Assignment 1 - Defining & Solving RL Environments

Sagar Jitendra Thacker
UBname: sagarjit
Person no: 50363834
Department of Computer Science and Engineering
University at Buffalo

1. Main Objective

The main objective of this project is to implement a reinforcement environment following OpenAI Gym standards. We focus on implementing deterministic and stochastic environments that are based on Markov Decision Process. The agent is trained using Q-learning and Double Q-learning tabular methods. The agent derives an optimal policy to navigate the environment to reach the goal and avoid obstacles. Hyper-parameters were tuned to find the optimal results and the performance of the agent on both deterministic and stochastic environments was analyzed.

2. Implementation

The project was implemented following OpenAI Gym standards.

2.1. Environment

The environment is a 5*5 Grid World that consists of 25 states. The environment is fully observable, Single-agent, Episodic, and Discrete.

s21	s22	s23	s24	s25
s16	s17	s18	s19	s20
s11	s12	s13	s14	s15
s6	s7	s8	s9	s10
s1	s2	s3	s4	s5

State set: It is the set of all possible states present in the environment.

$$S = \{s1, s2, s3, s4, s5, s6, s7, s8, s9, s10, s11, s12, s13, s14, s15, s16, s17, s18, s19, s20, s21, s22, s23, s24, s25\}$$

State s1 is the starting state and state s25 is the goal state. States s4, s13 contains reward of +5 and +10 respectively. State s16 contains a reward of -3. States s9, s12 are dead states i.e. when an agent reaches that state the environment resets and the agent receives a reward of -50 in each state. State s25 contains a reward of +50. All the other states have a reward of -1. An episode ends if the agent reached the goal state or any one of the dead states.

There are two types of environments:

- **Deterministic**: In the deterministic environment the probability of an action a in state s is certain i.e. there is no randomness or uncertainty. Example: If there are four actions (up, right, left, down) if the agent chooses to go up, the probability of that action in state s is 1 and all other actions are 0.
$$P(s', r|s, a) = \{0, 1\}$$
- **Stochastic**: In a stochastic environment there is some degree of randomness or uncertainty. Here the same action taken by the agent may produce different results based on how to model is defined i.e. the agent may enter different states based on the degree of randomness in the same state s.
$$\sum_{s',r} P(s', r|s, a) = 1$$

2.2. Action set

It is the set of actions that an agent can take in the environment. For the grid world environment the action set is given below:

$$A = \{Up, Down, Left, Right\}$$

2.3. Rewards

Reward set: It is the set of all possible rewards an agent can receive from the environment after taking an action. For the grid world environment the reward set is as follows:

$$R = \{-1, -3, 5, 10, -50, 50\}$$

Compared to part 1 there are changes in the reward set. It is ideal to have our highest reward thrice or five times the second-highest reward. Previously the highest reward was +100 and -100 for goal and dead states respectively. Now, the rewards for goal and dead states are +50 and -50 respectively.

3. Visualization of the environment

Below are some screenshots of the environment taken at different timesteps:

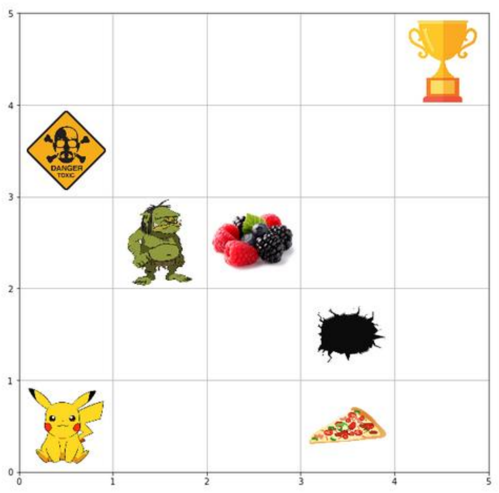

Figure 1: Initial State

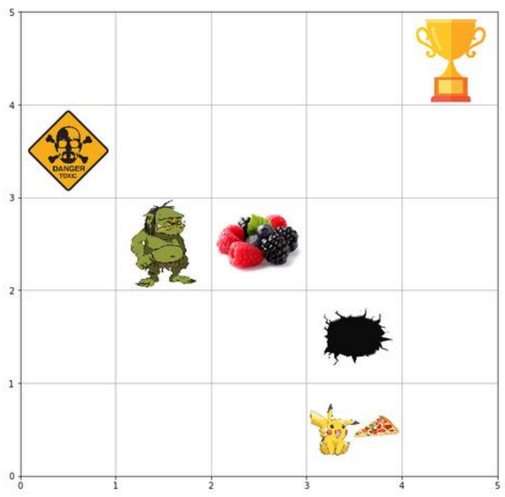

Figure 2: Agent received reward

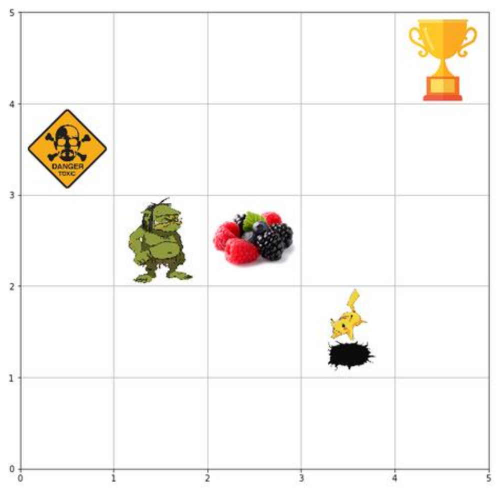

Figure 3: Agent fell in the pit

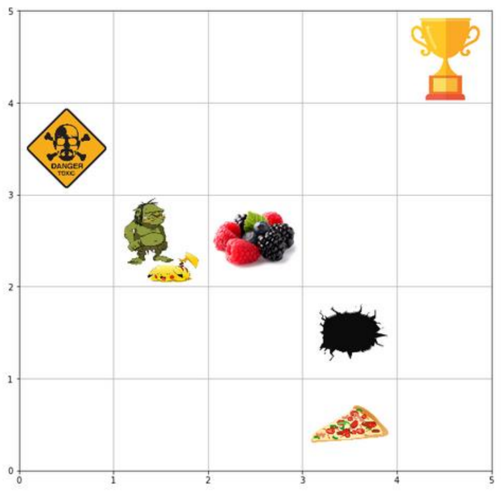

Figure 4: Agent killed by the monster

Pikachu represents the agent in the starting state (bottom left corner) and the trophy represents the goal state. Pizza and Berries represent rewards of +5 and +10 respectively. The danger sign represents a reward of -3. Monster and Pit are dead states i.e. when an agent reaches that state the environment resets and the agent receives a reward of -50 in each state. The trophy represents is our goal state and contains a reward of +50.

4. Define Stochastic Environment

An environment is stochastic if the agent is in state *s* and takes an action *a* can lead to different states. It contains some degree of randomness. Find the below table for the probability distribution for different actions:

Action	Up	Down	Left	Right
Up	0.94	0.02	0.02	0.02
Down	0.02	0.94	0.02	0.02
Left	0.02	0.02	0.94	0.02
Right	0.02	0.02	0.02	0.94

The probabilities for different actions in the above table are different from the one submitted in the part 1 report. This is because the agent was unable to learn in a stochastic environment and the degree of uncertainty was higher.

5. Q-learning

Q-learning is an off-policy reinforcement learning tabular method algorithm. Off policy, because there is no information about the transition probabilities i.e. there is no model of the environment. The action is not based on the policy but rather the best possible action that the agent can take in that state which is learned based on experience.

In q-learning, we maintain a q-table which is a table of dimensions (number of states * number of actions). The table is initialized to 0 before training and updated every episode. The agent takes action based on the epsilon greedy method while training based on the decay rate. The environment returns the next state and immediate reward. The q-value for the current state and action is updated based on the highest q-value for the action in the next state. We update the q-table until the values converge.

