# CSE 4/574: Introduction to Machine Learning Fall 2021
# Assignment 3 - Deep Reinforcement Learning

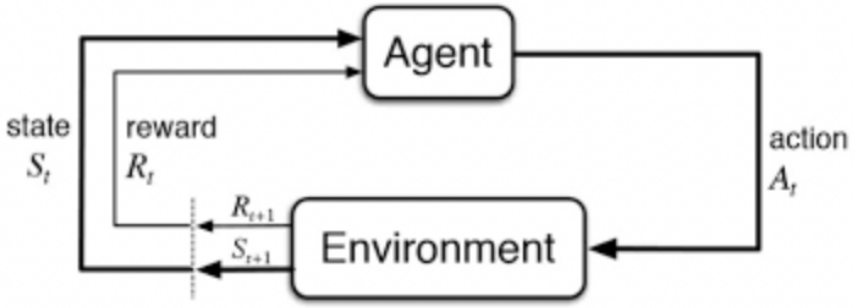
vidushis@buffalo.edu
Computer Science and Engineering Department

## Abstract

Q-learning is a model-free reinforcement learning algorithm to learn the quality of actions telling an agent what action to take under what circumstances. Q-learning finds an optimal policy in the sense of maximizing the expected value of the total reward over any successive steps, starting from the current state.[1]

## 1. Introduction

The project uses Q Learning techniques 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. The initial investment capital is $100,000 and performance is measured as a percentage of the return on investment. Q-Learning algorithm is used for reinforcement learning to train an agent to learn the trends in stock price and perform a series of trades.




## 2. Dataset / Environment

### 2.1 Dataset Introduction:

The dataset is based on the top of historical stock prices 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.

### 2.2 Environment Details:

**Possible Actions:**

```
Buy/Sell/Hold.
```

```
(Integer in the range 0 to 2 inclusive.)
```

The possible actions can be to Buy/Sell/Hold.

**States**:

```
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.
```

```
 [1, 0, 0, 1]: observation=0 =>{Price Increased:true,Price decreased:false, Agent Holding the stock:false, Agent Not Holding the stock:true }
[1, 0, 1, 0]):observation=1 => {Price Increased:true, Price decreased:false, Agent Holding the stock:true, Agent Not Holding the stock:false }
[0, 1, 0, 1]): observation=2 => {Price Increased:false, Price decreased:true, Agent Holding the stock:false, Agent Not Holding the stock:true }
[0, 1, 1, 0]):observation = 3 => {Price Increased:false, Price decreased:true, Agent Holding the stock:true, Agent Not Holding the stock:false }
```

**Goal** :

The goal of the problem is to approximate prices for stocks in a portfolio and return the account value after a particular number of days.

**Rewards**:

During the exploration step, we make sure that we keep a tracker of the rewards which we get during each episode and our goal is to take optimal steps so that the trend of the rewards is increasing as the number of episodes increase.

## 3. Reinforcement Learning

### 3.1 Introduction

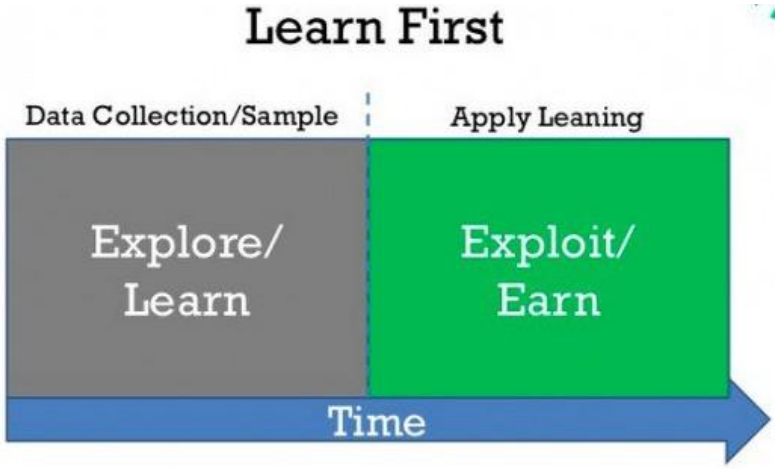
Reinforcement learning is the problem of getting an agent to act in the world so as to maximize its rewards.

