Designing Short-Term Trading Policies for Litecoin Cryptocurrency Using Reinforcement Learning Algorithms

Jenny Kim, Nguyet Minh Phu, Ellen Wang

December 7, 2018

1 Abstract

In this paper, we investigate ways to optimally trade Litecoin where short-selling is allowed. We compare a one-day greedy policy using a fully-connected neural network as a price estimator with policies found by modeling the problem as a Markov Decision Process (MDP) with model uncertainty. Our policy obtained using Maximum Likelihood model-based reinforcement learning outperforms that obtained from using Q-learning, a model-free reinforcement method. Overall, we found that short-selling is very difficult, with large risks and small returns.

2 Introduction

2.1 The History of Cryptocurrencies and Litecoin

Due to the increasing popularity of Bitcoin, the year 2011 saw an emergence of the rival cryptocurrency, Litecoin. Although Bitcoin is the World's first decentralized currency, Litcoin does have its own advantages. Bitcoin's network can only ever have 21 million coins while Litecoin can have up to 84 million [PS18]. Litecoin's larger maximum results in a lower price per unit. Additionally, Litecoin is capable of of confirming more transactions per minute than Bitcoin [PS18]. These properties make Litecoin a good short-term investment target, especially for traders who are starting with no money.

Furthermore, cryptocurrencies are known for their extreme price volatility. For example, it is well remembered that on December 28, 2017, Bitcoin was trading at $15,433.73 and a mere 40 days later on February 9, 2018, Bitcoin had fell to less than half of its December price, oscillating in the $6,000-$9,000 range.[1] Although instability is normally a concern for most investors, high volatility also means there are great gains to be had on a short-term basis. Therefore, we choose to exploit the volatile nature of Litecoin by engaging in short-selling tactics.

In order to do so, we model our program as a Markov Decision Process (MDP). Usually, financial trading lends itself naturally to a MDP framework. One can think of trading as a sequential decision problem in which market uncertainty can be modeled by a stochastic transition function. This unpredictability permeates into the reward function as well, where risk-taking actions may yield great rewards or serious losses [Rus10].

2.2 Understanding Short-Selling

Normally, one thinks a trader can only sell a security if he owns it. However, in trading, there is also the case of short-selling. Short-selling is the sale of a security that the seller has borrowed. In other words, a trader can sell a security even when he does not own it at the present moment. However, he will have to buy the security back at a later time to give the security to the buyer. A short-seller profits if a security's price declines. In other words, the trader sells to open the position and expects to buy it back later at a lower price and will keep the difference as a gain [Jag18].

Recently, short-selling has been frequently featured in the news because of Tesla. Tesla is one of the frequent targets of short-sellers, who through short-selling bet billions of dollars that Tesla's stock will

[1] https://www.forbes.com/sites/jayadkisson/2018/02/09/why-bitcoin-is-so-volatile/#7620ecfc39fb

eventually decline [Bou18]. These short-sellers have faced both ups and downs. In general, short-selling is considered extremely risky and should be done only by very experienced traders. Thus, we become interested in seeing whether we can use algorithms learned in CS238 to tackle this very challenging problem in finance.

3 Problem Statement

Our goal is to find a good policy for trading Litecoin over a period of 100 days where we begin with zero Litecoin. In our problem, we allow short-selling. To simplify the problem, we also define a trading cycle of 100 days. Since we allow short-selling, during each trading cycle, we can sell a Litecoin even when we do not currently own it. This means our position, which refers to the number of Litecoins we currently hold, can be negative.

On each day, we can either buy one unit of Litecoin, sell one unit of Litecoin, or do hold on to the units we have. However, on the one 100^{th} day, we need to settle our position. At this stage, if we sold coins that we did not have, we must buy back the coins now. If we bought coins and they have not yet been cashed out along the way, we must sell them now.

The utility is the sum of the reward at each time step (undiscounted) and the settled position at the 100^{th} day. To train our models and evaluate our the performance of our policies, we use historical data of Litecoin's daily opening price.[2] The data starts on April 27, 2013 and goes on until November 12, 2018; however a few dates towards the beginning were missing prices, so we backfilled any gaps we encountered.

