Reinforcement Learning for Portfolio Management

Daniil Chepenko
Big Data Institute
Hong Kong University of Science and Technology, Hong Kong
dchepenko@connect.ust.hk

Abstract – **Portfolio management is the art and science of decision-making process about investment for individuals and institutions. This project presents a Reinforcement Learning framework for cryptocurrency portfolio management. Cryptocurrency is a digital decentralized asset. The most well-known example are Bitcoin and Ethereum. To accomplish the same level of performance as a human-trader, agents have to learn for themselves how to create successful biased-free strategies. This work covers different reinforcement learning approaches. The framework is designed using Deterministic Policy Gradient using Recurrent Neural Networks (RNN). The robustness and feasibility of the system is verified with the market data from Poloniex exchange and compared to the the major portfolio management benchmarks.**

Index Terms – Cryptocurrency, Deep Learning, Machine Learning, Portfolio Management, Reinforcement Learning, Recurrent Neural Network

INTRODUCTION

The main principle of investing is a risk and return trade-off. But the future return can be rarely precisely predicted. There is almost always some risk associated with investment. The actual, or the realized return, will almost always deviate from initially expected at the begining. Portfolio management theory took a great improvement trough the last century from basically intuitive approaches to assets allocation to comprehensive quantitative approaches. Machine learning is widely used in financial science by now, as no wonder that accurate asset price prediction can lead to lucrative results.
However, until recently most of the algorithm used only straightforward supervised learning approach – take historical prices as an input and produce predicted prices for the selected period. As for unsupervised methods, in 1997 Moody and Wu[1] trained trading system using recurrent reinforcement learning algorithm. And in 1997 Moody and Saffel[2] presented the results of their model for foreign exchange market.
The recent successful attempts of unsupervised learning solve the portfolio management problem with deep learning techniques. Under some tests, Reinforcement Learning models even outperform human experts in conducting optimal control policies. The successful examples of its application include Atari game playing [3], robots navigation [4] and digital advertising [5]. Hence it leads the interest of quantitative traders to utilize reinforcement learning to solve portfolio management problems [6][7]. For example Jiang, Xu and Liang (2017)[7] proposed a framework for portfolio trading. This framework is implemented within a Convolutional Neural Network (CNN), a basic Recurrent Neural Network (RNN), and a Long Short-Term Memory (LSTM). The results are examined on cryptocurrency market with a step size of 30 minutes and a return period of 5 days. However, some failed to replicate their results [8]. Moreover, after the publication the whole cryptocurrency market witnessed a dramatic plunge of about 66%, losing over $553 billion in valuation. Bitcoin recorded a huge loss of over 50% in February, with valuation dropping below $7,000. Ethereum and Ripple also suffered similar drops, both recording losses of over 40% during the same month. Since that the updated data can be used to perform more accurate prediction.
In this project, I want to study Deterministic Policy Gradient using Recurrent Neural Networks (RNN) to maximize the log-return of cryptocurrency portfolio.
The motivation of using Reinforcement Learning steams from personal several experiments of using model-free algorithms such as Q-Learning for single-asset trading.

CRYPTOCURRENCY

1. *Assets description*

Cryptographic currencies, or simply cryptocurrencies, are electronic and decentralized alternatives to government-issued money (Nakamoto, 2008 [9]; Grinberg, 2012 [10]). While the best-known example of a cryptocurrency is Bitcoin, there are more than 100 other tradable cryptocurrencies competing each other and with Bitcoin. Comparable to traditional financial assets, the amount of liquid cryptocurrencies is small. Although there are more than 200 coins traded on different exchanges, the trading volume of some of them is low.
The developers use blockchain technology to create decentralized applications. Some of the application requires issuing a token. Such procedure called initial coin offering (ICO). A token can be issued on different available exchanges.
The researchers that have been worked on token valuation, apply both intrinsic [11], [12] and relative [13] approaches.

2. *Exchange description*

As there is no central regularity party, anyone can participate in trading with low entrance requirements. The majority of cryptocurrency exchanges has application programming interface for obtaining market data and trading around the clock. These features make cryptocurrency market interesting test-ground for algorithmic portfolio management

experiments. There has been an explosion of crypto assets in 2017, with trading volumes topping billions of dollars.
In this paper, the data from Poloniex exchange is used. In February 2018 it was ranked as the 14th largest crypto exchange with almost 100 tokens listed and daily volume of trading more than $150 millions.
In the experiments of this paper, the 8 most-volumed cryptocurrencies were selected. The reason for choosing the top-volumed cryptocurrencies is that this factor implies the better market liquidity.

3. *Trading period*

The algorithms are time-driven where time is divided into a period of $t = 1\,day$. The algorithm trained on a historical data of $T = 730\,days$. The four important factors characterize the asset price during the period: open, close, highest and lowest prices.

4. *Assumptions*

During this research three assumptions about the behavioral were taken:

1. Zero-slippage. The liquidity of all market assets is high enough that, each trade can be carried out immediately at the last price when an order is placed.Zero-slippage. The luquidity of all market assets is high enough that, each trade can be carried out immediately at the last price when a order is placed.
2. Zero market impact: The capital invested by the software trading agent is so insignificant that is has no influence on the market.
3. Poloniex exchange prices represent the relevant valuation of cryptocurrency asset.

This assumption 1 and 2 are valid for real-market data if the trading volume is high enough. However, in the further research, it would be interesting to study the influence of those hypotheses.
The last assumption reveals the current market problems. Arbitrage strategies among major exchanges are very popular due to the lack of central regularity party and immaturity of the market. However, by now there is no any research on the impact of arbitrage trading software in cryptocurrency market.

