Assignment 1 - Defining and Solving RL Environments

Adeline Grace George

March 7, 2022

1 Assignment Overview

The goal of the assignment is to acquire experience in defining and solving reinforcement learning environments, following OpenAI Gym standards. The first part of the assignment focuses on defining deterministic and stochastic environments that are based on Markov Decision Process. In the second part, two tabular methods (Q-Learning and SARSA in this case) are applied to solve the previously defined environments and the results are recorded and compared.

2 Python

Jupyter Notebook has been used for implementation.

3 Part 1 - Defining RL Environments

Reinforcement learning (RL) is an area of machine learning that focuses on how an agent might act in an environment in order to maximize some given reward. Reinforcement learning algorithms study the behavior of subjects in such environments and learn to optimize that behavior.

3.1 Markov Decision Process

An MDP consists of the following components:

- **agent:** the key decision maker
- **environment:** the surroundings where the agent navigates after interacting with it
- **state:** the current situation in the environment that the agent is positioned in

- **action:** choice of moves that the agent can take given it's current state
- **reward:** a feedback given to the agent (positive, negative or neutral) depending on the action it has taken

In reinforcement learning, the agent interacts with the environment it is placed in to make decisions about the next course of action to take from it's current state, receiving valuable feedback aka, rewards which in turn helps it navigate the environment better.

3.2 Environment

The details of the environment:
Actions: { Up, Down, Left, Right }
States: { Mine, Potion, Bomb, Gem, Goal }
Rewards: { -1, +1, -5, +3, +10 }
Objective: The main objective is to collect maximum rewards and reach the goal state in the least number of steps possible

3.2.1 Deterministic Environment

A deterministic environment is defined such that when an agent takes an action, the environment responds in the expected manner without any variability, ie., the response of the environment is as expected.

3.2.2 Stochastic Environment

A stochastic environment is defined such that when an agent takes an action, the environment responds in the expected manner 90% of the time, but has a different output, 10% of the time. In this environment, when the agent steps on a mine, the environment send back a reward of -1, 90% of the time, but returns a reward of 0, 10% of the time. Likewise, when the agent collects the potion, the environment rewards him with +1 point, 90% of the time and +3 points, 10% of the time.

3.3 Visualization

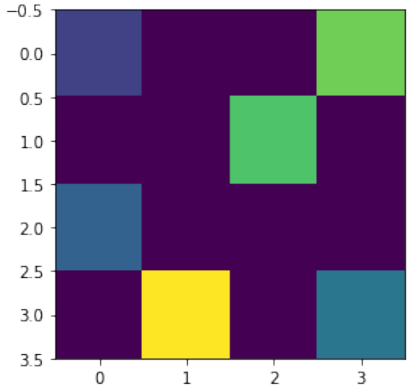

Figure 1: Deterministic Environment

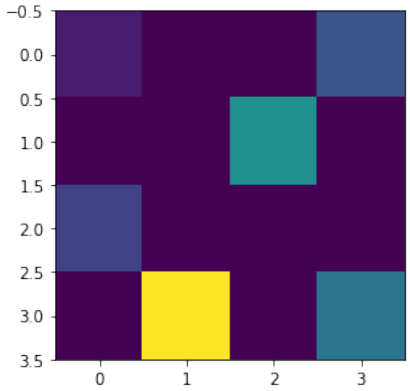

Figure 2: Stochastic Environment

4 Part 2 - Applying Tabular Methods

In this project, Q-learning and SARSA methods have been used to solve the previously defined environments.

4.1 Q-Learning Method

The objective of Q-learning is to find a policy that is optimal in the sense that the expected value of the total reward over all successive steps is the maximum achievable. So, in other words, the goal of Q-learning is to find the optimal policy by learning the optimal Q-values for each state-action pair.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t) \right]$$

The Q-table is updated iteratively as the agent takes actions. The agent has two choices:

- **exploration:** the act of interacting and studying the environment to find out information about the environment
- **exploitation:** the act of making use of information that is already known (Q-table values) about the environment to maximize return

The key to designing a good Q-learning model is striking a balance between exploration and exploitation - it's a trade-off. To get this balance between exploitation and exploration, we use what is called an **epsilon greedy strategy**. With this strategy, we define an exploration rate $\varepsilon = 1$ that we initially set to 1. This exploration rate is the probability that our agent will explore the environment rather than exploit it. With $\varepsilon = 1$, it is 100 percent certain that the agent will start out by exploring the environment. As the agent learns more about the environment, at the start of each new episode, ε will decay by some rate that we set so that the likelihood of exploration becomes less and less probable as the agent learns more and more about the environment. The agent will become "greedy" in terms of exploiting the environment once it has had the opportunity to explore and learn more about it.

