Reinforced Learning of Cryptocurrencies:
A Study in Optimal RL Techniques for Stock Market performance

Colton Hill

December 17, 2021

1 Abstract

Cryptocurrencies are an incredibly lucrative, yet risky investment which tend to disregard many of the rules of normal stocks or commodities. They go up or down when modern monetary policies say they shouldn't, and might boom or bust for no apparent reason. While certain cryptocurrencies have achieved enough popularity to begin to follow predictable trends, this is not the case for most. As a result, there is a need for systems capable of advising trading decisions for cryptocurrencies. This project seeks to make a reinforcement learning system capable to discovering and exploiting the patterns of cryptocurrency price fluctuations in order to turn a profit. This is done by training an agent with various reinforcement learning strategies, including Q-Learning, TDn, and neural net Q-functions. These agents are then trained on random slices of cryptocurrency price histories and evaluated to determine which method is the most successful. Ideally, review of this agents decisions will help reveal underlying patters of trading cryptocurrencies which have been overlooked by humans as of yet.

2 Intro

The problem of creating a stock trading agent is an incredibly intricate and multi variant problem. Parameters for every major/minor element of the stock market can be accounted for, ranging from general market trends, to correlative stocks, to current events. In ideal conditions, algorithmic traders using these parameters can be very profitable. However, real-world conditions requires the constant tuning/redefining of these parameters, rendering the efficacy of these traders rather questionable. When it comes to cryptocurrencies however, the inherent volatility renders this issue of re-tuning exponentially worse. This problem is so bad, some economists question the applicably of cryptocurrencies to modern economic equations. Thus, the insensitive for a general reinforcement learner.

Because it is questionable if economists are even considering the correct data when in comes to cryptocurrencies, a general problem solver is required to discover patterns that could be overlooked. Obviously, a truly general problem solver capable of acting on all the possible predictive data of a stock currently only exists in fantasy and is well beyond the scope of this project. Therefore, this project seeks to determine if an agent can learn to exploit a stock market based solely on the price history of that stock, and not on any meta data of the environment.

3 Environment & Action Space

Below is the general structure of the environment:

- **Master Price History:** a prepossessed raw price history striped of all technical indicators and meta data.
- **State Representation (Current Virtual Price History):** From the master price history, the environment can be “started/refreshed” to produce a new subsection of normalized price history to use for a round of trading. The size of this subset is by default twice the size of the requested ”window” size (the amount of the user wants accesses to at any given time state.) The environment will count the initial starting point half way though the data set, so the first state has a history actually equal to the requested “window” size. For example, if the requested size of the environment is 128, then the subset of history will be 254, with time zero starting at index 127 in the subset.
- **Action space:** The action space of the agent is only of size 2. 0 indicates that the agent does not want to hold the stock, 1 indicates that the agent does want to hold the stock. Whether the agent is or is not actually holding the stock in irrelevant when it comes to the action space. The agent is only allowed to voice it desires, so to speak. The environment, which can read if the agent is actually holding the stock, then extrapolate the desire of the agent into the appropriate sell/pass or hold/buy action.
- **Reward:** The reward is simply the net profit made by the agent since the last sell action. This is most fully utilized by the TDn agent, however is used by all agents, save the NN Actor-critic agent, who is directly trained by the custom critic.

4 Methods

For each of the methods listed below, the same general Environment (typically 128-512 time periods), Agent structure, and evaluation criteria where used. The agent is capable of receiving the state representation, tracking it’s own portfolio (binary single stock holding & net worth), and responding to the environment with a ”sell/buy” actions. The “state” that the agent responds to is unique for each agent, but is derived from the price history slice. For example, agent 2 uses only the current price as the state, while agent 3 uses the price difference between a given day and the last $(P(t_n) - P(t_{n-1}))$ as the “state”. Rewards are determined by a net increase in worth once a stock is sold, or at termination if stocks are not sold.

Below is a chronological order of RL methods applied to the above problem.