5. *Correlation matrix*

Compared to foreign exchange or equity market cryptocurrency market have different market demographics, so the behavioral patterns are less predictable.
Figure 1 shows the correlation matrix of 8 top-volumed cryptocurrencies.
To obtain significant diversification benefits, the assets need to have low-positive or negative correlations.
As there is no strong correlation between daily log-returns of any of the assets the performance of this portfolio can be studied

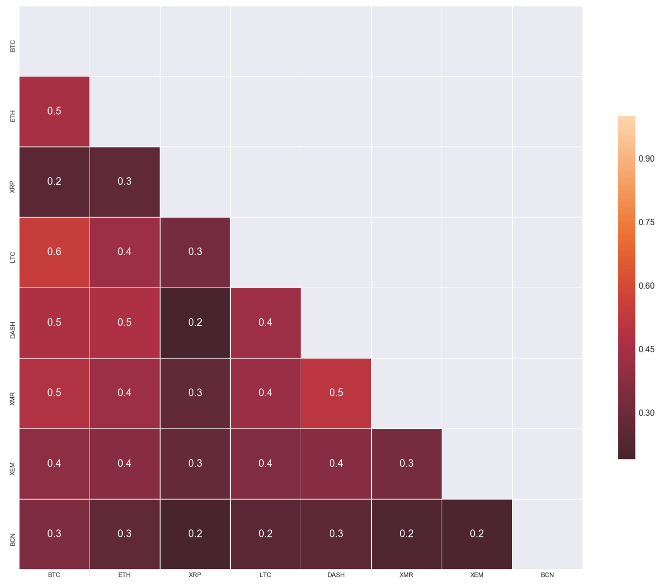

FIGURE 1
LOG-RETURN CORRELATION ANALYSIS OF 8 TOP-VOLUMED CRYPTOCURRENCIES BASED ON DAILY TICK DATA OVER THE PERIOD T

PORTFOLIO MANAGEMENT

Investment in M assets form a portfolio, and the $x_i, i \epsilon [1, m]$ are referred to portfolio shares or weights. The return of the portfolio is a random variable, and represent the linear combination of assets.

1. *Assets allocation*

Asset allocation is a common task in computational finance and financial engineering. This problem can be formulated as an optimization problem of balancing the weights of the assets to achieve certain targets. There are two major schools to this problem investigation:

- **Mean Variance Theory** (Markowitz, 1954). This theory focuses on a single-period (batch) portfolio selection to trade-off portfolio's expected returns and risk.
- **Capital asset pricing model** (Treynor, 1961, 1962; Sharpe, 1964; John Lintner, 1965; Mossin, 1966) a model that describes the relationship between systematic risk and expected return for assets, particularly stocks.

2. *CAPM for cryptocurrency*

The two components of risk, systematic (market) and unsystematic (company-specific), are considered within the CAPM framework. Since unsystematic risk is eliminated in a broadly diversified portfolio, investors are only compensated for systematic risk. The risk premium is then the product of systematic risk (beta) multiplied by the market price of risk, defined as the difference between the expected return of the market portfolio and the riskless interest rate. In the CAPM, the market portfolio is composed only of stocks and bonds. Under the CAPM framework, the market beta drives the prospective capital asset returns. However, it is negotiable that cryptocurrency is considered as capital assets. Thus, application of the CAPM model is of limited use.

3. MVT for cryptocurrency

As it steams from Figure 1 most of the correlations are below 0.5, as seen in the following table. The mean-variance theory the prescription consists of generating maximum Sharpe Ration (MSR) portfolio based on expected return, volatility and pairwise correlation parameters for all assets to be included.

4. Performance metrics

Different metrics are used to measure the performance of the particular portfolio selection strategy. The accumulative portfolio value (APV) p_t. APV is measured in the unit of their initial values, or equivalently $p_0 = 1$ and thus

$$p_t = p_t / p_0 \tag{1}$$

Under the same unit, final APV (fAPV) is the APV at the end of back-test experiments $p_f = \frac{p_f}{p_0} = p_{t_{f+1}} / p_0$

The main disadvantage of fAPV is that it doesn't consider the risk factors since it is merely sums up the periodic returns without considering the fluctuation of the asset. Sharpe ratio (SR) (Sharpe, 1964, 1994) can be used as a metric, that takes risk factor in an account.

$$S_t = \frac{Average(R_t)}{StandartDeviation(R_t)} = \frac{E[R_t]}{\sqrt{E[R_t^2] - (E[R_t])^2}} \tag{2}$$

$R_t = p_t - p_f$ is the return on investment for trading period t compare to the risk-free asset

$E[.]$ denotes the expectation.

This measurement shows the excess of return per unit of risk you are taken. It's basically the return on capital over the standard deviation, adjusted to risk. The higher is better.

In the traditional financial market usually, the government-issued bonds are concerned as a risk-free asset. However, the cryptocurrency market is more volatile and the identification of risk-free asset could be tricky. Jiang, Xu, and Liang [7] used Bitcoin as a risk-free cryptocurrency. This assumption needs to take more accurate research. However in this project the quoted currency is also Bitcoin, the risk-free return is zero. The SR equally measures the upwards and downwards movements. Alternative measurement it is the maximum dropdown. It is the maximum loss from a peak to a trough of a portfolio before a new peak is attained. It is an indicator of downside risk over a specified period.

$$D = max \frac{p_t - p_\tau}{p_\tau} \tag{3}$$

5. Benchmarks

To measure the performance of the portfolio management and further balancing, current strategy is compared to major benchmarks, such as:

- **Buy and Hold strategy (BAH).** The investor allocate weights to the assets and the beginning of the 1st period without further rebalancing. If the weights were allocated uniformly this strategy called uniform Buy and Hold (UBAH).
- **Best Stock strategy (BEST).** Through every rebalancing period the stock with the best performance is chosen and all capital put in this stock.

