A Deep Reinforcement Learning Framework for the Financial Portfolio Management Problem

Zhengyao Jiang   zhengyao.jiang15@student.xjtlu.edu.cn
Dixing Xu   dixing.xu15@student.xjtlu.edu.cn
Department of Computer Sciences and Software Engineering
Jinjun Liang   jinjun.liang@xjtlu.edu.cn
Department of Mathematical Sciences
Xi'an Jiaotong-Liverpool University
Suzhou, SU 215123, P. R. China

Abstract

Financial portfolio management is the process of constant redistribution of a fund into different financial products. This paper presents a financial-model-free Reinforcement Learning framework to provide a deep machine learning solution to the portfolio management problem. The framework consists of the Ensemble of Identical Independent Evaluators (EIIE) topology, a Portfolio-Vector Memory (PVM), an Online Stochastic Batch Learning (OSBL) scheme, and a fully exploiting and explicit reward function. This framework is realized in three instants in this work with a Convolutional Neural Network (CNN), a basic Recurrent Neural Network (RNN), and a Long Short-Term Memory (LSTM). They are, along with a number of recently reviewed or published portfolio-selection strategies, examined in three back-test experiments with a trading period of 30 minutes in a cryptocurrency market. Cryptocurrencies are electronic and decentralized alternatives to government-issued money, with Bitcoin as the best-known example of a cryptocurrency. All three instances of the framework monopolize the top three positions in all experiments, outdistancing other compared trading algorithms. Although with a high commission rate of 0.25% in the back-tests, the framework is able to achieve at least 4-fold returns in 50 days.

Keywords: Machine learning; Convolutional Neural Networks; Recurrent Neural Networks; Long Short-Term Memory; Reinforcement learning; Deep Learning; Cryptocurrency; Bitcoin; Algorithmic Trading; Portfolio Management; Quantitative Finance

1. Introduction

Portfolio management is the decision making process of continuously reallocating an amount of fund into a number of different financial investment products, aiming to maximize the return while restraining the risk (Haugen, 1986; Markowitz, 1968). Traditional portfolio management methods can be classified into four categories, "Follow-the-Winner", "Follow-the-Loser", "Pattern-Matching", and "Meta-Learning" (Li and Hoi, 2014). The first two categories are based on prior-constructed financial models, while they may also be assisted by some machine learning techniques for parameter determinations (Li et al., 2012; Cover, 1996). The performance of these methods is dependent on the validity of the models on different markets. "Pattern-Matching" algorithms predict the next market distribution based

on a sample of historical data and explicitly optimizes the portfolio based on the sampled distribution (Györfi et al., 2006). The last class, "Meta-Learning" method combine multiple strategies of other categories to attain more consistent performance (Vovk and Watkins, 1998; Das and Banerjee, 2011).

There are existing deep machine-learning approaches to financial market trading. However, many of them try to predict price movements or trends (Heaton et al., 2016; Niaki and Hoseinzade, 2013; Freitas et al., 2009). With history prices of all assets as its input, a neural network can output a predicted vector of asset prices for the next period. Then the trading agent can act upon this prediction. This idea is straightforward to implement, because it is a supervised learning, or more specifically a regression problem. The performance of these price-prediction-based algorithms, however, highly depends on the degree of prediction accuracy, but it turns out that future market prices are difficult to predict. Furthermore, price predictions are not market actions, converting them into actions requires additional layer of logic. If this layer is a hand-coded, then the whole approach is not fully machine learning, and thus is not very extensible or adaptable. For example, it is difficult for a prediction-based network to consider transaction cost as a risk factor.

Previous successful attempts of model-free and fully machine-learning schemes to the algorithmic trading problem, without predicting future prices, are treating the problem as a Reinforcement Learning (RL) one. These include Moody and Saffell (2001), Dempster and Leemans (2006), Cumming (2015), and the recent deep RL utilization by Deng et al. (2017). These RL algorithms output discrete trading signals on an asset. Being limited to single-asset trading, they are not applicable to general portfolio management problems, where trading agents manage multiple assets.

Deep RL is lately drawing much attention due to its remarkable achievements in playing video games (Mnih et al., 2015) and board games (Silver et al., 2016). These are RL problems with discrete action spaces, and can not be directly applied to portfolio selection problems, where actions are continuous. Although market actions can be discretized, discretization is considered a major drawback, because discrete actions come with unknown risks. For instance, one extreme discrete action may be defined as investing all the capital into one asset, without spreading the risk to the rest of the market. In addition, discretization scales badly. Market factors, like number of total assets, vary from market to market. In order to take full advantage of adaptability of machine learning over different markets, trading algorithms have to be scalable. A general-purpose continuous deep RL framework, the actor-critic Deterministic Policy Gradient Algorithms, was recently introduced (Silver et al., 2014; Lillicrap et al., 2016). The continuous output in these actor-critic algorithms is achieved by a neural-network approximated action policy function, and a second network is trained as the reward function estimator. Training two neural networks, however, is found out to be difficult, and sometimes even unstable.

This paper proposes an RL framework specially designed for the task of portfolio management. The core of the framework is the Ensemble of Identical Independent Evaluators (EIIE) topology. An IIE is a neural network whose job is to inspect the history of an asset and evaluate its potential growth for the immediate future. The evaluation score of each asset is discounted by the size of its intentional weight change for the asset in the portfolio and is presented to a softmax layer, whose outcome will be the new portfolio weights for the coming trading period. The portfolio weights define the market action of the RL agent.

An asset with an increased target weight will be bought in with additional amount, and that with decreased weight will be sold. Apart from the market history, portfolio weights from the previous trading period are also input to the EIIE. This is for the RL agent to consider the effect of transaction cost to its wealth. For this purpose, the portfolio weights of each period are recorded in a Portfolio Vector Memory (PVM). The EIIE is trained in an Online Stochastic Batch Learning scheme (OSBL), which is compatible with both pre-trade training and online training during back-tests or online trading. The reward function of the RL framework is the explicit average of the periodic logarithmic returns. Having an explicit reward function, the EIIE evolves, under training, along the gradient ascending direction of the function. Three different species of IIEs are tested in this work, a Convolutional Neural Network (CNN) (Fukushima, 1980; Krizhevsky et al., 2012; Sermanet et al., 2012), a basic Recurrent Neural Network (RNN) (Werbos, 1988), and a Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997).