4 Baseline policy: One-day Greedy Policy with Neural Net

Our baseline policy models what an average person who is risk-averse would do: use one-day greedy policy. In this section, we explain the one-day greedy policy as well as the use of a neural network to approximate an average person's ability to predict the Litecoin's next day opening price. We then analyze the performance of this short-sighted strategy.

4.1 One-day Greedy Policy

In our one-day greedy policy, the trader exploits the price difference between today and the next day. If the trader predicts that the price of Litecoin will increase tomorrow, they will choose to buy one unit of Litecoin today and sell one unit tomorrow to gain the difference in price. Similarly, if they predict the price will decrease, the trader will sell a unit of Litecoin today and buy a unit tomorrow, again earning the difference in price. Therefore, for the baseline, the trader always chooses their actions in pairs of $(buy, sell)$ or $(sell, buy)$, and given a pair of days, the second day's action is dictated by that of the first day.

4.2 Neural Network as a Price Predictor

To carry out the one-day greedy policy above, one will need to be able to predict how the price of Litecoin will change in the next day. We use a neural network to approximate an average person's ability to predict the change in price of Litecoin.

Given a historical window of 10 prices up to and including the current price, the neural network is trained to predict whether the price of Litecoin will increase, decrease or stay the same (within a 2% threshold of the current price). The neural network is fully connected with 100 hidden layers with 500 neurons each. Input o to the neural network is the history of 10 prices and output is one of classes $c \in \{1, 2, 3\}$ that corresponds to tomorrow's price increasing, decreasing or staying the same, respectively. The neural network is trained using stochastic gradient descent to minimize the cross-entropy loss for three classes of increase, decrease or stay the same.

[2] https://coinmetrics.io/data-downloads/

$$\text{Cross-Entropy Loss} = -\sum_{c=1}^{3} y_{o,c} \log(p_{o,c})$$

where $y_{o,c}$ is a binary indicator (0 or 1) if the given price history, o, shows that the price will increase, decrease or stay the same.

4.3 Performance of One-Day Greedy Policy

Using our one-day greedy policy to trade for period of 100 days, we see that the optimal amount of money varies as in figure 1. The spikes in the figure are due to when we buy then sell the next day, or sell then buy the next day. For example, the first downward spike at time step 0 to 1 in figure 1 happens because at time step 0, we predict that the price will increase at time step 1. Therefore, we buy one unit of Litecoin at time step 0, causing us to have a net loss, then at time step 1, we sell the coin at a higher price, increasing our overall position by a tiny amount. Thus, through a series of small increments, using a one-day greedy policy will allow us to make a small profit.

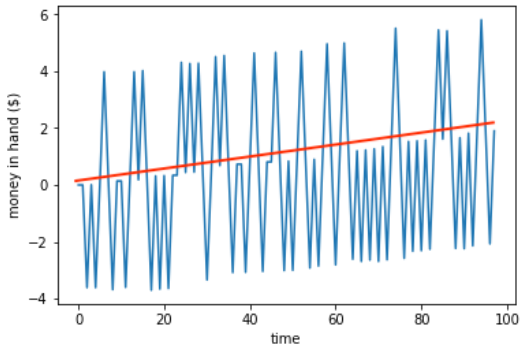


Figure 1: Given the exact price dynamics (or a perfect neural network), our net profit on each day from adopting the one-day greedy policy is captured by the above trend (shown in red). The visualization here is generated by choosing a random time in the history to begin short-selling Litecoin for a 100-day period.

The predictions from a perfect neural network are simply generated using the future prices in the actual data set. This shows us the upper bound on our utility of following this policy given that we can predict the price change with 100% accuracy. The red line indicates the change in our net position over time during the trading period of 100 days, and notice that our position strictly increases. In this particular example used for figure 1, we would earn $1.90.

However, after training with 2000 examples drawn from historical Litecoin data a with 90%/10% train/test split, our neural network only achieves 45% test accuracy. Again, applying our one-day greedy policy to the same trading period now with a neural network that was blind to the true next day price, we obtain figure 2. We found that for the same trading period, the neural network always predicted a decrease in price, prompting a sell action followed by a buy action. However, the price does not always decrease. The red line indicates the change in our net position throughout the trading period, and notice our position fluctuates. Overall, with an imperfect neural net trained to the predict tomorrow's price based on a 10-window price history, we would earn only $0.08 which is far less than the optimal utility for our one-day greedy policy.

