CSE 546 Reinforcement Learning (Spring 2024)
Assignment -1 Final Report

Name: Likhit Sastry
Person ID: 50545758

Part – 1 Defining RL Environments

Deterministic Environment:

The environment is a deterministic grid world based on Door dash delivery agent who must deliver food to the customer. On his way the agent is met with obstacles and rewards. The main objective is to deliver the food to the customer.
Actions available: The agent can move in 4 directions; Up, Down, Left and Right
Rewards available: There are 4 different types of rewards/ penalties.

1. The agent can be pulled over by a police officer for speeding and get a ticket. It is modelled by obtaining a negative reward of -1.
2. The agent can obtain free food on his way to deliver food. It is modelled by getting a reward of +1
3. The agent can also obtain free fuel on the way. It is modelled by a reward of +0.5
4. Finally, once the agent delivers the food, it gets a reward of +50.

States available: The grid size is 4x4, and hence there are 16 states in which the agent can be at any given point in time.

Stochastic Environment:

The environment is a stochastic grid world based on Door dash delivery agent who must deliver food to the customer. On his way the agent is met with obstacles and rewards. The main objective is to deliver the food to the customer. Stochasticity is introduced by providing transition probabilities for a given state action pair.

Actions available: The agent can move in 4 directions; Up, Down, Left and Right
Rewards available: There are 4 different types of rewards/ penalties.
1. The agent can be pulled over by a police officer for speeding and get a ticket. It is modelled by obtaining a negative reward of -1.
2. The agent can obtain free food on his way to deliver food. It is modelled by getting a reward of +1.
3. The agent can also obtain free fuel on the way. It is modelled by a reward of +0.5
4. Finally, once the agent delivers the food, it gets a reward of +50.

States available: The grid size is 4x4, and hence there are 16 states in which the agent can be at any given point in time.

Deterministic vs Stochastic Environments:

In a deterministic environment, the outcome of an action taken by the agent is entirely predictable. This means that a given state-action pair, the next state and corresponding reward are always the same.

In a stochastic environment, the outcome of an action is subject to randomness or uncertainty. The randomness can be introduced in various ways, such as probabilistic state transitions. As a result, the same action taken in the same state might lead to different outcomes each time.

Environment Visualization:

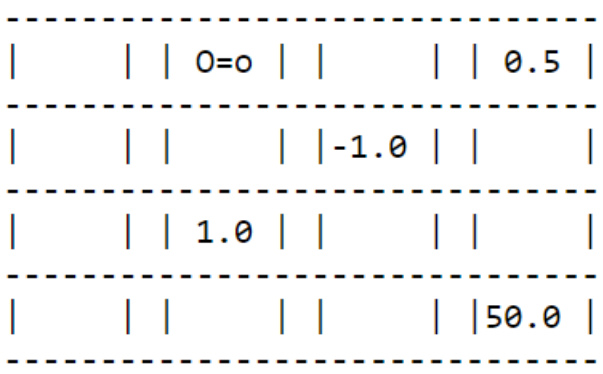

Fig1. Visualization of the environment

The above image is a simple visualization of an agent in the grid environment. Here we the agent is represented by sort of a car symbol(“O=o”). The rewards/ penalties are represented by constant values. The goal state is at the bottom right corner of the grid.

It is worth noting that, the environment is modelled in way that rewards stay alive at their respective positions even if the agent attains the reward at some point during the episode.

Safety in AI:

It is essential that the agent remains within the defined grid and does not move into any invalid state by taking a certain action. In order to make sure that the agent always ends up in a valid state, the *np.clip()* function is used while modelling the environment. The function accepts an interval and the values that outside the interval are clipped to the interval edges.

Part – 2 Applying Tabular Methods

Q-learning applied to agent in deterministic environment: In this case the agent is trained for 1100 episodes in a deterministic grid environment with a learning rate of 0.10 and discount factor of 0.98.