Being a fully machine-learning approach, the framework is not restricted to any particular markets. To examine its validity and profitability, the framework is tested in a cryptocurrency (virtual money, Bitcoin as the most famous example) exchange market, Polonix.com. A set of coins are preselected by their ranking in trading-volume over a time interval just before an experiment. Three back-test experiments of well separated time-spans are performed in a trading period of 30 minutes. The performance of the three EIIEs are compared with some recently published or reviewed portfolio selection strategies (Li et al., 2015a; Li and Hoi, 2014). The EIIEs significantly beat all other strategies in all three experiments

Cryptographic currencies, or simply cryptocurrencies, are electronic and decentralized alternatives to government-issued moneys (Nakamoto, 2008; Grinberg, 2012). While the best known example of a cryptocurrency is Bitcoin, there are more than 100 other tradable cryptocurrencies competing each other and with Bitcoin (Bonneau et al., 2015). The motive behind this competition is that there are a number of design flaws in Bitcoin, and people are trying to invent new coins to overcome these defects hoping their inventions will eventually replace Bitcoin (Bentov et al., 2014; Duffield and Hagan, 2014). There are, however, more and more cryptocurrencies being created without targeting to beat Bitcoin, but with the purposes of using the blockchain technology behind it to develop decentralized applications[1]. To June 2017, the total market capital of all cryptocurrencies is 102 billions in USD, 41 of which is of Bitcoin.[2] Therefore, regardless of its design faults, Bitcoin is still the dominant cryptocurrency in markets. As a result, many other currencies can not be bought with fiat currencies, but only be traded against Bitcoin.

Two natures of cryptocurrencies differentiate them from traditional financial assets, making their market the best test-ground for algorithmic portfolio management experiments. These natures are decentralization and openness, and the former implies the latter. Without a central regulating party, anyone can participate in cryptocurrency trading with low entrance requirements. One direct consequence is abundance of small-volume currencies. Affecting the prices of these penny-markets will require smaller amount of investment, compared to traditional markets. This will eventually allow trading machines to learn and

---

1. For example, Ethereum is a decentralized platform that runs smart contracts, and Siacoin is the currency for buying and selling storage service on the decentralized cloud Sia.
2. Crypto-currency market capitalizations, http://coinmarketcap.com/, accessed: 2017-06-30.

take advantage of the impacts by their own market actions. Openness also means the markets are more accessible. Most cryptocurrency exchanges have application programming interface for obtaining market data and carrying out trading actions, and most exchanges are open 24/7 without restricting frequency of tradings. These non-stop markets are ideal for machines to learn in the real world in shorter time-frames.

The paper is organized as follows. Section 2 defines the portfolio management problem that this project is aiming to solve. Section 3 introduces asset preselection and the reasoning behind it, the input price tensor, and a way to deal with missing data in the market history. The portfolio management problem is re-described in the language RL in Section 4. Section 5 presents the EIIE meta topology, the PVM, the OSBL scheme. The results of the three experiments are staged in Section 6.

2. Problem Definition

Portfolio management is the action of continuous reallocation of a capital into a number of financial assets. For an automatic trading robot, these investment decisions and actions are made periodically. This section provides a mathematical setting of the portfolio management problem.

2.1 Trading Period

In this work, trading algorithms are time-driven, where time is divided into periods of equal lengths T. At the beginning of each period, the trading agent reallocates the fund among the assets. $T = 30$ minutes in all experiments of this paper. The price of an asset goes up and down within a period, but four important price points characterize the overall movement of a period, namely the opening, highest, lowest and closing prices (Rogers and Satchell, 1991). For continuous markets, the opening price of a financial instrument in a period is the closing price from the previous period. It is assumed in the back-test experiments that at the beginning of each period assets can be bought or sold at the opening price of that period. The justification of such an assumption is given in Section 2.4.

2.2 Mathematical Formalism

The portfolio consists of m assets. The closing prices of all assets comprise the *price vector* for Period t, $\boldsymbol{v}_t$. In other words, the ith element of $\boldsymbol{v}_t$, $v_{i,t}$, is the closing price of the ith asset in the tth period. Similarly, $\boldsymbol{v}_t^{(\text{hi})}$ and $\boldsymbol{v}_t^{(\text{lo})}$ denote the highest and lowest prices of the period. The first asset in the portfolio is special, that it is the quoted currency, referred to as *the cash* for the rest of the article. Since the prices of all assets are quoted in cash, the first elements of $\boldsymbol{v}_t$, $\boldsymbol{v}_t^{(\text{hi})}$ and $\boldsymbol{v}_t^{(\text{lo})}$ are always one, that is $v_{0,t}^{(\text{hi})} = v_{0,t}^{(\text{lo})} = v_{0,t} = 1, \forall t$. In the experiments of this paper, the cash is Bitcoin.

For continuous markets, elements of $\boldsymbol{v}_t$ are the opening prices for Period $t+1$ as well as the closing prices for Period t. The *price relative vector* of the tth trading period, $\boldsymbol{y}_t$, is defined as the element-wise division of $\boldsymbol{v}_t$ by $\boldsymbol{v}_{t-1}$:

$$\boldsymbol{y}_t := \boldsymbol{v}_t \oslash \boldsymbol{v}_{t-1} = \left(1, \frac{v_{1,t}}{v_{1,t-1}}, \frac{v_{2,t}}{v_{2,t-1}}, ..., \frac{v_{m,t}}{v_{m,t-1}}\right)^{\mathsf{T}}. \tag{1}$$

The elements of $\boldsymbol{y}_t$ are the quotients of closing prices and opening prices for individual asset in the period. The price relative vector can be used to calculate the change in total portfolio value in a period. If p_{t-1} is the portfolio value at the begining of Period t, ignoring transaction cost,

$$p_t = p_{t-1}\,\boldsymbol{y}_t \cdot \boldsymbol{w}_{t-1}, \tag{2}$$

where $\boldsymbol{w}_{t-1}$ is the portfolio weight vector (referred to as the *portfolio vector* from now on) at the beginning of Period t, whose ith element, $w_{t-1,i}$, is the proportion of asset i in the portfolio after capital reallocation. The elements of $\boldsymbol{w}_t$ always sum up to one by definition, $\sum_i w_{t,i} = 1, \forall t$. The *rate of return* for Period t is then

$$\rho_t := \frac{p_t}{p_{t-1}} - 1 = \boldsymbol{y}_t \cdot \boldsymbol{w}_{t-1} - 1, \tag{3}$$

and the corresponding *logarithmic rate of return* is

$$r_t := \ln \frac{p_t}{p_{t-1}} = \ln \boldsymbol{y}_t \cdot \boldsymbol{w}_{t-1}. \tag{4}$$