1) **Random base line:**
   This strategy is common for projects such as this to establish a ”control” by which future experiments can be evaluated. Another common strategies also include doing all of one action, however because the only actions for this agent are buying & selling, only doing one won’t review any useful data. By using this baseline, general market trends can accounted for reviewed and determined to evaluate what is optimal and was isn’t.

2) **Q-Table learning with price based states:**
   For this strategy, the agent was initially dropped into a fresh market with no experience or history. However,

for the sake of having a standardized experiment, the agent was instead given a recent price history. This agents Q-table is a simple mapping of current prices ranges and actions to rewards. Naturally, the smaller these price history bin were, and the slower the training time proved to be. Do to the spurious evidence that this pattern mapping between discrete price ranges and future trends was useful, the following method was attempted.

3) Temporal Difference learning (with Δprice based states):
This method, rather then looking as normalized price bins, indexed states based on price difference from $P(t_1) - P(t_0)$. Here, the agent reviews the last n actions/state pair it made once a reward was received, which was decreasingly distributed from the present. While more successful then the previous incarnation, it was still very slow and only moderately more successful. This progress with relative price history spurred development for the next agent type.

4) NN Q-function with actor-critic evaluation:
Thought of as a natural extension of the TD(n) method, were $n \to \max$, using an NN in place of the agent's Q-table allows for the use of the entire simulated price history. Though this method should have a much greater start up lag than the former, the network should learn to place a much higher emphasis on more recent prices then distant ones. In order to take advantage of batch training, the normal reward structure had to be replaced by a critic so "ground-true" action could be assigned to each step. The critic in this case is simply a prescient traders who buys at every trough and sells a every peak, perfectly.

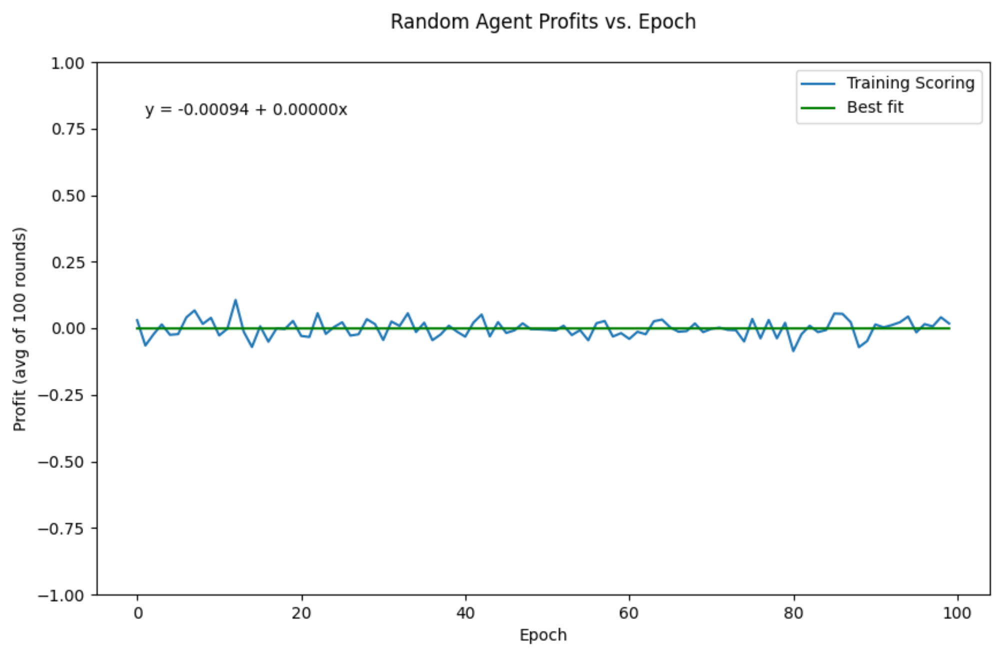
5) Gym wrapper with A2C/PPO:
Where as all of the other methods using were a result 100% person development, this method involves the use of predefined environment and agent template. This was done to test the validity of the development path, and to have a easy interface to use the A2C algorithm. For this wrapper, the technical indicators (daily high/low) were kept in, to see if those addition piece of info were necessary for the development of a successful agent.