4.1.1 Applying Q-Learning to the Deterministic Environment

Epsilon Decay Plot

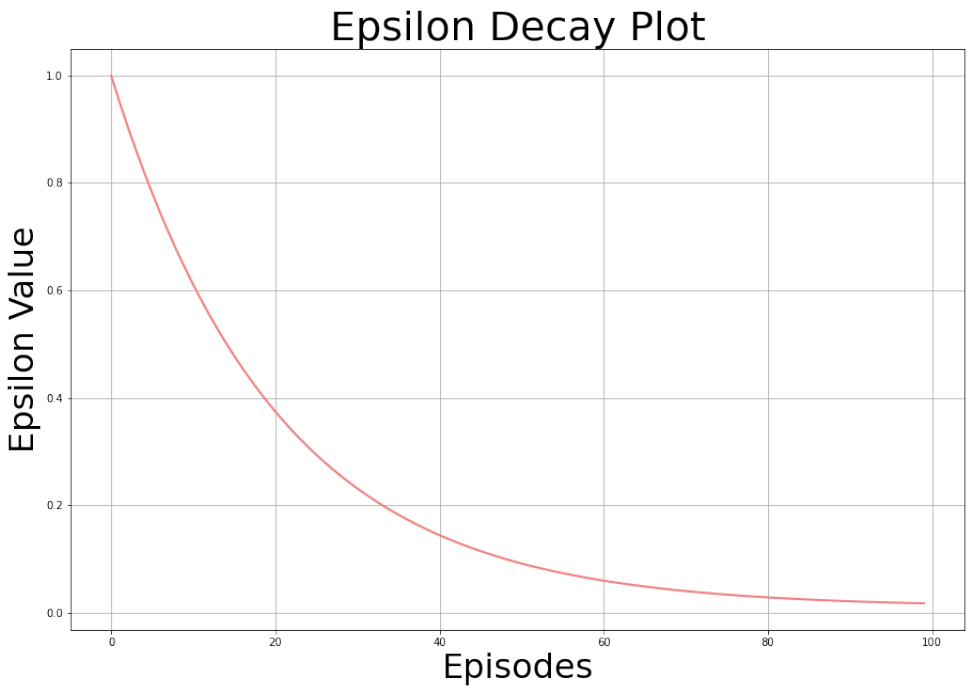

Figure 3: Epsilon Decay Plot

Total Rewards per Episode

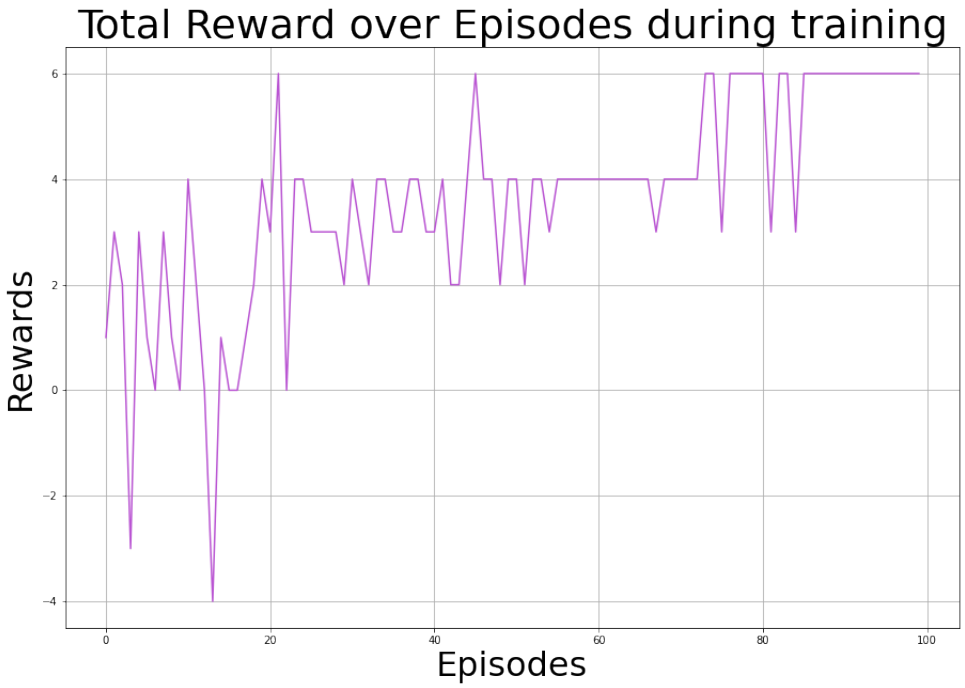

Figure 4: Rewards Plot

In Q-learning algorithm, as the agent continues training in the deterministic environment, the reward plot has a rising trend.