### 3.2 Exploration and Exploitation:

Reinforcement learning requires choosing between exploration and exploitation.

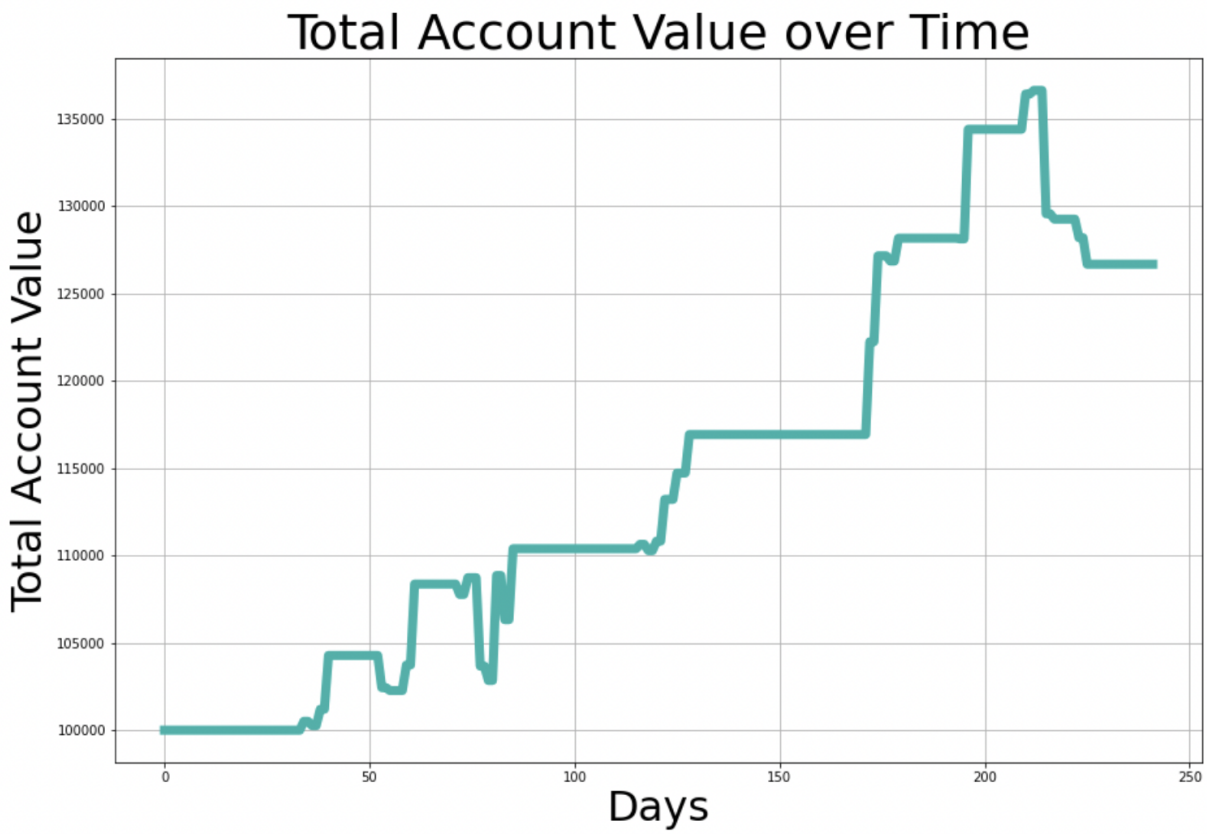
**Exploration** : Refers to taking actions specifically to obtain more training data.
In the exploration step in our problem, we train the data. We run 1000 iterations and try to update the Q Table with best values and keep on updating our Epsilon values accordingly. The rewards matrix is also computed in this step.

