Stocks prediction Deep Reinforcement Learning

Nandini Chinta
SUNY Buffalo, NY
nandinic@buffalo.edu

Abstract

This documentation refers to a project that predicts stocks using Q-Learning algorithm in deep Reinforcement Learning. This project has leveraged the usage of Nvidia's last 5 years (2017-2021) stock price dataset having 1258 entries of stock prices at it's opening, closed, highest and lowest in the day and the quantity of stocks shared on that day. We have used gym package to create the stock trading environment having observation space of 4 and action space of 3. Applying Q-Learning algorithm, this project has trained the agent to obtain total account value of 139095.3096

1 Dataset

In this project, the dataset given was the historical stock pricing from October 2017 to October, 2021 of Nvidia. It contains 1258 entries of information of stocks having features such as stock's opening price, closing price, highest and lowest of the day, and the volume of shares traded. This project has used this dataset to train the agent and predict the stocks.

2 Environment

Gym package from python has been used to create the stock trading environment. This environment has action space - {Buy, Sell, Hold} and Observation space – {0,1,2,3}. Possible observations contain four values having 0s and 1s. First two refer whether the stock's price has increased or decreased and the other two refer whether the agent holds the stock or not. The environment has three functions that purposes differently. To initialise the environment, we should provide the agent a path where it can find the dataset to train on. Also, other arguments such as whether the environment is for training or testing and an integer that indicates maximum number of days the agent should consider for deciding the prediction.
Below are few other methods that the environment holds.

Reset method: This method resets the environment to initialise a new episode. An episode is the path the agent takes to reach its goal. When called, this method returns a vector of observation space that contains values of 0s and 1s indicating if the prices have increased or decreased and if the agent already holds the stock or not.

Step method: This method is where the agent takes an action. This method takes an integer referring an action and returns the observation after making that action into the environment. This considers all possibilities of changes that can happen when the given actions take place at that point of time and returns the observation. The observation is an integer range between [0,3] where each integer indicates a vector of values.

Observation Vector = [price_increase, price_decrease, stock_held, stock_not_held]

0 = [1, 0, 0, 1]
1 = [1, 0, 1, 0]
2 = [0, 1, 0, 1]
3 = [0, 1, 1, 0]

Render method: This method is to plot various graphs as a result of agent's training and testing. It plots two graphs, rewards gained over time plotted against episodes. The other graphs is the total account value varied over time plotted against episodes.

3 Reinforcement Learning

Reinforcement Learning is one of the emerging fields of Artificial Intelligence. It makes use of combined Supervised and Unsupervised learning. Since AI is totally making good decisions, Reinforcement Learning is to learn to make good sequence of decisions. Using RL, AlphaGo defeated South Korean Go champion achieving 99.8% accuracy over other algorithms. RL is being use in various places like recommendation systems, playing various games, stocks & prices predictions etc. As it depends on sequence of decisions, it uses Markov Decision process to make the predictions.

Reinforcement Learning deals with agents, states, actions, rewards and penalties. The agent is who should be trained. It considers certain actions from the given action set and receives rewards and penalties, as necessary. The speciality of reinforcement learning is it learns from the prior knowledge. The final goal of the agent is to reach the target with maximum cumulative rewards. So, the agent must decide the sequence of actions that maximizes the completes the tasks. So RL provides the agents two methods to select the actions. One is to select the action randomly called as Exploration and the other is to select the action that best suits the environment called as Exploitation.

Reinforcement terminology:
Agent: The training model
Environment: It mimics the real-world situations virtually to the agent. It is what ana gent can interact with in one way or other. The environment changes as the agent performs certain actions.
States(s): It is the scenarios the agent encounters in the environment.

Actions(a): Agent's way of interaction with the environment. Actions can be taking right or left step, or can be anything.

Rewards(r): It is a compliment the agent receives for performing the best action in the environment. Technically, it is an integer. The Agent's goal is to maximize the reward.