4.1.2 Applying Q-Learning to the Stochastic Environment

Epsilon Decay Plot

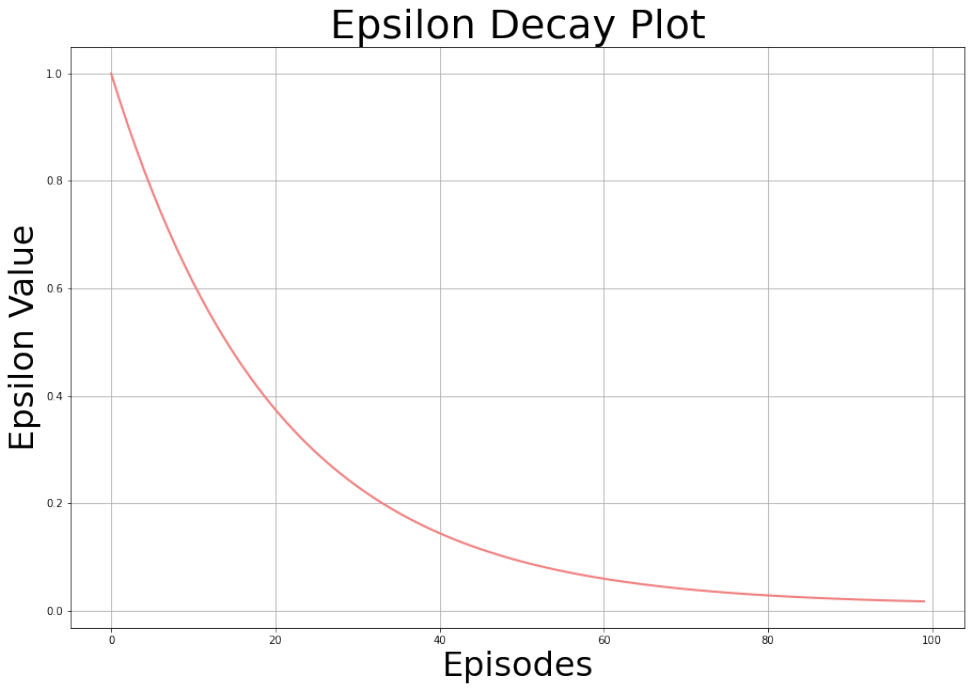

Figure 5: Epsilon Decay Plot

Total Rewards per Episode

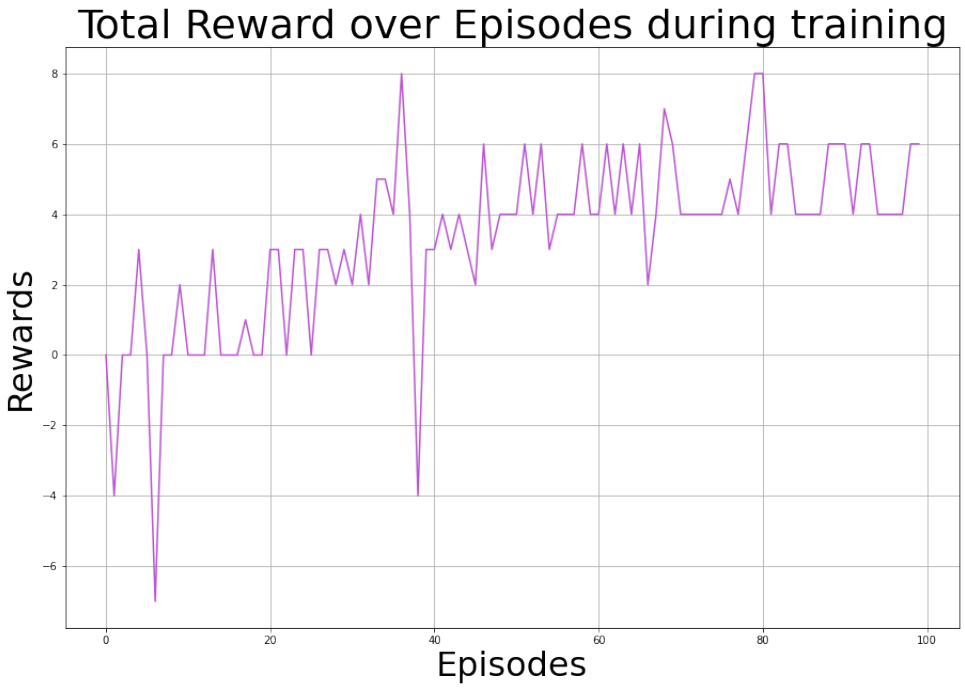

Figure 6: Rewards Plot

In Q-learning algorithm, as the agent continues training in the stochastic environment, the reward plot has a rising trend. The deviations and peaks are more pronounced in this case, owing to the stochastic nature.

4.2 SARSA Method

SARSA stands for S - State, A - Action, R - Reward, S - State, A - Action. This implies that the Q-table values are updated based on the current state of the agent (s_t), the action chosen by