placed by using clip function where if the robot is on the edge of the environment and tries to go out of bounds the position of the robot will be the same and additionally a negative reward is associated with such actions so that the robot doesn't try to go out of the environment.

Along with that there is a maximum timestep defined in the environment. This ensures that the robot doesn't take infinite number of steps to reach the goal or robot doesn't get stuck in a loop.

The action and the state space are defined properly which make it restricted thereby ensuring that the robot doesn't take any undefined action or reach an undefined state.

Changes done to environment for part 2:

Change of the reward value of the goal state

The Final Goal position reward was changed from +10 to +25. This is because since the initially set final reward was very low, the agent used to oscillate between two positions and then ran out of timesteps and didn't reach goal state. Even with high discount factor the final reward received on final goal position was so less that the path towards the goal was not optimal and instead it used to oscillate between two positions after collecting intermediate goal. Hence to make the agent learn, the value of final reward was increased so that the agent reaches the final goal position. Increasing the weight of the final reward made agent go towards final goal.

Change in the Stochasticity of the environment

Initially the environment stochasticity was set to 0.8 i.e. if robot decides to go left then the probability if environment doing the action left was 0.8 and environment doing some different action was 0.2. To have controlled stochasticity in the environment as suggested by the prof. Alina, the stochasticity of the environment was changed. The new probability of environment doing the action suggested by the agent is set to 0.95. E.g. if the agent selects action left then the probability of environment taking action left is 0.95, the probability of agent going right is 0.03 and the probability of agent going up or down is 0.01 each.

Part 2: Applying tabular methods to solve the Robot Grid World problem

To solve the Robot grid world problem two tabular methods were used. Tabular methods are the methods in which the State Value function [V(s)] or Action Value function [Q(s, a)] are represented using tabular data and are used for computation of optimal policy for the agent to gain maximum reward in the environment.

There are many tabular methods such as:

1. Dynamic Programming
2. Monte Carlo
3. Temporal Difference Learning
    a. TD(0)
    b. SARSA
    c. Q-Learning
    d. Double Q-Learning
    e. N-Step Bootstrapping
    f. TD(λ)

So to solve the above robot grid world problem Temporal Difference Learning method is used.

The two tabular methods used are 1. Q-Learning 2. SARSA

1. Q-Learning

Q-Learning uses Action Value functions for computation of optimal policy. Initially the all the values are set to 0 and then as the agent takes an action a in state s then the Q value for that state is calculated using the below formula

$$\mathbf{Q(s, a) = Q(s, a) + \alpha[R + \Upsilon \max_a Q(s', a) - Q(s, a)]}$$

R is the immediate reward received by the agent

$R + \Upsilon \max_a Q(s', a)$ is the target

$R + \Upsilon \max_a Q(s', a) - Q(s, a)$ is the loss

Here when an action a is taken and the agent reaches state s' from state s then the Q value of the maximum action in that state s' is selected for the computation. Hence it is independent of the policy. The next action to be taken in s' is selected using epsilon greedy action and this is continued till the agent reaches a terminal state

Key Features:

- Maximum Q value of the next state is used for computation
- Independent of the policy

2. SARSA

SARSA also uses Action Value Functions for computation of the optimal policy. All the values are set to 0 and then as the agent takes an action a in state s then the Q value of the state is calculated using the below formula

$Q(s, a) = Q(s, a) + \alpha[R + \Upsilon\, Q(s', a') - Q(s, a)]$

Here when an action a is taken and the agent reaches state s' from state s then the action a' is selected using the epsilon greedy policy and the action given by the policy is used for computation of Q value. The Q value of action a' in s' is used for computation of state value and since the action is generated using the policy, SARSA is policy dependent. Then in the next iteration the action a' is executed in the state s' and so on till the agent reaches the terminal state.

Key Features:

- Random action a' generated by policy in s' is used for computation of Q value
- It is policy dependent

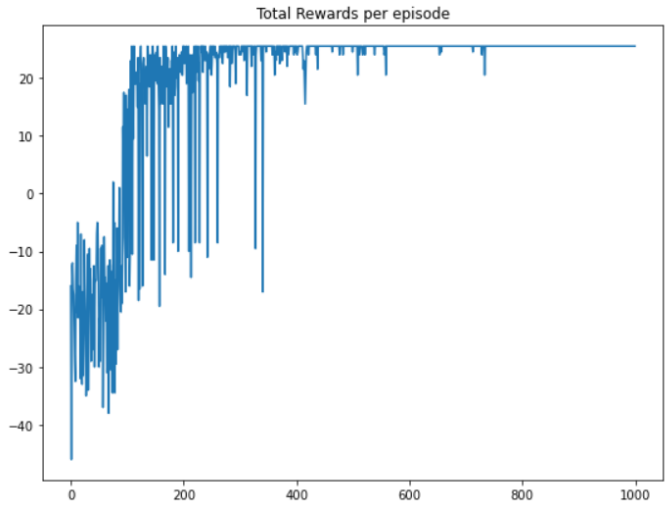
Q-Learning on Robot Grid World

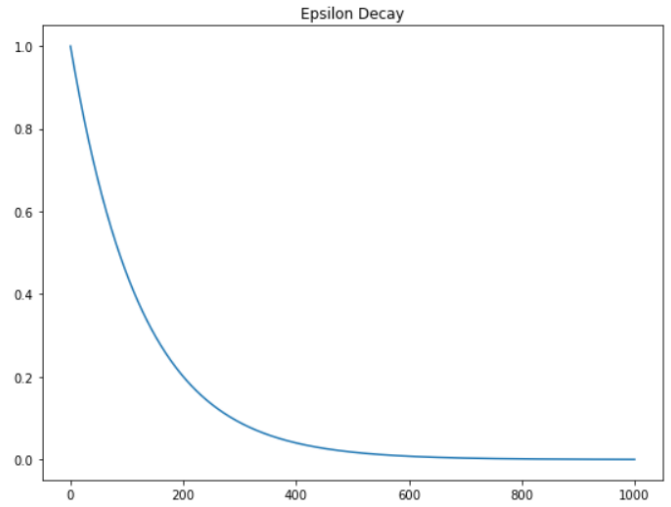
1. Deterministic Environment

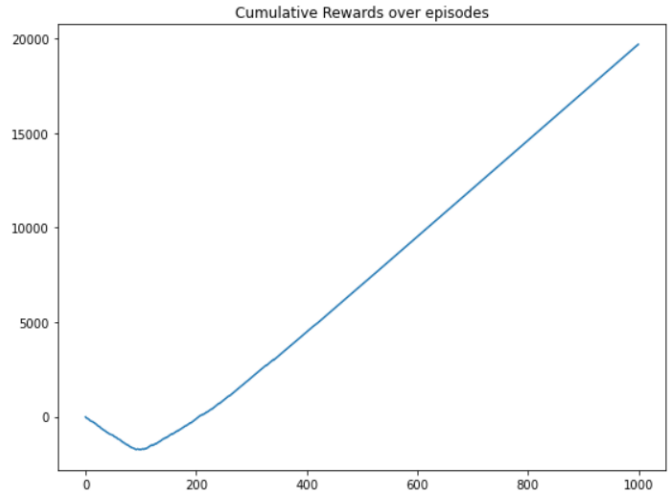
Following are the results of Q-Learning Algorithm used to find the optimal policy in Robot Grid World problem

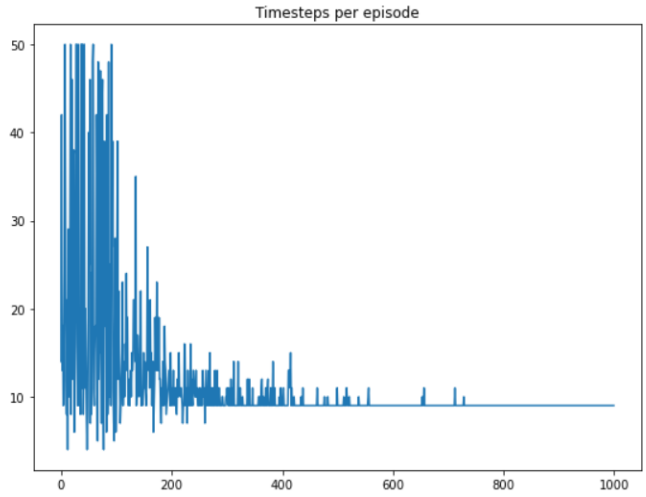
Parameters used for training:

alpha = 0.9, gamma = 0.9, epsilon = 1, epsilon decay = 0.008, training episodes = 1000