5.1. Update function for Q-learning

$$Q_{new}(s_t, a_t) = Q_{old}(s_t, a_t) + \alpha(r_t + \gamma * max_a Q(s_{t+1}, a) - Q_{old}(s_t, a_t))$$

Where,
s_t: Current state
s_{t+1}: Next state
a: Action taken
r_t: Immediate reward
α: Learning rate
γ: Discount factor

5.2. Hyperparameters

Epsilon: It defines the degree of randomness, whether the agent should explore different states or exploit the knowledge already learned. Initially, we would want the agent to explore and determine the action-value values for different states and actions. As the training progresses we want to exploit the knowledge and take the actions that produce the highest reward. Hence, we decay epsilon as the training progresses.

Alpha: It defines the learning rate of the agent i.e. how fast the agent is trained for the given problem. For a large value of alpha, the agent will train faster but may not produce the optimal results. On the other hand, for a small value of alpha, the agent may learn the optimal policy but it will take a longer time.

Gamma: It defines the discount factor which penalizes future rewards. It determines how important are the future rewards since these rewards can be uncertain and not give any immediate benefits.

Epsilon: The number of epochs the agent is trained.

5.3. Deterministic Environment

5.3.1. Experimenting with different values of gamma hyperparameter

5.3.1.1. For $\gamma = 1$

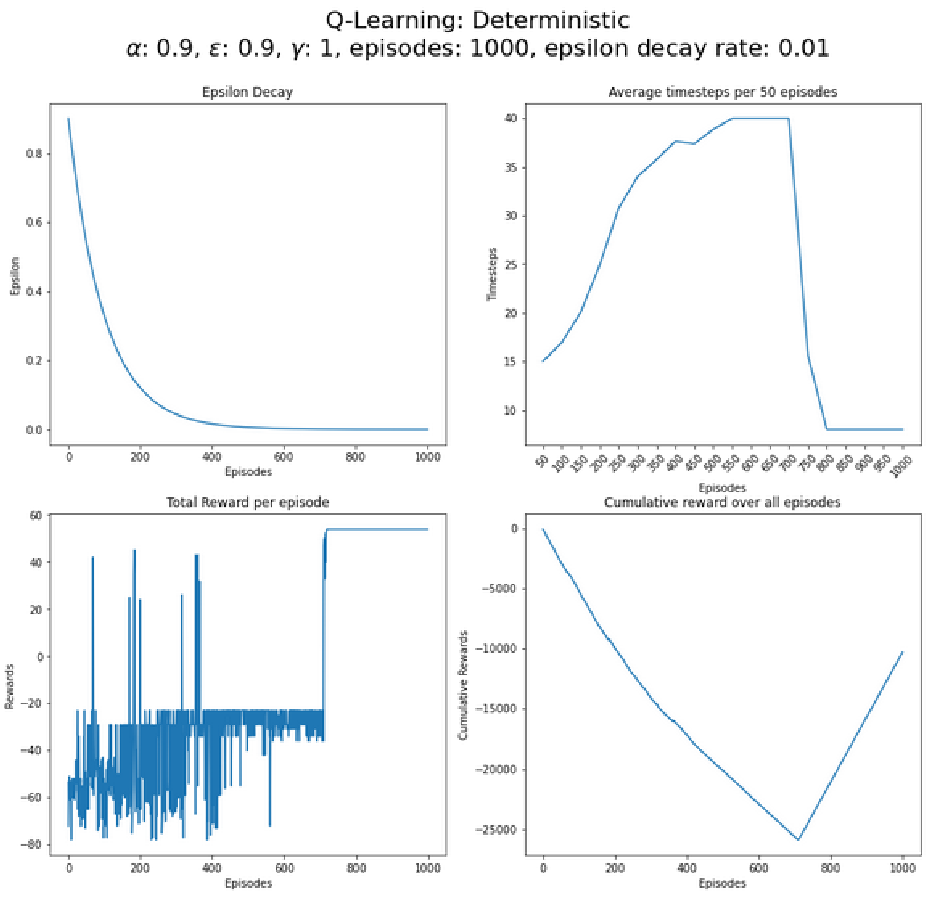

Figure 5: Q-learning Deterministic (gamma=1)

For gamma = 1 (Figure 5) we don't penalize the future rewards. Hence the agent tries to explore the whole grid space as much as possible. As seen in the total reward graph the agent reaches the goal and also encounters the dead states until 700 episodes. The agent also exhausted the maximum timesteps it is allowed to take any action in the environment which can be seen as a small band of ups and downs in the graph. Then the agent starts to take more greedy action and follow the path that reaches the goal which gives the maximum reward. This can also be validated from the average timesteps per 50 episodes, initially, the average timesteps are increasing as the agent explore more and more. Then as the agent takes more greedy actions the average timesteps converge to the optimal number of timesteps to reach the goal. Similarly, the graph for cumulative reward is decreasing and then increasing as more greedy actions are taken.

After training, evaluating the agent:

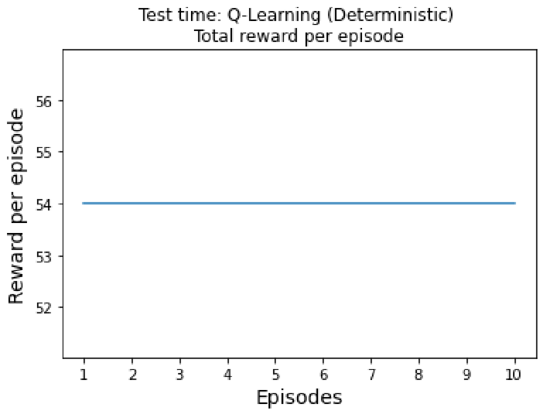

Figure 6: Evaluating agent for gamma=1

The above graph (Figure 6) showcases the total reward per episode for 10 episodes. The agent learns the optimal policy and follows the path to obtain the maximum reward.

5.3.1.2. For $\gamma = 0.9$

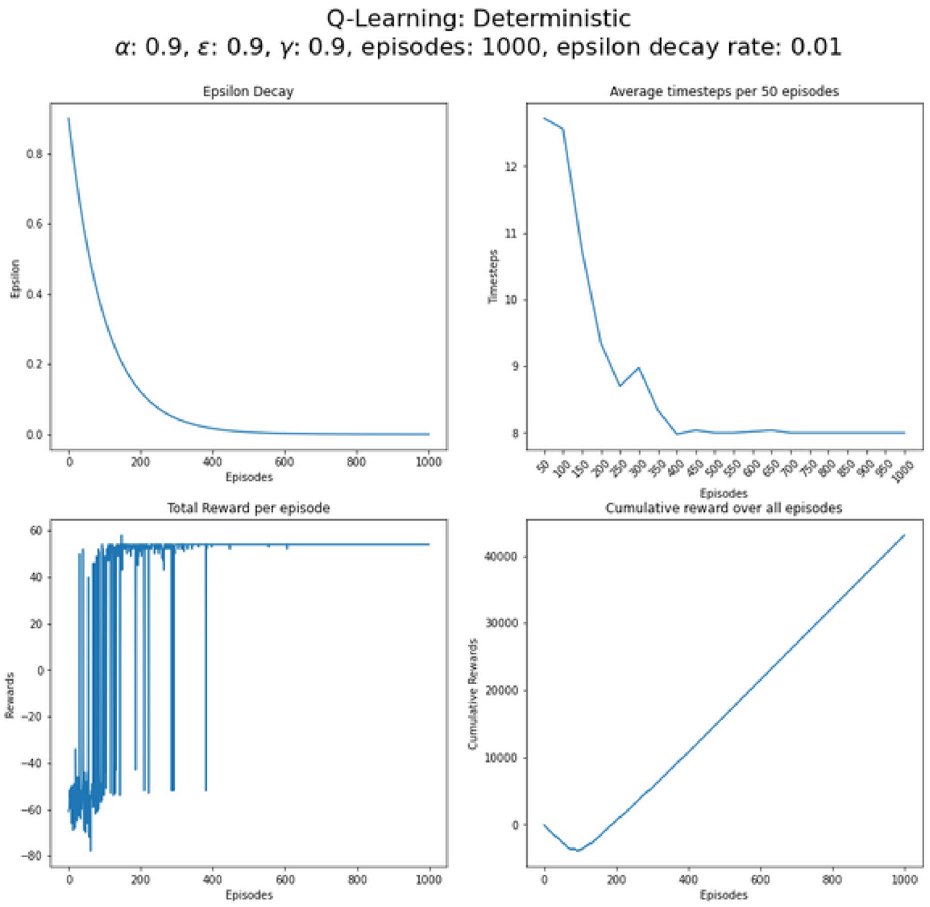

Figure 7: Q-learning Deterministic (gamma=0.9)