the agent given this state (a_t), the reward that is received by taking that action, the state that the action leads the agent to (s_{t+1}) and the next action taken by the agent (a_{t+1}).

$$Q\left(s_t, a_t\right) \leftarrow Q\left(s_t, a_t\right) + \alpha\left[r_{t+1} + \gamma Q\left(s_{t+1}, a_{t+1}\right) - Q\left(s_t, a_t\right)\right]$$

4.2.1 Applying SARSA to the Deterministic Environment

Epsilon Decay Plot

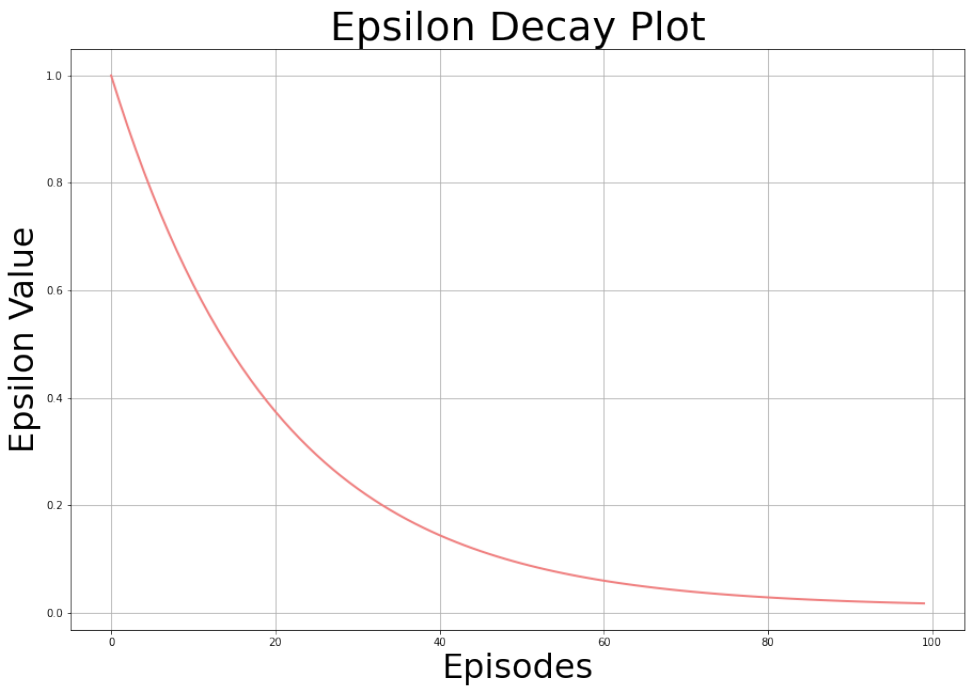

Figure 7: Epsilon Decay Plot

Total Rewards per Episode

Figure 8: Rewards Plot

In SARSA algorithm, as the agent continues training in the deterministic environment, the reward plot has a rising trend.