5 Results

To judge the effectiveness of each of these methods, each agent was periodically evaluated by placing it in a fresh environment and averaging it's returns over 100 runs. These runs are made using a random pair of indices which are illegal to use in training. This prevents the agents form learning the validation data. This average profit is plotted against the number of epochs the agent had been train on.

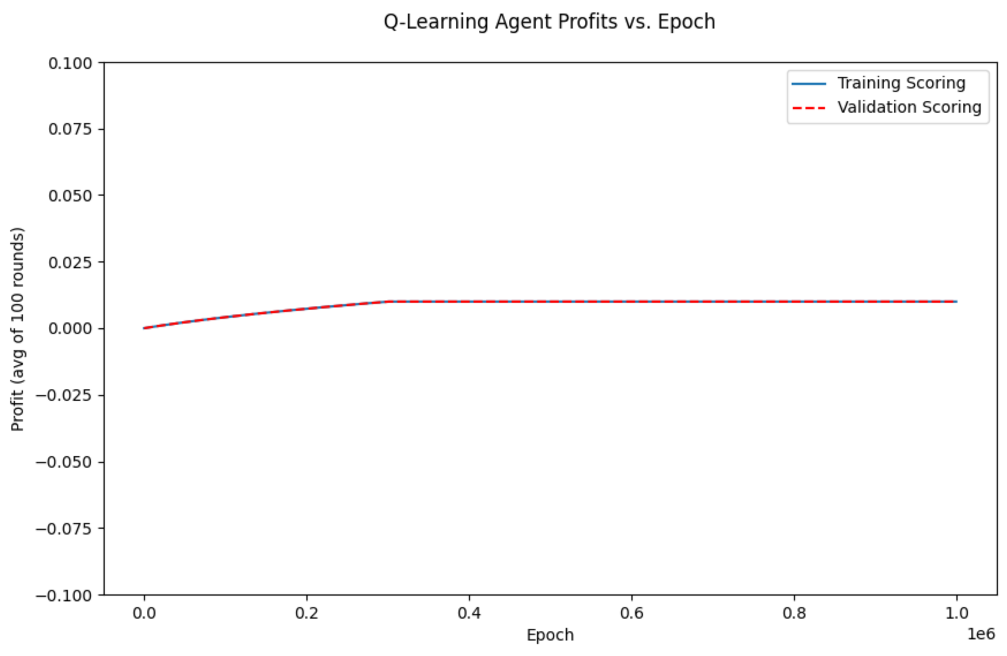
Note, for each of these graphs the exploration was slowly dialed down exponentially, similar to that of a half-life equation, such that by epoch 10,000, exploration is at 50%, and at epoch 20,000, exploration is at 25%.

1) Random base line:
Due to the general upwards trend of the cryptocurrency market, it is necessary to make a random agent to determine if this positive trend will have a statistically significant impact on the profits of the agents. Should it be found that the agents are being artificially boosted/hindered by the environment, this baseline random agent will provide a "neutral" axis which other agents will be measured against. The experiment represented by the following graph shows the average profit the random agent received in one Epoch (100 rounds) over 100 Epochs.