In a typical portfolio management problem, the initial portfolio weight vector $\boldsymbol{w}_0$ is chosen to be the first basis vector in the Euclidean space,

$$\boldsymbol{w}_0 = (1, 0, ..., 0)^{\mathsf{T}}, \tag{5}$$

indicating all the capital is in the trading currency before entering the market. If there is no transaction cost, the final portfolio value will be

$$p_{\mathrm{f}} = p_0 \exp\left(\sum_{t=1}^{t_{\mathrm{f}}+1} r_t\right) = p_0 \prod_{t=1}^{t_{\mathrm{f}}+1} \boldsymbol{y}_t \cdot \boldsymbol{w}_{t-1}, \tag{6}$$

where p_0 is the initial investment amount. The job of a portfolio manager is to maximize p_{f} for a given time frame.

2.3 Transaction Cost

In a real-world scenario, buying or selling assets in a market is not free. The cost is normally from commission fee. Assuming a constant commission rate, this section will re-calculate the final portfolio value in Equation (6), using a recursive formula extending a work by Ormos and Urbán (2013).

The portfolio vector at the beginning of Period t is $\boldsymbol{w}_{t-1}$. Due to price movements in the market, at the end of the same period, the weights evolve into

$$\boldsymbol{w}_t' = \frac{\boldsymbol{y}_t \odot \boldsymbol{w}_{t-1}}{\boldsymbol{y}_t \cdot \boldsymbol{w}_{t-1}}, \tag{7}$$

where $\odot$ is the element-wise multiplication. The mission of the portfolio manager now at the end of Period t is to reallocate portfolio vector from $\boldsymbol{w}_t'$ to $\boldsymbol{w}_t$ by selling and buying relevant assets. Paying all commission fees, this reallocation action shrinks the portfolio value by a factor μ_t. $\mu_t \in (0, 1]$, and will be called the *transaction remainder factor* from
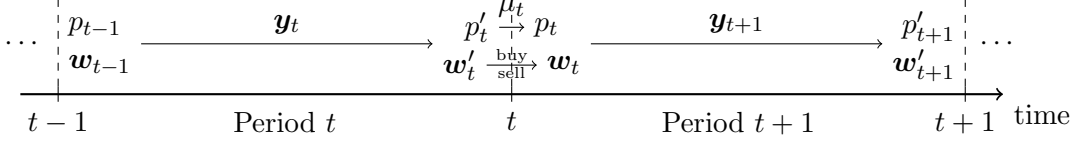
Figure 1: Illustration of the effect of transaction remainder factor μ_t. The market movement during Period t, represented by the price-relative vector $\boldsymbol{y}_t$, drives the portfolio value and portfolio weights from p_{t-1} and $\boldsymbol{w}_{t-1}$ to p'_t and $\boldsymbol{w}'_t$. The asset selling and purchasing action at time t redistributes the fund into $\boldsymbol{w}_t$. As a side-effect, these transactions shrink the portfolio to p_t by a factor of μ_t. The rate of return for Period t is calculated using portfolio values at the beginning of the two consecutive periods in Equation (9).

now on. μ_t is to be determined below. Denoting p_{t-1} as the portfolio value at the beginning of Period t and p'_t at the end,

$$p_t = \mu_t p'_t. \tag{8}$$

The rate of return (3) and logarithmic rate of return (4) are now

$$\rho_t = \frac{p_t}{p_{t-1}} - 1 = \frac{\mu_t p'_t}{p_{t-1}} - 1 = \mu_t \boldsymbol{y}_t \cdot \boldsymbol{w}_{t-1} - 1, \tag{9}$$

$$r_t = \ln \frac{p_t}{p_{t-1}} = \ln\left(\mu_t \boldsymbol{y}_t \cdot \boldsymbol{w}_{t-1}\right), \tag{10}$$

and the final portfolio value in Equation (6) becomes

$$p_{\mathrm{f}} = p_0 \exp\left(\sum_{t=1}^{t_{\mathrm{f}}+1} r_t\right) = p_0 \prod_{t=1}^{t_{\mathrm{f}}+1} \mu_t \boldsymbol{y}_t \cdot \boldsymbol{w}_{t-1}. \tag{11}$$

Different from Equation (4) and (2) where transaction cost is not considered, in Equation (10) and (11), $p'_t \neq p_t$ and the difference between the two values is where the transaction remainder factor comes into play. Figure 1 demonstrates the relationship among portfolio vectors and values and their dynamic relationship on a time axis.

The remaining problem is to determine this transaction remainder factor μ_t. During the portfolio reallocation from $\boldsymbol{w}'_t$ to $\boldsymbol{w}_t$, some or all amount of asset i need to be sold, if $p'_t w'_{t,i} > p_t w_{t,i}$ or $w'_{t,i} > \mu_t w_{t,i}$. The total amount of cash obtained by all selling is

$$(1 - c_{\mathrm{s}}) p'_t \sum_{i=1}^{m} (w'_{t,i} - \mu_t w_{t,i})^+ \tag{12}$$

where $0 \leqslant c_{\mathrm{s}} < 1$ is the commission rate for selling, and $(\boldsymbol{v})^+ = \mathrm{ReLu}(\boldsymbol{v})$ is the element-wise rectified linear function, $(x)^+ = x$ if $x > 0$, $(x)^+ = 0$ otherwise. This money and the original cash reserve $p'_t w'_{t,0}$ taken away the new reserve $\mu_t p'_t w_{t,0}$ will be used to buy new assets,

$$(1 - c_{\mathrm{p}}) \left[w'_{t,0} + (1 - c_{\mathrm{s}}) \sum_{i=1}^{m} (w'_{t,i} - \mu_t w_{t,i})^+ - \mu_t w_{t,0} \right] = \sum_{i=1}^{m} (\mu_t w_{t,i} - w'_{t,i})^+, \tag{13}$$

6

where $0 \leqslant c_{\mathrm{p}} < 1$ is the commission rate for purchasing, and p'_t has been canceled out on both sides. Using identity $(a - b)^+ - (b - a)^+ = a - b$, and the fact that $w'_{t,0} + \sum_{i=1}^{m} w'_{t,i} = 1 = w_{t,0} + \sum_{i=1}^{m} w_{t,i}$, Equation (13) is simplified to

$$\mu_t = \frac{1}{1 - c_{\mathrm{p}} w_{t,0}} \left[1 - c_{\mathrm{p}} w'_{t,0} - (c_{\mathrm{s}} + c_{\mathrm{p}} - c_{\mathrm{s}} c_{\mathrm{p}}) \sum_{i=1}^{m} (w'_{t,i} - \mu_t w_{t,i})^+ \right]. \tag{14}$$

The presence of μ_t inside a linear rectifier means μ_t is not solvable analytically, but it can only be solved iteratively.