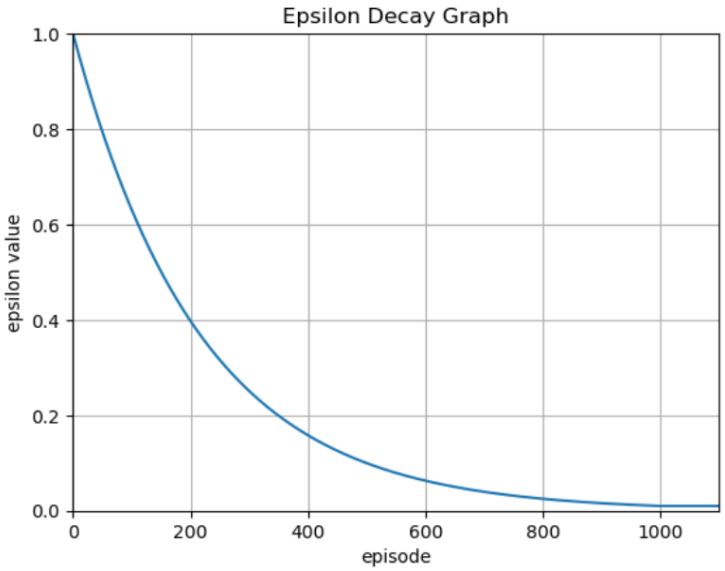

Fig2.1 epsilon decay graph

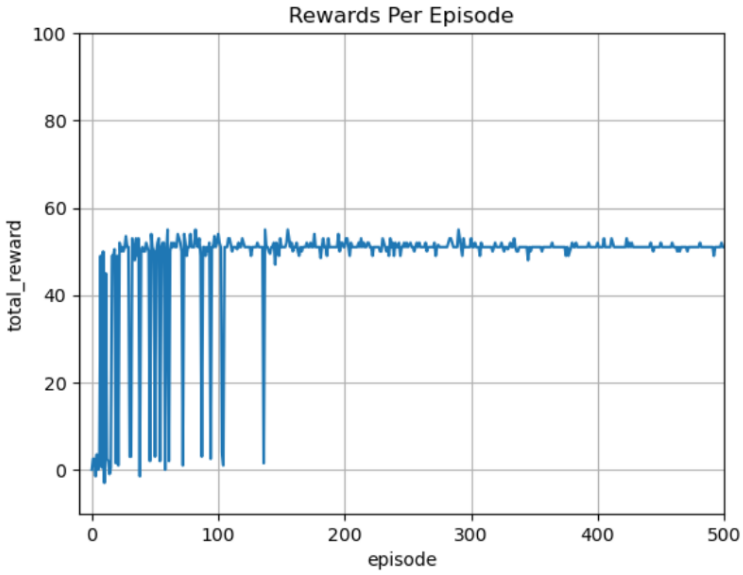

Fig2.2 rewards per episode

Results: When Q-learning is applied to the agent in a deterministic environment, the agent finds the optimal path within 7 timesteps and reaches the goal state using the optimal policy.

Q-learning applied to agent in stochastic environment: In this case the agent is trained for 1100 episodes in a deterministic grid environment with a learning rate of 0.10 and discount factor of 0.98.