5 Finding optimal policy via Markov Decision Process

The one-day greedy policy based on neural network predictions has rather poor performance (See Results Section). In this section, we analyze how to find the best trading policy by modeling the problem as a Markov Decision Process. We first explain how the problem can be modeled as a MDP with model uncertainty. Then, we explain how we applied both model-free and model-based reinforcement learning methods to approximate an optimal trading policy.

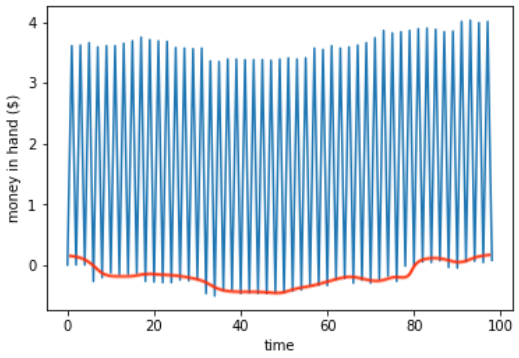


Figure 2: Using the price predictions made by our neural network, our net profit on each day from adopting the one-day greedy policy is captured by the above trend (shown in red). The visualization here is generated for the same time period as in Figure 1.

5.1 MDP Representation

5.1.1 State

Each state, $s \in S$, is a vector of attributes with length 2 where m is the gradient of a least squares regression line through a 10-window price history and n is the number of Litecoins held. As a result, the state also depends implicitly on the time t in history.

$$s = (m, n)$$

Because the gradient is a continuous variable, we divide the gradient values into 21 different, uniform intervals ranging from -1.0 to 1.0. When fitting the line, we fit a least squares regression line using equation (3) with the recent past ten prices that minimizes sum squared error and return the interval to which the gradient of the line belongs.

$$\hat{y} = a + bX \tag{1}$$

$$b = \frac{(\sum X_i - \bar{X})(\sum y_i - \bar{y})}{\sum (X_i - \bar{X})^2} \tag{2}$$

$$a = \bar{y} - a\bar{X} \tag{3}$$

5.1.2 Action

There are three possible actions: buy, hold, sell which are represented in the policy array as -1, 0, and 1, respectively. While our action does not directly affect the price of Litecoin, $s[m]$, it does affect the number of Litecoins we have at any given time, $s[n]$.

$$a \in \{-1, 0, 1\} \tag{4}$$

5.1.3 Transition

There is some stochasticity in our ability to take the desired action. Because new blocks need to be mined in order for a cryptocurrency transactions to be approved, miners receive a transaction fee that is a percentage of the highest bid. This means that one many not always be able to buy a unit of cryptocurrency should they are out-bidded. We model this uncertainty in how our buy action affects $s'[n]$ by setting the probability that our buy order actually goes through to be 0.9:

$$T(s[n],\ a = -1,\ s[n] + 1) = P(s'[n] = s[n] + 1 \mid a = -1) = 0.9 \tag{5}$$

Our model also limits the minimum and maximum number of coins that we are able to hold at any one time during our 100-day trading period. If $s[n]$ at any given time $t < 100$ reaches -50, meaning we

have sold 50 coins that we do not truly have, we will not be allowed to take buy or sell actions for the rest of the trading period. This encodes the fact that in reality, a limit is imposed on how much a trader can short-sell to prevent the case when they cannot buy back that many coins in the end. The same is true if $s[n]$ reaches +50. This encodes the fact that in reality, a trader only has a limited amount of money to buy coins.

$$\forall s \in S.\ s[n] = -50 \vee s[n] = 50 \implies T(s, a, s) = 1 \tag{6}$$

Because our horizon is 100, our generative models have also encoded at $t = 100$, $s_{t=100}$ is a terminating state. For all other cases in which $t < 100$, $-50 < s[n] < 50$, s' is uncertain because $s'[m]$ depends on tomorrow's price, and when $a = -1$ (buy) $s'[n]$ is also uncertain as previously noted.