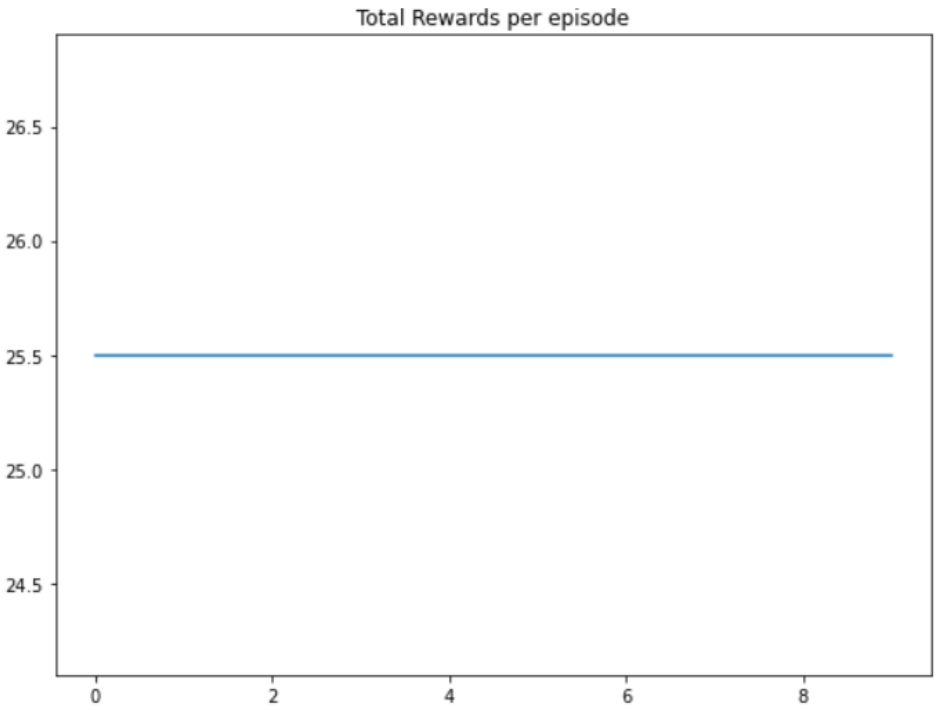


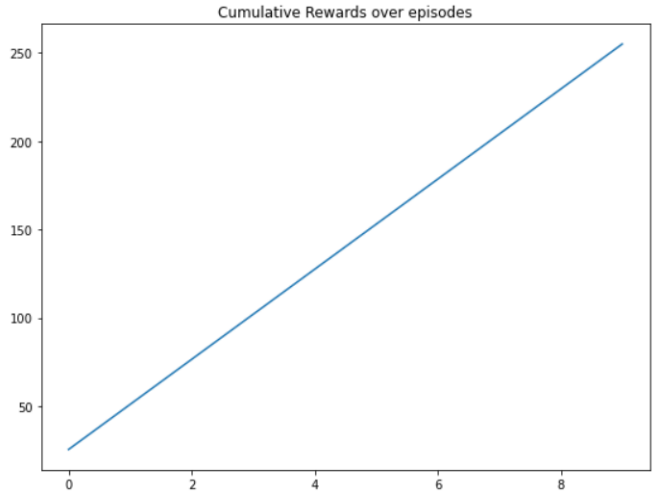


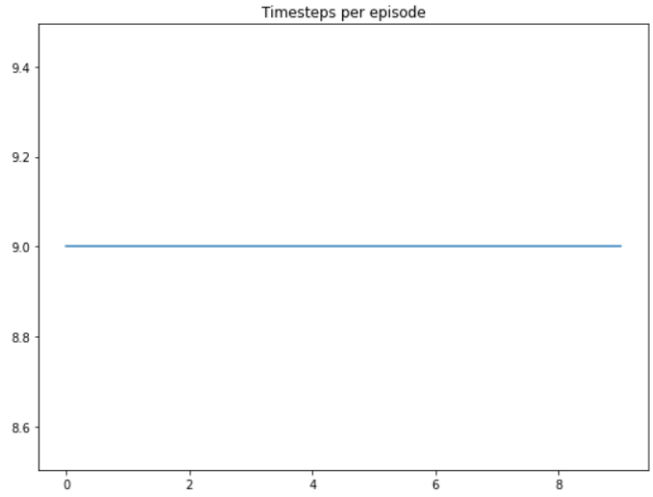
As seen above, initially the robot was exploring the grid world as the epsilon value was high. As the epsilon value kept on decreasing over the episodes the agent started to take greedy actions which led to building of the optimal policy. After around 400 episodes as seen from the graph the total rewards are almost a straight line as the agent had learned the optimal path and was following that to achieve the goal state. Also, we can see that the number of timesteps per episode decreased as the agent started taking optimal policy and the cumulative rewards increased linearly as the agent started learning.

After the training was completed the agent was tested in the environment for 10 episodes to see whether the agent had learned.

Testing Results:







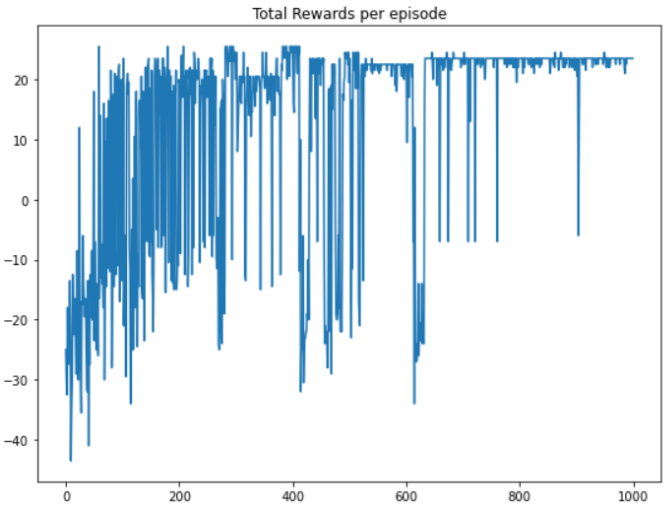
As seen above, since the environment is deterministic the agent has learned and is following the same path again and again. The reward obtained in all the episodes is same and cumulative reward is linear. The number of timesteps taken in each episode is also constant. Hence the agent has learned the optimal path.

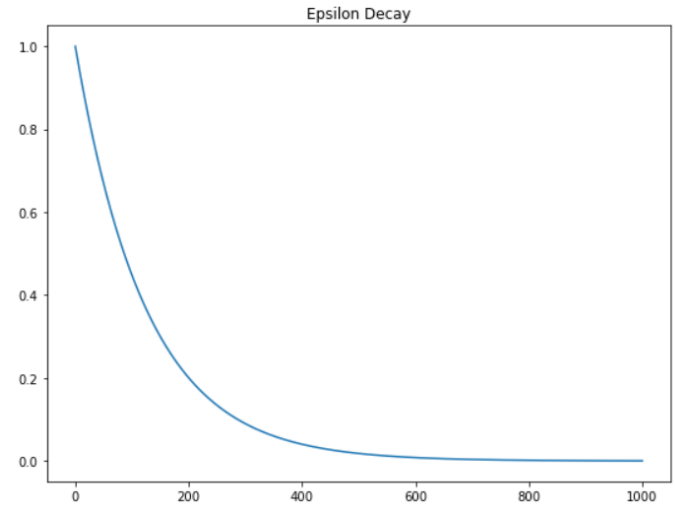
2. Stochastic Environment

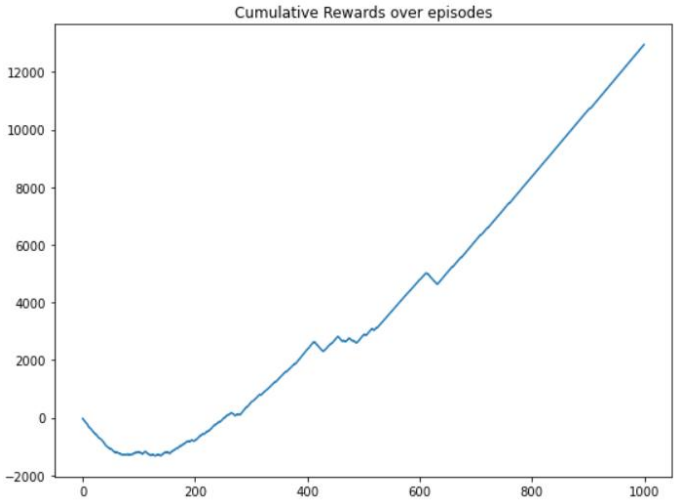
Following are the results of Q-Learning Algorithm used to find the optimal policy in Robot Grid World problem

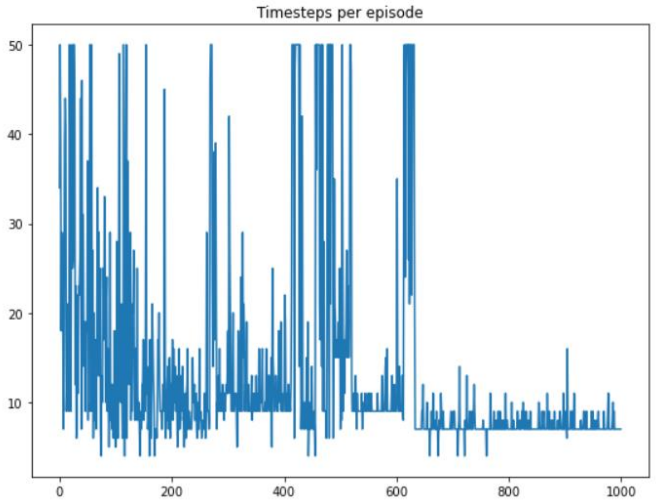
Parameters used for training:

alpha = 0.9, gamma = 0.9, epsilon = 1, epsilon decay = 0.008, training episodes = 1000





