Deep Reinforcement Learning

Dexson John D'Souza
Department of Computer Science
University at Buffalo
Buffalo, NY 14260
dexsonjo@buffalo.edu

1. Project Overview

The goal of this project is to using Reinforcement Learning to train an agent to learn trends in market and perform trading. We use will Q-learning algorithm to train the agent. We will also measure the performance of the agent.

2. Experimental Setup

We have used Jupyter Notebook which is an open-source software. We have used Gym which is a toolkit for developing and comparing reinforcement learning algorithms. We have used Scikit-learn which is open-source software for machine learning library in Python programming language to perform data partitioning and data normalization. We have used Numpy which is library for the Python programming language. We have also used the Python library Pandas. We have use Matplotlib to plot graphs.

3. Environment

There are three important components in defining a reinforcement learning environment: state, action, and reward. The state describes the current situation of the agent. In our environment, agent can be in one of the four states.

1. Price of the stock has increased and the agent does not hold any shares.
2. Price of the stock has increased and the agent holds shares.
3. Price of the stock has decreased and the agent does not hold any shares.
4. Price of the stock has decreased and the agent holds shares.

Action is what an agent can do in each state. There are 3 possible actions that the agent can perform.

1. BUY - Using the current capital investment amount, buy maximum amount of shares.
2. SELL - Sell all the shares that the agent holds.
3. HOLD - Do nothing. Don't buy or sell any share.

When agent takes an action in a state, it receives a reward. Reward is like a feedback from the environment. A reward can be positive or negative. When the reward is positive, it means that the action agent performed was good. When the reward is negative, it means that action had bad impact.

The goal of the agent is to increase the account value over the time.

4. Training and Testing

We have used a dataset on the historical stock price for Nvidia for the last 5 years. The dataset has 1258 rows. We used used 80% data for training and 20% data for testing the model.

We have used Q-learning algorithm for training purpose. The formula to update the Q-table is given as:

$$Q^{new}(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \left(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \max_a Q(s_{t+1}, a) - \underbrace{Q(s_t, a_t)}_{\text{old value}} \right)$$

We have used 50 episodes, 0.1 learning rate, 0.99 discount factor and 0.5 epsilon decay to train the agent. The graph for epsilon decay is depicted in figure 1 and graph for reward vs episode is given in figure 2.