Penalties: However, the Agent should not be greedy enough to collect maximum rewards forgetting its target and ethics.

Reward Table(R): This is a matrix of values having rows as many as states and columns as many as actions. It contains rewards for the agent taken an action given in a state.

In Reinforcement Learning, Agent performs an action in the environment and the environment sends the next state and reward/penalty to the agent as a result. Based on the received reward and state, agent decides the next action to perform. This is why, Reinforcement Learning is to learn good sequence of decisions. Below is an illustration of how Reinforcement Learning works.

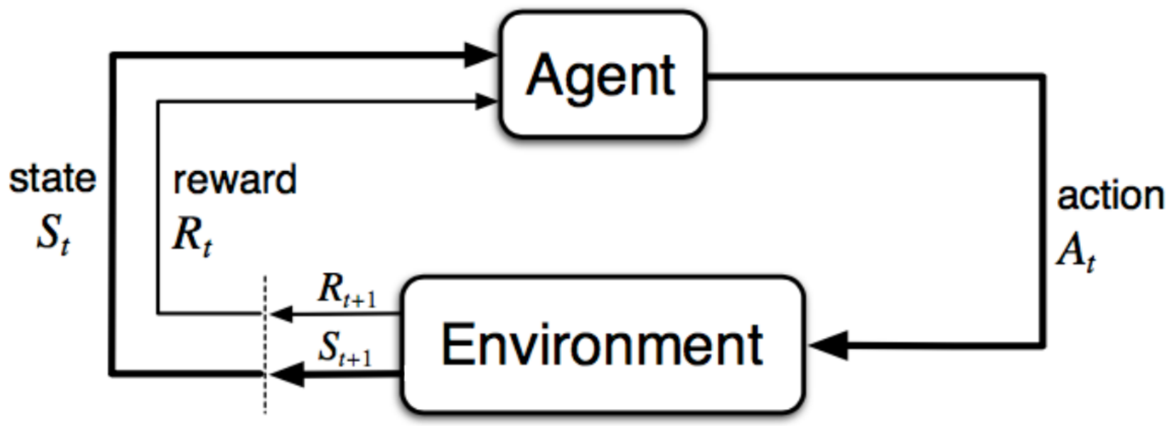


Fig 3.1: Reinforcement Learning illustration

3.1 Q-Learning Algorithm

Q-Learning is an algorithm where, the agent in Reinforcement Learning doesn't go greedy. This is one of the easiest algorithms in RL which was proven to converge to optimal solutions given conditions. The Q-value is calculated for every action taken given in certain state. Q-value represents the value for given (state, action) pair. Q(s,a) represents how worthy the action is for further sequence of actions to reach the target. Like reward table, Q-Learning algorithm has Q-table that gets updated for every timestep, with best results. Q-table is updated by a formula. In this project, we have used below formula to update the Q-values in the Q-Table.

$$Q^{new}(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\Bigg(\underbrace{\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}}}_{\text{new value (temporal difference target)}} - \underbrace{Q(s_t, a_t)}_{\text{old value}} \Bigg)}^{\text{temporal difference}}$$

Fig 3.1.1: Formula to update Q-table

Algorithm:

```
Initialise Q(s,a) arbitrary for all states and actions. Usually, it will be set to all zeroes initially.
Repeat (for each episode):
    Initialise s
    Repeat (for each step of episode):
        Choose an action from given policy.
            a = a1 if random_float_value(0,1) < epsilon else choose action with maximum Q-value. Receive next state s', reward r and Boolean value that indicates whether the episode has ended or not.
        Perform action a.
        Update Q-value for current state and action taken
        s 🡐 s'
        if episode ended: break
    Until s is terminated
```

Hyper-parameters: In Q-Learning we are using few hyper parameters while updating the Q-table.