4.2.2 Applying SARSA to the Stochastic Environment

Epsilon Decay Plot

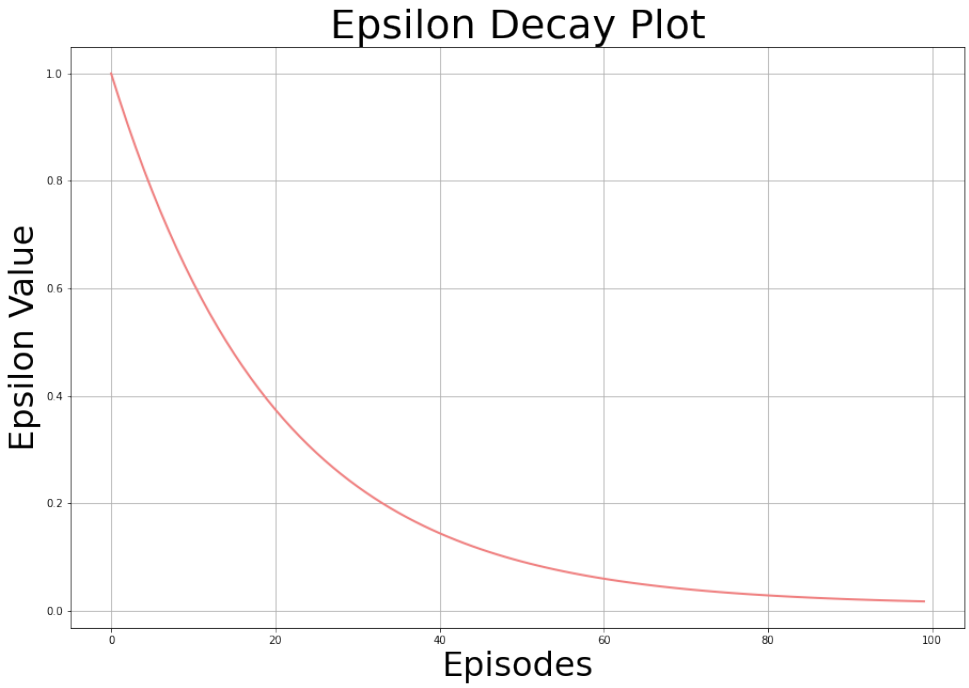

Figure 9: Epsilon Decay Plot

Total Rewards per Episode

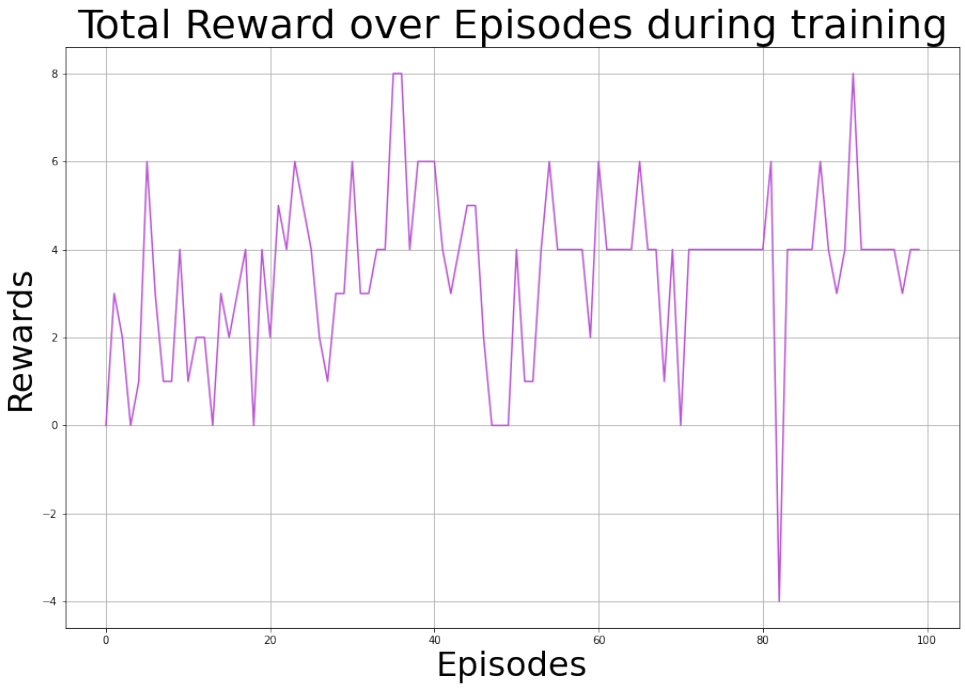

Figure 10: Rewards Plot

In SARSA algorithm, as the agent continues training in the stochastic environment, the reward plot has a rising trend, but with higher variations and peaks, owing to the stochastic nature of the environment.