In this paper, the performance of the strategy is compared to the BAH, BEST and MVT portfolio benchmarks. Other state-of-art strategies of portfolio management techniques are covered by Li and Ho (2014) [6]. They can be concerned with the further research.

6. Mathematical Formalism

The portfolio consists of M assets.

The actual return on a portfolio of assets over some period t can be described using price information

$$R_p = \sum_{n=1}^{N} \omega_m R_m \tag{4}$$

R_m- return on asset n over the period t

ω_m- weight of asset n in the portfolio at the begining of the period t

M - number of assets in the portfolio

To estimate the risk of the portfolio the variance of the portfolio is calculated. The variance is the measure of the dispersion of the possible rate of return outcomes around the expected return.

$$var(R_p) = \sum_{m=1}^{M} \omega_m var(R_m) + \sum_{m=1}^{M} \sum_{\substack{n=1 \\ n \neq m}}^{M} \omega_m \omega_n cov(R_m) \tag{5}$$

Markowitz diversification seeks to combine assets in portfolio with returns are less than perfectly correlated, in an effort to lower variance without sacrificing return. This approach leads to the development an efficient portfolio – portfolio that have highest expected return for a given level of risk.

Markowitz theory works under the assumption of normally distributed returns. In this case variance is a useful measure of risk. Also for many markets the return distribution have been found to be assymetric. To overcome this problem, Markowitz proposed to use semivariance, that is calculated by measuring the dispersion of all observations that fall below the mean or target value of a set of data.

PROBLEM STATEMENT

1. *Reinforcement Learning Environment*

To describe the Reinforcement Learning problem there are several frameworks used. The most common one is Markov Decision Process (MDP). Though, there are also exist Partially Observable Markov Decision Process (POMDP) and Multi-armed bandits. This project covers the MDP framework.
The MDP consists of the following components: agent, state, environment, state transition, action, reward.

2. *Agent*

An agent is a software portfolio-trading agent performing actions on the financial market.

3. *Environment*

The environment describes the exchange, where the orders for assets are stored. Traders, both software agents, and physical traders are placing orders on this exchange. An environment can be stochastic and discrete.

4. *Actions*

Agent's actions are defined as weights of every asset in the portfolio. Such as for time t actions are the following set of weights

$$A^t = \{\omega_1^t, \omega_2^t, .., \omega_n^t\} \tag{6}$$

${\omega_n}^t$- weight of asset n in the portfolio at the beginning of the period t

The agent is limited to perform portfolio management actions only on discrete time steps.

5. *State*

Under an assumption that price is a valid representation of the asset in the environment, the state depends on the historical prices from the period T for every asset from the portfolio. The procession of the whole historical data is very complex and huge task for the agent. In terms of efficiency sub-sampling techniques needed. This work covers the periodic feature-extraction and history cut-off. Periodic feature extraction discretizes the time into periods, and then extract the highest, lowest, and closing prices in each period. History cut-off simply takes the price-features of only a recent number of periods to represent the current state of the environment. As the result, the state is described as a matrix of periodic extracted features for assets array.
The state in time t can expressed in the following way:

$$S^t = (P^{t-N}, P^{t-N+1}, \dots, P^{t-1}, P^t, A^t) \tag{7}$$

P^t- price tensor
N −cut-off period

6. *Reward*

The reward is determined by profit and transaction cost together. The corresponding profit for time t is:

$$r_t = \omega_t(P^t - P^{t-1}) \tag{8}$$

Though in a real-world scenario asset trading is not free. The transaction cost is defined into two parts, the cost proportional to the change of stock assets, and the cost due to the change of portfolio:

$$C_t = \sum_{n=1}^{N} c_i \,|P^t - P^{t-1}| + c_0 1_{[P^t \neq P^{t-1}]} M \tag{9}$$

c_i −cost ratio of change asset i
c_0 −the cost ratio change in portfolio

$$R_t = r_t - C_t = \omega_t(P^t - P^{t-1}) - \sum_{n=1}^{N} c_i \,|P^t - P^{t-1}| + c_0 1_{[P^t \neq P^{t-1}]} N \tag{10}$$

R_t-cummulative reward over the time period $[0; t]$

7. *Transaction*

State transition represents the probability that agent will end up transitioning the state S^{t+1} given that in state S^t it performed an action A^t. The stationary transaction dynamics distribution $p(S^{t+1}|S^t, A^t)$ satisfying the Markov property $p(S^{t+1}|S^1, A^1, S^2, A^2, \dots, S^t, A^t) = p(S^{t+1}|S^t, A^t)$

8. *Policy*

The solution of MDP is called policy. It is a distribution of actions given state. It provide the agent with an action that can maximize long-term reward in particular state. In other words, it is a mapping from the state space to a probability distributions over the action space $\pi: S \rightarrow \mathcal{P}(A)$ Policy can be deterministic and stochastic and it depends on a current state.

METHODOLOGY

The goal of reinforcement learning to adjust the parameters to maximize the cumulative reward that is generated due to the action of the system. The optimal policy in timestamp t is the policy that maximizes cumulative rewards. As the policy and environment are stochastic we need to find an expectation of cumulative reward.
There are two main methods that are used to solve MDP: model-based and model-free.