Alpha(α): This as in other ML models, works as learning factor.
Gamma(γ): This is a discount factor. As steps increases the rewards value decreases exponentially to prevent the agent take longer duration to reach the target.
Epsilon(ε): To prevent the agent to become greedy about rewards, we use a hyperparameter epsilon that helps the agent to explore in the start and more exploit in the end of the iterations. This function is called Epsilon Decay function.

3.2 Epsilon Decay

Epsilon Decay function is used to prevent the agent become greedier and balance both exploration and exploitation for more optimization. In the Q-Learning algorithm the next actions will be chosen based on the epsilon value. We will consider a random value and if that value is less than epsilon, we choose exploration else we choose exploitation. We do this to make sure the agent doesn't always go for Exploitation/Exploration learning nothing. In this project, we have used Stretched Epsilon Decay function. In this function, we are using three hyper parameters that decay the epsilon in three different ways which we will be discussed further in this report.

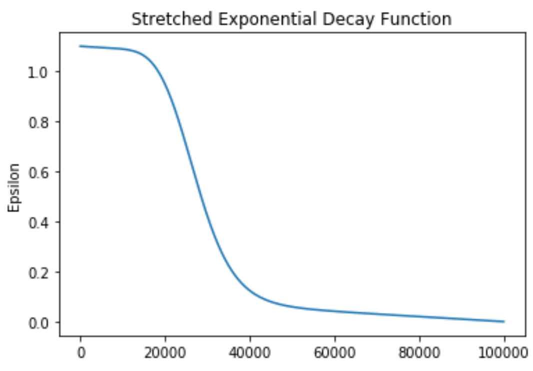
$$\text{Epsilon } \varepsilon = 1.1 - \frac{1}{cosh(x) + \frac{time - A * c}{Episodes}}$$

$$\text{Cosh(x)} = \frac{e^x + e^{-x}}{2} \qquad x = \frac{time - A * Episodes}{B * Episodes}$$

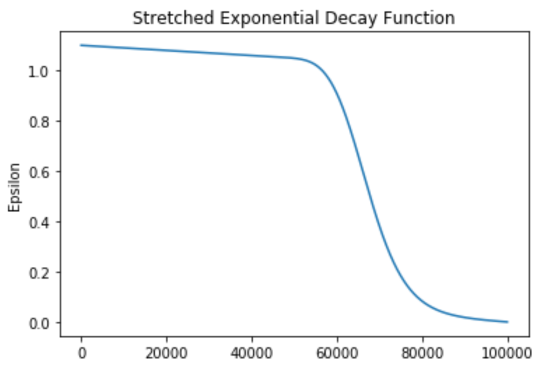
Episodes: The number of iterations the agent will be trained.

Time: This refers to the timesteps taken

A: It balances the exploration and exploitation. For values below 0.5, the agent will spend less time on exploration and more on exploitation.

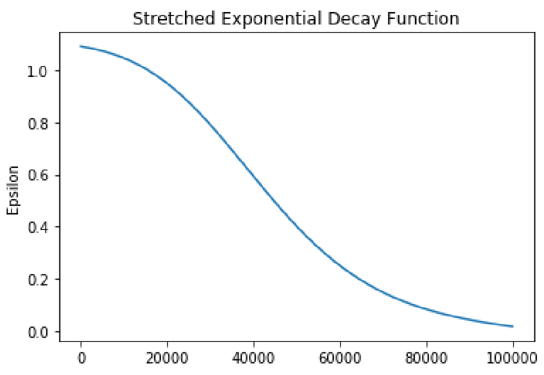


For A=0.3, the left tail is shortened



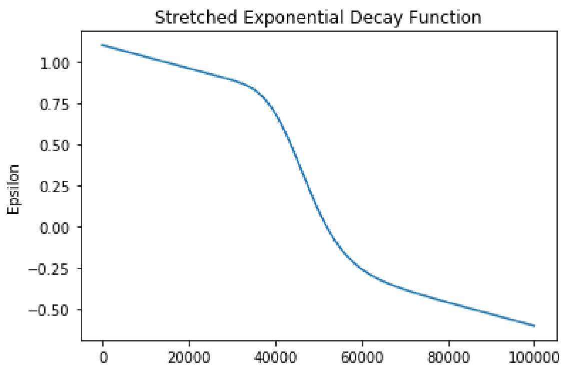
For A=0.5, the left tail has lengthened

B: This decides the slope of transition region between exploration and exploitation.