For gamma = 0.9 (Figure 7) we penalize the future rewards a little but there is a significant change compared to when the gamma value was 1. Here the agent converges faster. This can be seen as the average timesteps per 50 episodes starts to converge to its optimal value from the start and the total reward per episode converges a little before 400 episodes as the agent takes greedy actions to get the maximum reward. Similarly, cumulative reward per episode has a small decrease as the agent tries to explore and learn, then quickly increases linearly as the agent tries to follow the optimal path.

After training, evaluating the agent:

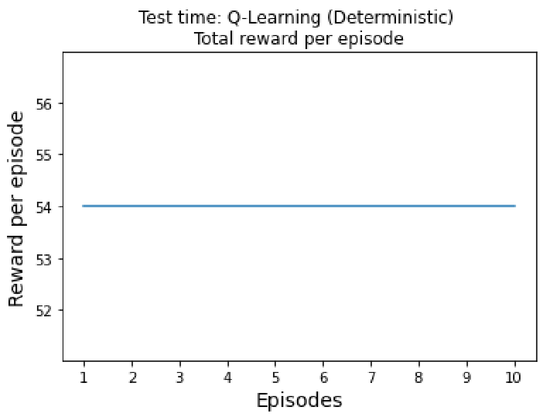

Figure 8: Evaluating agent for gamma=0.9

The above graph (Figure 8) showcases the total reward per episode for 10 episodes. The agent learns the optimal policy and follows the path to obtain the maximum reward.

5.3.1.3. For $\gamma = 0.5$

For gamma=0.5 (Figure 9) we penalize the future rewards even more. The average timesteps per 50 episodes drop much faster when compared to gamma=1 and gamma=0.9. Similarly, the total reward per episode starts to converge at its maximum value and cumulative reward per episode follows a decline while exploration and then increases as the agents take actions to receive the highest reward.

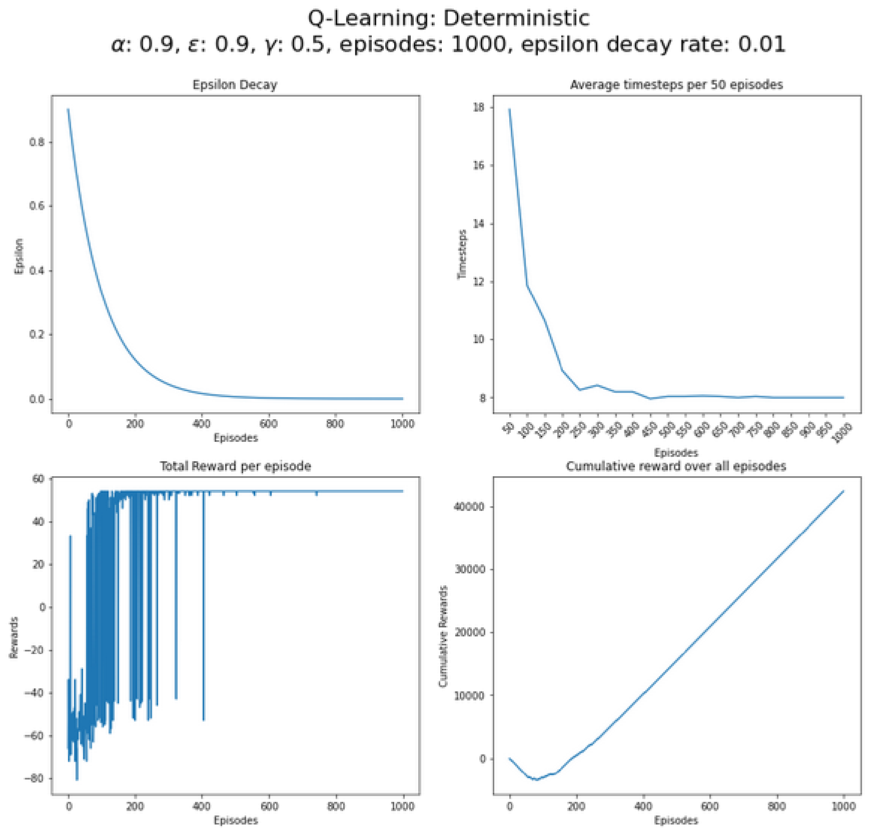

Figure 9: Q-learning Deterministic (gamma=0.5)

After training, evaluating the agent:

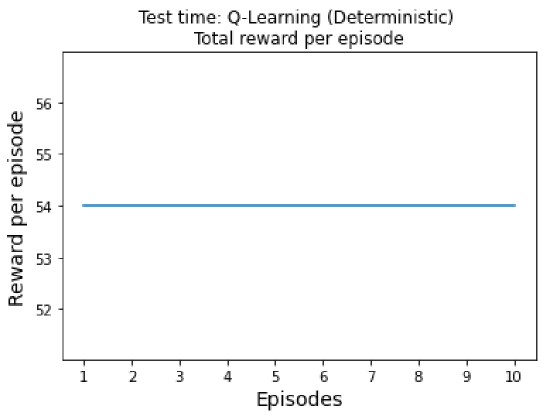

Figure 10: Evaluating agent for gamma=0.5

The above graph (Figure 10) showcases the total reward per episode for 10 episodes. The agent learns the optimal policy and follows the path to obtain the maximum reward.

Combining graphs (Figure 11) for all three gamma values we can see the effect of how gamma affects the way the agent learns. Similar results can be seen for gamma values 0.9 and 0.5 but greatly differences for gamma=1. Here we can also see a graph for success rate i.e. how many times the agent reached the goal per 100 episodes. For gamma = 0.9 and 0.5 as the number of episodes increases success rate increasing but for gamma=1 there is more exploration and the success rate is low and then increases as more greedy actions are taken. As we know future rewards can be uncertain and penalizing the future rewards helps us to better understand the value for the current state-action pair.