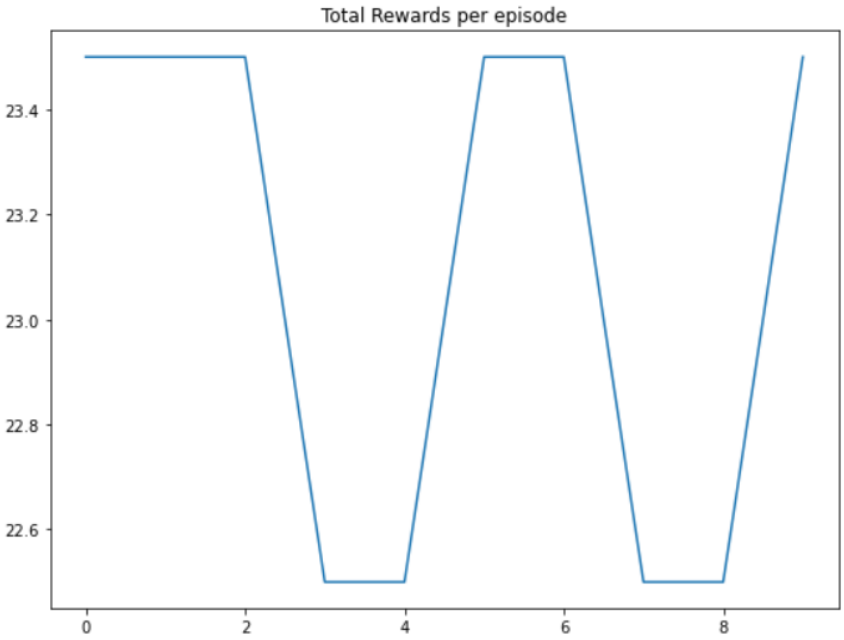


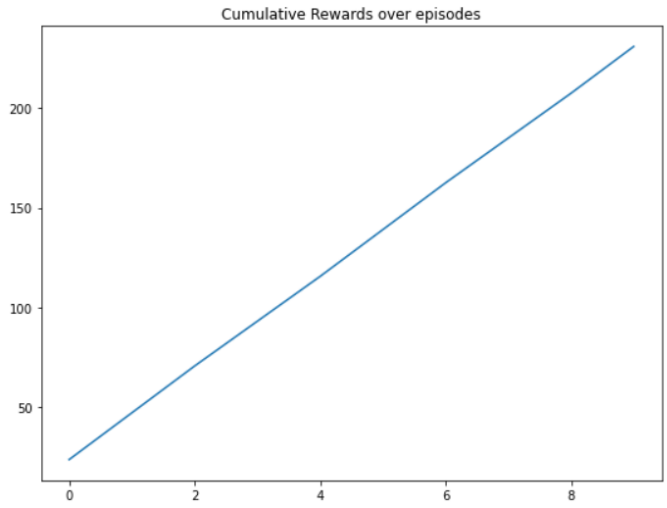


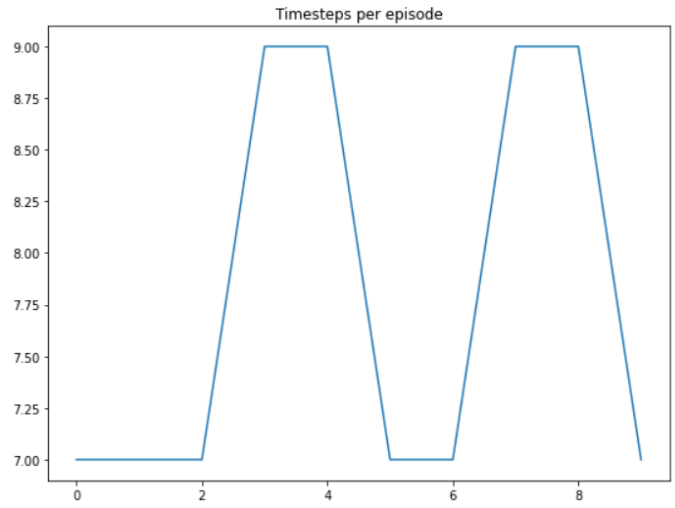
As seen above, initially the robot was exploring the grid world as the epsilon value was high. As the epsilon value kept on decreasing over the episodes the agent started to take greedy actions. But since the environment was stochastic, even though the agent was taking the best action the environment used to take some random action. This can be seen in the total rewards graph. As to reach the goal position the agent has to go from close to pit and monster sometimes even after selecting the best action the agent used to fall in the pit or got eaten by the monster because the environment took some random action. The graph of cumulative rewards is not linear and the timestep graph varies because of this stochasticity.

After the training was completed the agent was tested in the environment for 10 episodes to see whether the agent had learned, and the performance of the agent was optimal.

Testing Results:







Since the environment is stochastic the total rewards per episode is not constant and the number of timesteps to reach the goal state also varies. But since the agent always ended up in the goal state during the testing phase without falling into the pit or getting eaten by the monster the graph of cumulative rewards is linear. Chances are that when the agent is tested again in the environment because of the stochastic actions taken by the environment robot might fall into the pit or get eaten by monster and receive a negative reward. During that the cumulative reward graph won't be linear also the total reward graph might have a negative value in it.

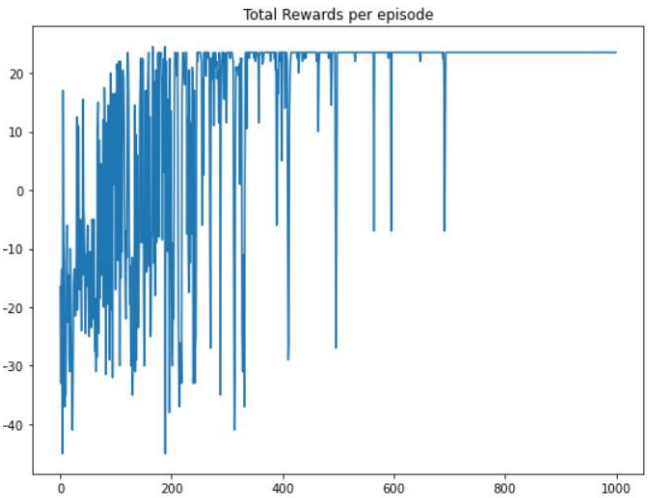
SARSA on Robot Grid World

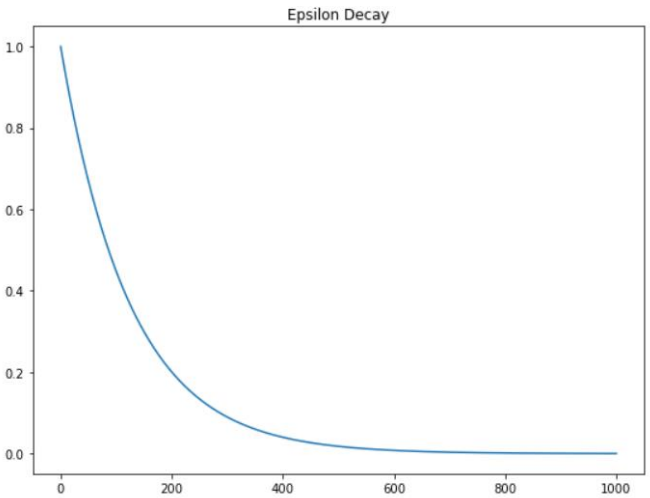
1. Deterministic Environment

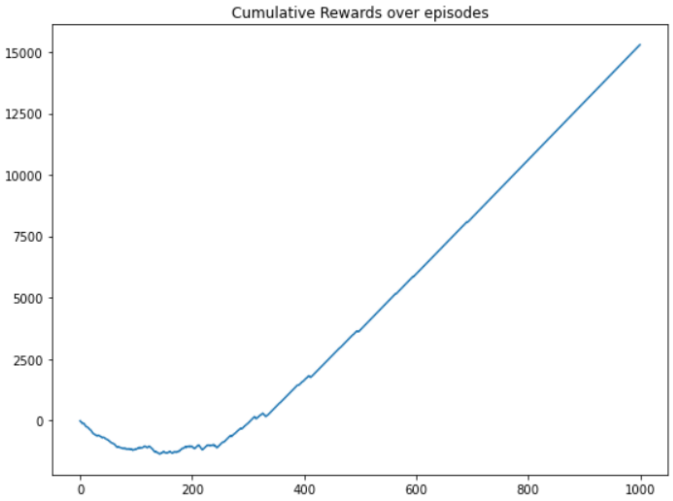
Following are the results of SARSA Algorithm used to find the optimal policy in Robot Grid World problem