1. *Model-based methods*

Model-based method learns the transaction probabilities for all state-action pairs. After several iterations, the agent will know how likely to enter a specific state given current state and action. The main disadvantage, that they become impractical as the state space and action space grows.

2. Model-free methods

Model-free methods rely on trial-and-error to update its knowledge. As a result, it does not require space to store all the combination of states and actions. The main objective of this class of algorithms is to compute the value of the chosen policy. Model-free methods are those in which we do not have the transition probabilities and these methods can be divided into value-based and policy-based.

3. Value-based methods

The Value Learning can be written through Bellman expectation equations. There are two functions, that represent the value of the chosen policy.
State-value function is expected return conditional on state. It is value of following policy π from state S.

$$v_\pi(s) \triangleq \mathrm{E}_\pi[G_t|S_t = s] = \sum_a \pi(a|s) \sum_{r,s'} p(r,s'|s,a)[r + \gamma v_\pi(s')] \quad (7)$$

$G_t -$ environment space at time t
$S_t -$ state space at time t
$s -$ state value
$s' -$state value at time t'
$r -$ reward
$\gamma -$ discount factor

The action-value function is used to calculate an expected return conditional on state and action. It is a value of following policy π after committing action A in state S.

$$q_\pi(s,a) \triangleq \mathrm{E}_\pi[G_t|S_t = s, A_t = s] = \sum_{r,s'} p(r,s'|s,a)[r + \gamma v_\pi(s')] \quad (8)$$

The optimal value function $q^*_\pi(s,a) = max_\pi\ q_\pi(s,a)$ gives the maximum action value for state s and action a achievable by any policy.
Eq.(7) and Eq.(8) are called Bellman Expectation Equation. Solving this equation for large MDP is a hard computational problem. But most of the time the precise solution of the system is not necessary. It is sufficient to obtain a good approximation of the solution. Approximation of state-value function is $v_\pi(s,a,\theta)$, and for action-value function is $q_\pi(s,a,\theta)$. Both functions fits the orginal functions across the state and action-state sapce correspondingly. θ is a parameter vector. As for function approximator there can be used decision trees, neural networks, linear combinations of features, fourier/wavelets bases and other statistical learning approaches.
The updates to θ are derived from various reinforcemenet learning algorithms. The researchers used Q-learning algorithm, which aims to directly approximate the optimal action value function $q^*_\pi(s,a) \approx q_\pi(s,a,\theta)$. To learn the parameters θ backpropagation algorithm minimizes a sequence of loss function:

$$L_i(\theta_i) = \mathrm{E}[R_t + \gamma max_{A^{t+1}} q_\pi(S^{t+1}, A^{t+1}, \theta_{i-1}) - q_\pi(S^t, A^t, \theta_i)] \quad (9)$$

The above method is called one-step Q-leaning and it updates action-value function towards one step return. A one-step Q-leaning method doesn't specify what actions the agent should take as it updates its estimates. To find the optimal action-value function the agent must perform every action in all states many times. Mnih et al.[14][15] adapted the Q-learning algorithm in order to make effective use of large neural networks as function approximators.
$Q(\lambda)$ -learning algorithm combines $TD(\lambda)$ returns for λ and Q-learning. In this case the n-step return is defined as $R^t + \gamma R^{t+1} + \cdots + \gamma^{n-1} R^{t+n-1} + \gamma max_a q_\pi(R^{t+1}, A, \theta)$. This result is affecting n state-action pairs.

4. Policy-based methods

Policy gradient algorithms are widely used in reinforcement learning problems with continuous action spaces. In continuous action space finding the greedy policy requires an optimization at A_t at every timestamp t. The idea is to represent policy as a parametric probability distribution $\pi_\theta(a|s) = \mathrm{P}[a|s;\theta]$ that stohastically selects action a in state s according to parameter θ.
Indeed policy gradient algorithm is an optimization problem.The basic idea is to adjust the parameter θ of the policy in the direction of performance gradient $\nabla_\theta J(\pi_\theta)$

$$J(\pi_\theta) = \int_S \rho^\pi(s) \int_A \pi_\theta(s,a) r(s,a) da ds = \mathrm{E}_{s\sim\rho^\pi, a\sim\pi_\theta}[r(s,a)] \quad (10)$$

The fundamental result underlying these algorithms is the policy gradient theorem (Sutton et al. , 1999) [18]

$$\nabla_\theta J(\pi_\theta) = \int_S \rho^\pi(s) \int_A \nabla_\theta \pi_\theta(s,a) Q^\pi(s,a) da ds = \mathrm{E}_{s\sim\rho^\pi, a\sim\pi_\theta}[\nabla_\theta log \pi_\theta(a|s) Q^\pi(s,a)] \quad (11)$$

$J(\pi_\theta) -$ performance objective function
$\pi_\theta -$stochastic policy

The good overview of policy learning methods is provided by Athey and Wager in Efficient Policy Learning[15]. The latest breakthrough in policy gradient methods is connected with deep learning. Andrej Karpathy provides an excellent walkthrough on using deep reinforcement learning to learn policy for the Atari game Pong[16]. In this game, the states

were the raw pixels from the game and the output – the probability of moving the paddle up or down.
In continuous action space, the curse of dimensionality dominates – the total number of actions increases exponentially as the discretization precision increase.