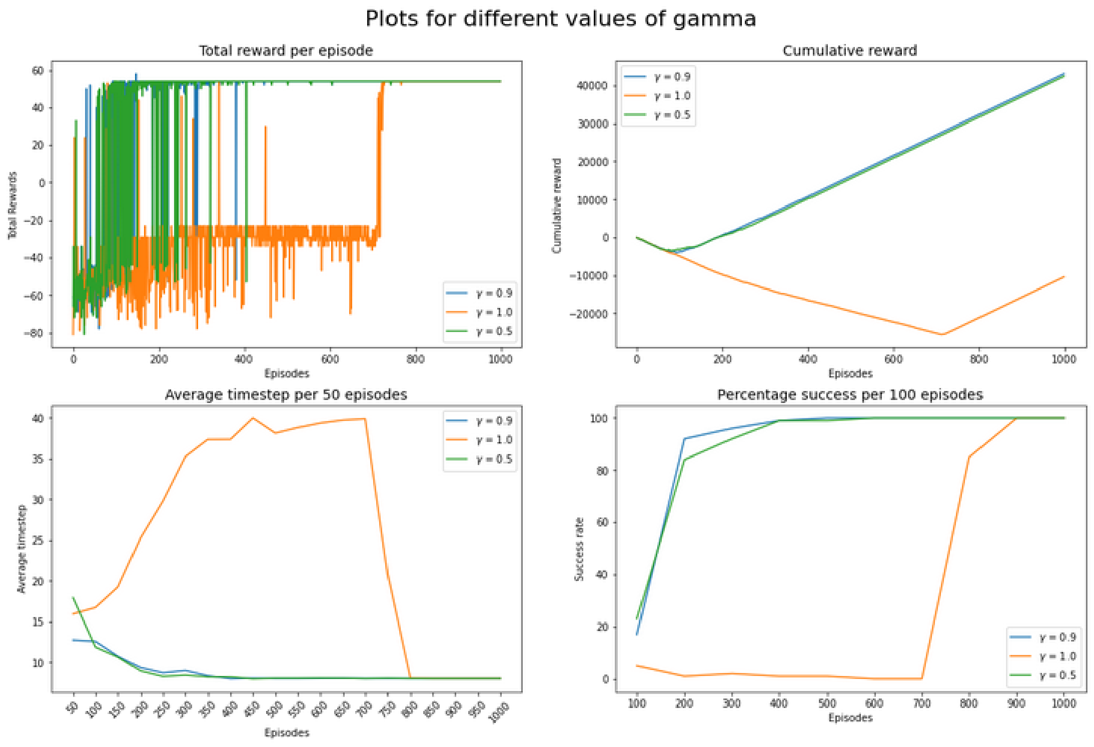

Figure 11: Comparing graphs for different gamma values

5.3.2. Experimenting with different values of epsilon decay rate (edr)

5.3.2.1. For epsilon decay rate = 0.01

Epsilon decay has been calculated using the formula:

$$epsilon = epsilon - (epsilon * epsilonDecayRate)$$

Initially, the agent takes more random actions to explore and as the epsilon decreases the agent starts to take more greedy action to receive the maximum reward. The total reward per episode graph varies more in the beginning as the agent is exploring and then flattens out to the maximum value as it takes more greedy actions. Similar behavior can be seen in cumulative reward where the cumulative reward decreases initially and then increases. As the agent explores it takes more timesteps but while choosing greedy actions it converges to the optimal number of time steps required to reach the goal which can be seen in the average timesteps per 50 episodes graph.

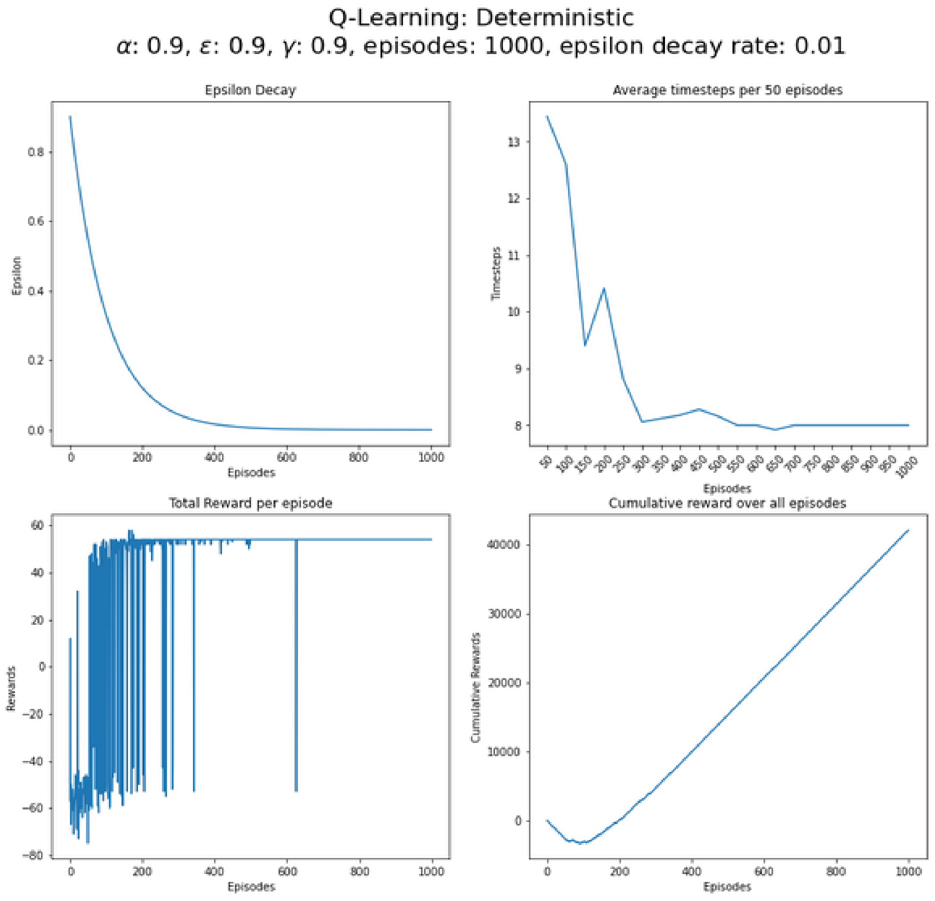

Figure 12: Q-learning Deterministic (edr = 0.01)

After training, evaluating the agent:

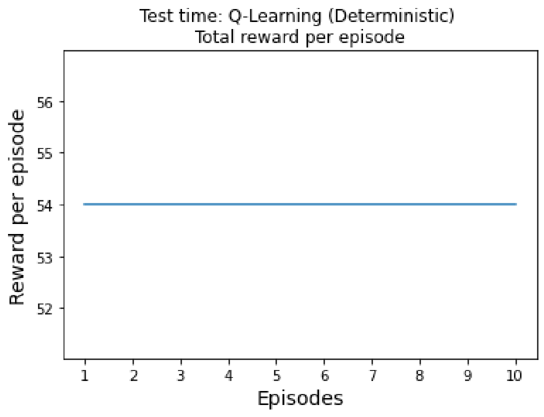

Figure 13: Evaluating agent for edr=0.01

The above graph (Figure 13) showcases the total reward per episode for 10 episodes. The agent learns the optimal policy and follows the path to obtain the maximum reward.

5.3.2.2. For epsilon decay rate = 0.001

As shown in figure 14 the epsilon decays much slower than when the epsilon decay rate = 0.01. Hence, here the chance of taking a greedy action as epsilon decreases in less compared to before. This effect can be seen in the other graphs. The total reward per episode graph moves up and down a lot because of random actions being taken even after it was trained on 1000 episodes. Average timesteps per 50 episodes can be seen to have a decline with ups and downs in between due to the randomness. Similarly, cumulative reward the graph decreases, and as the number of episodes increases, agents try to take greedy action but are unable to do so due to slower epsilon decay rate.