Theorem 1 *Denoting*

$$f(\mu) := \frac{1}{1 - c_{\mathrm{p}} w_{t,0}} \left[1 - c_{\mathrm{p}} w'_{t,0} - (c_{\mathrm{s}} + c_{\mathrm{p}} - c_{\mathrm{s}} c_{\mathrm{p}}) \sum_{i=1}^{m} (w'_{t,i} - \mu w_{t,i})^+ \right],$$

the sequence $\left\{ \tilde{\mu}_t^{(k)} \right\}$, defined as

$$\left\{ \tilde{\mu}_t^{(k)} \,\middle|\, \tilde{\mu}_t^{(0)} = \mu_\odot \text{ and } \tilde{\mu}_t^{(k)} = f\left(\tilde{\mu}_t^{(k-1)} \right), \, k \in \mathbb{N}_0 \right\} \tag{15}$$

converges to μ_t, the solution to Equation (14), for any $\mu_\odot \in [0, 1]$.

While this convergence is not stated in Ormos and Urbán (2013), its proof will be given in Appendix A. This theorem provides a way to approximate the transaction remainder factor μ_t to an arbitrary accuracy. The speed on the convergence depends on the error of the initial guest $\mu_\odot$. The smaller $|\mu_t - \mu_\odot|$ is, the quicker Sequence (15) converges to μ_t. When $c_{\mathrm{p}} = c_{\mathrm{s}} = c$, there is a practice (Moody et al., 1998) to approximate μ_t with $c \sum_{i=1}^{m} |w'_{t,i} - w_{t,i}|$. Therefore, in this work, $\mu_\odot$ will use this as the first value for the sequence, that

$$\mu_\odot = c \sum_{i=1}^{m} \left| w'_{t,i} - w_{t,i} \right|. \tag{16}$$

In the training of the neural networks, $\tilde{\mu}_t^{(k)}$ with a fixed k in (15) is used. In the back-test experiments, a tolerant error δ dynamically determines k, that is the first k, such that $\left| \tilde{\mu}_t^{(k)} - \tilde{\mu}_t^{(k-1)} \right| < \delta$, is used for $\tilde{\mu}_t^{(k)}$ to approximate μ_t. In general, μ_t and its approximations are functions of portfolio vectors of two recent periods and the price relative vector,

$$\mu_t = \mu_t(\boldsymbol{w}_{t-1}, \boldsymbol{w}_t, \boldsymbol{y}_t). \tag{17}$$

Throughout this work, a single constant commission rate for both selling and purchasing for all non-cash assets is used, $c_{\mathrm{s}} = c_{\mathrm{p}} = 0.25\%$, the maximum rate at Poloniex. The purpose of the algorithmic agent is to generate a time-sequence of portfolio vectors $\{\boldsymbol{w}_1, \boldsymbol{w}_2, \cdots, \boldsymbol{w}_t, \cdots\}$ in order to maximize the accumulative capital in (11), taking transaction cost into account.

2.4 Two Hypotheses

In this work, back-test tradings are only considered, where the trading agent pretends to be back in time at a point in the market history, not knowing any "future" market information, and does paper trading from then onward. As a requirement for the back-test experiments, the following two assumptions are imposed:

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 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.

In a real-world trading environment, if the trading volume in a market is high enough, these two assumptions are near to reality.

3. Data Treatments

The trading experiments are done in the exchange Poloniex, where there are about 80 tradable cryptocurrency pairs with about 65 available cryptocurrencies[3]. However, for the reasons given below, only a subset of coins is considered by the trading robot in one period. Apart from coin selection scheme, this section also gives a description of the data structure that the neural networks take as their input, a normalization pre-process, and a scheme to deal with missing data.

3.1 Asset Pre-Selection

In the experiments of the paper, the 11 most-volumed non-cash assets are preselected for the portfolio. Together with the cash, Bitcoin, the size of the portfolio, $m + 1$, is 12. This number is chosen by experience and can be adjusted in future experiments. For markets with large volumes, like the foreign exchange market, m can be as big as the total number of available assets.

One reason for selecting top-volumed cryptocurrencies (simply called coins below) is that bigger volume implies better market liquidity of an asset. In turn it means the market condition is closer to Hypothesis 1 set in Section 2.4. Higher volumes also suggest that the investment can have less influence on the market, establishing an environment closer to the Hypothesis 2. Considering the relatively high trading frequency (30 minutes) compared to some daily trading algorithms, liquidity and market size are particularly important in the current setting. In addition, the market of cryptocurrency is not stable. Some previously rarely- or popularly-traded coins can have sudden boost or drop in volume in a short period of time. Therefore, the volume for asset preselection is of a longer time-frame, relative to the trading period. In these experiments, volumes of 30 days are used.

However, using top volumes for coin selection in back-test experiments can give rise to a *survival bias*. The trading volume of an asset is correlated to its popularity, which in turn is governed by its historic performance. Giving future volume rankings to a back-test, will inevitably and indirectly pass future price information to the experiment, causing
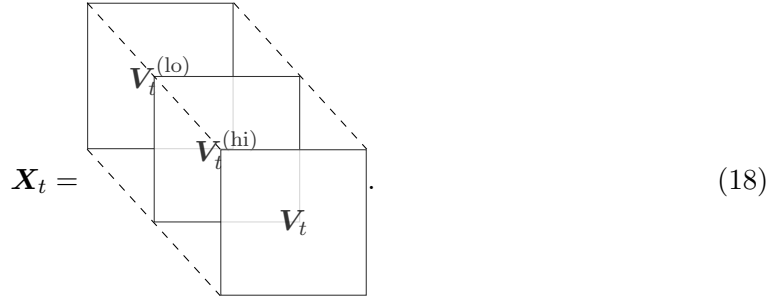
---

3. as of May 23, 2017.

unreliable positive results. For this reason, volume information just before the beginning of the back-tests is taken for preselection to avoid survival bias.