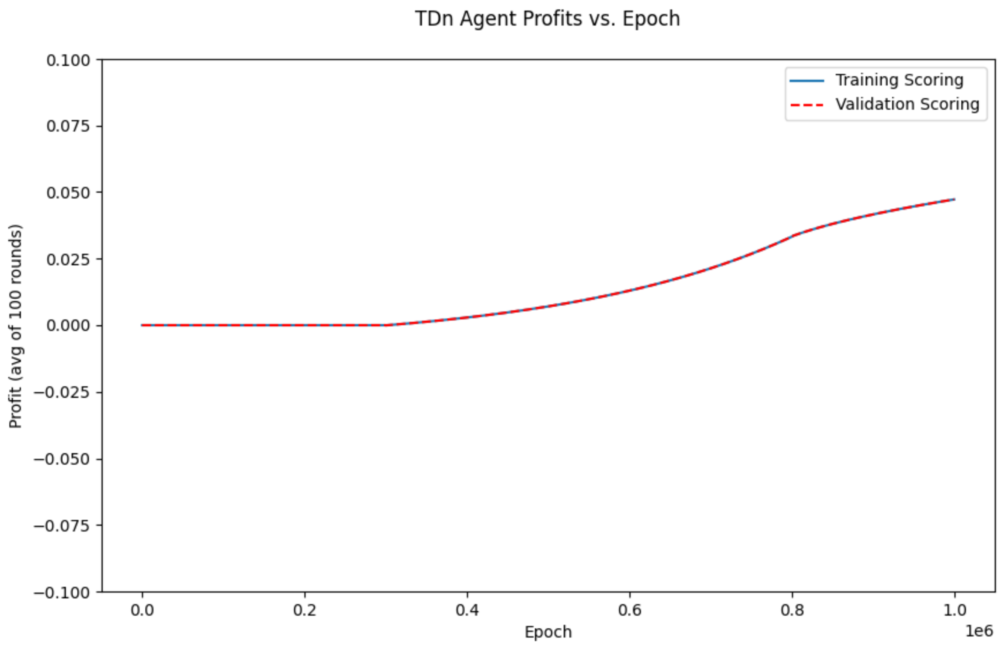
As one can see from this graph, there is no significant trend upwards or downward. This mean that each following agent's profit margin can be taken at face value. Scoring a 1 will indicate a 100% increase, and conversely scoring -1 indicates a 100% decrease.

2) Q learning with profit based rewards:
Below is the graph which represent the evaluation of the simple Q-table agent under the same experiment conditions, except that it was allowed to run for 100,000 epochs. Clearly, these results indicate the patterns of price histories (if they exist) are more complex then a direct price mapping.



Here, one can see that this agent is not learning to solve the environment well. It believes it is finding a pattern associated with the absolute price bins, however, it is not significantly different from he baseline agent.

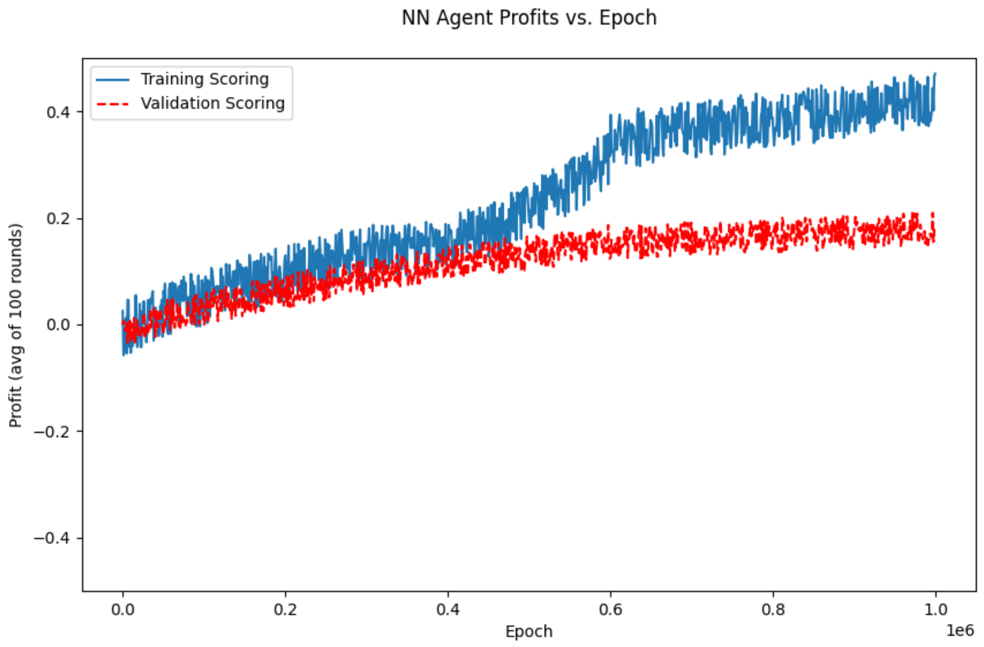
3) Temporal Difference learning:
Below is the graph representing the results of the TDn agent experiment, also run for 100,000 epochs.



