Trading agent with virtual money/Bitcoins

Course Project. COMP-767: Reinforcement Learning Learning, Winter 2018

Ahmadreza Godarzvandchegini
ahmadreza.godarzvandchegini@mail.mcgill.ca
McGill ID: 260795422

Project Repository Available at: https://github.com/madarez/ReinforcementLearningProject

I. Introduction

Financial data, despite been studied by many, do not seem to reasonably be tractable on how investment scheduling should be for a rather maximum profit. This is no surprise given that exchange markets are large zero-sum games meaning that the profits one makes is at the expense of someone else's loss [1]. If certain cue in market indicates a buying signal, then there remains no seller to sustain losses of that trade. That yields the rationale to keep the trading algorithms a secret in order to make a potential advantage over other traders.

Theoretically, reinforcement learning attempts at finding a policy that can produce optimal value of rewards in stochastic environments [2]. With a surging attention that machine learning is receiving, it is worthwhile to attempt to solve old practical problems such as market trading. Putting the stealthy nature of strategies exercised by traders into perspective, it is not clear if one has not yet found a trading policy yet.

To the best of author's knowledge, other than cryptocurrencies the only market that is continuously open is Forex. These markets do not experience interruptions of a regular hours market which make them closer to Markov Decision Processes' assumptions. Furthermore, decentralized cryptocurrencies are not regulated by central banks [3] which could sound as a compelling alternative to Forex. Despite the mentioned advantage, one should bear in mind that cryptocurrencies' time series are non-stationary just as any other financial data. For instance, in January 2018, cryptocurrencies' volatile prices burst bubble and made a historical market crash [4].

Bitcoin [5] is by far the most popular cryptocurrency surveyed by Forbes [6]. Active exchange markets such as Bitfinex [7] make an ideal environment for an agent to learn to trade as unmet buying orders and the selling orders have a narrow gap. This is thanks to a high traffic of Bitfinex users and keeps the agent's actions executable.

All of the aforementioned properties encourages the current study on how to transform the reinforcement knowledge into this practical domain that respect most of theoretical assumptions. In an effort to train a reinforcement learning agent to trade in an exchange market, the states' representations are engineered and Dyna-Q algorithm [8] is applied and presented in the following sections. Lastly, we concluded with a discussion about the results and future work.

II. Problem representation

In this work, the agent's interaction with the environment has been modeled as a tabular MDP with buy, sell, and hold actions [2], and the environment is taken to be a basic simulation of Bitcoin's exchange market. The historical traded price of Bitcoin was collected from Bitfinex's historical data until April 18, 2018. Trading fees in Bitfinex is approximately 0.2 percent of each transaction.

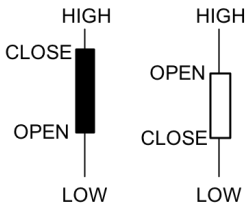
The curated database contains 172137 candlestick tuples whose entries come from opening price, closing price, minimum price, and maximum price during 15 minute intervals 2. These data are normalized to 1 millionth of a Bitcoin also known as Satoshi (฿$1 = 10^6$ satoshi). Then, this information is used to form state representation of at each timestamp as follows:

$$\phi_0(s) = (coin = 0)$$
$$\phi_1(s) = (balance > CLOSE)$$
$$\phi_2(s) = (\frac{\text{OPEN}}{\text{LOW}} < 1.5)$$
$$\phi_3(s) = (\frac{\text{CLOSE}}{\text{LOW}} < 1.5)$$
$$\phi_4(s) = (\frac{\text{HIGH}}{\text{LOW}} < 1.5)$$
$$\phi_5(s) = (\frac{\text{VOLUME}}{\text{LOW}} < 4)$$
$$\phi_6(s) = (\frac{\text{CLOSE}}{\text{OPEN}} < 1)$$
$$\phi_7(s) = (\text{isWeekend})$$

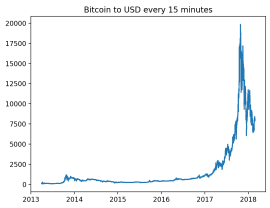
First two features indicate affordability of the agent at each time and the last feature is True only if the trade day is Saturday or Sunday. The remaining features are only set at heuristic thresholds to indicate whether the feature is relatively high or low.

Actions are set to only trade one satoshi at a time or abstain trading for one timestamp. This is particularly set limit the range of the actions to learn as with MDP formalizations, it is not straightforward to credit similar actions similarly and the agent is to learn each action independently. These simplifications are taken in order to assist with the convergence of agent's policy iterations.

The reward accrued at each timestamp is simply the amount of money the agent cashes. This makes sure that the agent liquefies its coins at right times to earn profits in USD. This



(a) Japanese candlesticks



(b) BTC-USD in Bitfinex over time

Fig. 1: (a) Candlesticks tuple is an excerpt of the traded information in a certain interval (b) Opening price of exchanged Bitcoin price as of the first available 15 minute candlestick on Bitfinex platform

is to essentially move towards the ultimate goal the agent should have, i.e., to secure the investments while taking risk to maximize profits.