3.2 Price Tensor

Historic price data is fed into a neural network to generate the output of a portfolio vector. This subsection describes the structure of the input tensor, its normalization scheme, and how missing data is dealt with.

The input to the neural networks at the end of Period t is a tensor, $\boldsymbol{X}_t$, of rank 3 with shape (f, n, m), where m is the number of preselected non-cash assets, n is the number of input periods before t, and $f = 3$ is the feature number. Since prices further back in the history have much less correlation to the current moment than that of recent ones, $n = 50$ (a day and an hour) for the experiments. The criterion of choosing the m assets were given in Section 3.1. Features for asset i on Period t are its closing, highest, and lowest prices in the interval. Using the notations from Section 2.2, these are $v_{i,t}$, $v_{i,t}^{(\mathrm{hi})}$, and $v_{i,t}^{(\mathrm{lo})}$. However, these absolute price values are not directly fed to the networks. Since only the changes in prices will determine the performance of the portfolio management (Equation (10)), all prices in the input tensor will be normalization by the latest closing prices. Therefore, $\boldsymbol{X}_t$ is the stacking of the three normalized price matrices,

$$\boldsymbol{X}_t = \begin{array}{c} \boldsymbol{V}_t^{(\mathrm{lo})} \\ \boldsymbol{V}_t^{(\mathrm{hi})} \\ \boldsymbol{V}_t \end{array}. \tag{18}$$

where $\boldsymbol{V}_t$, $\boldsymbol{V}_t^{(\mathrm{hi})}$, and $\boldsymbol{V}_t^{(\mathrm{lo})}$ are the normalized price matrices,

$$\boldsymbol{V}_t = \left[\boldsymbol{v}_{t-n+1} \oslash \boldsymbol{v}_t \middle| \boldsymbol{v}_{t-n+2} \oslash \boldsymbol{v}_t \middle| \cdots \middle| \boldsymbol{v}_{t-1} \oslash \boldsymbol{v}_t \middle| \boldsymbol{1} \right],$$

$$\boldsymbol{V}_t^{(\mathrm{hi})} = \left[\boldsymbol{v}_{t-n+1}^{(\mathrm{hi})} \oslash \boldsymbol{v}_t \middle| \boldsymbol{v}_{t-n+2}^{(\mathrm{hi})} \oslash \boldsymbol{v}_t \middle| \cdots \middle| \boldsymbol{v}_{t-1}^{(\mathrm{hi})} \oslash \boldsymbol{v}_t \middle| \boldsymbol{v}_t^{(\mathrm{hi})} \oslash \boldsymbol{v}_t \right],$$

$$\boldsymbol{V}_t^{(\mathrm{lo})} = \left[\boldsymbol{v}_{t-n+1}^{(\mathrm{lo})} \oslash \boldsymbol{v}_t \middle| \boldsymbol{v}_{t-n+2}^{(\mathrm{lo})} \oslash \boldsymbol{v}_t \middle| \cdots \middle| \boldsymbol{v}_{t-1}^{(\mathrm{lo})} \oslash \boldsymbol{v}_t \middle| \boldsymbol{v}_t^{(\mathrm{lo})} \oslash \boldsymbol{v}_t \right],$$

with $\boldsymbol{1} = (1, 1, \cdots, 1)^{\mathsf{T}}$, and $\oslash$ being the element-wise division operator.

At the end of Period t, the portfolio manager comes up with a portfolio vector $\boldsymbol{w}_t$ using merely the information from the price tensor $\boldsymbol{X}_t$ and the previous portfolio vector $\boldsymbol{w}_{t-1}$, according to some policy π. In other words, $\boldsymbol{w}_t = \pi(\boldsymbol{X}_t, \boldsymbol{w}_{t-1})$. At the end of Period $t + 1$, the logarithmic rate of return for the period due to decision $\boldsymbol{w}_t$ can be calculated with the additional information from the price change vector $\boldsymbol{y}_{t+1}$, using Equation (10), $r_{t+1} = \ln(\mu_{t+1} \boldsymbol{y}_{t+1} \cdot \boldsymbol{w}_t)$. In the language of RL, r_{t+1} is the immediate reward to the portfolio management agent for its action $\boldsymbol{w}_t$ under environment condition $\boldsymbol{X}_t$.

3.3 Filling Missing Data

Some of the selected coins lack part of the history. This absence of data is due to the fact that these coins just appeared relatively recently. Data points before the existence of a coin are marked as Not A Numbers (NANs) from the exchange. NANs only appeared in the training set, because the coin selection criterion is the volume-ranking of the last 30 days before the back-tests, meaning all assets must have existed before that.

As the input of a neural network must be real numbers, these NANs have to be replaced. In a previous work of the authors (Jiang and Liang, 2017), the missing data was filled with fake decreasing price series with a decay rate of 0.01, in order for the neural networks to avoid picking these absent assets in the training process. However, it turned out that the networks deeply remembered these particular assets, that they avoided them even when they were in very promising up-climbing trends in the back-test experiments. For this reason, in this current work, flat fake price-movements (0 decay rates) are used to fill the missing data points. In addition, under the novel EIIE structure, the new networks will not be able to reveal the identity of individual assets, preventing them from making decision based on the long-past bad records of particular assets.

4. Reinforcement Learning

With the problem defined in Section 2 in mind, this section presents a reinforcement-learning (RL) solution framework using a deterministic policy gradient algorithm. The explicit reward function is also given under this framework.

4.1 The Environment and the Agent

In the problem of algorithmic portfolio management, the agent is the software portfolio manager performing trading-actions in the environment of a financial market. This environment comprises of all available assets in the markets and the expectations of all market participants towards them.