5.1.4 Reward

Let $price_t$ be the current price of Litecoin in state s_t. Then for all states s_t where $-50 < s_t[n] < 50$ and time $t < 100$, the reward is as follows:

$$r(s_t, a) = \begin{cases} -price_t & a = -1 \\ 0 & a = 0 \\ price_t & a = 1 \end{cases} \tag{7}$$

Recall that if $s[n] = -50$ or $s[n] = 50$, the transition model ensures we will hold until time $t = 100$. Finally, for any s at time $t = 100$ we must settle our position, meaning the reward is calculated as follows:

$$r(s_{t=100}, a) = price_{t=100} * s_{t=100}[n] \tag{8}$$

6 Methods

6.1 Model-free RL: Q-learning

Q-learning is one of the model-free reinforcement learning algorithms, which can be derived from the Bellman Update formula. Based on the modeling of MDP, we used Q-learning to extract the best policy. The formula to update Q-learning is as follows:

$$Q(s, a) \leftarrow Q(s, a) + \alpha(r + \gamma max_{a'} Q(s', a') - Q(s, a)) \tag{9}$$

In order to train the Q-learning, we first defined a simulator according to the MDP as described above. More specifically, given timestamp, action, state, and current price, the simulator returns the reward and next state.

We chose the gamma value (discount factor) to be 0.9 and alpha value (learning rate) 1.0. With hundred iterations through the data, Q-learning take about five minutes to complete. To balance exploration and exploitation, we adopt an ϵ-greedy exploration strategy where $\epsilon = 0.2$.

6.2 Model-based RL: Maximum Likelihood with Dyna

Since Q-learning is a model-free technique, we also implemented the Maximum Likelihood model-based reinforcement learning algorithm. We wanted to compare the two and see whether Litecoin's price history alone is enough to inform the simplest possible model.

We trained for 10,000 trials across our training data using the simulator we developed for Q-learning. At each step of a trial, we chose the action based on the ϵ-greedy exploration strategy where $\epsilon = 0.2$. We then estimated the transition and reward models directly from our simulated experience by keeping a count of the number of times we've encountered a transition using a $N(s, a, s')$ table and the sum of rewards from taking an action in a state using a $\rho(s, a)$ table. We then estimated the transition and reward model using the following equations provided in the course textbook [Koc15]:

$$N(s, a) = \sum_{s'} N(s, a, s')$$

$$T(s'|s,a) = N(s,a,s')/N(s,a)$$
$$R(s,a) = \rho(s,a)/N(s,a)$$

To update our Q values, we used the Dyna algorithm by performing the following update at the current state as well as 25 random states from our history:

$$Q(s,a) \leftarrow R(s,a) + \gamma \sum_{s'} T(s'|s,a) \max_{a'} Q(s',a') \quad \text{where } \gamma = 0.9$$

This reduces our computational expense of running a dynamic programming algorithm by only solving a portion of our MDP at each time step.

7 Results

We use two evaluation metrics to evaluate the performance of our policies: money gained and percentage gained (compared to the worst case scenario). In this section, we explain each metric, the reason why we use it, and the performance of each policy using the metric. We also discuss the implications of our results.

7.1 Evaluation Metrics

7.1.1 Money gained

The money gained by a policy refers to the average amount of money that the trader can gain during a one-hundred-day trading period if he follows the policy. We compute the money by using equation 10, where k is the start date of the ith trial, p is the price of Litecoin, and S_i is the money gained during the one-hundred-day trading window. $\pi_k + j$ is the policy for the $k + j$ timestamp, which could be -1 (buy), 0 (hold), and 1 (sell).

$$S_i = \sum_{j=1}^{100} \pi_{k+j} \cdot p_{k+j} \tag{10}$$

We generate a policy for thirty random days to calculate the average amounts of money we could gain with different methods. In other words, formula for the average amount of money gained is $P_{avg} = \frac{1}{30} \sum_{i=1}^{i=30} S_i$. We compute these values and report the average gain as well as the standard deviation for three different methods in the Table 1.

	Neural Network	Q-learning	MLE
Money Gained ($)	-0.30 ± 1.06	4.32 ± 13.00	12.91 ± 29.67

Table 1: This table shows the money gained for different methods.

7.1.2 Percentage gained

In short-selling with a trading cycle of 100 days without any restriction on our position, assuming we can settle any position at the end of the trading cycle, in the best possible case, we will buy whenever the price is below the price on the 100^{th} day and sell whenever the price is above the price on the 100^{th} day. Thus the maximum amount of money that we can gain, or the utility of the optimal policy π^*, will be the sum of the absolute value of the difference between the price on each day and the price on the 100^{th} day.