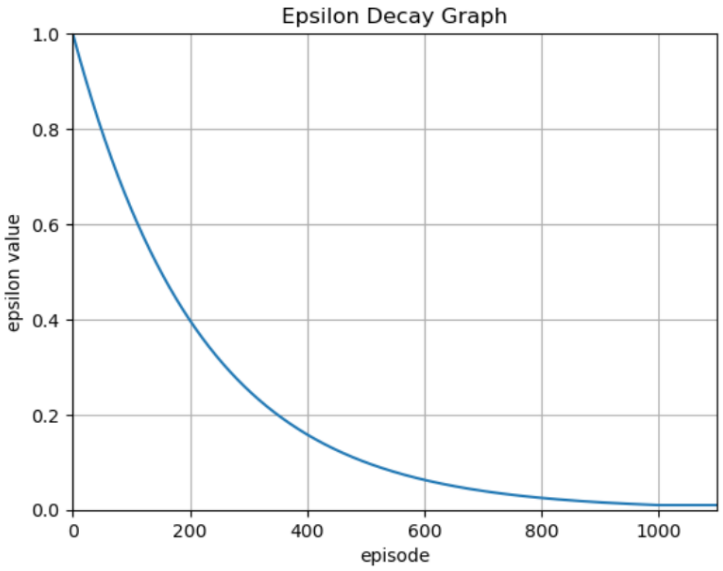

Fig2.3 epsilon decay graph

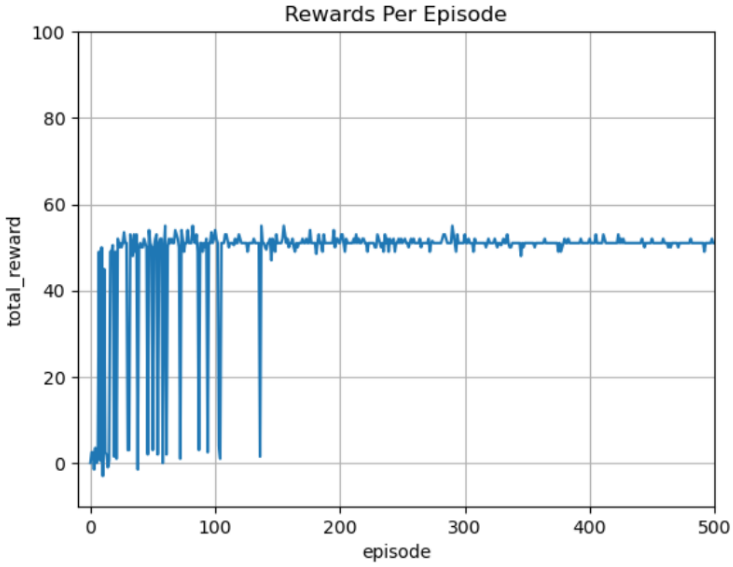

Fig2.4 rewards per episode

Results: When Q-learning is applied to the agent in stochastic environment, the agent finds the optimal path within few timesteps and reaches the goal state using the optimal policy

SARSA applied to agent in deterministic environment: In this case the agent implements a SARSA algorithm to updates its estimates formed within a deterministic environment, with a learning rate of 0.10 and discount factor of 0.98.

