Assignment 3 - Q-Learning

Baasit Sharief

December 11, 2021

1 Dataset and Understanding the problem statement

1.1 Introduction

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 can either buy/sell/hold. We start with an investment capital of $100,000$ USD and the performance is measured as a percentage of the return on investment.

We use Q-Learning algorithm for reinforcement learning to train an agent to learn the trends in stock price and perform a series of trades. The purpose of this assignment is to understand the benefits of using reinforcement learning to solve the real world problem of stock trading.

1.2 Dataset

We have been given a dataset on the historical stock price for Nvidia for the last 5 years. The dataset has 1258 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.

1.3 Environment

The environment class is initialized using OpenAI gym Env class involving the basic functionality like reset, step, init and sample method which can be used for exploring. The environment divides the stock value dataset into an $train : test$ split of 80:20. We start with an investment capital of 100000 USD. We have an action space of 3 Discrete actions, namely,

1. Buy, $action = 0$
2. Sell, $action = 1$
3. Hold, $action = 2$

We also have an observation space of size 4,

1. Stock price increase and stock not held, $observation = 0, observation\_vector = [1, 0, 0, 1]$
2. Stock price increase and stock held, $observation = 1, observation\_vector = [1, 0, 1, 0]$
3. Stock price decrease and stock not held, $observation = 2, observation\_vector = [0, 1, 0, 1]$
4. Stock price decrease and stock held, $observation = 3, observation\_vector = [0, 1, 1, 0]$

The environment has a function called render which plots the Total Account Value over days. The goal of our task is to maximize the total account value accumulated.

2 Q-Learning

2.1 Introduction

$Q - Learning$ is a model free algorithm which requires a single look-up table of rows equal to the size of the observation space and columns equal to the size of the action space. For any Finite Markov Decision Process (FMDP), Q-Learning tries to find an optimal policy to maximize the expected value of total rewards for any or all successive steps from a given current step. Q here refers to the function that the algorithm tries to compute - the expected reward for a given state and the given action.

Algorithm 1 Q-Learning: Learn Function

Require:
States $\mathcal{X} = \{1, \ldots, n_x\}$
Actions $\mathcal{A} = \{1, \ldots, n_a\}$, $A : \mathcal{X} \Rightarrow \mathcal{A}$
Reward function $R : \mathcal{X} \times \mathcal{A} \to R$
Black-box (probabilistic) transition function $T : \mathcal{X} \times \mathcal{A} \to \mathcal{X}$
Learning rate $\alpha \in [0, 1]$, typically $\alpha = 0.1$
Discounting factor $\gamma \in [0, 1]$
procedure QLEARNING($\mathcal{X}$, A, R, T, α, γ)
  Initialize $Q : \mathcal{X} \times \mathcal{A} \to R$ arbitrarily
  while Q is not converged **do**
    Start in state $s \in \mathcal{X}$
    while s is not terminal **do**
      Calculate π according to Q and exploration strategy (e.g. $\pi(x) \leftarrow \arg\max_a Q(x, a)$)
      $a \leftarrow \pi(s)$
      $r \leftarrow R(s, a)$ ▷ Receive the reward
      $s' \leftarrow T(s, a)$ ▷ Receive the new state
      $Q(s, a) \leftarrow (1 - \alpha) \cdot Q(s, a) + \alpha \cdot (r + \gamma \cdot \max_{a'} Q(s', a'))$
      $s \leftarrow s'$
    end while
  end while
return Q
end procedure

An episode of the algorithm ends when state s_{t+1} is a final or terminal state. However, Q-learning can also learn in non-episodic tasks (as a result of the property of convergent infinite series). If the discount factor is lower than 1, the action values are finite even if the problem can contain infinite loops.

2.2 Learning the Q-Table: Hyperparameter setup and observations

For the exploration and exploitation strategy during learning, I have implemented ε-greedy policy as follows:

Algorithm 2 ε-greedy policy

Require: state $s \in$ States $\mathcal{X}$, $\varepsilon \in [0, 1]$, $\mu(0, 1)$ ▷ $\mu(0, 1)$ represent a random generator function from a uniform distribution between (0,1)
procedure ε-GREEDY CHOICE(s, ε, $\mu(0, 1)$)
  if $\mu(0, 1) < \varepsilon$ **then**
    $s' \leftarrow sample(\mathcal{X})$
  else
    $s' \leftarrow \arg\max_a Q(x, a)$
  end if
end procedure

I have used epsilon decay after every 10 episode on a condition that the mean of the last 10 episodic rewards is less than one before that. The pseudocode is defined on the next page.

