-

Assignment 3- (Q-Learning in Stock Trading)

Ramesh Pavan Pothamsetty

8th December 2021

1 Assignment Overview

The goal of the assignment is to work with Nvidia dataset and train an agent to learn the trends in trade and perform trading using Q-Learning algorithm to increase the total account value over time. Thereby understanding how Reinforcement Learning can be used in real world problems.

2 Dataset

The dataset has the stock details of the company for the past five years with 1258 entries. The different attributes of the stock that were given are stock opening price, stock closing price, price fluctuations in a day and number of shares traded per day etc.

The environment to perform trading is already given and I am supposed to implement Q-Learning from scratch to devise an efficient trading strategy.

Training data: 80 percent of 1258 entries.

Testing data: 20 percent of 1258 entries.

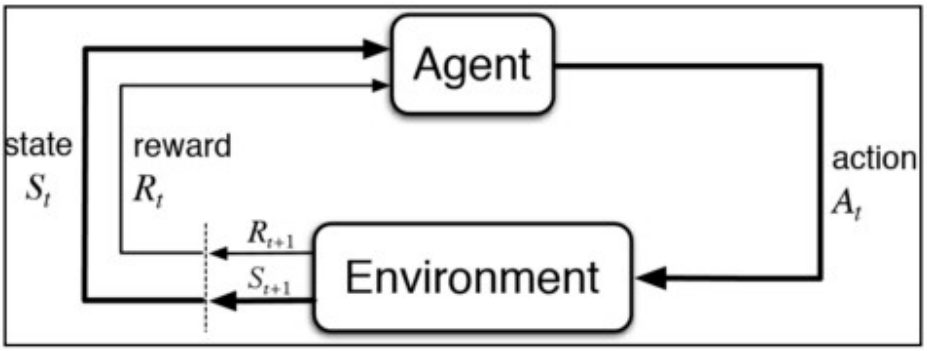
3 Python Editor

I have used Jupiter Notebook IDE for implementation.

4 Reinforcement Learning

Reinforcement Learning is branch of machine learning in which an agent is trained to make optimal decisions from past experiences. In which the environment provides rewards or penalties to the agent depending upon the actions that the agent takes in the environment. Eventually the objective of the agent is to maximize its rewards.

-



Q-Learning

Q-Learning is a technique in RL in which agent learns optimal policy depending upon the quality of the action that it takes in a particular state. We setup a matrix of size (states*actions) in which we fill each entry which gives the quality value depending upon the past Q value, immediate reward and discounted future reward. For doing this we use the below training rule/equation.

$$\hat{Q}_n(s,a) \leftarrow (1-\alpha_n)\hat{Q}_{n-1}(s,a) + \alpha_n\left[r + \gamma \max_{a'} \hat{Q}_{n-1}(s',a')\right]$$

We do this process of updating Q-table over several iterations in order to obtain the best possible version of the table so that when agent starts performance in the real world it starts exploiting mechanism to take optimal decisions depending upon the Q-table obtained over all those iterations.

5 Environment

(class StockTradingEnvironment(gym.Env))

Given environment has a class named stock_trading_environment which has the following methods.

-

Init method:

This method defines the variables that are required by the stock trading environment and initializes them, and calls reset method. Hence reset the values of the variables to initial ones after every episode.

Reset method:

As the name suggests this method resets the environment and returns an observation to the agent. This observation is one of the states that agent can be in, and the value is between 0-3 as we have four states in our environment which can be described from the observation vector.

observation= [price_increase, price_decrease, stock_held, stock_not_held]

returns 0 if the stock price increases and the agent does not hold any stocks

returns 1 if the stock price increases and the agent holds the stock

returns 2 if the stock price decreases and the agent does not hold any stocks.

returns 3 if the stock price decreases and the agent holds the stock.

Step method:

Based on the observation, the agent takes the action which is passed to the step function and later returns reward for the action, the next observation, whether the episode is done or not and some additional info. These actions are selected based on the value of epsilon and are of two types- exploratory actions and exploiting actions.

The three possible actions that the agent can take are buy, sell and hold which are represented in the form of integer values 0,1and 2 respectively.

Render method:

-

This method just plots the agent's performance during training in terms of plot between total account over time which is 80 percent of the given entries in the dataset.

Goal:

The goal of the agent is to learn an optimal decision-making mechanism using Q-Learning algorithm by building a Q-table on the training data and use that table for exploiting max rewards while performing on the testing data.

Rewards:

Initial penalty for the agent is 0.

If the action is buy and the no of shares that the agent hold are greater than zero, the penalty is -10. If the agent has capital and buys shares, then the reward is penalty+1 and if the agent buys shares when there is no capital then the reward is -10.

If the action is sell and the value for which the agent bought the shares (book value) is greater than zero the agent gets the reward equal to the amount of profit that it gets for selling the current shares i.e positive value if the share price increases and negative value if the share price decreases. Else if the agent sells the shares when there are no shares it gets a reward of -10.

If the action is hold and the value for which the agent bought the shares (book value) is greater than zero then the agent gets the positive reward if it holds the stock when stock price decreases than purchased value and negative reward when it holds the stock when stock price increases. Else if the agent holds the stock when there is no stock at all then it gets reward of -1.

class QLearning

For the QLearning class skeleton is already provided for the following methods.

-

Init method:

In the method I have defined variables that are required for calculating Q values and defined a matrix with size (4*3) and filled it with zeros.

alpha = learning rate

gamma= discount factor

epsilon = a constant which is helpful in setting exploitation and exploration modes.

Train method:

In the train method I have defined number of episodes as 500 and two lists for storing rewards and epsilon decay to plot it later

The agent chooses exploration if the epsilon value is below a threshold in which actions are based on randomness. As the number of episodes increases, I have implemented epsilon decay which forces the agent to go towards exploitation which means taking the actions based on best possible Q value which is calculated by the below equation which determines quality of the state.

$$\hat{Q}_n(s,a) \leftarrow (1-\alpha_n)\hat{Q}_{n-1}(s,a) + \alpha_n\left[r + \gamma \max_{a'} \hat{Q}_{n-1}(s',a')\right]$$

Evaluate method:

In this method I have defined one episode to test the trained agent on the testing data. In this the agent chooses best action based on highest Q values of the state taken from the matrix that was built during training.

Plot method:

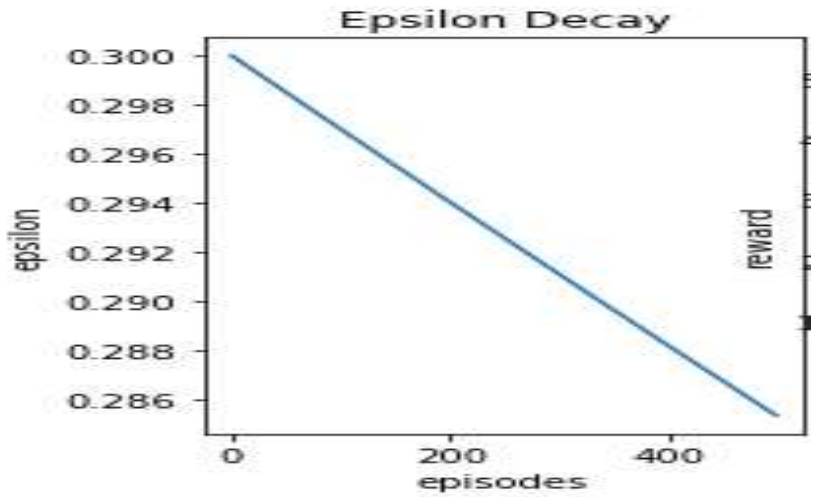
This method is for plotting the graphs for rewards and epsilon decay during training.

-

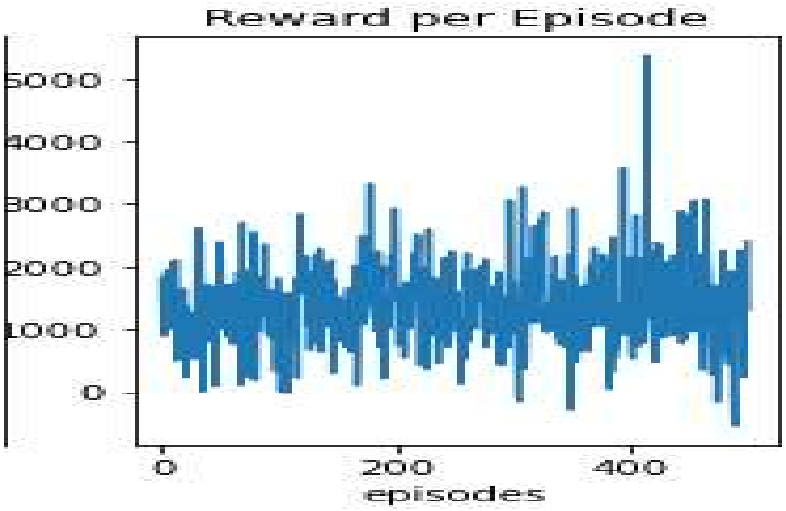
6 Results

Below are the results during training the agent where the agent chooses exploration in the beginning and as the episodes increase it tends to choose exploitation.

Epsilon Decay:



Reward per Episode:



-

The final plot which is generated by the given render function in the environment. The agent runs for one episode on the test dataset and makes decisions based on the best Q values from the table that is built during training. The below graph gives total account value over time=139095.30966699996