It is impossible for the agent to get total information of a state of such a large and complex environment. Nonetheless, all relevant information is believed, in the philosophy of technical traders (Charles et al., 2006; Lo et al., 2000), to be reflected in the prices of the assets, which are publicly available to the agent. Under this point of view, an environmental state can be roughly represented by the prices of all orders throughout the market's history up to the moment where the state is at. Although full order history is in the public domain for many financial markets, it is too huge a task for the software agent to practically process this information. As a consequence, sub-sampling schemes for the order-history information are employed to future simplify the state representation of the market environment. These schemes include asset preselection described in Section 3.1, 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 periods. History cut-off simply takes the price-features of only a recent number of periods to represent the current state of the environment. The resultant representation is the price tensor $\boldsymbol{X}_t$ described in Section 3.2.

Under Hypothesis 2 in Section 2.4, the trading action of the agent will not influence the future price states of the market. However, the action made at the beginning of Period t

will affect the reward of Period $t + 1$, and as a result will affect the decision of its action. The agent's buying and selling transactions made at the beginning of Period $t+1$, aiming to redistribute the wealth among the assets, are determined by the difference between portfolio weights $\boldsymbol{w}_t'$ and $\boldsymbol{w}_t$. $\boldsymbol{w}_t'$ is defined in term of $\boldsymbol{w}_{t-1}$ in Equation (7), which also plays a role in the action for the last period. Since $\boldsymbol{w}_{t-1}$ has already been determined in the last period the action of the agent at time t can be represented solely by the portfolio vector $\boldsymbol{w}_t$,

$$\boldsymbol{a}_t = \boldsymbol{w_t}. \tag{19}$$

Therefore a previous action does have influence on the decision of the current one through the dependency of r_{t+1} and μ_{t+1} on $\boldsymbol{w}_t$ (17). In the current framework, this influence is encapsulated by considering $\boldsymbol{w}_{t-1}$ as a part of the environment and inputting it to the agent's action making policy, so the state at t is represented as the pair of X_t and $\boldsymbol{w}_{t-1}$,

$$\boldsymbol{s}_t = (\boldsymbol{X}_t, \boldsymbol{w}_{t-1}), \tag{20}$$

where $\boldsymbol{w}_0$ is predetermined in (5). The state $\boldsymbol{s}_t$ consists of two parts, the external state represented by the price tensor, $\boldsymbol{X}_t$, and the internal state represented by the portfolio vector from the last period, $\boldsymbol{w}_{t-1}$. Because under Hypothesis 2 of Section 2.4, the portfolio amount is negligible compared to the total trading volume of the market, p_t is not included in the internal state.

4.2 Full-Exploitation and the Reward Function

It is the job of the agent to maximize the final portfolio value p_f of Equation (11) at the end of the $t_\mathrm{f} + 1$ period. As the agent does not have control over the choices of the initial investment, p_0, and the length of the whole portfolio management process, t_f, this job is equivalent to maximizing the average logarithmic cumulated return R,

$$R(\boldsymbol{s}_1, \boldsymbol{a}_1, \cdots, \boldsymbol{s}_{t_\mathrm{f}}, \boldsymbol{a}_{t_\mathrm{f}}, \boldsymbol{s}_{t_\mathrm{f}+1}) := \frac{1}{t_\mathrm{f}} \ln \frac{p_\mathrm{f}}{p_0} = \frac{1}{t_\mathrm{f}} \sum_{t=1}^{t_\mathrm{f}+1} \ln \left(\mu_t \boldsymbol{y}_t \cdot \boldsymbol{w}_{t-1} \right) \tag{21}$$

$$= \frac{1}{t_\mathrm{f}} \sum_{t=1}^{t_\mathrm{f}+1} r_t. \tag{22}$$

On the right-hand side of (21), $\boldsymbol{w}_{t-1}$ is given by action $\boldsymbol{a}_{t-1}$, $\boldsymbol{y}_t$ is part of price tensor $\boldsymbol{X}_t$ from state variable $\boldsymbol{s}_t$, and μ_t is a function of $\boldsymbol{w}_{t-1}$, $\boldsymbol{w}_t$ and $\boldsymbol{y}_t$ as stated in (17). In the language of RL, R is the cumulated reward, and r_t/t_f is the immediate reward for an individual episode. Different from a reward function using accumulated portfolio value (Moody et al., 1998), the denominator t_f guarantees the fairness of the reward function between runs of different lengths, enabling it to train the trading policy in mini-batches. With this reward function, the current framework has two important distinctions from many other RL problems. One is that both the episodic and cumulated rewards are exactly expressed. In other words, the domain knowledge of the environment is well-mastered, and can be fully exploited by the agent. This exact expressiveness is based upon Hypothesis 1 of Section 2.4 that an action has no influence on the external part of future states, the price tensor. This isolation of action and external environment also allows one to use the

same segment of market history to evaluate difference sequences of actions. This feature of the framework is considered a major advantage, because a complete new trial in a trading game is both time-consuming and expansive.

The second distinction is that all episodic rewards are equally important to the final return. This distinction, together with the zero-market-impact assumption, allows r_t/t_f to be regarded as the action-value function of action $\boldsymbol{w}_t$ with a discounted factor of 0, taking no consideration of future influence of the action. Having a definite action-value function further justifies the full-exploitation approach, since exploration in other RL problems is mainly for trying out different classes of action-value functions.

Without exploration, on the other hand, local optima can be avoided by random initialisation of the policy parameters which will be discussed below.

4.3 Deterministic Policy Gradient

A policy is a mapping from the state space to the action space, $\pi : \mathcal{S} \rightarrow \mathcal{A}$. With full exploitation in the current framework, an action is deterministically produced by the policy from a state. The optimal policy is obtained using a gradient ascent algorithm. To achieve this, a policy is specified by a set of parameter $\boldsymbol{\theta}$, and $\boldsymbol{a}_t = \pi_{\boldsymbol{\theta}}(\boldsymbol{s}_t)$. The performance metric of $\pi_{\boldsymbol{\theta}}$ for time interval $[0, t_\text{f}]$ is defined as the corresponding reward function (21) of the interval,

$$J_{[0,t_\text{f}]}(\pi_{\boldsymbol{\theta}}) = R\left(\boldsymbol{s}_1, \pi_{\boldsymbol{\theta}}(s_1), \cdots, \boldsymbol{s}_{t_\text{f}}, \pi_{\boldsymbol{\theta}}(s_{t_\text{f}}), \boldsymbol{s}_{t_\text{f}+1}\right). \tag{23}$$

