Selection Submission Report

Vipul Kumar Rathore

August 23, 2019

1 Problem Statement

The task is to create a deep Reinforcement Learning agent to make profits on bitcoin trading. The same model can also be extended to any cryptocurrency apart from bitcoin also. In general, RL seems a bad idea for this kind of problem as it is risky to incorporate exploration at the cost of losing some reward in the trading world. However, recent advances propose that RL agents are capable of learning much more than supervised learning agents within the same problem domain. This is our major motivation behind exploring the efficacy of deep RL models even for this kind of sensitive task.

2 Contribution

Our contributions are three-fold :-

- Create a reinforcement trading environment using gym library in python
- Created a simple visualization of the environment using gym and Matplotlib
- Trained the RL agent to earn profits in bitcoin trading

3 Dataset

We experiment upon 2 bitcoin trading datasets-

- Coinbase BTC/USD daily database (daily data available)
- Coinbase BTC/USD hourly database (hourly data available)

These datasets can be found at this link

4 Data Engineering

We use Pandas to load the data into dataframe and maintain an optional *initial_balance* variable and a variable *lookback_window_size*, indicating how many time steps from the past are observed by the agent for learning. The variable *commission* is set to 0.075 % as default value and the variable *serial* is false, indicating that the slices of dataframe are traversed in a random fashion.
Apart from this, we remove NaN values using *dropna()* and reset the frame indices using *reset_index()*.

5 Model Details

- **Action space** - A discrete set of 3 options - buy, sell, or hold, and another discrete set of 10 amounts - 1/10, 2/10, 3/10,....., 10/10 for each of these 3 options i.e. a total of 3*10 = 30 possible actions. When the buy action is selected, we will buy amount*self.balance worth of BTC. For the sell action, we will sell amount*self.btc_held worth of BTC. The hold action will ignore the amount and do nothing.
- **Observation space** - A continuous set of floats between 0 and 1, with the shape (10, lookback_window_size + 1). The + 1 is to account for the current time step. For each time step in the window, we will observe the OHCLV values, our net worth, the amount of BTC bought or sold, and the total amount in USD we've spent on or received from those BTC.
- **No. of episodes(trading sessions)** = 100000. End of the episode happens when we run out of the BTC held i.e. btc_held becomes 0.
- **RL model** - We experiment with 2 inbuilt models from stable-baselines library - PPO2 and A2C. Experiments show that switching from PPO2 to A2C can significantly improve our performance.
- **Reward** - We try 2 rewards - current net worth and (current net worth - previous net worth). Experiments show that the second reward function helps in achieving better performance than the first one. This somehow indicate that not just achieving total reward as net worth and staying there is sufficient, but it is necessary to increase your net worth with each episode and hence use the incremental net worth as the reward fucntion.

6 Training Details

We use 80% as training and 20% as test data. The reason behind not using validation is because we don't have too many option for cross-validation on time series data. In fact, k-fold cross-validation doesn't work for time-series data because later data is highly dependent on previous data and hence the ordering of the data points matters a lot. Therefore, splitting the data into random

sets doesn't make sense for time-series data. Also, we want to demonstrate the efficacy of the model by getting good results on data (test data) never seen before.
Next, since our environment is only set up to handle a single data frame, we will create two environments, one for the training data and one for the test data.

7 Experiments and Results

- Link to Matplotlib visualization of our agent trading Bitcoin - here
- **With PPO2 model and current net worth as reward -**

