Assignment -3
Deep Reinforcement Learning (Q Learning)

Nikhil Bandari (Person Number: 50418501) 10 December 2021

1 Assignment Overview:

The goal of the assignment is to learn the trends in stock price and perform a series of trades over a period of time and end with a profit. In each trade we buy/sell/hold. By using Q-Learning algorithm training an agent to learn the trends in stock price and perform a series of trades.

2 Data Pre-Processing:

By using NVDA data set for historical stock price from last 5 years, comprises with1258 entries starting 10/27/2016 to 10/26/2021. The features include information such as the price at which the stock opened, the intraday high and low, the price at which the stock closed, the adjusted closing price and the volume of shares traded for the day

3 Python Editor

I have used Google Colab IDE for implementation and Comparing results.

4 Environment Structure:

Input parameters:

1. **file_path:** Path of the CSV file containing the historical stock data.
2. **train**: - Boolean indicating whether the goal is to train or test the performance of the agent.
3. **number_of_days_to_consider** = Integer representing whether the number of days the for which the agent considers the trend in stock price to make a decision

Reset method: This method resets the environment and returns the observation.

1 **Returns**: observation: - Integer in the range of 0 to 3 representing the four possible observations that the agent can receive. The observation depends upon whether the price increased on average in the number of days the agent considers, and whether the agent already has the stock or not.
Step method: This method implements what happens when the agent takes the action to Buy/Sell/Hold. Input parameter: action: - Integer in the range 0 to 2 inclusive.

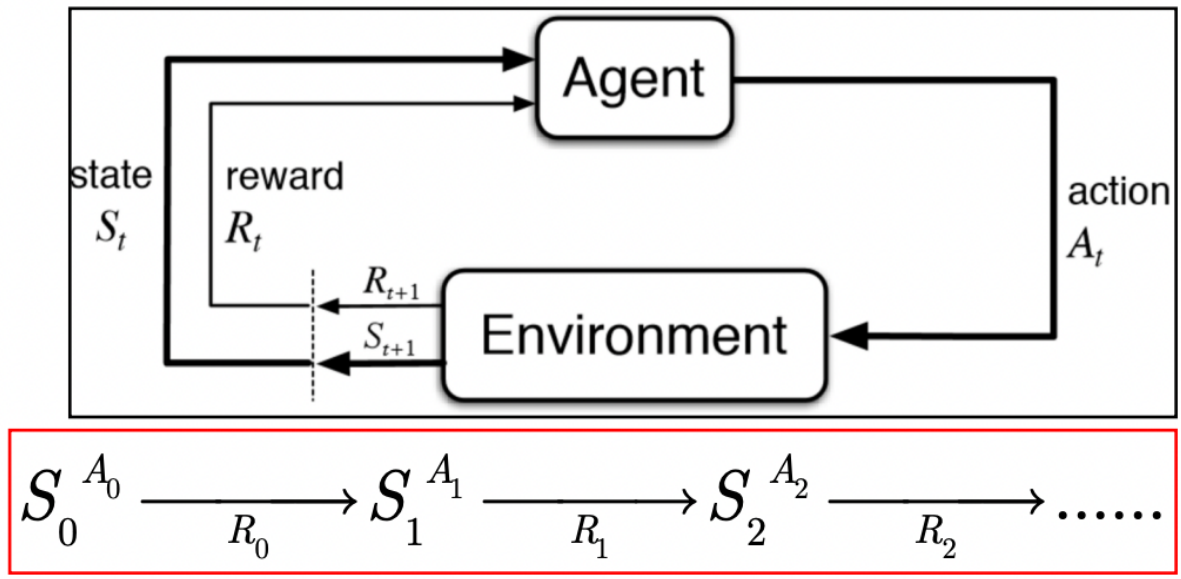
Returns:
1. observation: - Integer in the range of 0 to 3 representing the four possible observations that the agent can receive. The observation depends upon whether the price increased on average in the number of days the agent considers, and whether the agent already has the stock or not.
2. reward: - Integer/Float value that's used to measure the performance of the agent.
3. done: - Boolean describing whether or not the episode has ended.
4. info: - A dictionary that can be used to provide additional implementation information.

Render method: This method renders the agent's total account value over time. Input parameter: mode: 'human' renders to the current display or terminal and returns nothing.

5 **Train the Model**:

Agent exists in environment with set of states S. It can perform any of a set of actions A – Performing action At in state St receives reward Rt.

Agent's task is to learn control policy π :
SàA • That maximizes expected sum of rewards – with future rewards discounted exponentially.



The state/observation description as mentioned in the assignment problem statement:
"observation: - Integer in the range of 0 to 3 representing the four possible observations that the agent can receive. The observation depends upon whether the price increased or decreased on average in the number of days the agent considers, and whether the agent already has the stock or not."