In the worst possible case, we will sell whenever the price is below the price on the 100^{th} day and buy whenever the price is above the price on the 100^{th} day. In this case, we will lose an amount of money equal to the sum of the absolute value of the difference between the price on each day and the price on the 100^{th} day.

We will then compare the utility of our policy with this upper and lower bound. Having this upper and lower bound is important because the utility of the optimal policy starting at different time will be different.

The equation to calculate the average percentage gained is in (13). S_i stands for the ith trial's utility. The $\overline{U}$ stands for the upper bound of the utility for the ith trial's day. The $\underline{U}$ stands for the lower bound of the utility for the ith trial's day, and thus $\underline{U} \leq S_i \leq \overline{U}$. Table 2 shows the average percentage gain as well as the standard deviation using our neural network-informed one-day greedy policy, Q-learning, and MLE.

$$\overline{U_i} = \sum_{j=100}^{j=1} |p_{k+j} - p_{k+100}| \tag{11}$$

$$\underline{U_i} = -\sum_{j=100}^{j=1} |p_{k+j} - p_{k+100}| \tag{12}$$

$$\%_{avg} = \frac{1}{30} \sum_{i=1}^{i=30} \frac{S_i - \underline{U_i}}{\overline{U_i} - \underline{U_i}} * 100 \tag{13}$$

	Neural Network	Q-learning	MLE
Percentage Gained	49.80% ± 1.45%	51.3 ± 7.58%	58.25 ± 21.52%

Table 2: This table shows the average percentage gains for different methods.

7.2 Discussion on performance of each policy

Overall, we realized it is hard to gain significant amount of money with short-selling. Historical price data alone is not sufficient information to make successful investments. Still, we were able to get a better performance using Q-learning and MLE with Dyna than with the one-step greedy policy.

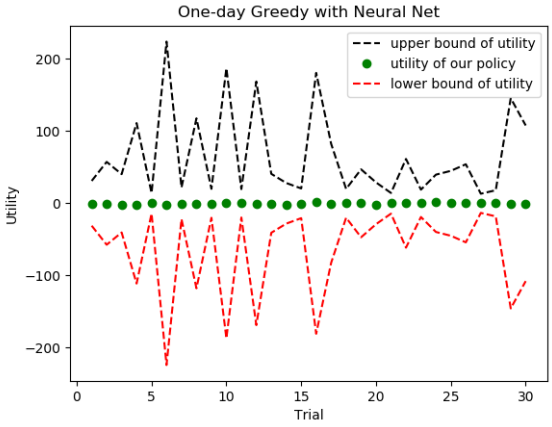
The poor performance of one-step greedy policy with neural network is expected as the greedy nature of the policy means that we can only gain through the small difference in price of every two consecutive days. In fact, even if we know exactly how the price change on the next day, we will only be able to gain a limited amount of money. In our simulation for evaluation, when we replaced the prediction of the neural net with the actual direction of price change, the average money gained over 30 trials was only \$2.96 ± 1.65. Due to inaccuracy of the neural net (it has a test accuracy of 45%), the money gained was even lower as expected.

Moreover, MLE with Dyna updates performed better than Q-learning. Because Q-learning is a model-free method whereas MLE with Dyna is a model-based method, a possible explanation of this could be that the price trend was not too complicated, making the transition and the reward function reasonable to learn. Figure 3 shows that MLE with Dyna follows the trend of the upper and the lower bound more closely than Q-learning does. This indicates that model-free learning method may have failed to capture the trend in the price while model-based learning method effectively learns the trend.

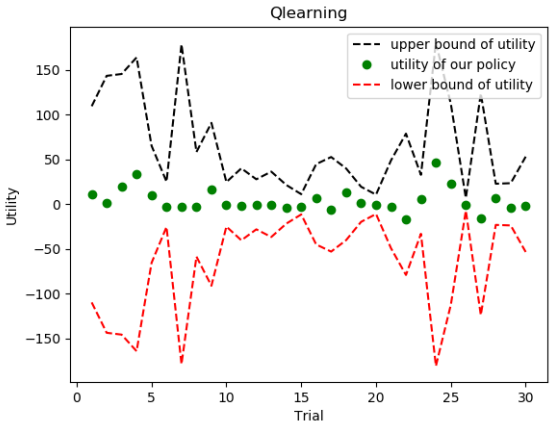
Still, for all three methods, we see large standard deviations in the average money gained. This suggests that all three policies sometimes lose a great amount of money and sometimes earn a great amount of money. Since cryptocurrencies have volatile prices, using these algorithms may result in large losses or large gains even though on average they result in positive net profit.