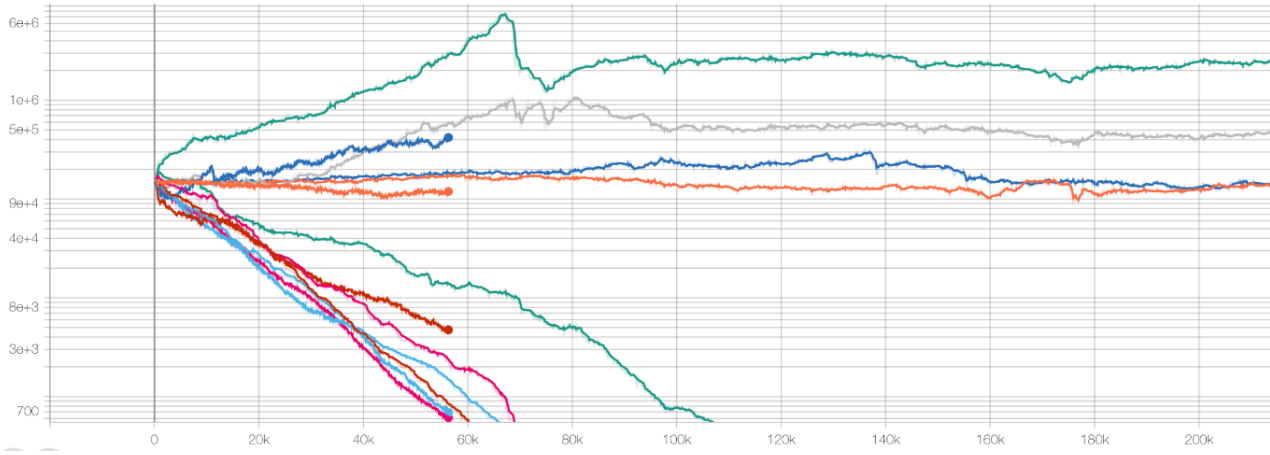

Figure 1: *Discounted rewards (multiple agents) v/s time on training data*

Clearly, according to figure 1, only a couple of agents are doing well and rest are running towards bankruptcy. The best agents are able to do 10x and even upto 60x initial balance. According to figure 2, these agents are running poorly towards bankruptcy on test data.

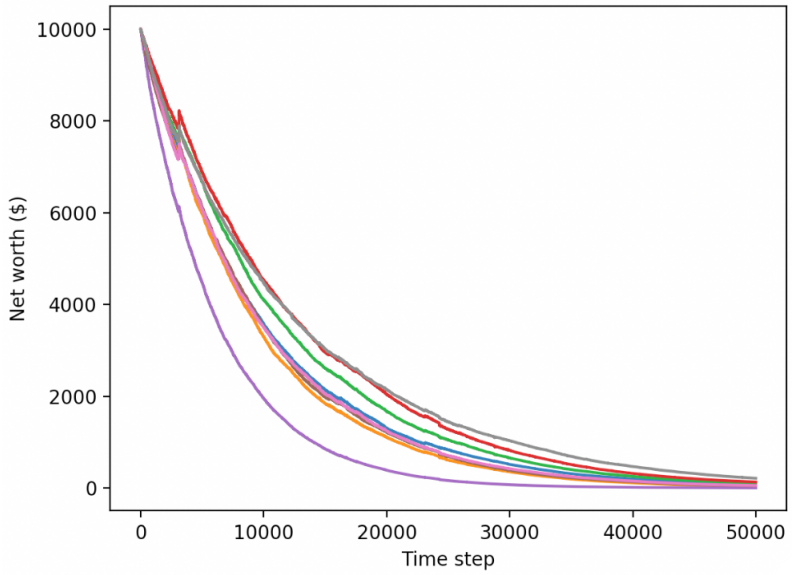

Figure 2: *Trained agents race to bankruptcy when trading on fresh, test data*

- **With A2C model and incremental worth (profit) as reward -**

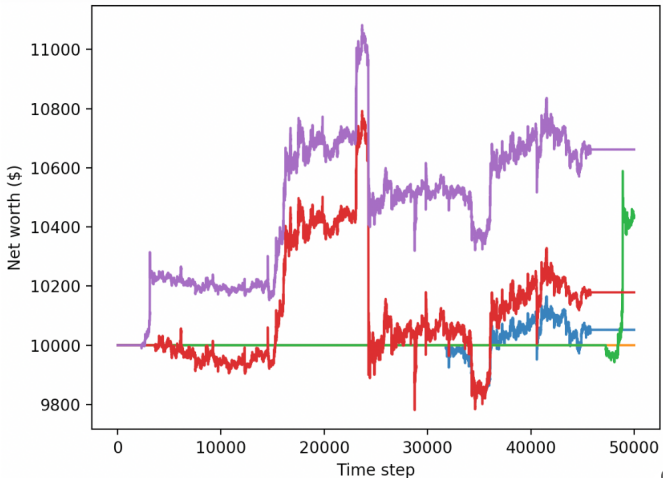

Figure 3: *The mentioned 2 changes significantly increase the profitability on test, fresh data*

8 Code link

https://github.com/rathorevipul28/RLTrader

9 Intallation and training Commands

Readme