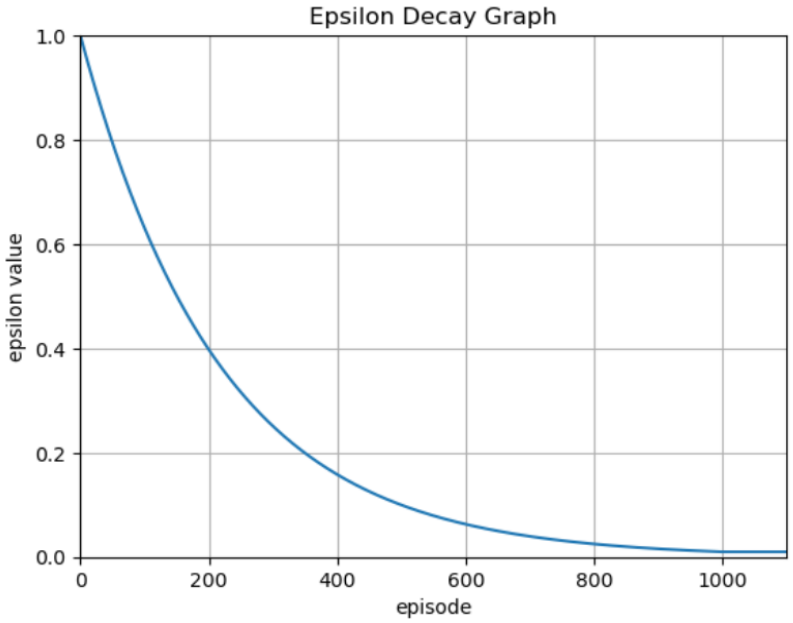

Fig2.5 epsilon decay graph

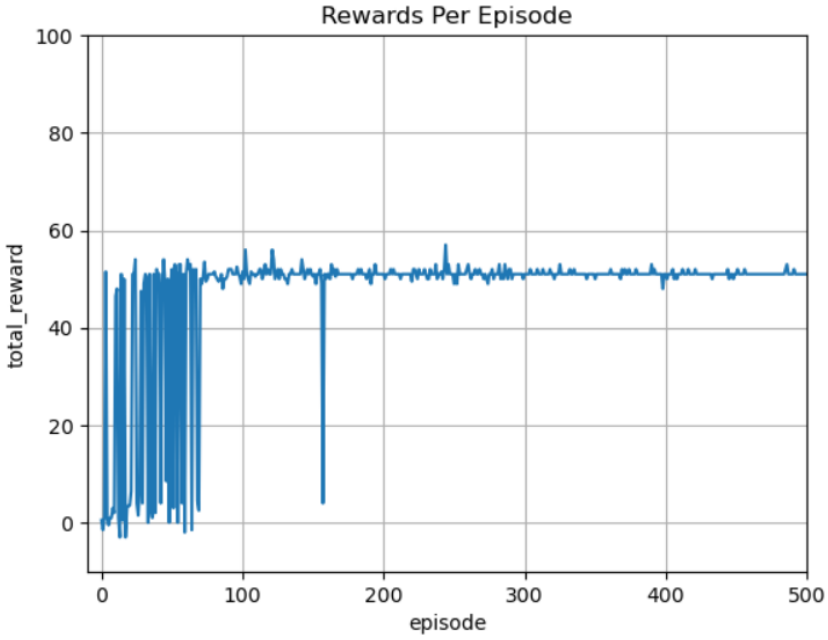

Fig2.6 rewards per episode

Results: When SARSA algorithm is applied to the agent in deterministic environment, the agent finds the optimal path and reaches the goal state using the optimal policy.

SARSA applied to agent in stochastic environment: In this case the agent implements a SARSA algorithm to updates its estimates formed within a stochastic environment, with a learning rate of 0.10 and discount factor of 0.98.