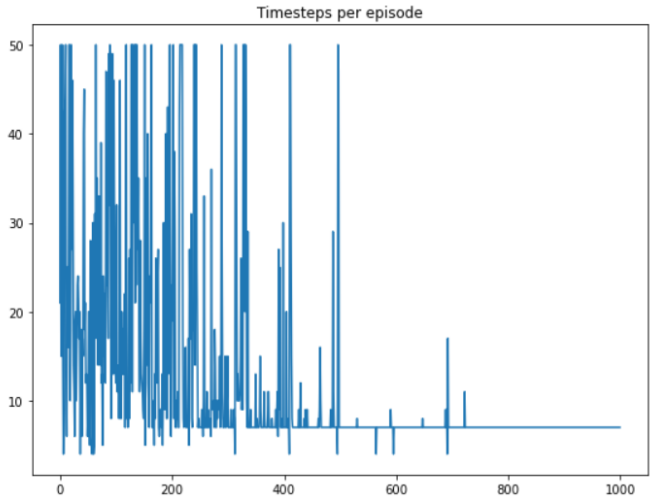
Parameters used for training:

alpha = 0.9, gamma = 0.9, epsilon = 1, epsilon decay = 0.008, training episodes = 1000





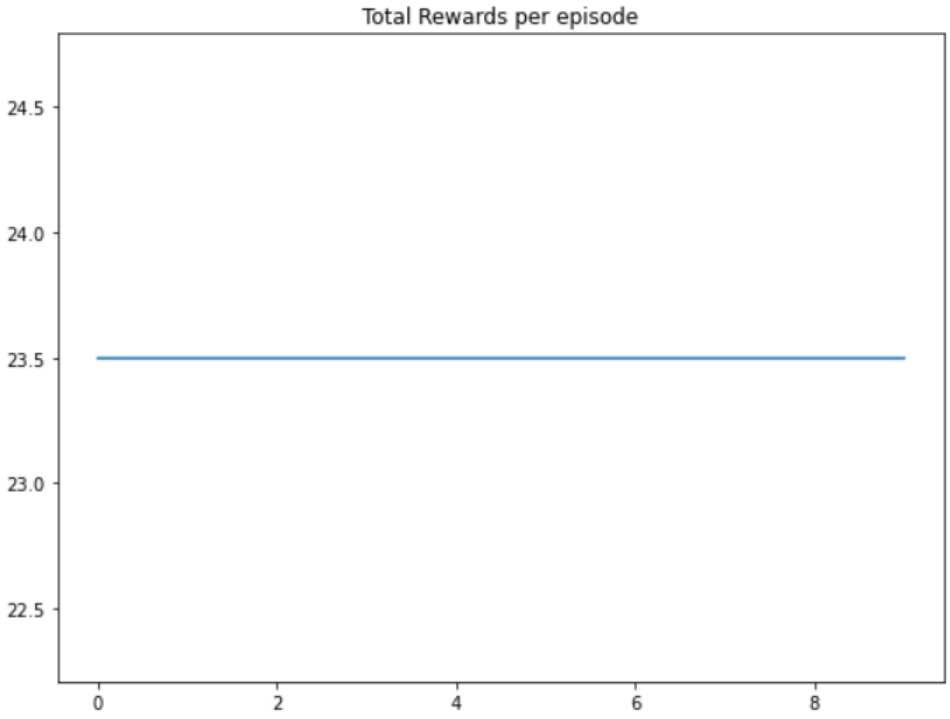


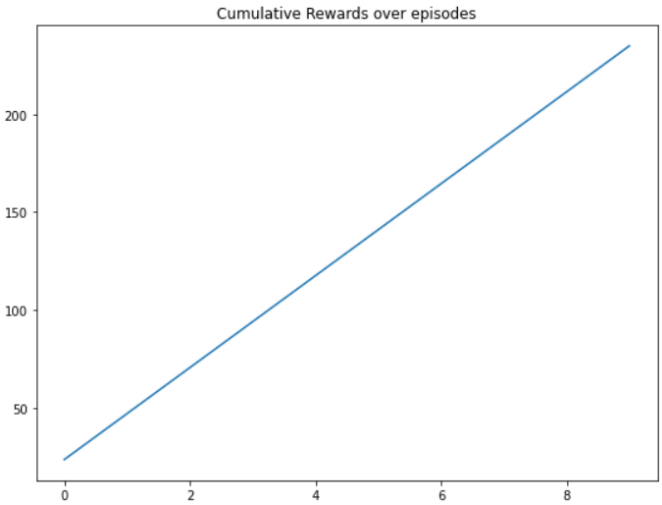


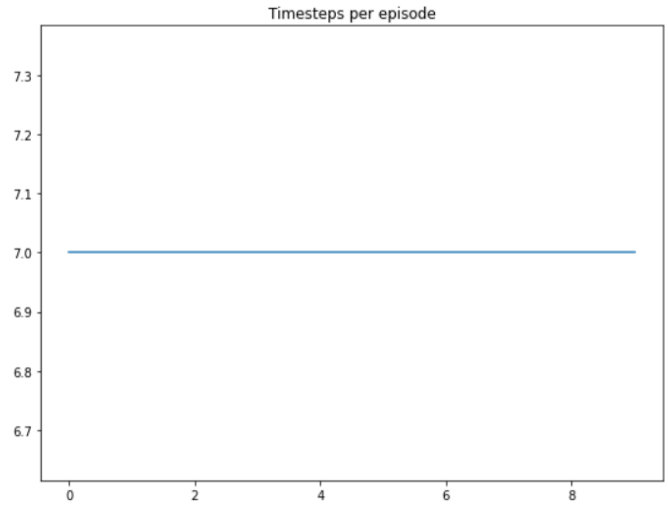
As seen above, initially the robot was exploring the grid world as the epsilon value was high. As the epsilon value kept on decreasing over the episodes the agent started to take greedy actions which led to building of the optimal policy. After almost 350 episodes as seen from the graph the total rewards are almost a straight line as the agent had learned the optimal path and was following that to achieve the goal state. Also, we can see that the number of timesteps per episode decreased as the agent started taking optimal policy and the cumulative rewards increased linearly as the agent started learning.

After the training was completed the agent was tested in the environment for 10 episodes to see whether the agent had learned, and the performance of the agent was optimal.

Testing Results:







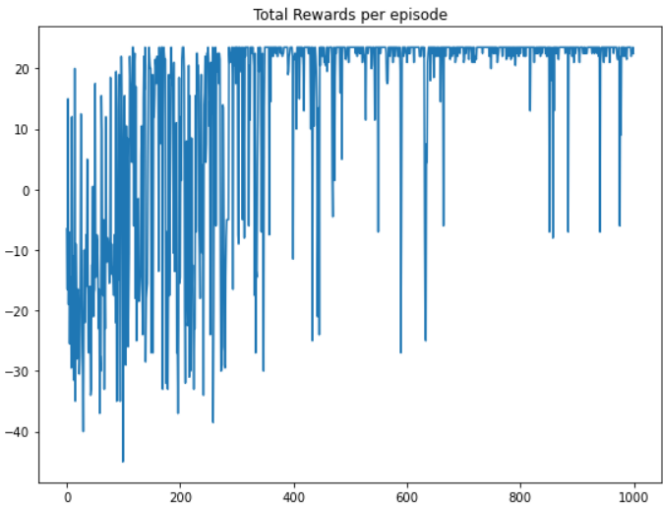
As seen above, since the environment is deterministic the agent has learned and is following the same path again and again. The reward obtained in all the episodes is same and cumulative reward is linear. The number of timesteps taken in each episode is also constant. Hence the agent has learned the optimal path.

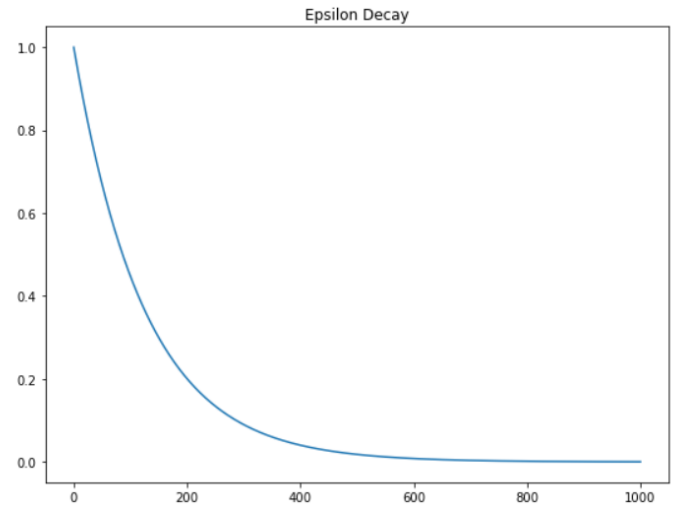
2. Stochastic Environment

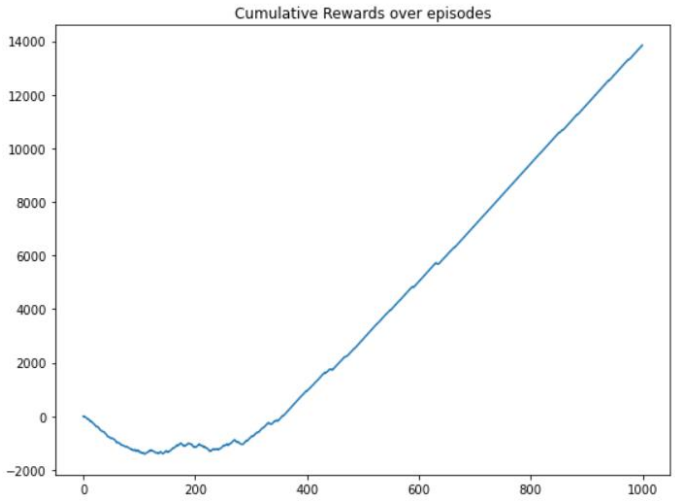
Following are the results of SARSA Algorithm used to find the optimal policy in Robot Grid World problem