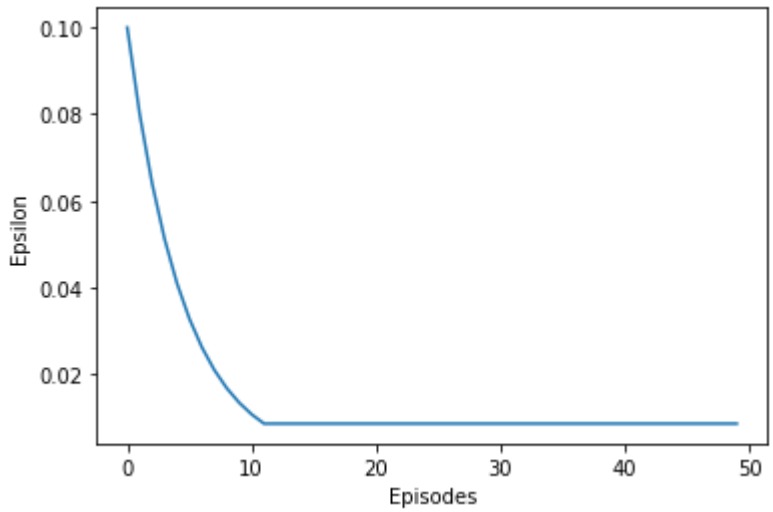


Fig 1: Epsilon v/s Episode

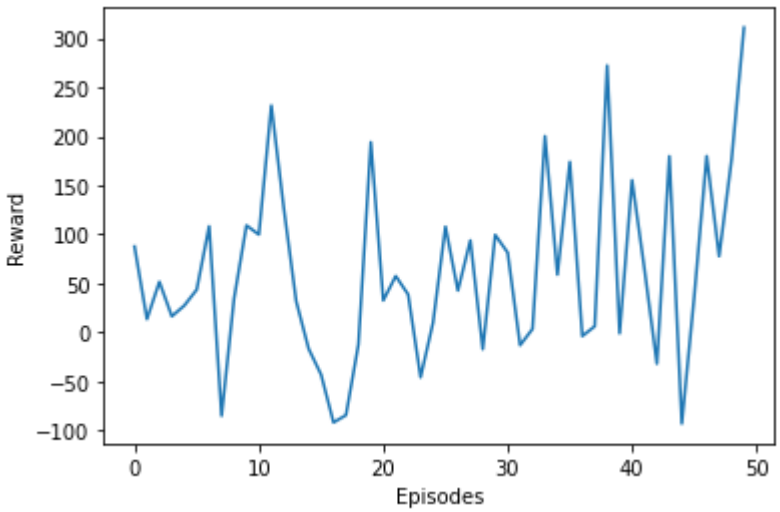
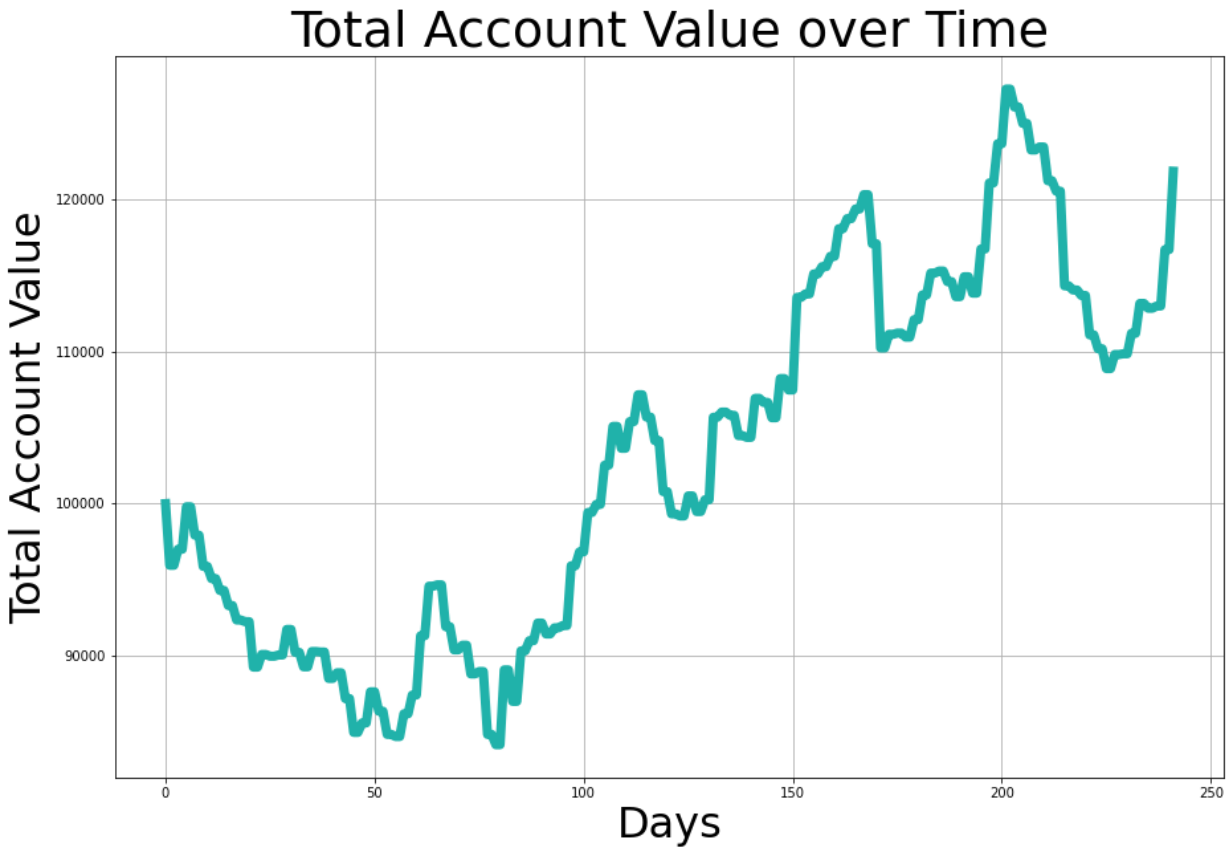


Fig 2: Reward v/s Episode

5. Result Analysis

Our agent was successful in using Q-learning algorithm to understand the trends in stock price and perform a series of trades over a period of time and end with a profit.The goal to end up with higher account value over the time was achieved. The plot of agent's performance is depicted in figure 3.



Total Account Value = 121874.10041700007

Fig 3: Account Value v/s Days

We can see that q-learning uses future rewards to influence the current action on a given state and therefore helps the agent select the optimal action that can maximize total reward.