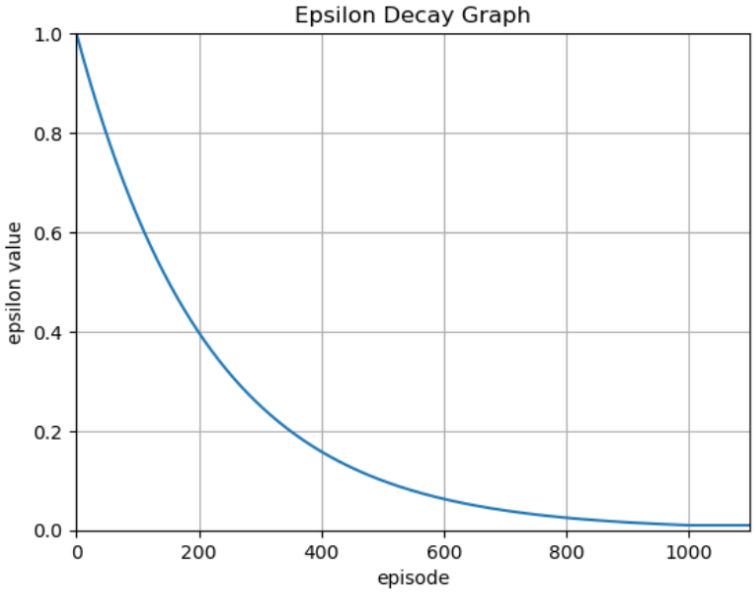

Fig2.7 epsilon decay graph

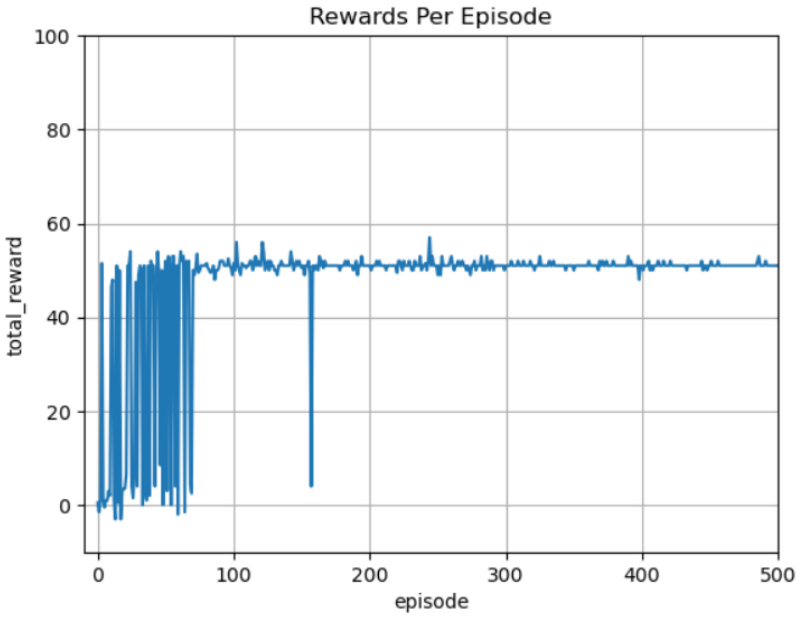

Fig2.8 rewards per episode

Results: When Q-learning is applied to the agent in deterministic environment, the agent finds the optimal path within 7 timesteps and reaches the goal state using the optimal policy

Evaluation of agent implementing Q-learning chooses the greedy actions in deterministic environment:

```
action: moved right
new_state: (2, 2)
reward obtained: 0
-------------------------------
|     | |     | |     | | 0.5 |
-------------------------------
|     | |     | |-1.0 | |     |
-------------------------------
|     | | 1.0 | | O=o | |     |
-------------------------------
|     | |     | |     | |50.0 |
-------------------------------
action: moved down
new_state: (3, 2)
reward obtained: 0
-------------------------------
|     | |     | |     | | 0.5 |
-------------------------------
|     | |     | |-1.0 | |     |
-------------------------------
|     | | 1.0 | |     | |     |
-------------------------------
|     | |     | | O=o | |50.0 |
-------------------------------
action: moved right
new_state: (3, 3)
reward obtained: 50
-------------------------------
|     | |     | |     | | 0.5 |
-------------------------------
|     | |     | |-1.0 | |     |
-------------------------------
|     | | 1.0 | |     | |     |
-------------------------------
|     | |     | |     | | O=o |
-------------------------------
```

Fig 2.9 Evaluation of the agent

Comparing the performance of Q-learning and SARSA in a Deterministic environment:

Q-learning vs SARSA in Deterministic Environment

```
In [59]: reward_dynamic((varQD, varSARSAD),"Q-learning", "SARSA", "Cumulative reward in Deterministic Environment", 1000)
```

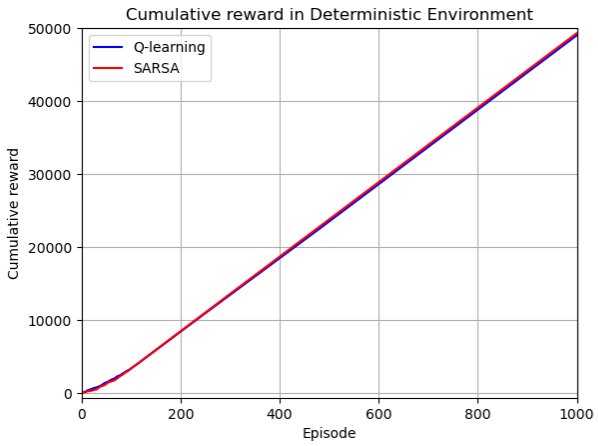

Fig 2.10 Reward dynamics of Q-learning vs SARSA

We can see that the agent almost obtains a similar reward structure in both the cases. In both the cases the rewards tend to increase over the episodes which is a good sign that the agent is learning