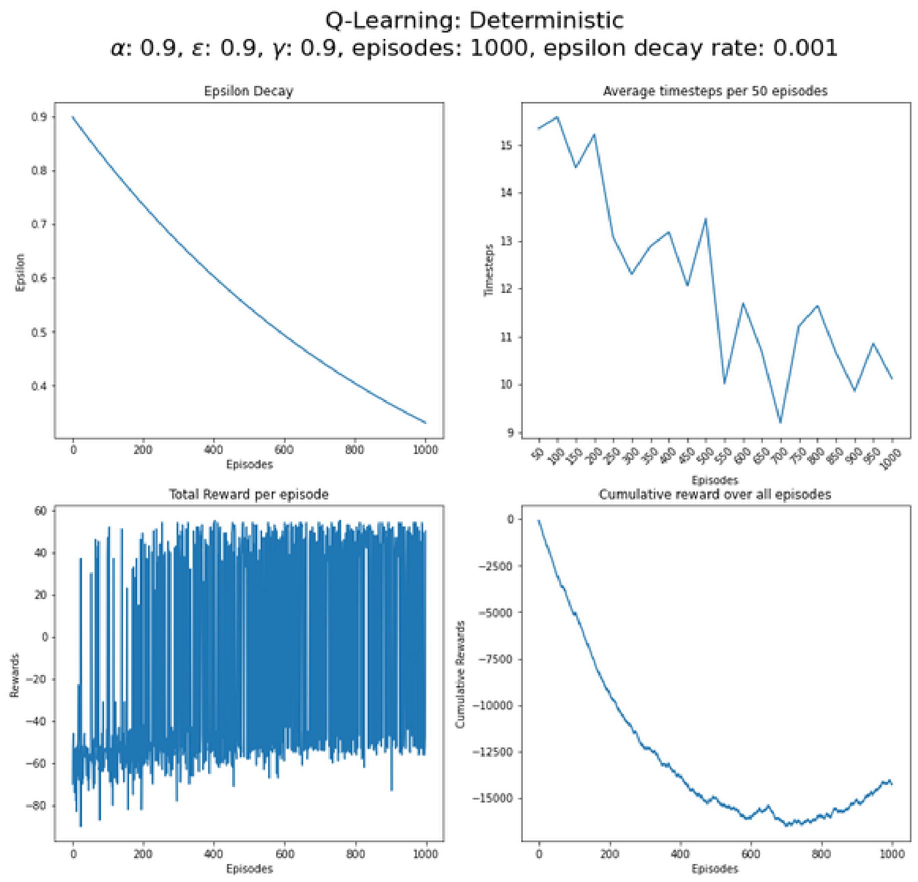

Figure 14: Q-learning Deterministic (edr=0.001)

After training, evaluating the agent:

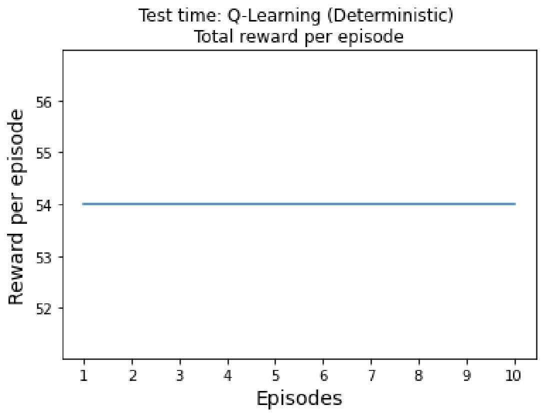

Figure 15: Evaluating agent for edr=0.001

The above graph (Figure 15) showcases the total reward per episode for 10 episodes. Since the epsilon is zero there is the absence of randomness hence, while evaluating the agents only takes greedy action. The agent learns the optimal policy and follows the path to obtain the maximum reward.

5.3.2.3. For epsilon decay rate = 0.005

Epsilon decay for epsilon decay rate 0.005 is not as steep as when decay rate is 0.01 and not as slow as 0.001, it is somewhere in between them. Hence, there is more randomness compared to edr=0.01 but less than 0.001. This effect can be seen in figure 16 total reward per episode graph varies a lot from the beginning toward the middle but then slowly converges towards the maximum reward that the agent can receive. Similarly, average timesteps per 50 episodes show a decline with few ups and downs but compared to edr=0.001 the variations are less and it converges to the optimal number of timesteps to read the goal towards the end. The cumulative reward can be seen with a decline while exploring but increases as the agent take more greedy steps.

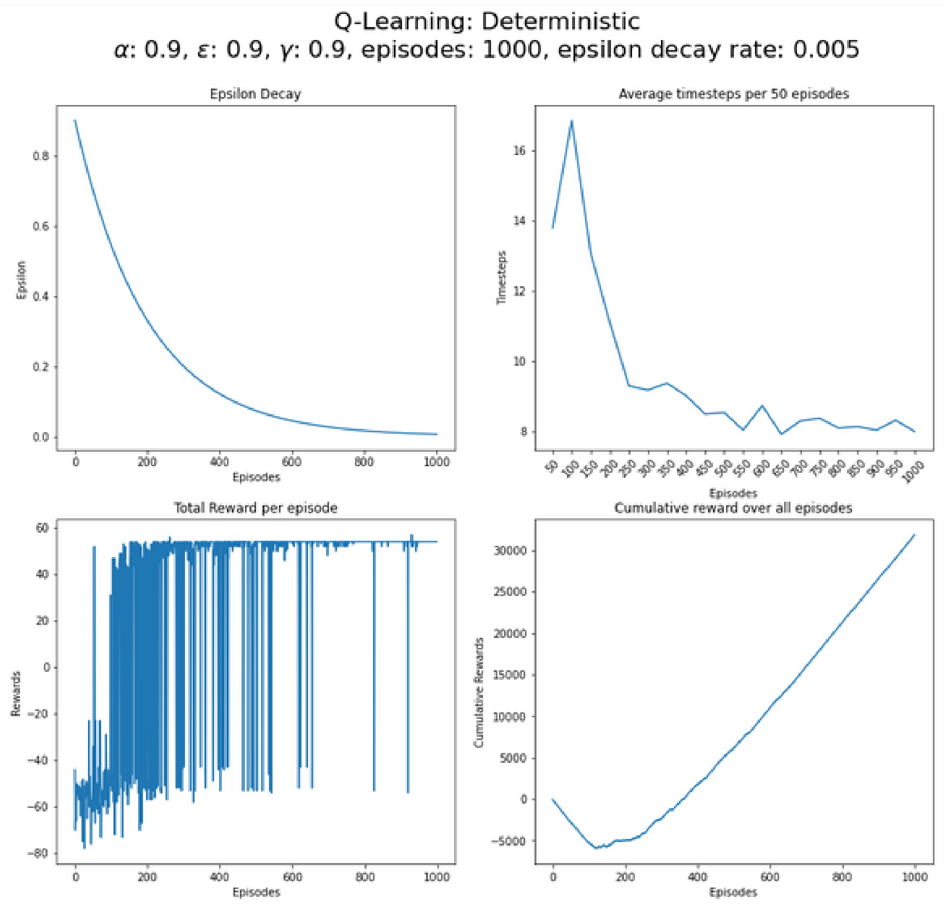

Figure 16: Q-learning Deterministic (edr=0.005)

After training, evaluating the agent:

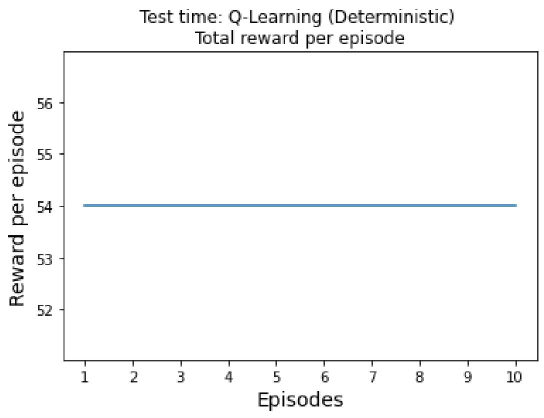

Figure 17: Evaluating agent for edr=0.005

The above graph (Figure 17) showcases the total reward per episode for 10 episodes. Since the epsilon is zero there is the absence of randomness hence, while evaluating the agents only takes greedy action. The agent learns the optimal policy and follows the path to obtain the maximum reward.

Comparing results for all three epsilon decay rates (Figure 19) total reward per episode graph varies more when the decay rate is 0.001 than when the decay rate is 0.005. It varies the least when the decay rate is 0.01. Similar behavior can be seen in average timesteps per 50 episodes as agents move around more when the decay rate is 0.001 than when the decay rate is 0.005 and the lease in 0.01. The cumulative reward will below are decay rate 0.001 than decay rate 0.005 than 0.01. The success rate is lowest when the decay rate is 0.001 due to more random actions than greedy compared to the decay rate of 0.005. The best success rate can be seen when the decay rate is 0.01.