4.3 Evaluation Results

SARSA Plot - Green
Q-Learning Plot - Violet

4.3.1 Deterministic Environment

Figure 11: Evaluation Result Plot

SARSA method has a higher reward than Q-learning for identical hyper-parameters in the deterministic environment.

4.3.2 Stochastic Environment

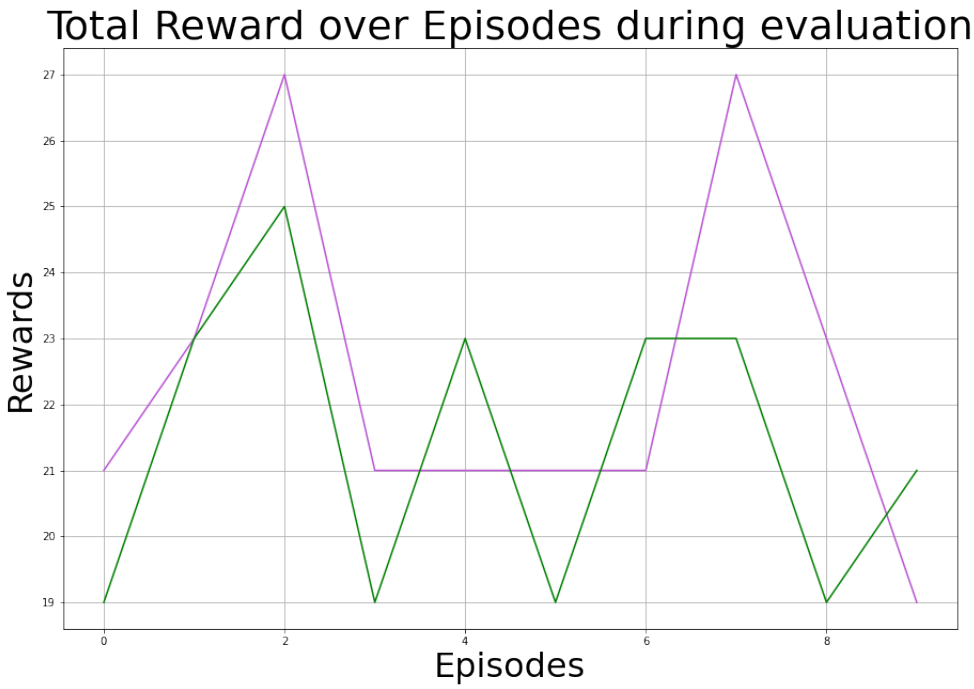

Figure 12: Evaluation Result Plot

Q-Learning method has a higher reward than SARSA for identical hyper-parameters in the stochastic environment.

5 Hyper-Parameter Tuning

5.0.1 Gamma Tuning

Gamma is the "discount factor" used to maximize the effect of immediate rewards on the agent's learning process. An optimum value is necessary to help the agent choose the optimum and quickest and most rewarding path to reach the goal.