**Exploitation**: Refers to taking actions that come from the current best version of the learned policy.
In the exploitation step , we use Greedy Approach to get the most optimal Account value. In this step we do not use the epsilon decay to update the epsilon values.



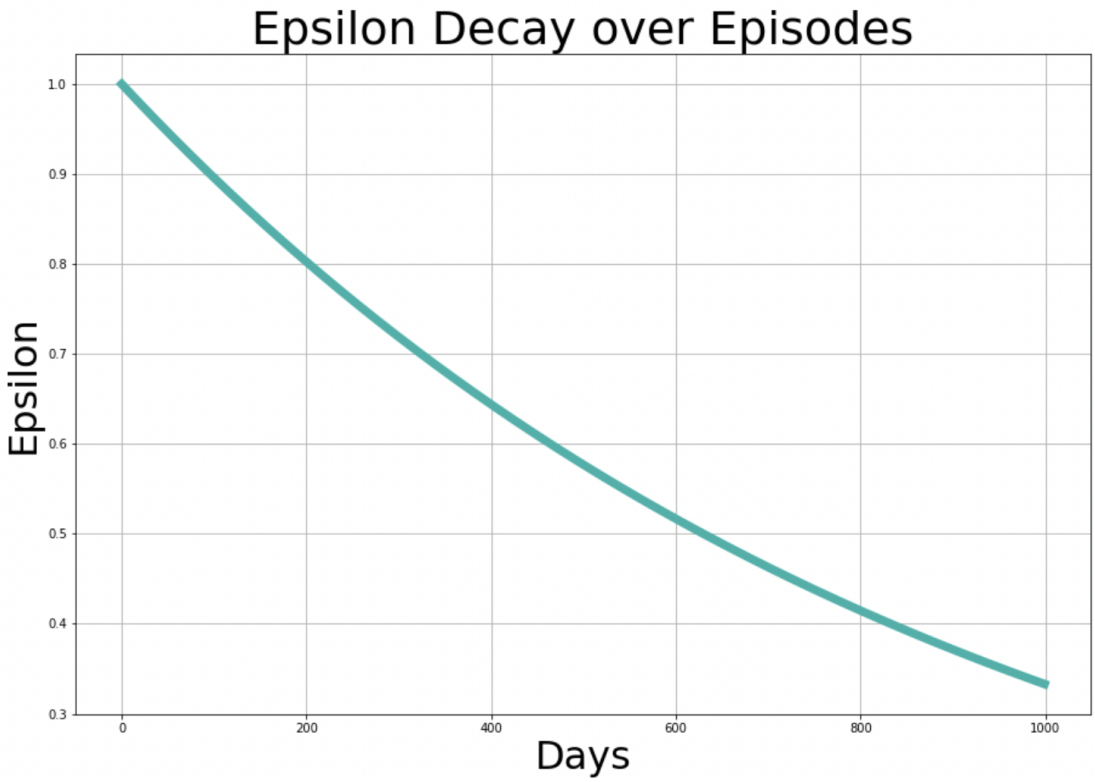

## 4. Findings

| | |
|---|---|
| Learning Rate | 0.9 |
| Decay Rate | 0.9989 |
| Initial Epsilon | 1.0 |
| Iterations for Training | 1000 |
| Total Account Value | `126672.15922499992` |




```
-------------------------------Total Account Value-----------------------------------
Final Value  126672.15922499992
```


Rewards over Episodes
Rewards
Days
1000
0
−1000
−2000
−3000
0
200
400
600
800
1000


Epsilon Decay over Episodes
Epsilon
Days
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0
200
400
600
800
1000

5. References

1.https://www.analyticsvidhya.com/blog/2020/10/reinforcement-learning-stock-price-prediction/#:~:text=Q%2Dlearning%20is%20a%20model,starting%20from%20the%20current%20state.

2.https://prakhartechviz.blogspot.com/2019/02/epsilon-greedy-reinforcement-learning.html

3.https://cedar.buffalo.edu/~srihari/CSE574/Chap15/15.1-Reinf-Learning.pdf