After random initialisation, the parameters are continuously updated along the gradient direction with a learning rate λ,

$$\boldsymbol{\theta} \longrightarrow \boldsymbol{\theta} + \lambda \nabla_{\boldsymbol{\theta}} J_{[0,t_\text{f}]}(\pi_{\boldsymbol{\theta}}). \tag{24}$$

To improve training efficiency and avoid machine-precision errors, $\boldsymbol{\theta}$ will be updated upon mini-batches instead of the whole training market-history. If the time-range of a mini-batch is $[t_{\text{b}_1}, t_{\text{b}_2}]$, the updating rule for the batch is

$$\boldsymbol{\theta} \longrightarrow \boldsymbol{\theta} + \lambda \nabla_{\boldsymbol{\theta}} J_{[t_{\text{b}_1}, t_{\text{b}_2}]}(\pi_{\boldsymbol{\theta}}), \tag{25}$$

with the denominator in the corresponding R defined in (21) replaced by $t_{\text{b}_2} - t_{\text{b}_1}$. This mini-batch approach of gradient ascent also allows online learning, which is important in online trading where new market history keep coming to the agent. Details of the online learning and mini-batch training will be discussed in Section 5.3

5. Policy Networks

The policy functions $\pi_{\boldsymbol{\theta}}$ will be constructed using three different deep neural networks. The neural networks in this paper differ from a previous version (Jiang and Liang, 2017) with three important innovations, the mini-machine topology invented to target the portfolio management problem, the portfolio-vector memory, and a stochastic mini-batch online learning scheme.

5.1 Network Topologies

The three incarnations of neural networks to build up the policy functions are a CNN, a basic RNN, and a LSTM. Figure 2 shows the topology of a CNN designed for solving the current portfolio management problem, while Figure 3 portrays the structure of a basic RNN or LSTM network for the same problem. In all cases, the input to the networks is the price tensor $\boldsymbol{X}_t$ defined in (18), and the output is the portfolio vector $\boldsymbol{w}_t$. In both figures, an hypothetical example of output portfolio vector is used, while the dimension of the price tensor and thus the number of assets are actual values deployed in the experiments. The last hidden layers are the voting scores for all non-cash assets. The softmax outcomes of these scores and a cash bias become the actual corresponding portfolio weights. In order for the neural network to consider transaction cost, the portfolio vector from the last period, $\boldsymbol{w}_{t-1}$, is inserted to the networks just before the voting-layer. The actual mechanism of storing and retrieving portfolio vectors in a parallel manner is presented in Section 5.2.

A vital common feature in all three networks is that the networks flow independently for the m assets while network parameters are shared among these streams. These streams are like independent but identical networks of smaller scopes, separately observing and assessing individual non-cash assets. They only interconnect at the softmax function, just to make sure their outputting weights are non-negative and summing up to unity. We call these streams mini-machines or more formally Identical Independent Evaluators (IIE), and this topology feature Ensemble of IIE (EIIE) nicknamed mini-machine approach, to distinguish with the wholesome approach in an earlier attempt (Jiang and Liang, 2017). EIIE is realized differently in Figure 2 and 3. An IIE in Figure 2 is just a chain of convolution with kernels of height 1, while in Figure 3 it is either a LSTM or a Basic RNN taking the price history of a single asset as input.

EIIE greatly improves the performance of the portfolio management. Remembering the historic performance of individual assets, an integrated network in the previous version is more reluctant to invest money to a historically unfavorable asset, even if the asset has a much more promising future. On the other hand, without being designed to reveal the identity of the assigned asset, an IIE is able to judge its potential rise and fall merely based on more recent events.

From a practical point of view, EIIE has three other crucial advantages over an integrated network. The first is scalability in asset number. Having the mini-machines all identical with shared parameters, the training time of an ensemble scales roughly linearly with m. The second advantage is data-usage efficiency. For an interval of price history, a mini-machine can be trained m times across different assets. Asset assessing experience of the IIEs is then shared and accumulated in both time and asset dimensions. The final advantage is plasticity to asset collection. Since an IIE's asset assessing ability is universal without being restricted to any particular assets, an EIIE can update its choice of assets and/or the size of the portfolio in real-time, without having to train the network again from ground zero.

5.2 Portfolio-Vector Memory

In order for the portfolio management agent to minimize transaction cost by restraining itself from large changes between consecutive portfolio vectors, the output of portfolio weights
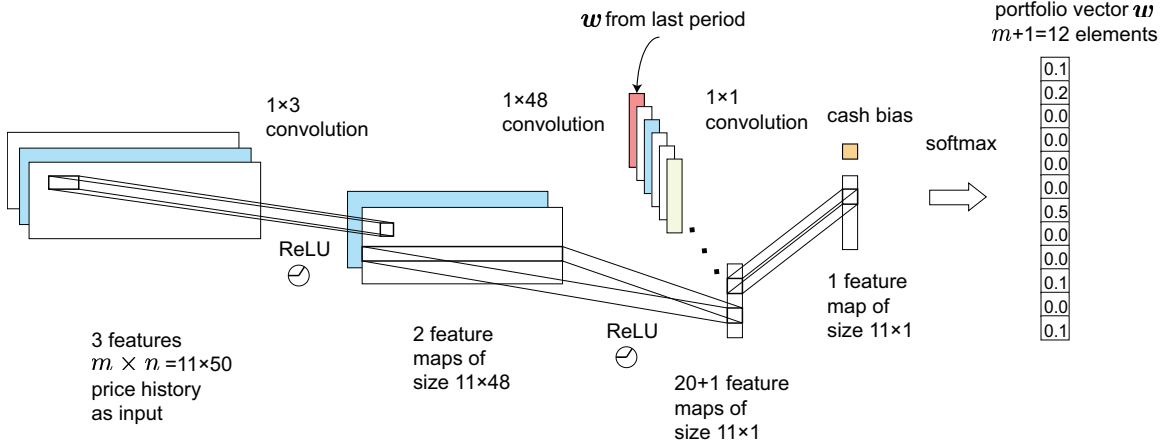
Figure 2: CNN Implementation of the EIIE: This is a realization the Ensemble of Identical Independent Evaluators (EIIE), a fully convolutional network. The first dimensions of all the local receptive fields in all feature maps are 1, making all rows isolated from each other until the softmax activation. Apart from weight-sharing among receptive fields in a feature map, which is a usual CNN characteristic, parameters are also shared between rows in an EIIE configuration. Each row of the entire network is assigned with a particular asset, and is responsible to submit a voting score to the softmax on the growing potential of the asset in the coming trading period. The input to the network is a $3 \times m \times n$ price tensor, comprising the highest, closing, and lowest prices of m non-cash assets over the past n periods. The outputs are the new portfolio weights. The previous portfolio weights are inserted as an extra feature map before the scoring layer, for the agent to minimize transaction cost.