5. *Actor-Critic Algorithm*

However, deep learning can solve the problem of high observation dimensional data it only handles discrete and low-dimensional action-spaces. This algorithm relies on finding the action that maximizes the action-value function, which in continuous spaces requires solving a complex optimization problem. Obvious solutions like action-space discretization lead to the explosion of the numbers of discrete actions. One of the solutions is used actor-critic approach [17].
The actor-critic is a widely used architecture based on the policy gradient theorem (11). It consists of two components: actor adjust parameters θ of the stophastic policy $\pi_\theta(s)$ by stochastic gradient ascent. But instead of unknown $q_\pi(s,a), q_\omega(s,a)$ is used. A critic estimates action-value function $q_\omega(s,a) \approx q_\pi(s,a)$ using an appropriate policy evaluation algorithm.
In order not to produce a bias after evaluation there are two assumptions made:

1) $q_\omega(s,a) = \nabla_\theta log\pi_\theta(a|s)^T\omega$
2) parameter ω is chosen to minimize the mean-squared error $\epsilon^2(\omega) = \mathrm{E}_{s\sim\rho^\pi, a\sim\pi_\theta}[(q_\omega(s,a) - q_\pi(s,a))^2]$

$$\nabla_\theta J(\pi_\theta) = \mathrm{E}_{s\sim\rho^\pi, a\sim\pi_\theta}[\nabla_\theta log\pi_\theta(a|s) q_\omega(s,a)] \quad (12)$$

The actor-critic algorithm maintains a parameterized actor function $\mu(s|\theta^\mu)$ which specifies the current policy by deterministically mapping states to a specific action. The critic is learned using Belman equation (7)

6. *Deterministic Policy Gradient Algorithms*

The majority of model-free algorithms are based generalized policy iteration: iterating policy evaluation with policy improvement. Policy evaluation estimates the action-value function q_ω or q_π and policy improvent algorithm updates policy with respect to estimated action-value function. It can be achieved using greedy maximization.
In continuous action space, greedy policy becomes problematic as it requires maximization every step. Instead, the better alternative is to move in the direction of the gradient. Specifically for each visited state s the policy parametrs θ^{k+1} are updated in proportion to the gradient $\nabla_\theta q^{\mu^k}(s, \mu_\theta(s))$
With a parameter vector $\theta \in R^n$ a deterministic policy can be formally considered. Silver (2014) [17] proved a deterministic policy gradient theorem that states that the actor is updated by applying the chain rule to do extend return from to the expected return from the start distribution J with respect to the actor parameters:

$$\nabla_{\theta^\mu} J(\mu_\theta) = \mathrm{E}_{s\sim\rho^\mu}[\nabla_\theta \mu_\theta(s) \nabla_a q_\mu(s,a)|_{a=\mu_\theta(s)}] \quad (13)$$

μ_θ - deterministic policy $\mu_\theta : S \to A$

In this project RNN is used as function approximator and allows to learn in large state and action spaces online. The main challenge when using the neural network in reinforcement learning is that most optimization algorithms assume that the samples are independently and identically distributed. To address these issues the replay buffer, finite-sized cache, is used. Transitions sampled from the environment according to the exploration policy and the tuple (S^t, A^t, R^t, S^{t+1}) stored in replay buffer. At each timestamp t, the actor and critic are updated by sampling a minibatch uniformly from the buffer. To treat the problem of continuous action space exploration the exploration policy is constructed μ' by adding noise sampled from a noise process $\mathcal{N}$ to actor policy

$$\mu'(s^t) = \mu\left(s^t \middle| \theta_t^\mu\right) + \mathcal{N} \quad (14)$$

$\mathcal{N}$ is chosen to suit the environment. The gradients will be calculated by first backpropagating through 6 to the action and then back- propagating in the neural nets (which is automatically taken care of by deep learning libraries such as TensorFlow).

7. *Recurrent Neural Network*

A recurrent neural network is a class of artificial neural network where connections between units form a directed graph along a sequence. This allows it to exhibit dynamic temporal behavior for a time sequence. For basic RNN the relation can be expressed:

$$h_t = f(W_h * h_{t-1} + W_x * x_t) \quad (15)$$

h_t -hidden state of the cell at timestamp t
x_t- the input at t
f - non-linear activation function

Based on N previous days actions or calculated based on the output states of RNN cell. As an input log returns of each time step are taken. Weights are initialize using Xavier initialization. The advantage of that technique - the weights are pick up randomly from a Gaussian distribution.
To prevent falling into the pitfalls of numerical precision [18] that is crucial for financial application in this paper the inputs of the RNN cell are magnified by 500 [19].

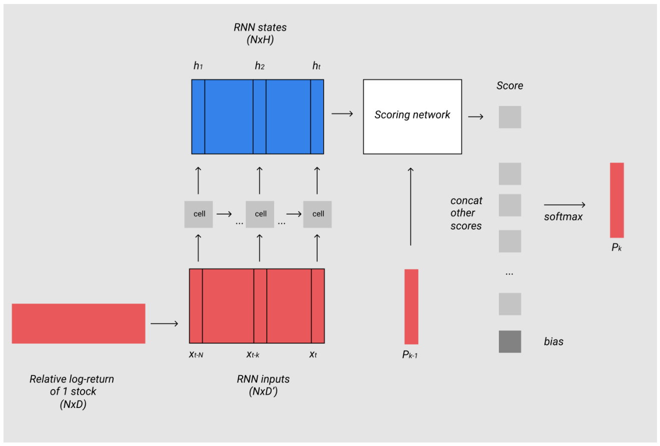

FIGURE 2
ARCHITECTURE DIAGRAM OF RNN MODEL

The scoring network consists of two-feed layers and another bias denoting the 'cash'. Distribution for allocation then produced by a softmax at the timestep t and represent the action state A^t or weight of the asset in the portfolio.