Algorithm 3 ε-decay

Require: λ, ε, *prev_rewards*, *cur_rewards* ▷ *cur_rewards* and *prev_rewards* represent the mean of rewards of the last 10 episodes and the one before that respectively

procedure ε-DECAY(λ, ε, *prev_rewards*, *cur_rewards*)

if *prev_rewards* > *cur_rewards* **then**

$\varepsilon \leftarrow \varepsilon * \lambda$

else

prev_rewards ← *cur_rewards*

end if

end procedure

For the training setup which takes each day as a step, we learn the Q-table for 5000 episode with $\alpha = 0.1$, $\gamma = 0.9$, $\varepsilon = 1$, $\lambda = 0.99$ which is the rate of decay of ε. Given below are the observations:

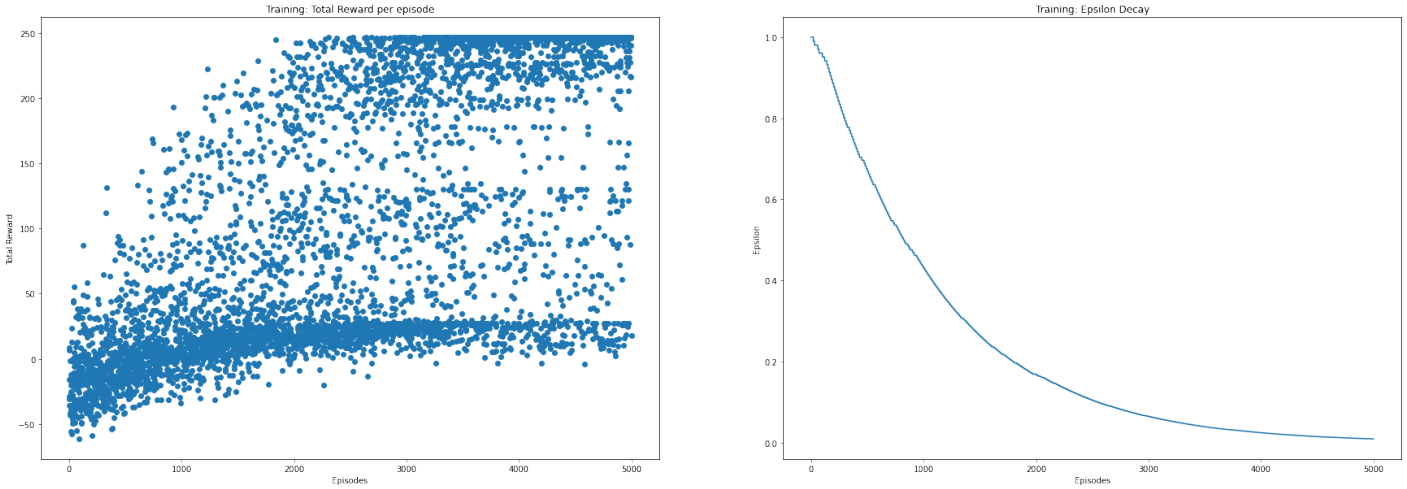

Figure 1: Training Curves

2.3 Evaluation

For the evaluation, we take a step for each 10 days instead of 1 day. We start with the same amount investment capital and always choose the greedy step based on the information learnt stored in the Q-table. Given are the observations

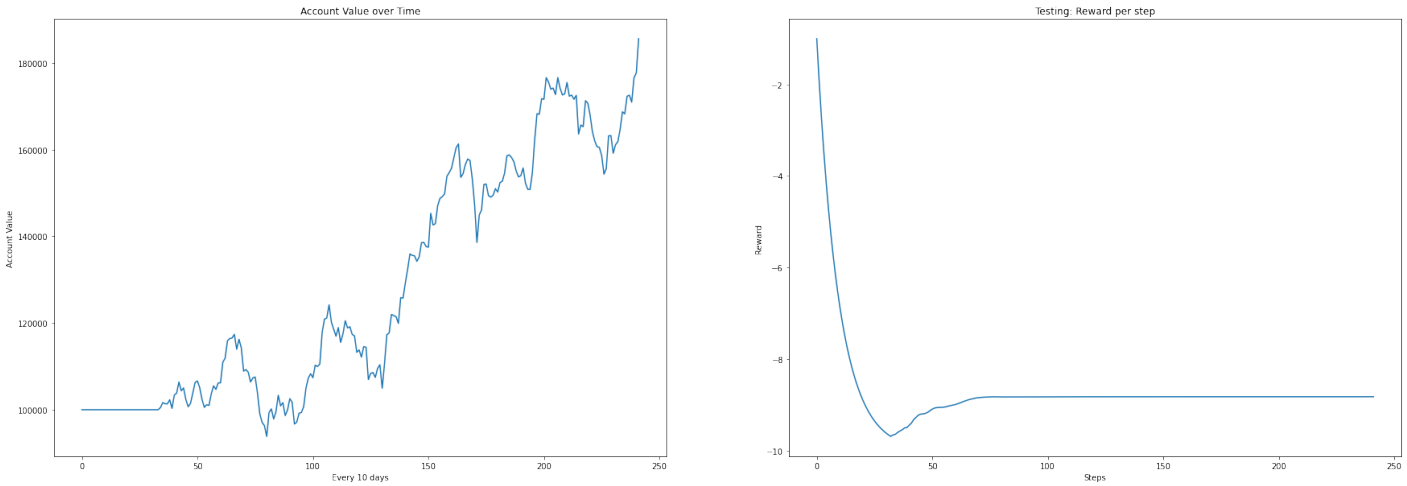

Figure 2: Evaluation Curves

Even though I get a final account value greater than 120000 USD by learning for 100 episodes, I trained it for more 5000 episodes to check if it can reach a value even higher than that which I did.

Episodes	Total Account Value
100	139095.309
500	176207.547
1000	176207.547
5000	185524.719

3 Conclusion

Through this assignment, I was introduced to Q-Learning and Reinforcement Learning in general. The assignment also introduced us to tools like OpenAI Gym which can be used to solve multiple Reinforcement Learning tasks.