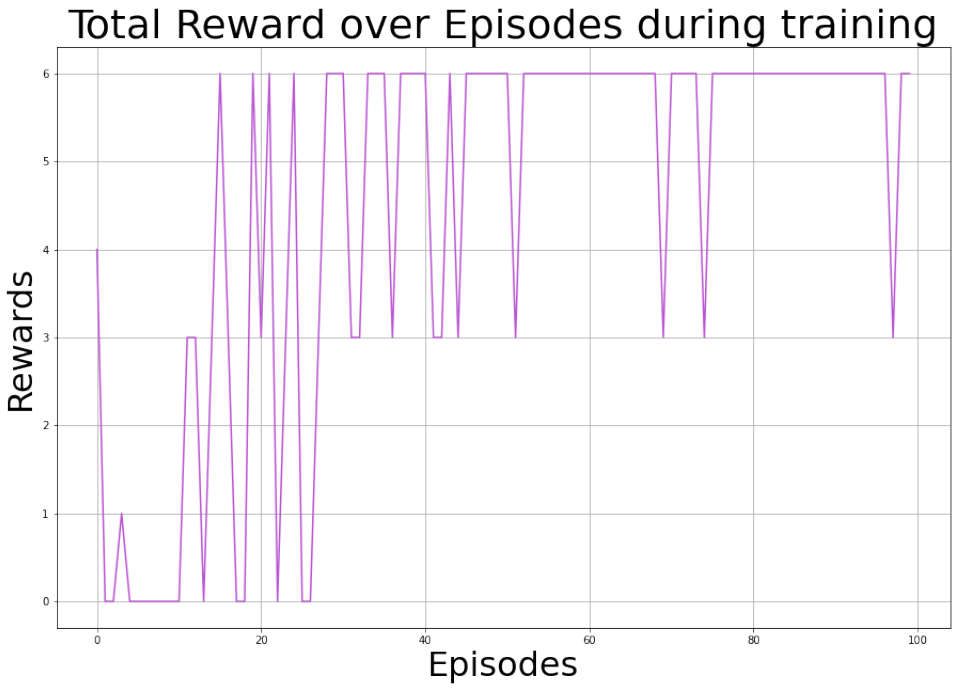

Figure 13: Reward plot with gamma = 0.4

Gamma is way too low in this case as is evident in the plot. The agent is not able to learn well.

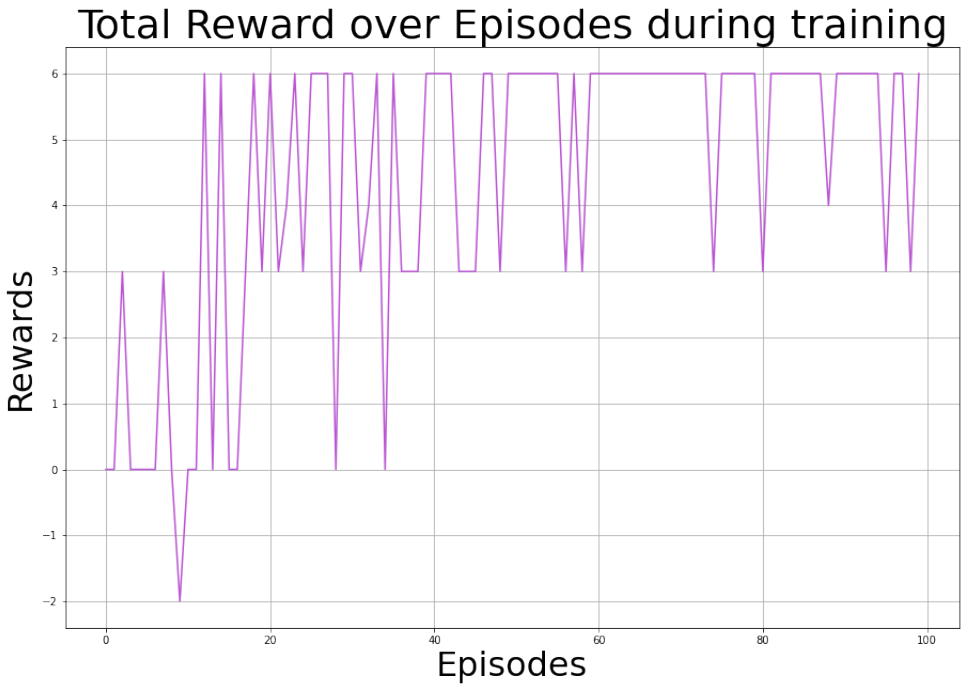

Figure 14: Reward plot with gamma = 0.7

Gamma is on the lower side in this case as is evident in the plot. The agent is still not able to learn well.