Experiments

In evaluating trading systems and their performance there are four main ways to produce results:

1. Historical backtesting
2. Out-of-sample testing
3. Walk-forward testing
4. Recall-time testing

In this project only historical backtesting used. The testing period is from 2018-02-23 to 2018-05-21. The performance metrics and benchmarks are covered in Portfolio Management section of this paper.

TABLE I
UNIFORM BUY AND HOLD (UBAH)

Period	APV	Sharpe ratio	MDD
2018-02-23 : 2018-05-21	-0.19	-0.78	0.71

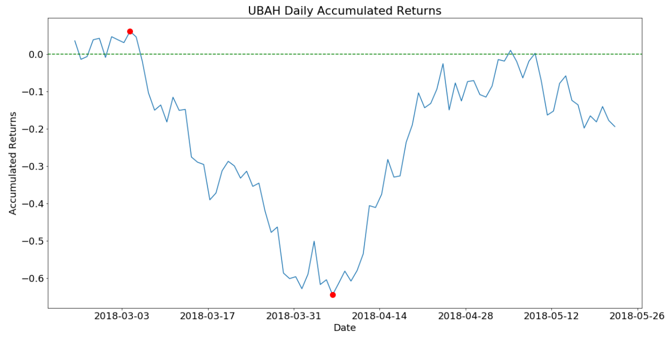

FIGURE 3
UBAH DAILY ACCUMULATED RETURNS DURING HISTORICAL BACKTESTING

Best Stock Strategy (BEST) strategy performed with rebalancing period = 30 days. After that, the overall performance measured and used as a benchmark.

TABLE 2
BEST STOCK STRATEGY (BEST)

Period	APV	Sharpe ratio	MDD
2018-02-23 : 2018-03-25	-0.2	-2.7	0.36
2018-02-23 : 2018-04-24	0.05	0.63	0.25
2018-04-24 : 2018-05-22	0.59	2.48	0.49
2018-02-23 : 2018-05-22	0.59	1.0	1.32

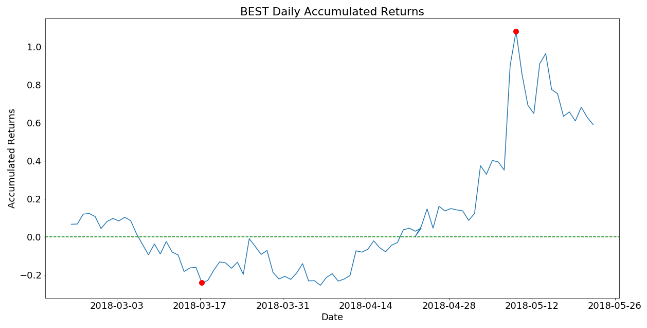

FIGURE 4
BEST DAILY ACCUMULATED RETURNS DURING HISTORICAL BACKTESTING WITH REBALANCING PERIOD EVERY = 30 DAYS

The mean-variance theory provides an investor with a choice of using either portfolio with the highest Sharpe-ratio or with minimal volatility. In this project the performance is measured with both allocation methods. According to Markowitz theory an optimal portfolio can be designed with a perfect balance between risk and return. The points on the plot of risk versus expected returns where optimal portfolio lie called efficient frontier.

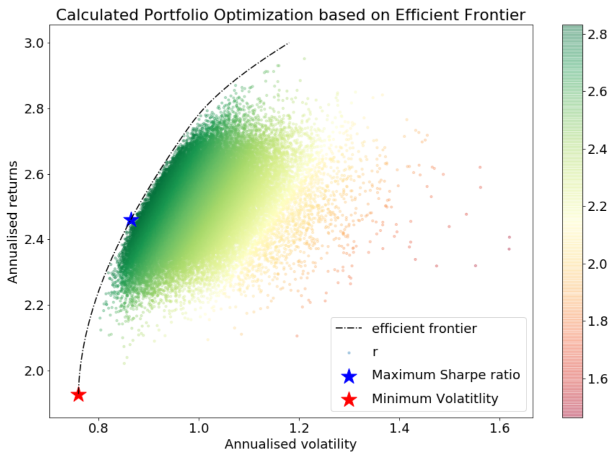

FIGURE 5
MVT RISK-RETURN PLOT ON TRAIN PERIOD T

Table 3 and Table 4 shows the result of MVT portfolio trained on train period T performance.

TABLE 3
MAXIMUM SHARPE RATIO STRATEGY (MSR)

Period	APV	Sharpe ratio	MDD
2018-02-23 : 2018-05-21	-0.31	-1.32	0.72

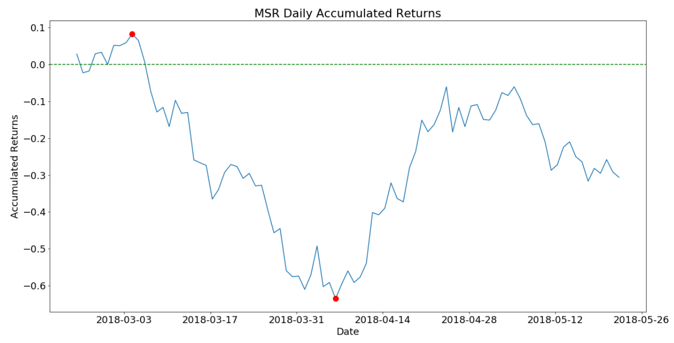

FIGURE 6
MAXIMUM SHARPE RATIO ACCUMULATED RETURNS DURING HISTORICAL BACKTESTING

TABLE 4
MINIMUM VARIANCE STRATEGY (MV)

Period	APV	Sharpe ratio	MDD
2018-02-23 : 2018-05-21	-0.23	-1.08	0.64

Table 5 and Table 6 shows the result of MVT portfolio trained on train period T performance with rebalancing period = 30 days.