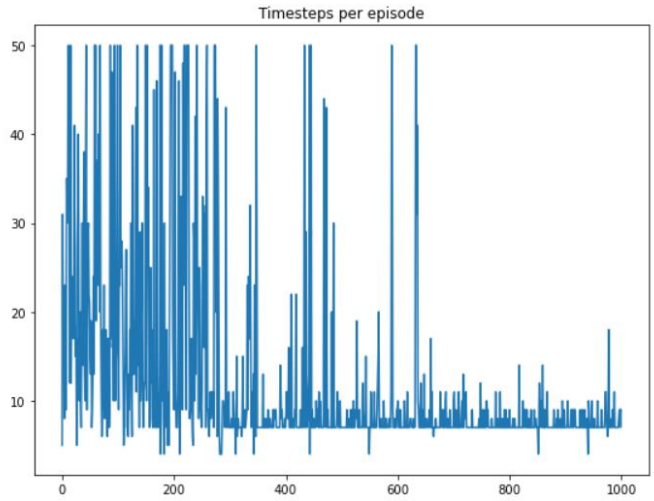
Parameters used for training:

alpha = 0.9, gamma = 0.9, epsilon = 1, epsilon decay = 0.008, training episodes = 1000





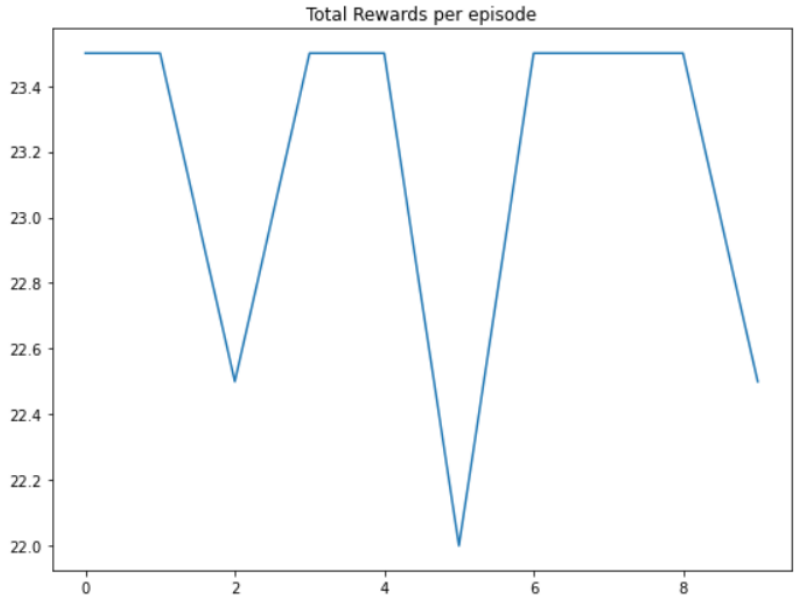


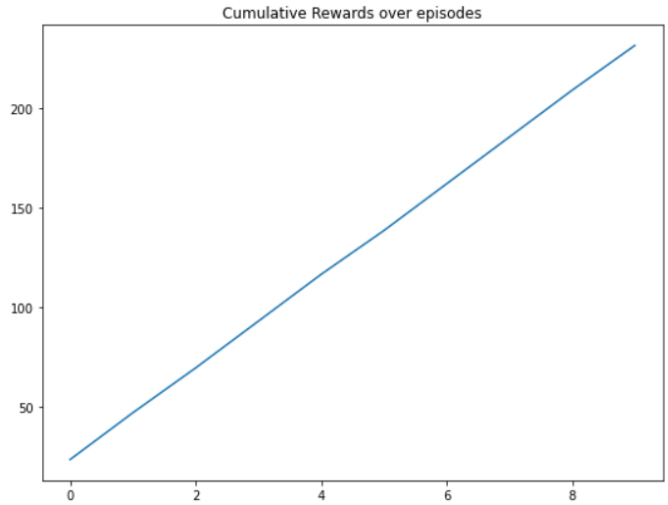


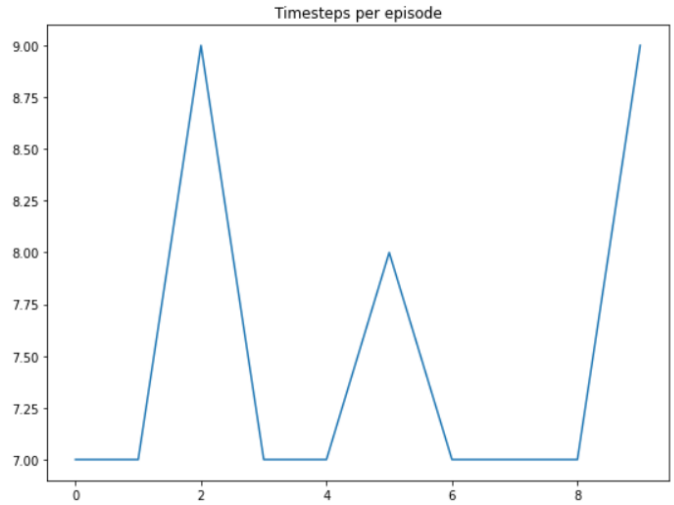
As seen above, initially the robot was exploring the grid world as the epsilon value was high. As the epsilon value kept on decreasing over the episodes the agent started to take greedy actions. But since the environment was stochastic, even though the agent was taking the best action the environment used to take some random action. This can be seen in the total rewards graph. After episode 450 even though the robot was taking best action the total rewards received is sometimes negative. As to reach the goal position the agent has to go from close to pit and monster sometimes even after selecting the best action the agent used to fall in the pit or got eaten by the monster because the environment took some random action. The graph of cumulative rewards is not linear and the timestep graph varies because of this stochasticity.

After the training was completed the agent was tested in the environment for 10 episodes to see whether the agent had learned, and the performance of the agent was optimal.

Testing Results:



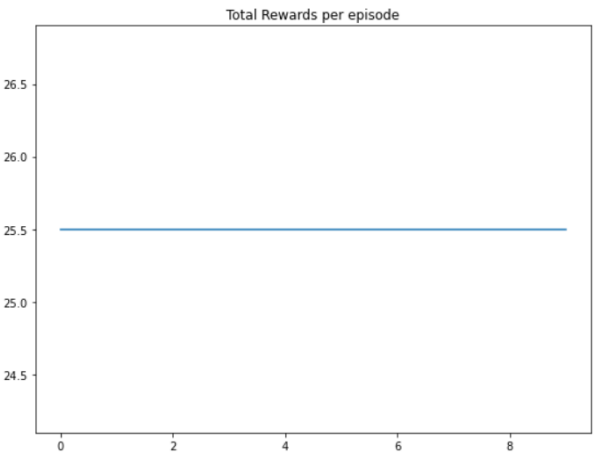


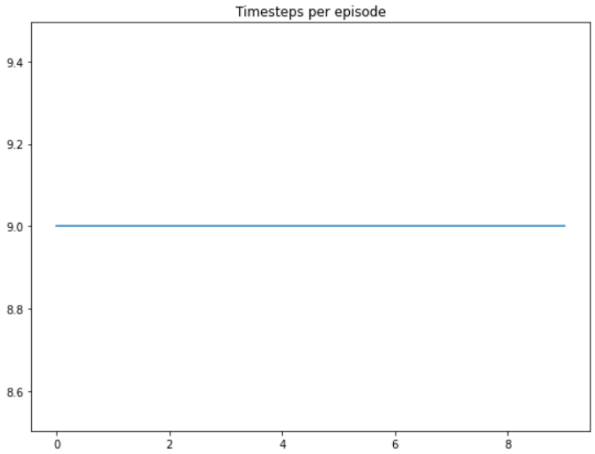


Since the environment is stochastic the total rewards per episode is not constant and the number of timesteps to reach the goal state also varies. But since the agent always ended up in the goal state during the testing phase without falling into the pit or getting eaten by the monster the graph of cumulative rewards is linear. Chances are that when the agent is tested again in the environment because of the stochastic actions taken by the environment robot might fall into the pit or get eaten by monster and receive a negative reward. During that the cumulative reward graph wont be linear also the total reward graph might have a negative value in it.