8 Conclusion

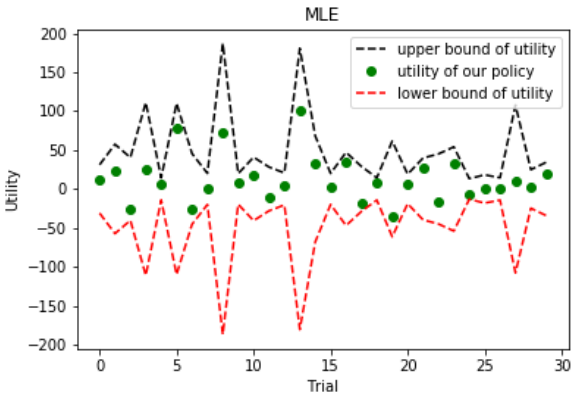
In this paper, we discuss how Q-learning and MLE algorithms performed better in cryptocurrency trading than a naive one-day greedy strategy. MLE with Dyna captures the price history trend more accurately, followed by Q-learning and the one-day greedy selling baseline strategy. Still, we can see large standard deviations for both reward metrics, indicating that investing using the Q-learning or MLE with Dyna could lead to both large gain or loss. Thus, there is a high risk involved in trading.



((a)) Neural Network



((b)) Q learning



((c)) MLE with Dyna

Figure 3

We propose two main ideas for future work to improve the predicting accuracy and thus increasing the expected gain. As mentioned in the above section, the price history of cryptocurrency itself may not be a useful indicator on how the next day's price would change. We could incorporate other data such as economic trends and current events to predict the litecoin prices as researched in [PG17] [Kim+16]. We expect having relevant features such as media data on cryptocurrency would help us determine the next day's cryptocurrency prices more effectively.

We could also try to use policy learning algorithms to increase the performance. In the 2001 publication in the IEEE Transactions on Neural Networks journal, authors Moody and Saffell presented an adaptive direct reinforcement (DR) for optimizing trading systems that outperformed supervised methods such as Q-learning when simulating with real financial data [MS01]. Their algorithm called recurrent reinforcement learning (RLL) differs from dynamic programming and reinforcement algorithms in that it does not learn a value function. Instead, their approach makes use of instantaneous performance feedback at each time step to adjust their policy. One of the most notable limitations of our current value function-based approaches is that Q-learning and MLE require the context of discrete state and action spaces. We could use the similar approach to replicate the positive results of Moody and Saffell by conducting an adaptive policy search rather than being confined to a model or precomputed value function.

References

[MS01] John Moody and Matthew Saffell. "Learning to trade via direct reinforcement". In: *IEEE transactions on neural Networks* 12.4 (2001), pp. 875–889.

[Rus10] Andrzej Ruszczyński. "Risk-averse dynamic programming for Markov decision processes". In: *Mathematical programming* 125.2 (2010), pp. 235–261.

[Koc15] Mykel J Kochenderfer. *Decision making under uncertainty: theory and application*. MIT press, 2015.

[Kim+16] Young Bin Kim et al. "Predicting fluctuations in cryptocurrency transactions based on user comments and replies". In: *PloS one* 11.8 (2016), e0161197.

[PG17] Ross C Phillips and Denise Gorse. "Predicting cryptocurrency price bubbles using social media data and epidemic modelling". In: *Computational Intelligence (SSCI), 2017 IEEE Symposium Series on*. IEEE. 2017, pp. 1–7.

[Bou18] Neal E. Boudette. *Betting Against Tesla: Short-Sellers Make Their Case*. Sept. 2018. URL: `https://www.nytimes.com/2018/09/17/business/tesla-stock-shorts.html`.

[Jag18] John A. Jagerson. *Short Selling*. Nov. 2018. URL: `https://www.investopedia.com/terms/s/shortselling.asp`.

[PS18] M Padmavathi and RM Suresh. "Secure P2P Intelligent Network Transaction using Litecoin". In: *Mobile Networks and Applications* (2018), pp. 1–9.