With B=0.3, the transition portion has gradient = -45 degree

C: This controls the steepness of both the tails.



With C=0.7, the left and right tail tends to get steeper

4 Testing

Now that the algorithm is implemented on the training data, the agent is evaluated against test data. The agent is trained with learning rate 0.3, discount factor(gamma) 0.4, maximum iterations1217 and for epsilon decay, A=0.4, B=0.4, C=0.1. Fig 4.1 shows the plot drawn against episodes and total account value over time during training.

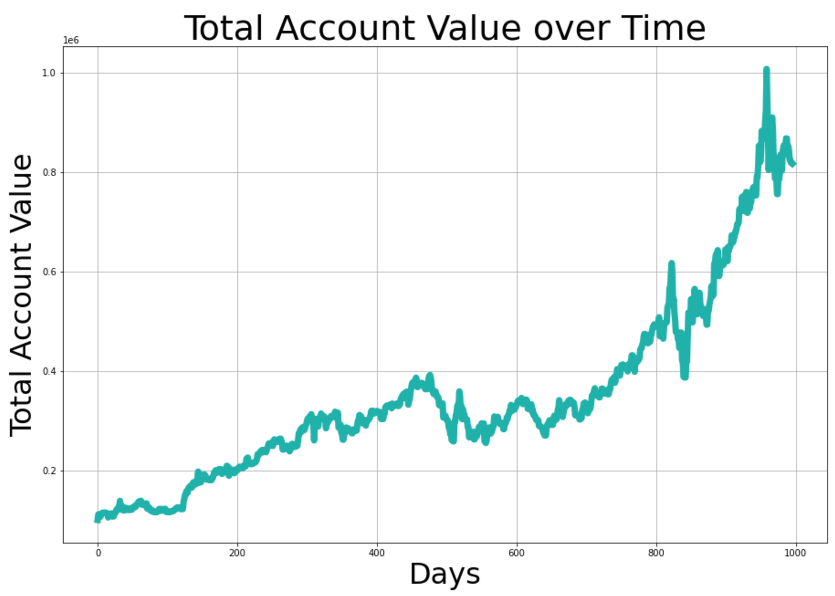


Fig 4.1: Total Account Value over time during training

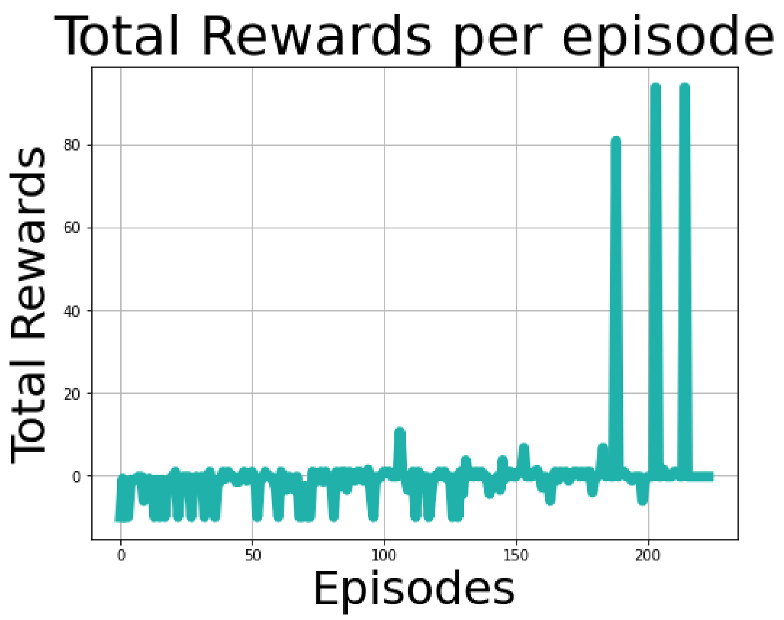


Fig 4.2: Rewards gained over time during training

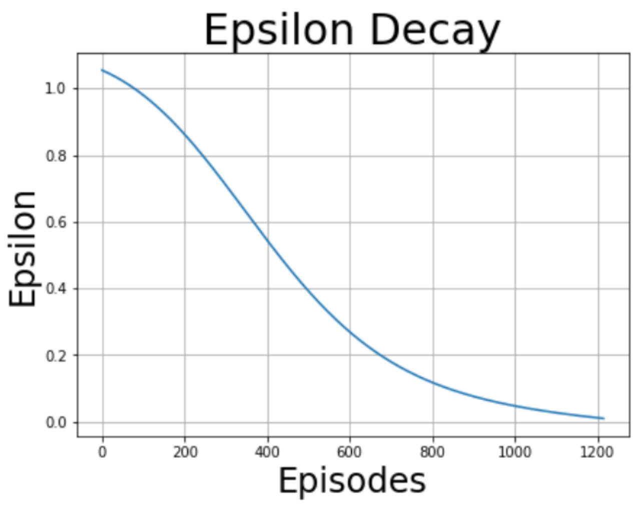


Fig 4.3: Stretched Epsilon Decay

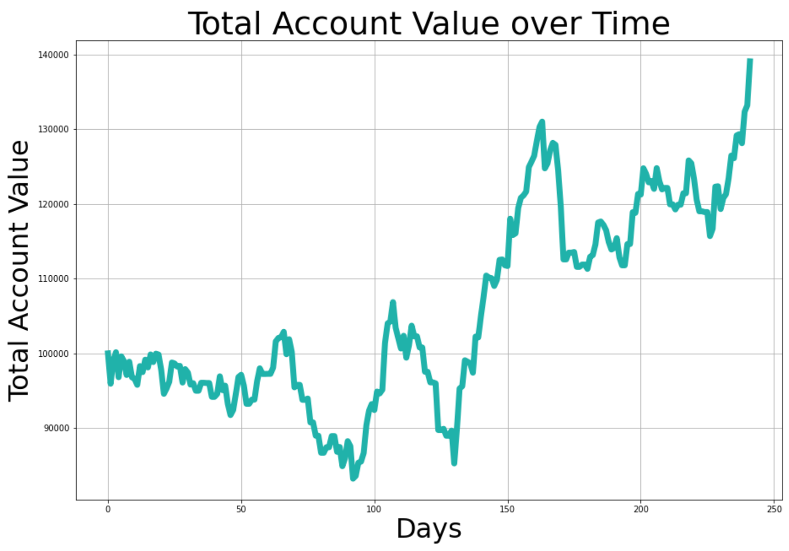


Fig 4.4: Total Account Value over time during testing

5 Conclusion

To conclude, the agent was trained on different hyper parameters and different numericals for maximum iterations. With hyper parameter A, 0.5 and above the agent explored more than it did the exploitations. I have played with the model with different hyper parameters and at certain points, the agent showed better performance and obtained total account value of 139095.3095

References

[1] https://en.wikipedia.org/wiki/Q-learning
[2] https://www.eecs.tufts.edu/~mguama01/post/q-learning/
[3] https://www.learndatasci.com/tutorials/reinforcement-q-learning-scratch-python-openai-gym/
[4] https://www.freecodecamp.org/news/diving-deeper-into-reinforcement-learning-with-q-learning-c18d0db58efe/
[5] https://medium.com/analytics-vidhya/stretched-exponential-decay-function-for-epsilon-greedy-algorithm-98da6224c22f

[3]