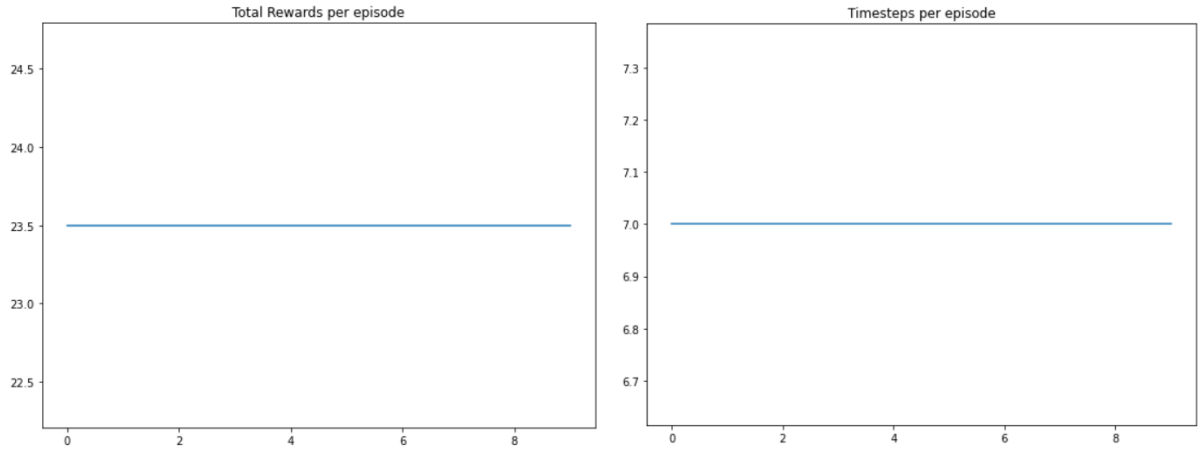
Comparison of methods on Deterministic Environment

Q-Learning





SARSA



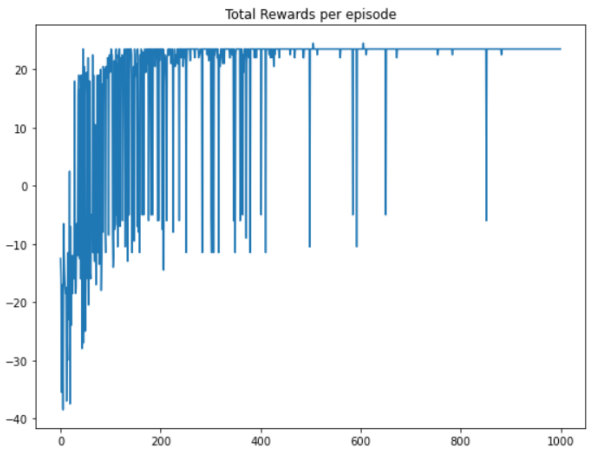
As we can see the performance of the agent in the environment using both the methods is very similar. The agent performed same actions in each episode of the method. But the only difference is that in Q-Learning the agent took more timesteps to reach the goal and the reward obtained was higher than the reward obtained in SARSA. This is because in Q-Learning the agent went to position (0, 4) first to and got the reward of +5 and then reached the goal at position (4, 3) taking 9 timesteps in total. Whereas in the SARSA agent went sideways and reached position (4,0) to collect +2 reward and then went to goal position (4, 3) in 7 timesteps.

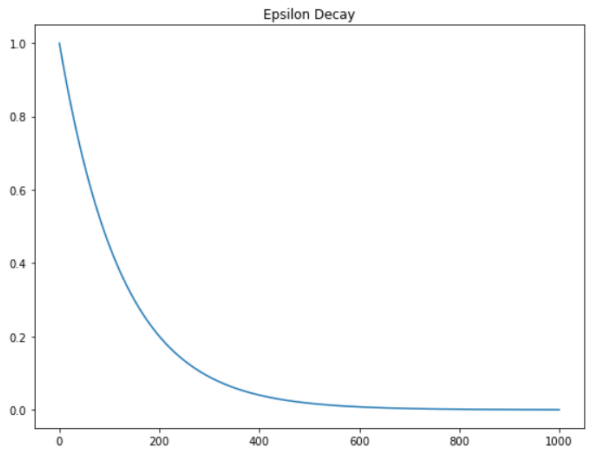
This is because in the exploration phase the random actions selected in the Q-learning were upwards and in SARSA they were sideways and because of this when the Q values were updated and best actions were selected, the robot used to select the path which it explored more. In Q learning since we take the max of Q of next state the agent used to select the path of going up towards +5 reward and then going to the goal state. Whereas in the SARSA the next action is selected by the policy randomly the randomness generated more actions towards left in training and the agent used to follow that path to collect +2 reward and then reach the goal.

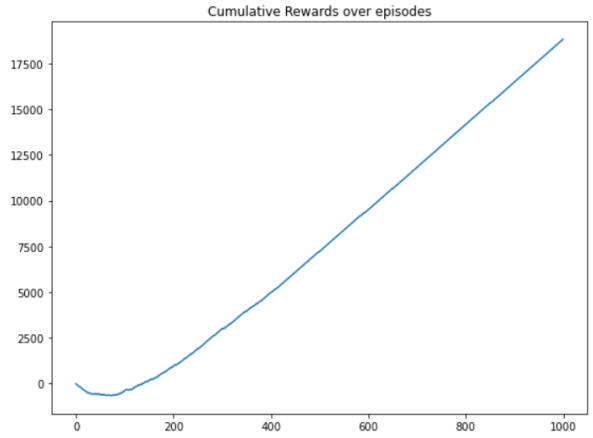
It is possible that when the agent is trained again in the deterministic environment using Q-Learning the agent will follow the same path as SARSA. This is because of the randomness of the actions taken during the exploration phase. Also, since the intermediate reward of +2 are closer to negative terminal states the agent takes the suboptimal path of going upwards and collecting the +5 reward like it did here. If the negative terminal states were placed near the +5 reward, then the agent would have always selected the best optimal path i.e. taking intermediate reward of +2 and then reaching goal state. So, the difference in the environment training using Q learning and SARSA is because of the positions of the negative terminal state and the randomness of the exploration phase happening initially.

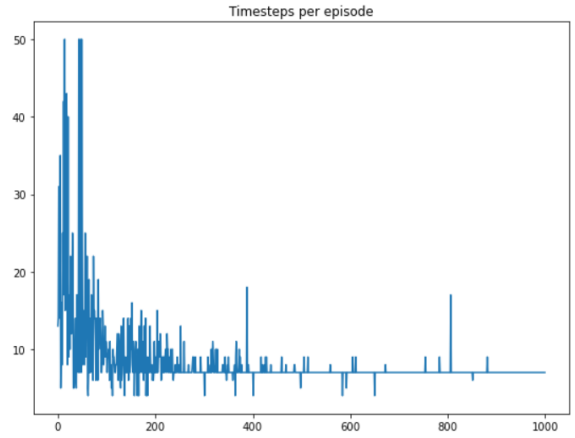
When the agent was trained in Q-Learning environment again following were the results:

Training results:

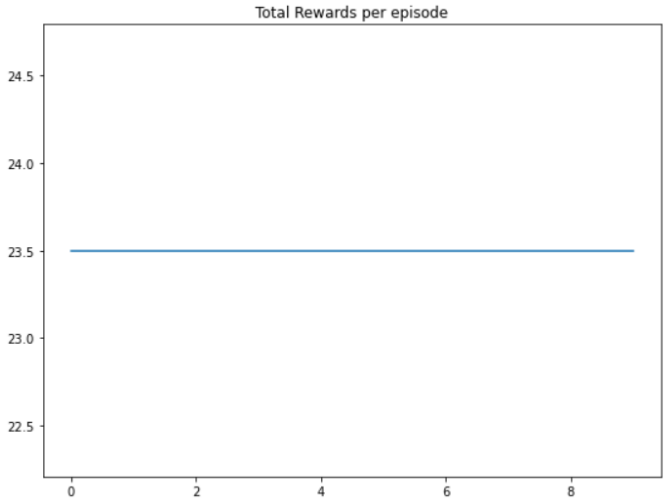


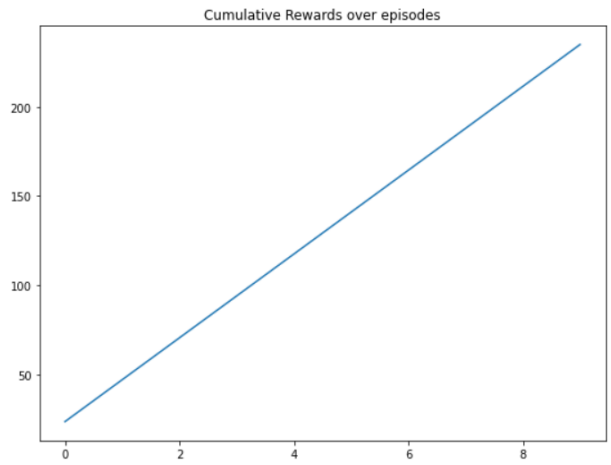


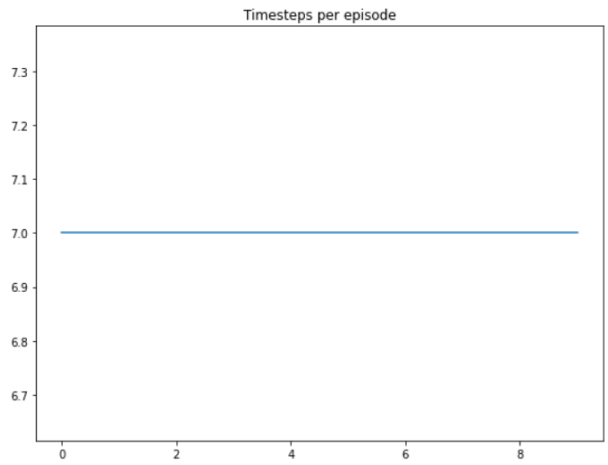




Testing Results:



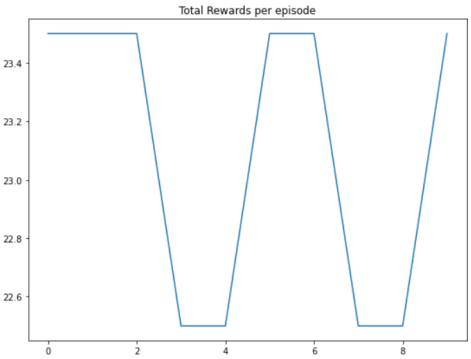


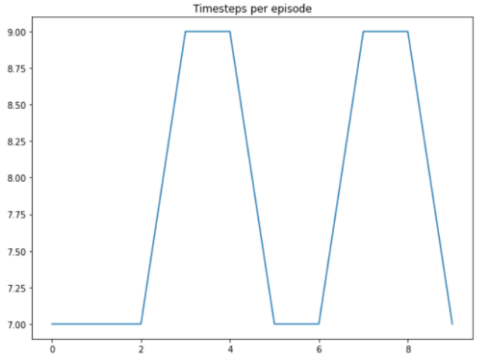


As seen above the results are same as that of SARSA. The maximum reward achieved during this is 23.5 and the timesteps the agent took is 7 which is same as that of the reward received and timesteps the agent took using SARSA method.

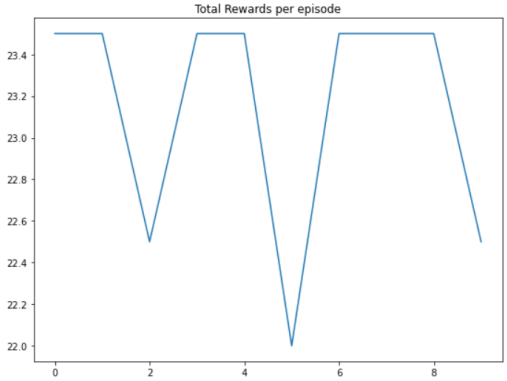
Comparison of methods on Stochastic Environment

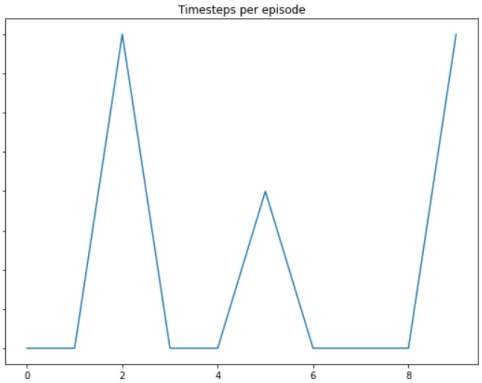
Q-Learning





SARSA





For stochastic environment the performance of the robot using both the methods is very similar too. In both the methods the agent reached the goal state without falling into the pit or getting eaten by the monster. Like in the deterministic environment the Q-Learning follows a path to go upward and collect +5 reward and then reach goal state in 9 timesteps compared to SARSA which goes sideways towards left and collects +2 reward and reaches the goal in 7 timesteps, same can be observed here. The minimum timestep taken in Q-Learning is 9 and in SARSA it is 7. The difference in timestep per episode in each method is because of stochasticity in the environment

Tuning of Hyperparameters

For Tuning of Hyperparameters to evaluate its effect on learning Q-Learning method is used. The environment which is used is Deterministic environment. The parameters used for tuning are:

1. Epsilon Decay Rate
2. Max Timesteps

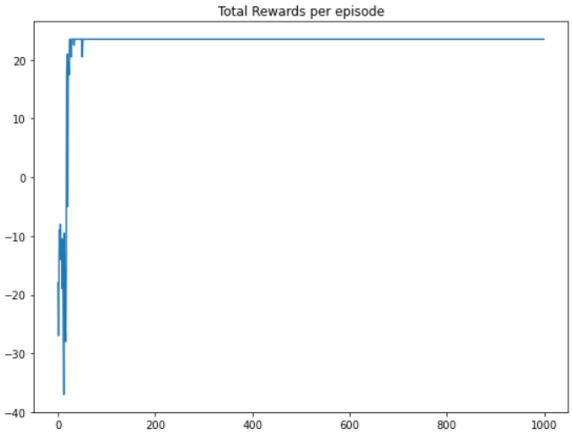
Epsilon Decay Rate

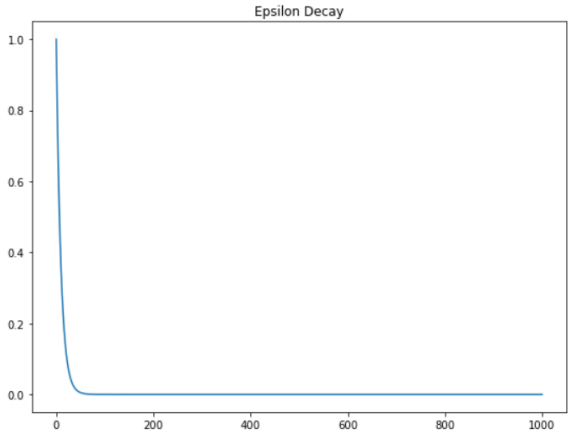
Epsilon decay rate is the rate at which the epsilon value decreases over time. Epsilon (ε) value defines the probability that a random action is selected and 1 – ε value defines the probability that a greedy action is selected. Epsilon value initially is set high so that the agent explores the environment and decreases over time so that the agent selects the best action possible to reach the goal state.

Following are the results of tuning Epsilon Decay Rate using Q-Learning Algorithm used to find the optimal policy in Robot Grid World problem

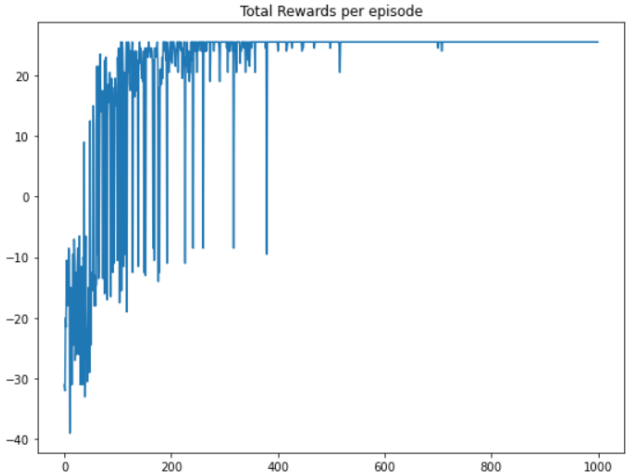
Parameters used for training: alpha = 0.9, gamma = 0.9, epsilon = 1, epsilon decay values = [0.1, 0.01, 0.001], training episodes = 1000