Comparing the performance of Q-learning and SARSA in a Stochastic environment:

Q-learning vs SARSA in Stochastic Environment

```
In [60]: reward_dynamic((varQS, varSARSAS),"Q-learning", "SARSA", "Cumulative reward in Stochastic Environment", 1000)
```

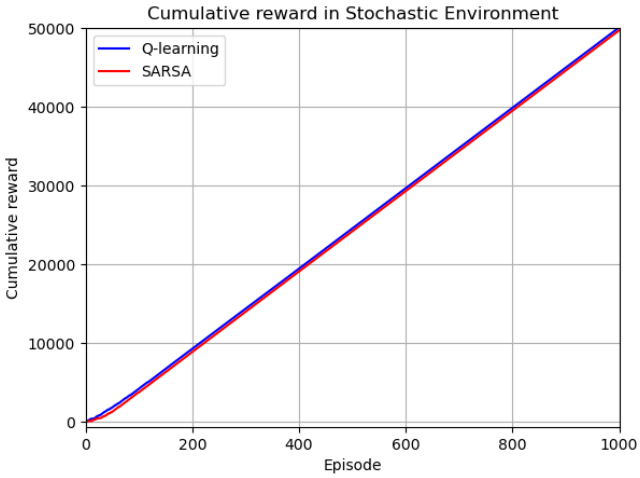

Fig 2.11 Reward Dynamics of Q-learning vs SARSA

We can see that the agent almost obtains a similar reward structure in both the cases. In both the cases the rewards tend to increase over the episodes which is a good sign that the agent is learning

Q-learning tabular method:

1. The objective of Q-learning,a reinforcement learning algorithm is to learn the optimal action-value function (Q-function), denoted as Q(s, a), which represents the expected cumulative reward of taking action 'a' in state 's' and then following the optimal policy.
2. The Q-Learning Algorithm is as follows:
   Initialize Q-values: Initialize Q-values for all state-action pairs to zero.
   Exploration vs. Exploitation: The agent explores the environment by utlizing the epslion greedy approach which utilizes the eploration-exploitation dilemma.
   Update Q-values: Use the Q-learning update rule to update Q-values
   Once the Q-values are learned, the optimal policy can be derived by selecting the action with the highest Q-value for each state.
3. Update rule for Q-learning is: Q(s, a) = (1 - α) * Q(s, a) + alpha* [r + gamma * max(Q(s', a'))]
   Q(s, a) is the Q-value of state-action pair (s, a)
   alpha is the learning rate, controlling the rate of learning
   r is the reward received after taking action 'a' in state 's'
   gamma is the discount factor
   max(Q(s', a')) is maximum Q-value among the possible actions in the next state 's'.

SARSA tabular method:

1. The SARSA (State-Action-Reward-State-Action) algorithm, which is an on-policy reinforcement learning algorithm. The Pseudo code is as follows
   Initialize Q-table: Initialize the Q-values for all state-action pairs in the Q-table.
   Initialize parameters: Set the value of epsilon, the learning rate and the discount factor.
   For each episode:
   Initialize state: Set the initial state of the environment.
   Choose action: Use epsilon-greedy policy to select the action for the current state.
   While episode is not terminated:
   Take action: Execute the chosen action in the environment.
   Observe reward and next state: Receive the reward and observe the next state.
   Choose next action: Use epsilon-greedy to select the next action for the next state.
   Update Q-value: Update Q-value of the current state-action pair using update rule
   Update state and action: Set the current state & action to the next state & action
2. Repeat the process for a predefined number of episodes or until convergence.
3. The update rule for SARSA is as follows:
   Q(s, a) = Q(s, a) + alpha * [r + gamma * Q(s', a') - Q(s, a)]
   Q(s, a) is the Q-value of the current state-action pair.
   alpha is the learning rate.
   r is the immediate reward received after taking action a in state s.
   gamma is the discount factor.
   Q(s', a') is the Q-value of the next state-action pair.