#Observation vector:
observation = [price_increase, price_decrease, stock_held, stock_not_held]

```
if np.array_equal(observation, [1, 0, 0, 1]):
    observation = 0
if np.array_equal(observation, [1, 0, 1, 0]):
    observation = 1
if np.array_equal(observation, [0, 1, 0, 1]):
    observation = 2
if np.array_equal(observation, [0, 1, 1, 0]):
    observation = 3
```

The four observations represent the following scenarios:
0. Price of the stock has increased and the agent doesn't hold any shares.
1. Price of the stock has increased and the agent holds shares.
2. Price of the stock has decreased and the agent doesn't hold any shares.
3. Price of the stock has decreased and the agent holds shares.

Differnet function is QLearning :

Init :

This method instantiates the Q-learning parameters. Our agent will randomly select its action at first by a certain percentage, called 'exploration rate' or 'epsilon'. This is because at first, it is better for the agent to try all kinds of things before it starts to see the patterns. Select a random uniform number.

Train :

when we call reset, we initialize the environment with a fresh episode. This allows us to effectively run through episodes (only needing to call reset at the beginning of an episode), but, more importantly, reset() returns the environment's initial state.

The step method accepts an action as a parameter (which, for this example, is an integer in [0, 3]), processes the action, and returns the new state, the reward for performing the action, and a boolean indicating if the run is over.

Evaluate:

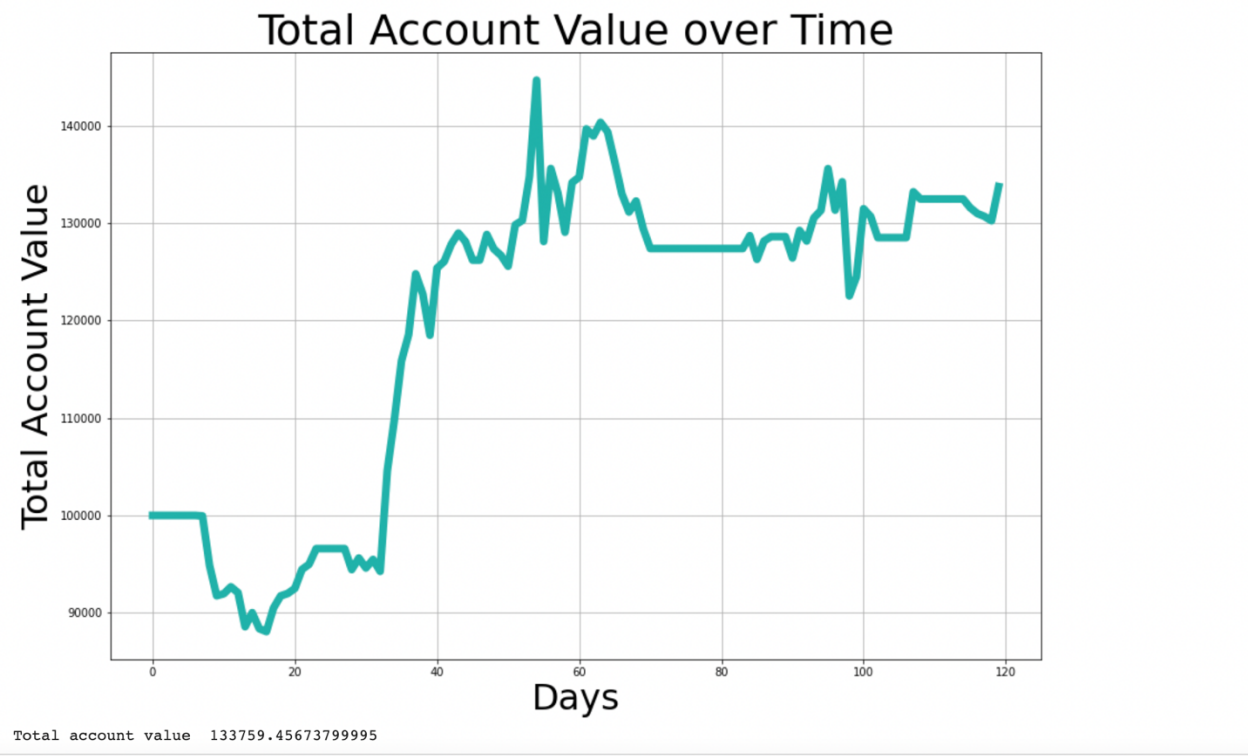
This method evaluates the trained agent's performance. Evaluate the trained agent's performance by selecting only the greedy/best action in each state.

Plot :

After training and evaluating the agent performance , Plotting all the graphs for epsilon and total rewards and epsilon decay .

Results :

After training the data, we have evaluated agent performance where the agent chooses only greedy actions from the learnt policy , and plotted agent's account over time with value : 133759.45




```
Total account value  133759.45673799995
```

Displayed graphs for epsilon decay and total reward per episode.