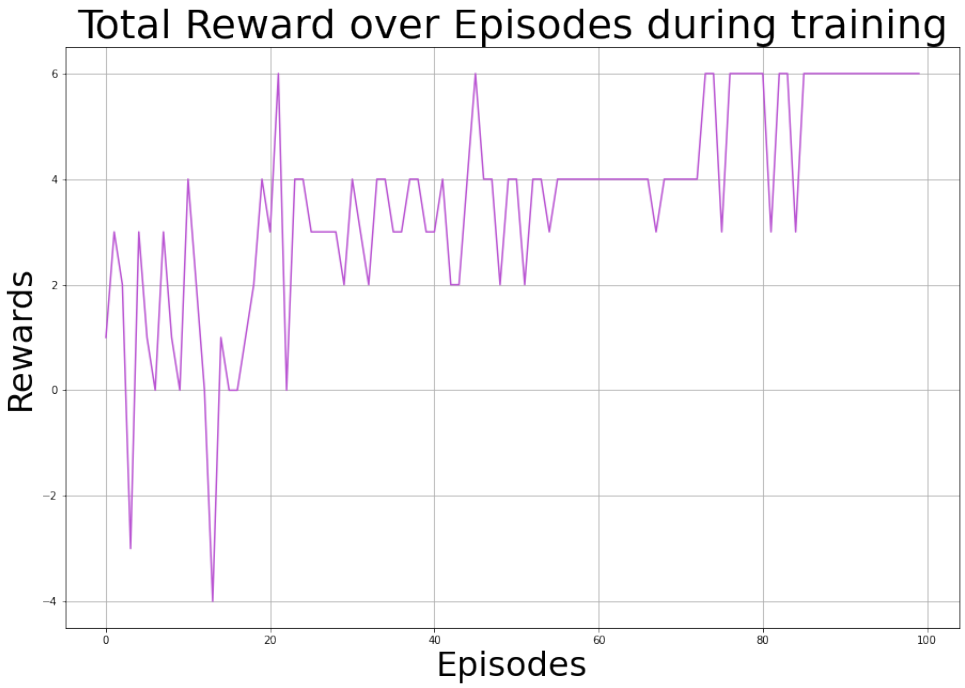

Figure 15: Reward plot with gamma = 0.85

Gamma value is optimum and the agent is able to maximize it's rewards for the given environment. This gamma value was chosen in the final implementation.

5.0.2 Epsilon decay rate

The epsilon decay rate is crucial to help the agent strike a balance between exploration and exploitation. Choosing an optimum value is key to help the agent be aware of it's surrounding quickly and also apply it by making greedy decisions to get to the goal as soon as possible.

Figure 16: Reward plot with epsilon decay rate = 0.1

The epsilon decay rate is too high for this agent and model. This will cause the agent to stop exploring too soon.

Figure 17: Reward plot with epsilon decay rate = 0.0005

The epsilon decay rate is too low in this case. The agent will continue exploring instead of applying the knowledge gained to make greedy decisions until much later, thereby increasing the time taken to maximize it's reward and to reach the goal.

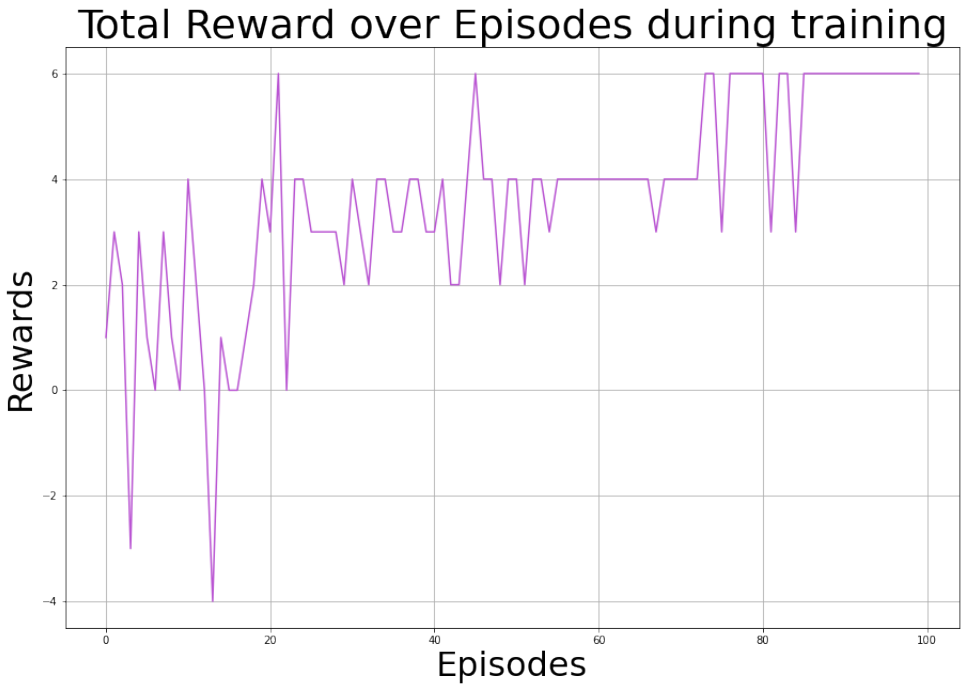

Figure 18: Reward plot with epsilon decay rate = 0.05

This epsilon decay rate is ideal and allows the agent to strike a balance between exploring the environment and exploiting it. This value has been fixed in the final implementation.

6 Safety in AI

This environment is defined in such a way that the agent is only able to explore actions within the scope and boundaries of the environment and is forbidden from venturing out. This is done by limiting the agent's position (by using numpy.clip function) to the maximum index size of the environment.

Negative rewards for the bomb (-5) and and mine (-1) are implemented to reinforce to the agent that these need to be avoided.