TABLE 5
MAXIMUM SHARPE RATIO STRATEGY (MSR)

Period	APV	Sharpe ratio	MDD
2018-02-23 : 2018-03-25	-0.33	-4.62	0.45
2018-02-23 : 2018-04-24	0.69	5.31	0.28
2018-04-24 : 2018-05-22	0.59	2.48	0.49
2018-02-23 : 2018-05-22	0.59	1.98	1.45

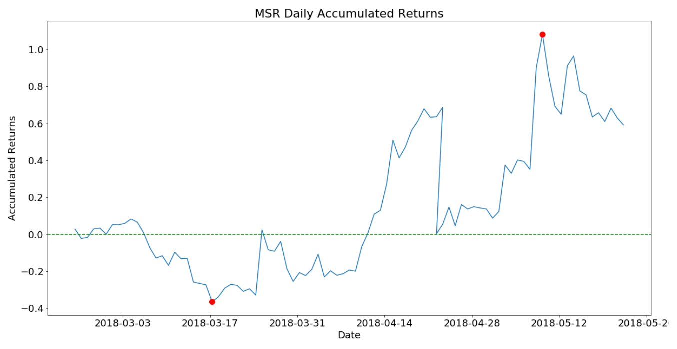

FIGURE 7
MAXIMUM SHARPE RATIO ACCUMULATED RETURNS DURING HISTORICAL BACKTESTING WITH REBALANCING PERIOD = 30 DAYS

TABLE 6
MINIMUM VOLATITLITY STRATEGY (MSR)

Period	APV	Sharpe ratio	MDD
2018-02-23 : 2018-03-25	-0.23	-3.27	0.41
2018-02-23 : 2018-04 24	0.05	0.63	0.25
2018-04-24 : 2018-05-22	-0.07	-1.3	0.2
2018-02-23 : 2018-05-22	-0.07	-1.31	0.41

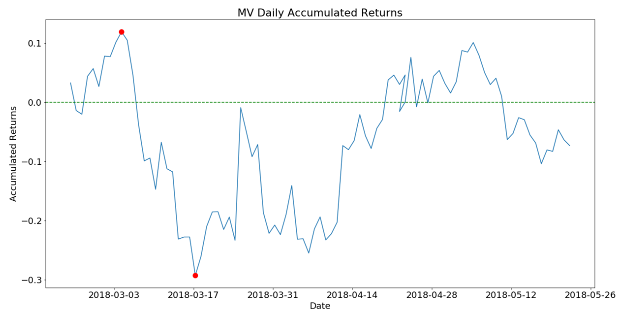

FIGURE 8
MINIMUM VOLATILITY ACCUMULATED RETURNS DURING HISTORICAL BACKTESTING WITH REBALANCING PERIOD = 30 DAYS

For Deterministic Policy Gradient using Recurrent Neural Networks (RNN) model, the extensive analysis was provided. For hyperparameters tuning such as the number of days to look back N and hidden cell size, Hs run on validation period 2017-11-23 to 2018-02-23.
For historical backtesting the model with $N = 15$ and $Hs = 10$

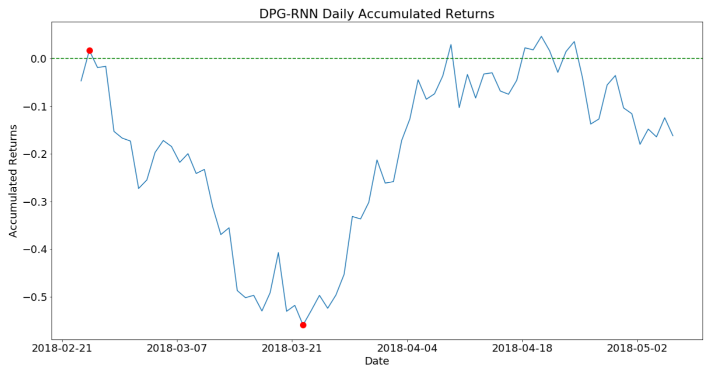

FIGURE 8
DPG-RNN ACCUMULATED RETURNS DURING HISTORICAL BACKTESTING

TABLE 7
MINIMUM VARIANCE STRATEGY (MV)

Period	APV	Sharpe ratio	MDD
2018-02-23 : 2018-05-21	-0.16	-0.76	0.58

Conclusion

In this project, the portfolio management problem was formulated as a Deep Reinforcement Learning problem with Deep Policy Gradients. The model was implemented using Tensorflow library and trained on data from Poloniex exchange. For DPG Recurrent Neural Networks were used as a function approximator.

At the end of test results, DPG-RNN model has slightly surpassed the UBAH baseline and performed better that MVT models. However, the results are still far from models with rebalancing.