Part – 3 Stock Trading Environment

After training, the agent, based on the Q-learning algorithm learns the optimal actions and does so efficiently. Below are the evaluation results:

Note: The agent was trained for 1100 episodes with a learning rate of 0.10 and gamma of 0.99

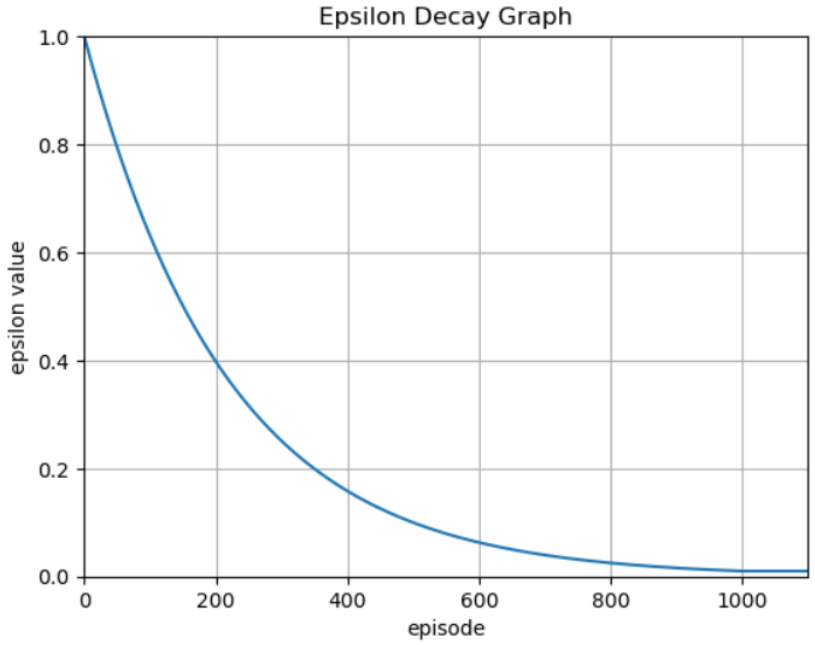

Fig 1. Epsilon decay graph

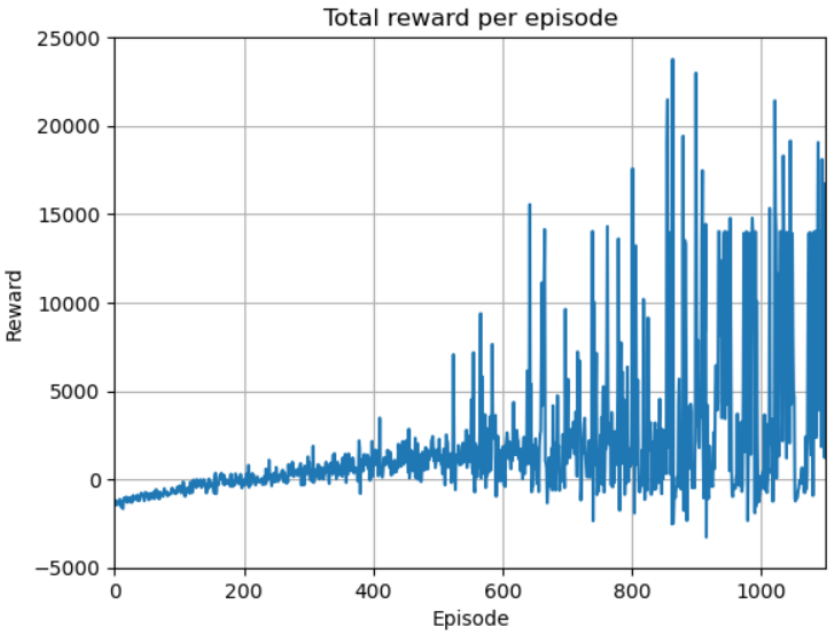

Fig 2. Total rewards per episode

Evaluation results for the stock trading agent: After training, the agent is evaluated to see how it performs based on its experience. In this case, the performance can be measured by plotting the agent's total account value over time.