These results are interesting as the improvement in the graph doesn't occur until exploration is approximately 10% or less. Normally, we would expect a some what linear increase in performance as random action are diminished. This shows how the TDn method continues to improve and learn, despite when exploration is "off". Despite this continued progress, it is still far from what would one would expect for a smarter method. However, this shows that this relational approach to prices is more sounds and should be explored more.

4) NN Q-table with actor-critic evaluation:

Below are the results for the NN actor-critic agent, also run for 100,000 epochs. Due to the way the agent is trained with batches, the graph is not averaged and sampled as heavily as the other agents, hence the noisy training and validation scores.



Again, while more stable then previous agent, the validation score is not matching the training scores, indicating that there might be meta data still slipping in which the neural net is capitalizing on, or the Net is over-fitting to the train data. Despite this disparity between the training and validation sets, this agent is preforming about 3 times better then the previous agent, pleating around 15% profit. Still, this is not the 40% purported by other actor critic method's. So, this begs the question, is there a better way to do this method, or is this dataset particularly difficult.

5) Gym-Wrapper Agents (A2C & PPO):

Because these two methods where on developed as part of this project, they could not be subjected to the same experiment as the rest of the agents. These methods come courtesy of the "stable-baselines, gym-anytrading" wrapper. One thing to note is that these predefined agents both use the technical indicators in there assessment, and come with learning transfer from other stocks. Despite these advantages, the A2C agent was only able to return 9% after several hours of training, and the PPO only 11%. I suspect this is a result of the learning transfer not being applicable to the cryptocurrencies, but it might also be a manifestation of the difficultly of this dataset.

6 Summary

Below is a recap of the final evaluations for the above methods:

Method	**Profit**
Baseline:	0%
Q-table:	1%
TD(n):	5%
NN-actor/critic:	15%
Gym-A2C:	9%
Gym-PPO:	11%

From review of these findings, particularly the Q-Learning agent, one can assert that the absolute price of a cryptocurrency has little to no bearing on the future of the stock. Next, based on the TDn agent's improvement, one can also say that there is a slight influence previous price changes have on current price trends. Finally, based on the success for the NN agents, one can surmise that there is a pattern in previous stock prices which can help predict future prices.

7 Conclusions

From the viewed experiments, there does appear to be a weak link between the raw price histories of a cryptocurrency, and it's future trends. What that exact pattern is likely impossible to extrapolate, however, that pattern is present enough for a reinforcement learning to take advantages of it, particularly agents with a Neural Net learning function. This advantage is very weak however, and needs to be averaged over many hundreds of iteration before it can be seen. At the stage of development of this project, this advantage is very difficult to exploit and is not ready for real world application.

8 Sources

- Git Hub for this project:
  https://coltonchill.github.io/
- Example stock trading agent:
  https://towardsdatascience.com/deep-reinforcement-learning-for-automated-stock-trading-f1dad0126a02
- Tutorial RL-stock trader:
  https://www.baeldung.com/cs/reinforcement-learning-neural-network
- Gym wrapper RL-stock trader:
  https://www.youtube.com/watch?v=D9sU1hLT0QY
- Tensorflow Documentation:
  https://www.tensorflow.org/api$_docs/python/tf$
- Pandas Documentation:
  https://pandas.pydata.org/docs/