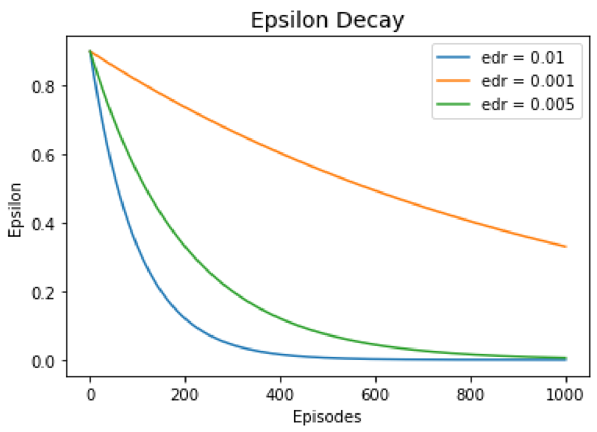

Figure 18: Epsilon Decay rate graphs

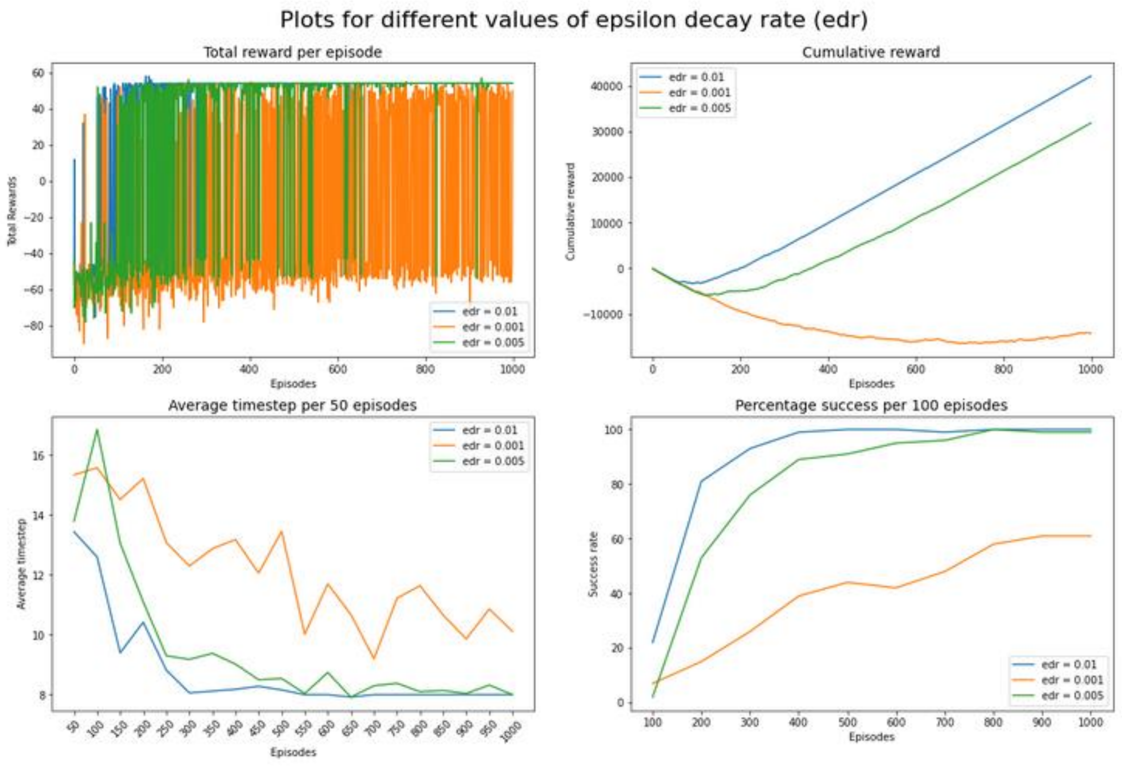

Figure 19: Comparing results for different edr values

Looking at all the graphs for different combinations of gamma and epsilon decay rates. The best results can be seen for when alpha=0.9, gamma=0.9, epsilon=0.9, episodes=1000 and epsilon decay rate=0.01.

5.4. Stochastic Environment

Results of the stochastic environment, when compared to the deterministic environment, are different because of the stochasticity introduced in the stochastic environment. Here even though the epsilon decay happens and the agent tries to perform an action that gives the highest reward, the environment can perform the same or different action based on the probability distribution for that action space.

This effect can be seen in figure 20 total rewards per episode graph tries to move toward the maximum reward but due to the stochastic environment either it runs out of timesteps or falls into one of the dead states and terminates. Ups and downs in average timesteps per 50 episodes can be seen due to this stochasticity.

The reason the total reward per episode has negative rewards even if the agent has trained for 1000 episodes is that as the agent tries to follow the optimal path i.e. to receive the berry or pizza, there are two dead states adjacent to those rewards. Consider if the agent is following the optimal path and reaches the berry position then due to stochastic in the environment choose to go left where the agent might have chosen to go up. Due to which the episode terminates and the agent receives a negative reward. The graph for cumulative reward also shows the same variations.

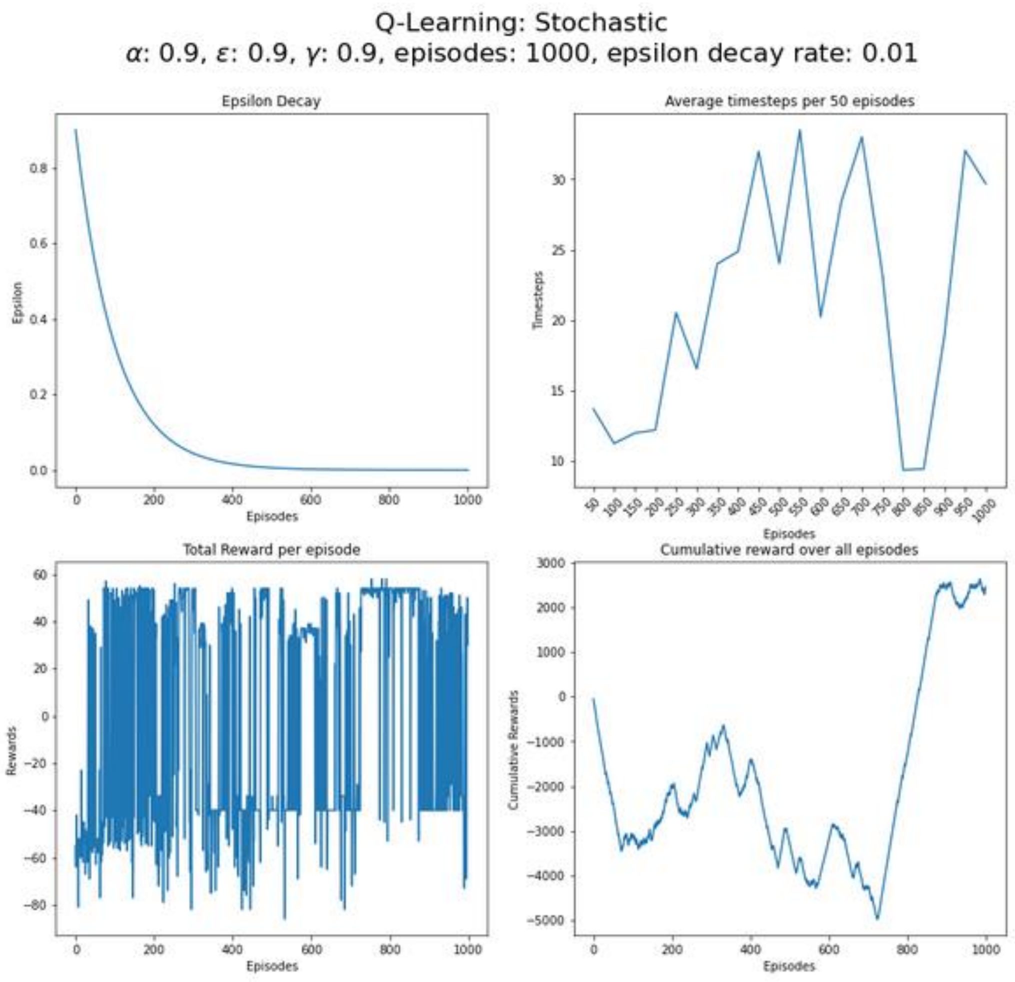

Figure 20: Q-learning Stochastic

After training, evaluating the agent:

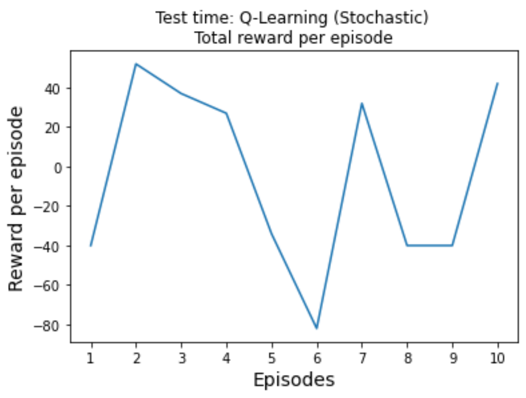

Figure 21: Evaluating agent for Stochastic Environment

Here for episode 1 the agent tried to move towards the optimal path and reaches the berry with +10 and then rather than going up, the agent went to the left due to stochasticity and received a negative reward of -50 which resulted in -40, and the episode terminated. Due to this reason, we can see a lot of ups and downs in the graph. Reward -80 can be seen because the agent is moving around and then either fell in the pit or encountered the monster due to stochasticity.

6. Double Q-learning

Double Q-learning is also an off-policy-based tabular method. The idea behind double q-learning is to ask the question "Is one estimator good enough?"

Since in Q-learning the optimal policy is a greedy policy based on the current action values, which is defined based on the maximum operation. Consider if the right action is taken by the agent and by using max we estimate the value to be positive but if the reward is drawn from a normal distribution with a mean of -1 and variance 0.5. Thus, the expected return for taking a right from that state is a mistake. But still, the agent will take action to go right. This is call maximization bias. To overcome this issue we maintain two q-tables so that the chances of overestimation for the same action are reduced.

Estimator Q1: Obtains the best action from the Q1 table for that state
Estimator Q2: Evaluates the Q value for the action we got from above

6.1. Update function for Double Q-learning

$$\boldsymbol{Q_1(S,A) = Q_1(S,A) + \alpha(R + \gamma * Q_2(S', argmax_a Q_1(S',a)) - Q_1(S,A))}$$

$$\boldsymbol{Q_2(S,A) = Q_2(S,A) + \alpha(R + \gamma * Q_1(S', argmax_a Q_2(S',a)) - Q_2(S,A))}$$

The hyperparameters are similar to that of Q-learning.

6.2. Deterministic Environment

As seen in figure 22 below as the number of episodes increases the epsilon value decreases. Initially, the agent takes more random actions and tries to explore the environment hence the total reward per episode varies more. As the epsilon values decrease and force the agent to take more greedy action the value for total reward per episode converges to the maximum reward that the agent can take.

There is a drop in reward somewhere near 400 episodes which can be due to the best action received from one of the q tables when evaluated on the 2nd q table might not give the best q value and hence saw a decline. Upon learning further the agent learns to follow the optimal path.

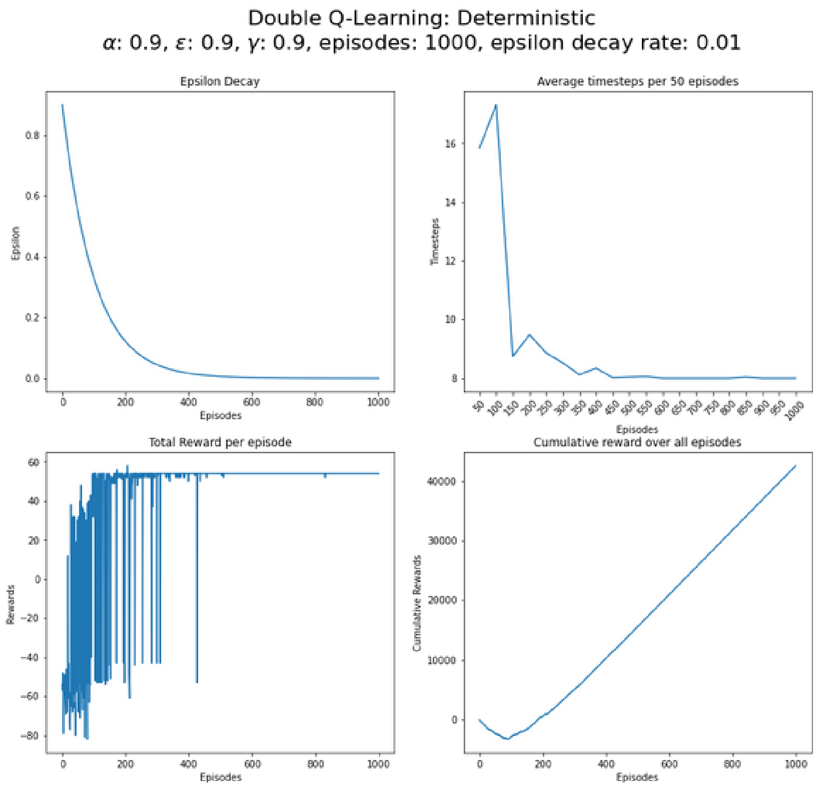

Figure 22: Double Q-learning Deterministic Environment

After training, evaluating the agent:

The below graph (Figure 23) showcases the total reward per episode for 10 episodes. The agent learns the optimal policy and follows the path to obtain the maximum reward.

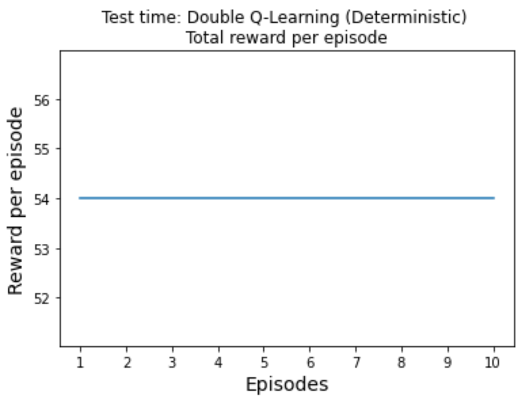

Figure 23: Evaluating agent using Double q-learning (Deterministic)

6.3. Stochastic Environment

Here even though the epsilon decay happens and the agent tries to perform an action that gives the highest reward, the environment can perform the same or different action based on the probability distribution for that action space.

This effect can be seen in figure 24 total rewards per episode graph tries to move toward the maximum reward but due to the stochastic environment either it runs out of timesteps or falls into one of the death states and terminates. There is an overall decline in average timesteps per 50 episodes but ups and downs can be seen due to this stochasticity.

The reason the total reward per episode has negative rewards even if the agent has trained for 1000 episodes is that as the agent tries to follow the optimal path i.e. to receive the berry or pizza, there are two dead states adjacent to those rewards. Consider if the agent is following the optimal path and reaches the berry position then due to stochastic in the environment choose to go left where the agent might have chosen to go up. Due to which the episode terminates and the agent receives a negative reward. The graph for cumulative reward also shows the same variations.

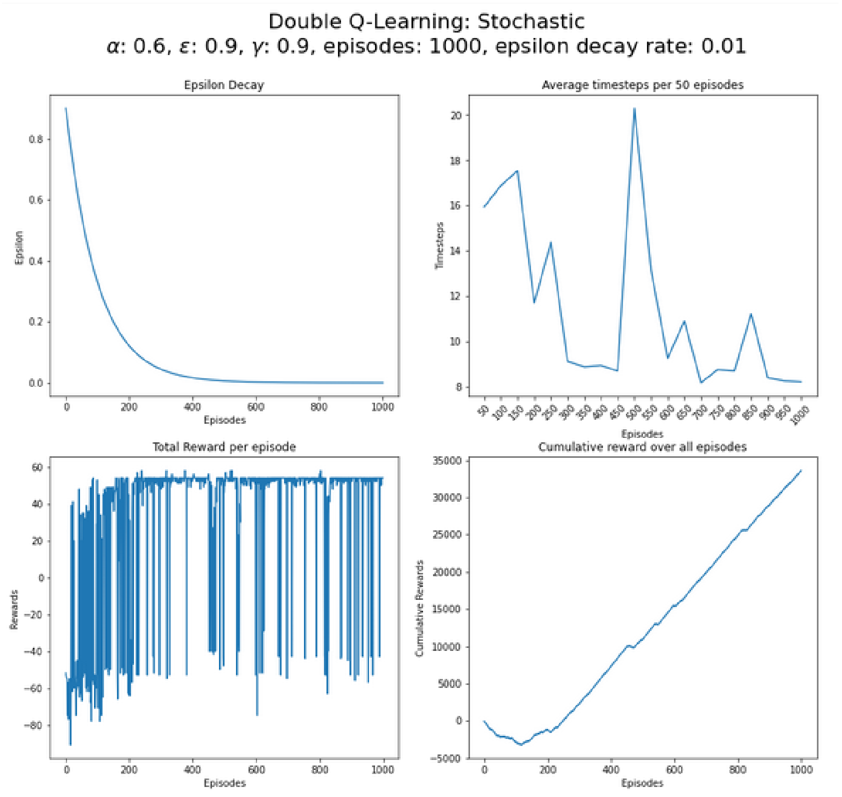

Figure 24: Double Q-learning Stochastic Environment

After training, evaluating the agent:

In most of the episodes, the agent can reach the goal state and receive the maximum reward (Refer to figure 25). But in some cases such as in episode 8, there is more random movement i.e. not following the optimal path but few different actions are being performed due to stochasticity and the overall reward received for that episode decreases.

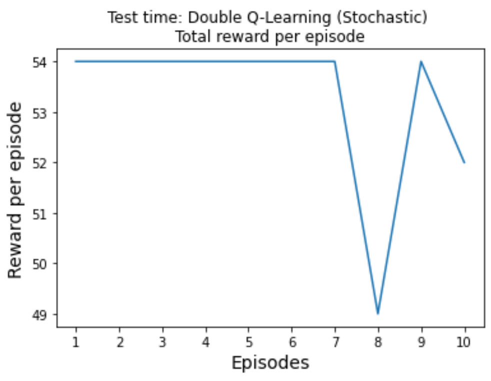

Figure 25: Evaluating agent using Double Q-learning (Stochastic)

7. Compare the performance of both algorithms in the same deterministic environment

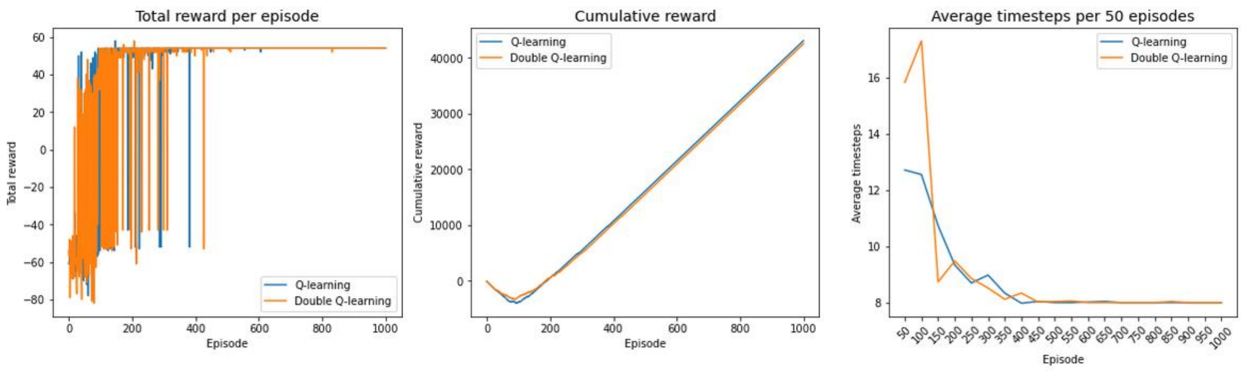

Figure 26: Q-learning vs Double Q-learning in Deterministic environment

The results for both q-learning and double q-learning are similar in the deterministic environment. Since the environment is simply the performance in the deterministic environment for both are similar. The only difference that can be seen is that double q-learning even though initially average timesteps per 50 episodes is higher at the beginning but it starts to converge to the optimal timestep faster compared to q-learning.

8. Compare the performance of both algorithms in the same stochastic environment

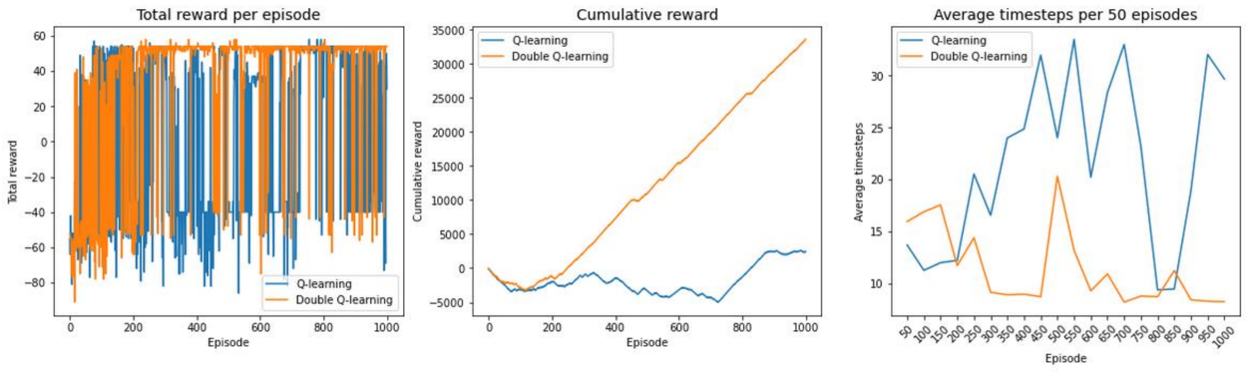

Figure 27: Q-learning vs Double Q-learning in Stochastic Environment

Double q-learning converges faster as compared to q-learning in a stochastic environment. Total reward per episode for double q-learning converges towards the maximum reward more compared to the q-learning algorithm. Also, this can be seen that the cumulative reward for double q-learning is increasing whereas for q-learning is increasing and decreasing. The average number of timesteps per 50 episodes for double q-learning is decreasing with an increase in episodes with some few ups and downs but that is not the case with q-learning.

Note:

- Due to the presence of bad states such as pit and monster being very close to the rewards the total reward per episode graph does not converge to the optimal values as expected.
- Here's why even though the agent tries to follow the optimal path and reach the intermediate reward action in the direction of any of the above-mentioned bad states causes the episode to terminate and give a huge negative reward for the episode. Hence the graph for stochastic environments when developed again might not be the same.
- Also, training the agent again might give different results such as following a different path with the third-best reward than the second (while considering different hyperparameter values), increase in the number of timesteps as the episode increases.

- This can be mitigated either by trial and error method to get the best q-table, proper tuning of each hyperparameter to achieve the goal, or a change in the position of rewards or the bad states so that the effect of stochasticity can be reduced for the scenario mentioned above.