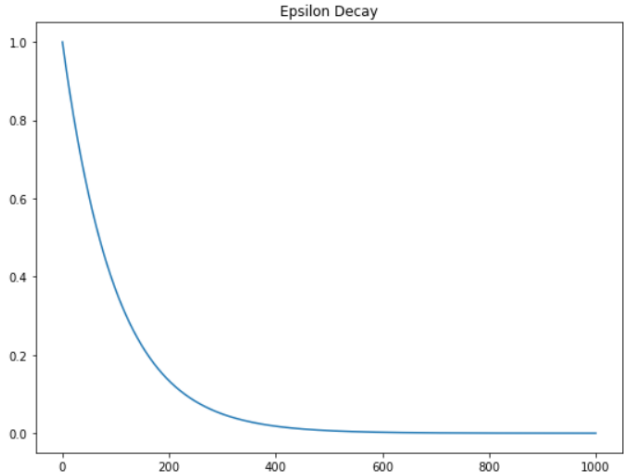
Epsilon Decay Value: 0.1



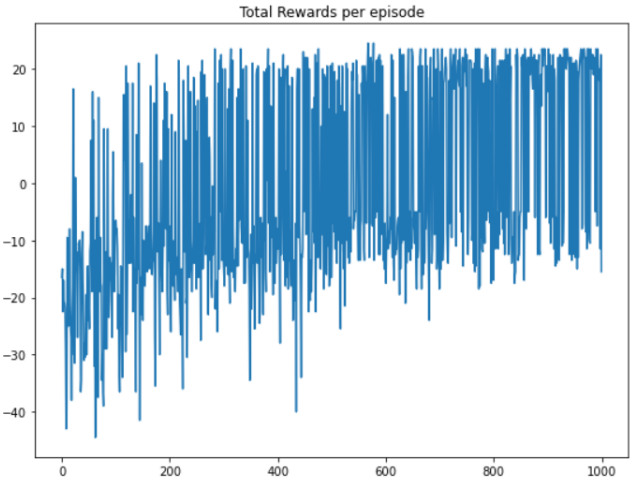


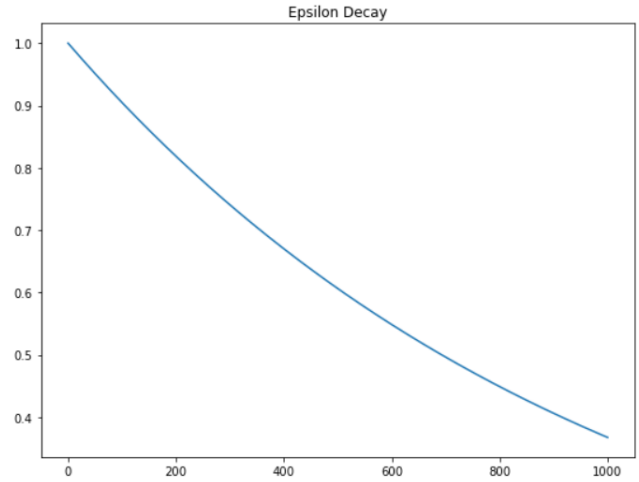
Epsilon Decay Value: 0.01





Epsilon Decay Value: 0.001





As we can see that when the decay value is 0.1 the probability of robot exploring decreases quickly and after almost less than 100 episodes the robot starts taking greedy actions because of this the rewards obtained after 100 episodes is almost a straight line

When the decay value is 0.01 the probability of robot exploring decreases over time exponentially. After almost 400 episodes the robot starts taking only greedy actions. Till then it explores the environment and because of this the reward graph converges after almost 400 episodes and then it's a straight line.

When the decay value is 0.001 the decreasing rate is low and over 1000 episodes the epsilon value doesn't go below 0.4. Hence the agent is just exploring for most of the episodes. The reward graph shows that most of the times because of exploring the robot either falls in the pit or gets eaten by the monster and only sometimes reaches the goal state. Because of exploration and not taking greedy actions the reward graphs does not converge

Hence the most efficient decay rate out of the three is 0.01 since during this the agent explores the environment and then after some episodes starts taking greedy actions to learn the optimal path to reach the goal state.

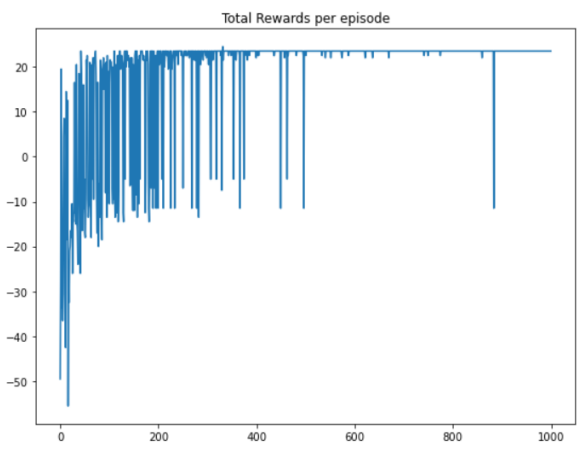
Max Timesteps

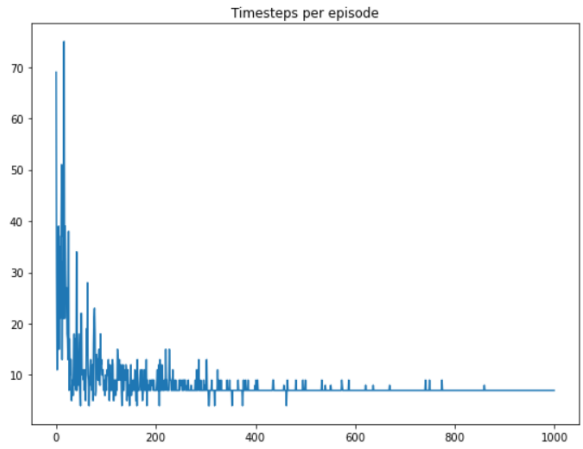
Maximum timesteps is the maximum number of times the agent can act in the environment. This is used so that while learning if the agent doesn't reach any terminal state, the episode must end somewhere. This is to make sure that the agent doesn't get stuck in an infinite loop and terminates. After each action taken the timestep is updated and when the agent reaches maximum timestep the episode terminates

Following are the results of tuning Max timesteps using Q-Learning Algorithm used to find the optimal policy in Robot Grid World problem:

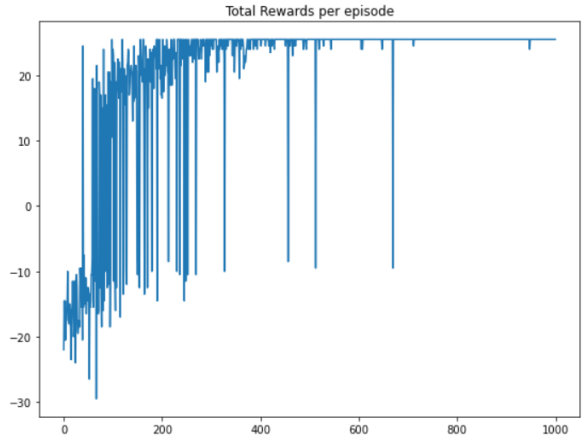
Parameters used for training: alpha = 0.9, gamma values = 0.9, epsilon = 1, Epsilon decay = 0.008, training episodes = 1000, Timestep values = [75,25,12]

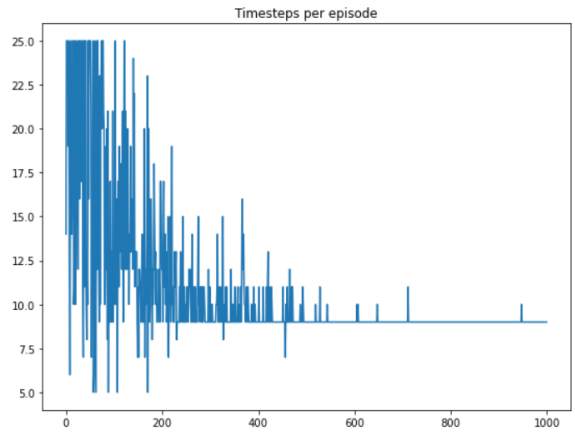
Max Timesteps: 75



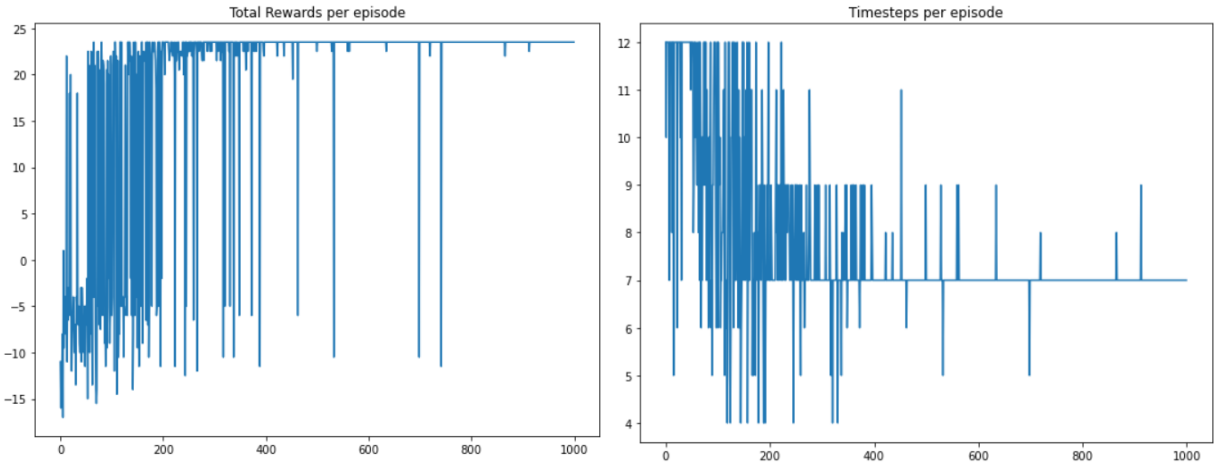


Max Timestep: 25





Max Timestep: 12



When the max number of timesteps is 75 we can see that initially while exploring the agent explores a lot of states initially and because of this the rewards received in the beginning of episodes are almost close to -50. The robot explores a lot of states initially and Q values of those states are updated and then the agent reaches the terminal states faster as compared to other two graphs.

When the max number of timesteps is 25 we can see that the agent can't explore much of the states like when max timesteps was 75. Q values of the states are updated over time and to reach the terminal state the timesteps taken by the agent decreases slowly.

When the max number of timesteps is 12 we can see that agent cannot explore all the states and takes more episodes exploring. Because of this the reward graph also converges slowly.

Hence the optimal max timesteps out of the three would be 75 as the agent explores a lot of states during the exploration using epsilon greedy policy and the rewards will converge quickly.

GITHUB REPOSITORY LINK:

https://github.com/anupthakkar/Robot-Grid-World