The possible explanation of such results are:

1. Deep learning models tend to have the high variance which makes them hard to train of financial data with noise.
2. To analyze cryptocurrency market behavior other factor models should be concerned but not only historical prices.
3. The data can be insufficient. Jiang, Xu, Liang [7] trained their model on 30 minutes data, while in this work daily data is used
4. Network structure requires more refinement.
5. RNN might be not an efficient way for function approximation

As a future work the following improvements are proposed:

1. Integrate more outside information other that historical prices. This can be achieved by inventing special cryptocurrency indicators: Blockchain monitoring indicator, Github commit indicator, Sentiment analysis indicators.

- *Blockchain monitoring indicator* – The transparency of a blockchain allows the analysis on the movement of every token. Especially those held by major holders, otherwise known as "whales." When the distribution is not uniform and the majority of tokens are held by few holders, tools which monitor "whale" movements can show when major holders are buying or selling their holdings.
- *Github commit indicator* – As some of the projects disclose their product in the open-source, tracking the activities might be a useful feature. The investors have the ability to monitor Github commits and changes so that they can immediately see all updates on coins and tokens they hold and trade.
- *Sentiment analysis indicators* - Use natural language processing and opinion mining based on media monitoring

2. Work on network structure. For example, softmax operation on the output of the network might be problematic because it changes a possibly linear scale in the scores to exponential. Also, the way how calculated action and reward can be changed
3. RNN appraoch can be improved by using LSTM and GRU network architectures.
4. To reduce the variance of the model can be applied attention[22] or dropout[23] mechanism.
5. Evaluate the performance of strategy using out-of-sample testing, walk-forward testing, recall-time testing on different time periods.
6. Explore the influence of zero-slippage and zero-impact market assuptions

Reinforcement learning has a good potential for portfolio management problem, since it can capture unevident events and dependencies. It definitely requires further studies..

References

[1] Optimization of Trading Systems and Portfolios, John Moody, Lizhong Wu, 1997

[2] Reinforcement Learning for Trading, John Moody, Mathew Saffel, 1998

[3] H. R. Beom and K. S. Cho, "A sensor-based navigation for a mobile robot using fuzzy logic and reinforcement learning," IEEE Trans. Syst., Man, Cybern., vol. 25, no. 3, pp. 464–477, Mar. 1995.

[4] V. Mnih et al., "Human-level control through deep reinforcement learning," Nature, vol. 518, no. 7540, pp. 529–533, 2015.

[5] Deep Reinforcement Learning for Sponsored Search Real-time Bidding, Jun Zhao, Guang Qiu, Ziyu Guan, Wei Zhao, Xiaofei He

[6] Signal Representation and Trading, Yue Deng, Feng Bao, Youyong Kong, Zhiquan Ren, and Qionghai Dai, *Senior Member, IEEE,*

[7] Jiang, Xu, Liang , A Deep Reinforcement Learning Framework for the Financial Portfolio Management Problem, 2017

[8] https://github.com/wassname/rl-portfolio-management

[9] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.

[10] Reuben Grinberg. Bitcoin: An innovative alternative digital currency. Hastings Sci. & Tech. LJ, 4:159, 2012.

[11] Chris Burniske, Cryptoasset Valuations

[12] Alex Evans https://medium.com/blockchannel/on-value-velocity-and-monetary-theory-a-new-approach-to-cryptoasset-valuations-32c9b22e3b6f

[13] Willy Woo http://woobull.com/nvt-signal-a-new-trading-indicator-to-pick-tops-and-bottoms/

[14] Mnih, Volodymyr, Kavukcuoglu, Koray, Silver, David, Graves, Alex, Antonoglou, Ioannis, Wierstra, Daan, and Riedmiller, Martin. Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602 , 2013.

[15] Mnih, Volodymyr, Kavukcuoglu, Koray, Silver, David, Rusu, Andrei A, Veness, Joel, Bellemare, Marc G, Graves, Alex, Riedmiller, Martin, Fidjeland, Andreas K, Ostrovski, Georg, et al. Humanlevel control through deep reinforcement learning. Nature , 518(7540):529–533, 2015.

[16] Andrej Karpathy, Deep Reinforcement Leanring: Pong from Pixels http://karpathy.github.io/2016/05/31/rl/

[17] Silver, David, Lever, Guy, Heess, Nicolas, Degris, Thomas, Wierstra, Daan, and Riedmiller, Martin. Deterministic policy gradient algorithms. In ICML , 2014.

[18] Why .1 + .1 Might Not Equal .2 and Other Pitfalls of Floating-Point Arithmetic, Clarke Thacher, SAS Institute Inc., Cary, NC

[19] CS221 Project Final Report, Deep Reinforcement Learning in Portfolio Management, Ruohan Zhan, Tianchang He, Yunpo Li

[20] Sutton, R. S., McAllester, D. A., Singh, S. P., and Mansour, Y. (1999). Policy gradient methods for reinforcement learning with function approximation

[21] Volodymyr Mnih, Adrià Puigdomènech Badia , Mehdi Mirza, Alex Graves, Tim Harley, Timothy P. Lillicrap, David Silver, Koray Kavukcuoglu, Asynchronous Methods for Deep Reinforcement Learning, 2016

[22] Effective Approaches to Attention-based Neural Machine Translation, Luong et al, arXiv:1508.04025

[23] Improving deep neural networks for lvcsr using rectified linear units and dropout, Dahl et al, IEEE 2013, DOI: 10.1109/ICASSP.2013.6639346

[24] The sharpe ratio, Sharpe, W. F. (1994), The Journal of Portfolio Management, 21(1), 49-58

AUTHOR INFORMATION

Daniil Chepenko, Master of Science student, Big Data Institute, Hong Kong University Of Science And Technology. Code implementation can be found: