**Using Reinforcement Learning for Managing Portfolio**

Introduction:

Portfolio optimization is the process of selecting the best portfolio from a set of available ones which minimize risk and maximise returns. An investor’s success lies in managing a profitable portfolio. The inapplicability of Modern Portfolio theory in actual market conditions has led to an introduction of machine learning for solving the portfolio problem and has proved lucrative for quite some time now. Apart from using supervised and unsupervised learning algorithms, the recent interest towards application of reinforcement learning in finance industry is noteworthy.

In this research, we present a rudimentary approach to portfolio optimization using reinforcement learning techniques, comparing a deep Q-Learning model with a baseline model.

Related Work:

Previous works in the field focus on the processing of stock data from various sources, modeling portfolio system of two stocks, use of actor-critic methods but none of them compare the performance of reinforcement learning algorithm with a baseline strategy on optimizing portfolio of more than 2 stocks.

Approach:

Dataset:

We are trying to optimize a portfolio of Top 15 cryptocurrencies. The training dataset consists of closing prices of the cryptocurrencies over a duration of 375 days, i.e from 1st Jan 2018 to 31th Dec 2018 and we have assumed a holding period of no more than 180 days.

The test dataset consists of closing proces of the same 15 cryptocurrencies from 1st Jan 2019 to 30th June 2019.

The dataset includes closing prices of crypotcurrencies Cardano, Bitcoin Cash, Binance, Bitcoin, Dash, EOS,Ether, IOT, Chainlink, Litecoin, Tron, Tether, Stellar, Monero, Ripple cryptocurrency for the specified period. We have consolidated the data from the Yahoo Finance repository into CSV format.

The structure of CSV consists of a Date field as index followed by 15 columns representing the closing prices of the mentioned cryptocurrencies.

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Start Date | End Date | Total Data Points |
| Train | 1st January 2018 | 31st December 2018 | 365 |
| Test | 1st January 2019 | 30th June 2019 | 180 |
| Total |  |  | 545 |

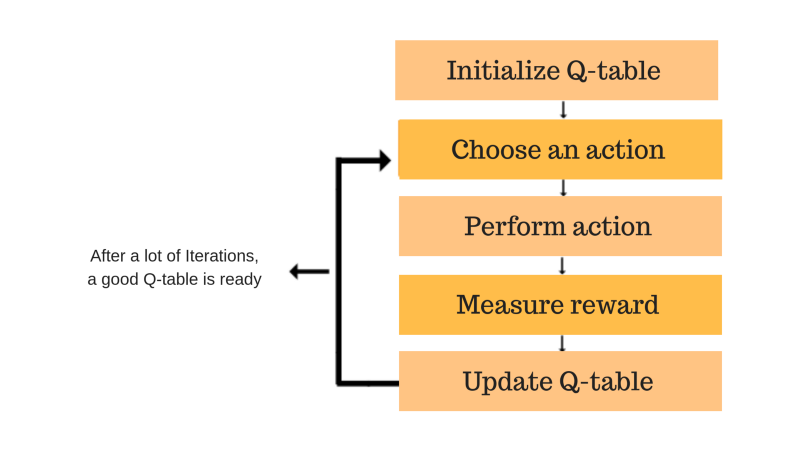
Methodology:

Like all reinforcement learning goals, we are aiming to train our agent to be capable of successfully interacting with the environment by taking appropriate actions to influence the environment state and gain rewards. Our agent employs discrete actions as it choses between a fixed number of actions – to hold, sell or buy. Our agent is working towards an aim of being able to allocate weights to the stocks in the porfolio optimally.

The agent holds a Deep Neural Network model to predict the closing prices of the cryptocurrencies which are the states on which the agent bases it’s actions.

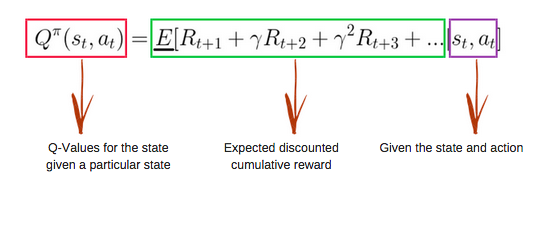
The states of environment are the percentage changes in the closing price of a fixed window size from any specied point of time. The job of the agent is to act on the information present in the current state and move to the next state acquiring a reward. It has to take into consideration not just the present rewards but also the future rewards with a discount factor gamma.

We used a Deep Q-Learining strategy to train the agent to act in a way to optimize it’s rewards. Q-Learning is an iterative process of updating the values. As the agent starts to explore, it starts getting better rewards and these are updated in the Q-table. The Q-function, which is a function of state-action pair needs to be updated.



In our case, we have fixed the batch size as 32. For each batch, we store the state-action pairs in Qtable and at the end of each batch, use the Q-table to update the Q-function.

The following gives insights about Q-function:



Our agent exploit known information with a probabilty of 1 – epsilon and explores new possibilities otherwise. As the number of iterations proceed, we decrease the value of epsilon so that the agent starts relying more on gained knowledge.

Experiment Setup:

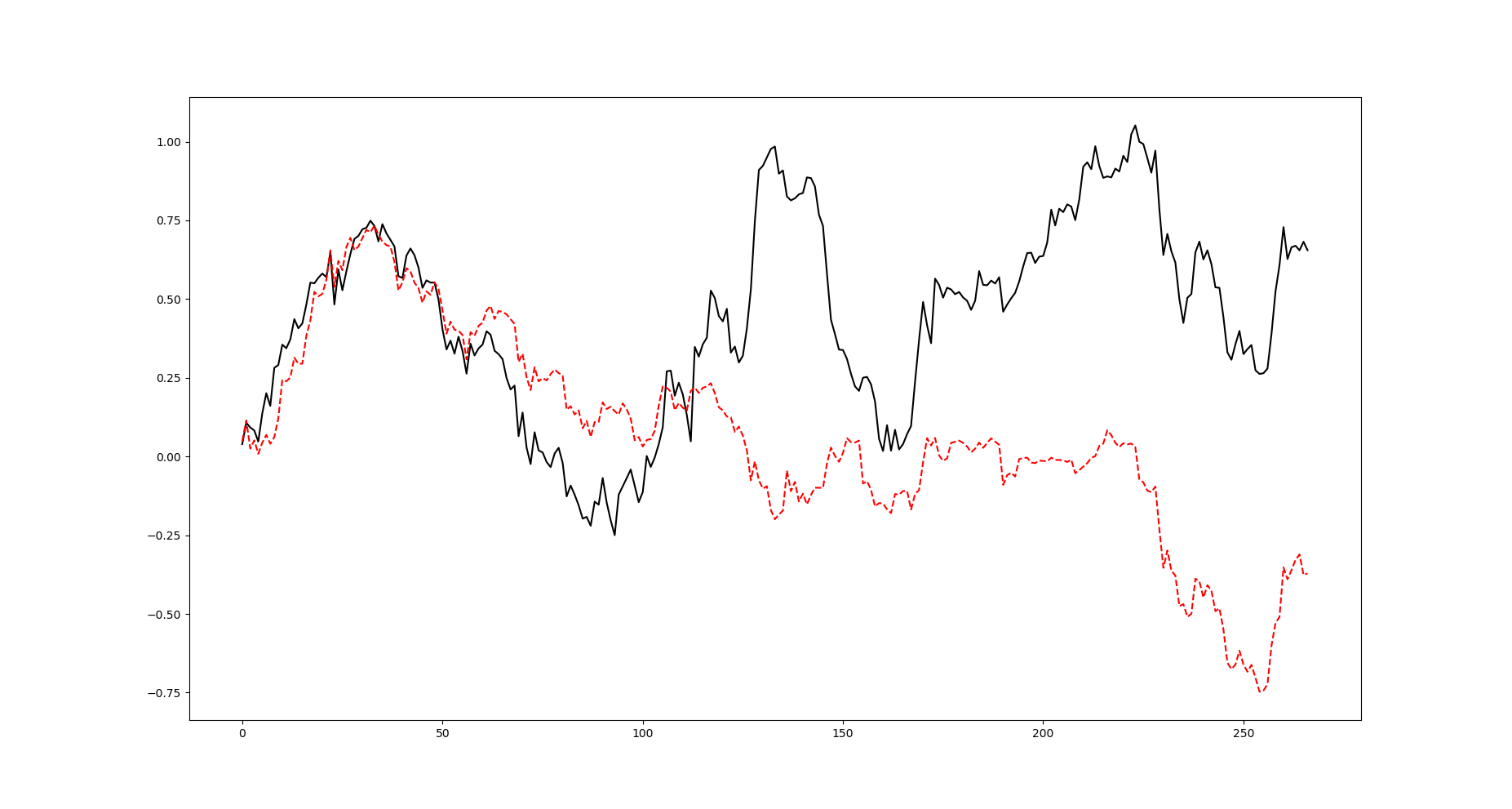
We have trained a Deep Q-Learning network which changes the weights of the portfolio dynamically according to the changing market. We have compared the outcomes from this strategy with a baseline strategy which allocates equal weights to all stocks always. To begin with, we employ a complete exploration strategy but keep on decreasing current epsilon by a factor of 0.99 after each iteration. Our agent takes into consideration the future rewards with a discount factor of 0.95. Finally, we have used a learning rate of 0.5.

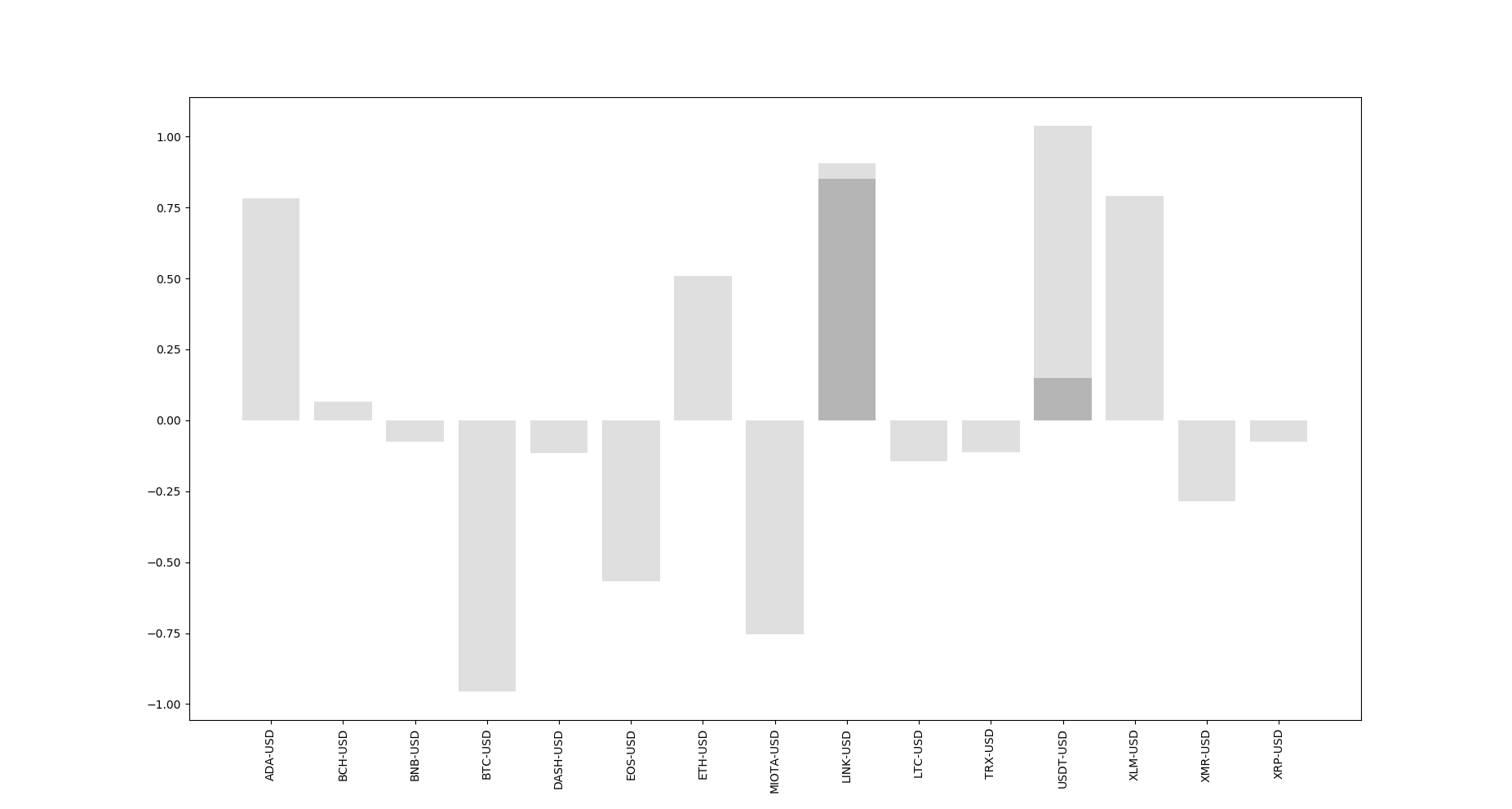
Evaluation:

We have used stock prices data from first half of 2019 for evaluation. We use the acquired returns as a metric to evaluate the performance of our agent. For the baseline model, we analyse the returns gained by allocating equal weight to the portfolio. Further, we compare the performance of the agent for both the strategies.

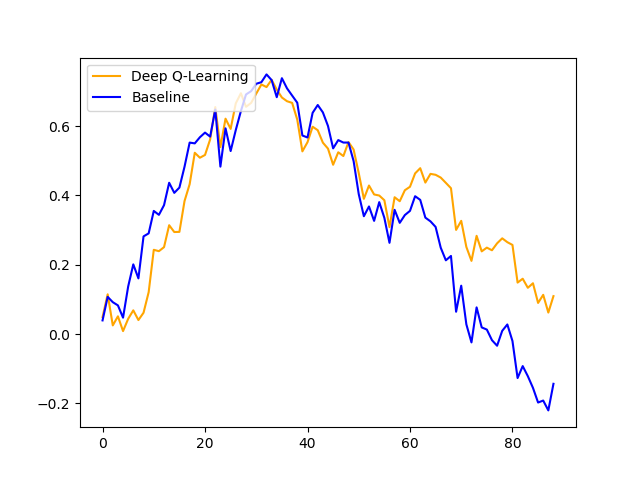
Results and Contribution:

Performance on training set:

The red dotted line is the performance of the baseline model whereas the black line is that of reinforcement learning agent. The graph shows the sum of all returns obtained across the time duration of the training set data for the last iteration. As we can see clearly, the Deep Q-Learning strategy has outperformed the baseline strategy in the long run.



Performance on Test set:



The above graph shows the returns obtained for the time duration of test data using both baseline model and reinforcement learning model.

References:

[1] <https://medium.com/@dennybritz/exploration-vs-exploitation-f46af4cf62fe>

[2] <https://arxiv.org/pdf/1909.09571.pdf>

[3] <https://www.freecodecamp.org/news/an-introduction-to-q-learning-reinforcement-learning-14ac0b4493cc/>