A. *Rewards*

Let R_t, B_t, C_t, and P_t respectively denote reward, balance, number of coins, and last closing price at time t. Three types of rewards were considered:

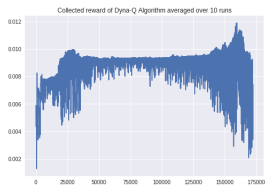
1) $R_t = B_t - B_0$
2) $R_t = B_t + \bar{B}_t - B_0$ where $\bar{B}_t$ here acts as a queue for each transaction the agent executes before it can sell the just bought coin or buy with a just yielded money out of a sell. It is a withholding strategy preventing the agent to frequently involve in selling and buying coins (augmented reward).
3) $R_t = B_t + \bar{B}_t - B_0 + P_t \times (C_t + \bar{C}_t)$ where $\bar{B}_t$ and $\bar{C}_t$ are queues similar to the previous reward. It is noteworthy that this reward is an imaginary reward in terms of real gains of one's capital. Assets bought and not yet liquidated are not real cash though this is an indicator most of the traders rely on when trading (imaginary reward).

III. Algorithm Selection and Implementation

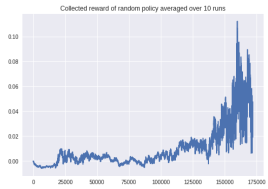
As the curated database is very limited in size, and the data cannot be generated freely, it is crucial to leverage maximum usage of data. If the problem was defined in a function approximation, we could follow LSPI [9] or fitted-Q a variant of fitted value iterations [10]. LSPI mixes different features with different units into a value representation. As it does not have a justifiable interpretation of investors, this was not chosen to further study this model. Fitted-Q was not finished in the time frame of this project, but remains an interesting line of study. Along tabular MDP algorithms, model-based Dyna-Q was implemented considering the constraint described earlier in regards to the historical data size.

A. *Hyperparameter scheduling*

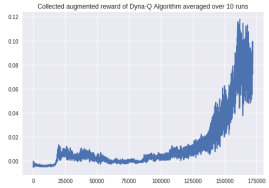
The hyperparameters in the current project would be ϵ, exploration probability, and α, learning rates. It is known that in order to have a convergence in the TD updates,



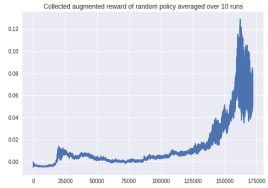
(a) Regular reward with learned policy-final capital: 0.046537



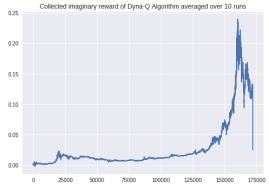
(b) Regular reward with random policy-final capital: 0.010614



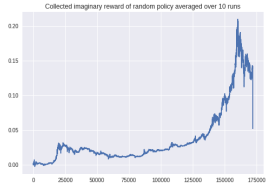
(c) Augmented reward with learned policy-final capital: 0.045200



(d) Augmented reward with random policy-final capital: 0.006030



(e) Imaginary reward with learned policy-final capital: 0.010068



(f) Imaginary reward with random policy-final capital: 0.005572

Fig. 2: In all of the cases, the agent's learning incurs higher profit than random policy. The agent may achieve higher rates maximum as it learns with augmented reward and yet better with imaginary reward. However, as can be inferred from the total capital averaged, the agent is also most likely losing capital with market crashes afterwards.

the learning rates should follow the stochastic approximation theory condition.

$$\sum_{n=1}^{n} a_n(A) = \infty \text{ and } \sum_{n=1}^{n} a_n^2(A) < \infty$$

Based on above, $\alpha_t = \frac{1}{10+t}$ and $\epsilon_t = \begin{cases} 0.5 & t \leq 500 \\ 0.1 & o.w \end{cases}$.

IV. Testing and Validation

The agent were given 1 cent in the beginning and traded at next timestamp based on previous timestamp's information. In the following, one can see a summarization of the evaluating the policies on rewards defined above:

TABLE I: Final profit made at each reward

	profit	
	validation set	test set (Kaggle)
Multinomial	0.874	0.778
tf-idf multinomial	0.863	0.769

V. Discussion

In this work, actions are limited to trade in one satoshi only. This could be generalized to

Also, Dyna-Q is aptly capable of considering non-stationarity in data by bootstrapping historical data in a queue

In long term, ideally, one should look for is an algorithm that turn the trading world into an arms race. Acting in certain way should constantly change the behavior of the users and the optimal policy should be varying and convergent. These are the challenges that need to be addressed when trying to design algorithms for continuous time series.

References

[1] D. McCrum, "Profits still to be made in zero-sum game," 2012. [Online]. Available: https://www.ft.com/content/9520ec4a-0629-11e2-bd29-00144feabdc0

[2] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1, no. 1.

[3] "Forex4noobs - when not to trade." [Online]. Available: https://www.forex4noobs.com/forex-education/when-not-to-trade/

[4] A. Kharpal, "Most cryptocurrencies will crash to zero, goldman sachs says," 2018. [Online]. Available: https://www.cnbc.com/2018/02/07/most-cryptocurrencies-will-crash-to-zero-goldman-sachs-says.html

[5] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[6] P. Mourdoukoutas, "Bitcoin, ethereum and litecoin are the most popular cryptocurrency investments among millennials," 2018. [Online]. Available: https://tinyurl.com/y8qznfer

[7] "Bitfinex - bitcoin, litecoin and ethereum: Exchange and margin trading platform." [Online]. Available: https://www.bitfinex.com/

[8] R. S. Sutton, "Integrated architectures for learning, planning, and reacting based on approximating dynamic programming," in *Machine Learning Proceedings 1990*. Elsevier, 1990, pp. 216–224.

[9] M. G. Lagoudakis and R. Parr, "Least-squares policy iteration," *Journal of machine learning research*, vol. 4, no. Dec, pp. 1107–1149, 2003.

[10] G. J. Gordon, "Approximate solutions to markov decision processes," *Robotics Institute*, p. 228, 1999.