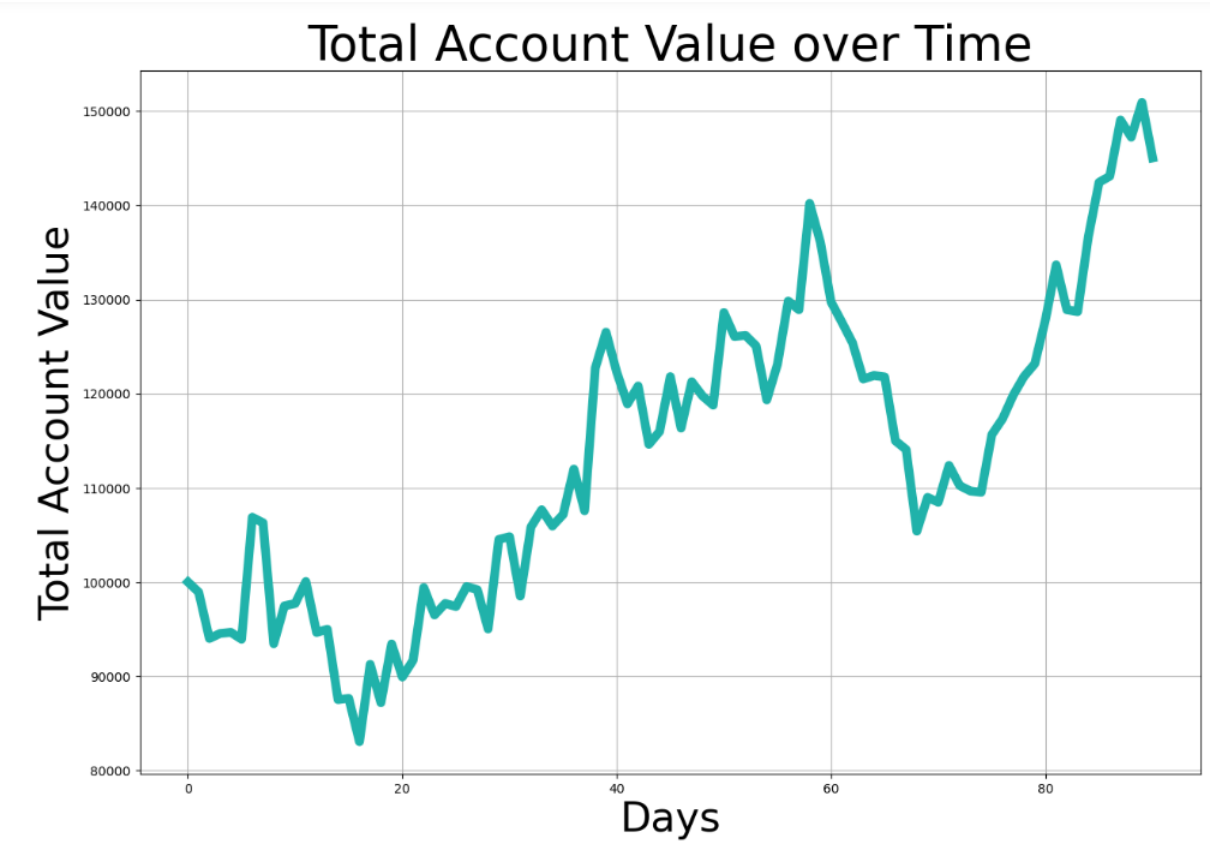

Fig 3. Account value of agent over time

The agent's total account value begins with a capital of $ 100,000 and eventually increases as the agent makes better decisions over time to increase its value.

Good reward function: A good reward function is the one that incentivizes good actions and penalizes bad actions with more weightage. The distribution of rewards is crucial for the agent to learn the sequence of decisions.

References used: Prof. Alina's class slides and video demonstration