from the previous trading period is input to the networks. One way to achieve this is to rely on the remembering ability of RNN, but with this approach the price normalization scheme proposed in (18) has to be abandoned. This normalization scheme is empirically better performing than others. Another possible solution is Direct Reinforcement (RR) introduced by Moody and Saffell (2001). However, both RR and RNN memory suffer from the gradient vanishing problem. More importantly, RR and RNN require serialization of the training process, unable to utilize parallel training within mini-batches.

In this work, inspired by the idea of experience replay memory (Mnih et al., 2016), a dedicated Portfolio-Vector Memory (PVM), is introduced to store the network outputs. As shown in Figure 4, the PVM is a stack of portfolio vectors in chronological order. Before any network training, the PVM is initialized with uniform weights. In each training step, a policy network loads the portfolio vector of the previous period from the memory location at $t-1$, and overwrites the memory at t with its output. As the parameters of the policy networks converge through many training epochs, the values in the memory also converge.

Sharing a single memory stack allows a network to be trained simultaneously against data points within a mini-batches, enormously improving training efficiency. In the case of
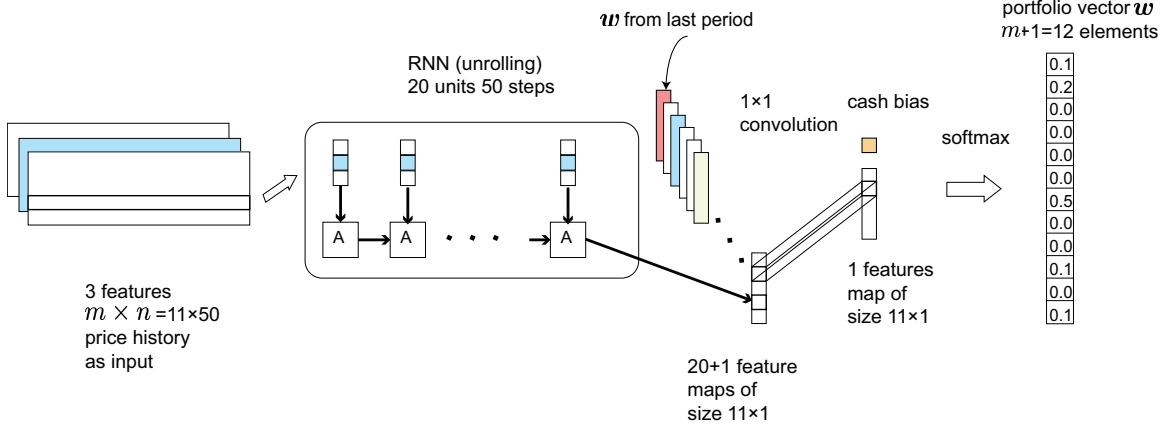
Figure 3: RNN (Basic RNN or LSTM) Implementation of the EIIE: This is a recurrent realization the Ensemble of Identical Independent Evaluators (EIIE). In this version, the price inputs of individual assets are taken by small recurrent subnets. These subnets are identical LSTMs or Basic RNNs. The structure of the ensemble network after the recurrent subnets is the same as the second half of the CNN in Figure 2.
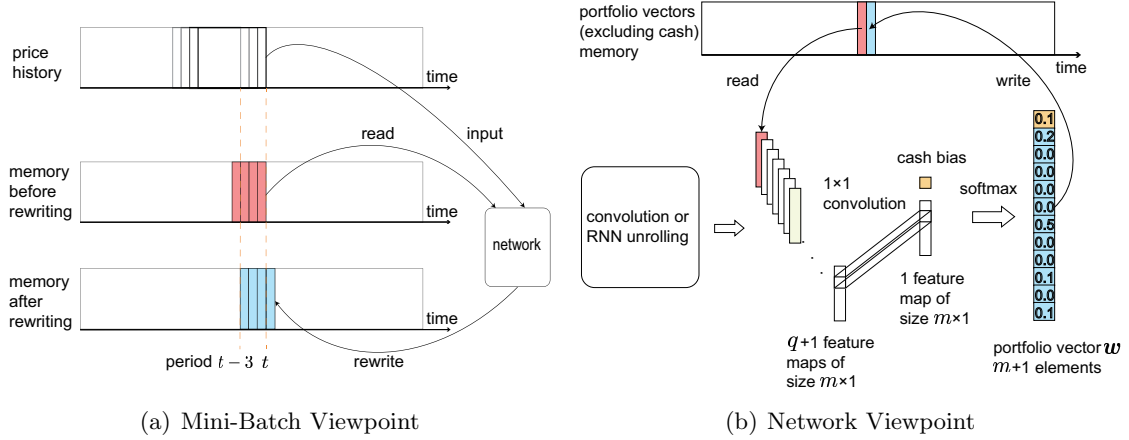

(a) Mini-Batch Viewpoint

(b) Network Viewpoint

Figure 4: A Read/Write Cycle of the Portfolio-Vector Memory: In both graphs, a small vertical strip on the time axis represents a portion of the memory containing the portfolio weights at the beginning of a period. Red memories are being read to the policy network, while blue ones are being overwritten by the network. The two colored rectangles in (a) consisting of four strips are example of two consecutive mini-batches. While (a) exhibits a complete read and write circle for a mini-batch, (b) shows a circle within a network (omitting the CNN or RNN part of the network).

RNN versions of the networks, inserting last outputs after the recurrent blocks (Figure 3) avoids passing the gradients back to the deep RNN structures, circumventing the gradient vanishing problem.

5.3 Online Stochastic Batch Learning

With the introduction of the network output-memory, mini-batch training becomes plausible, although the learning framework requires sequential inputs. However, unlike supervised learning, where data points are unordered and mini-batches are random disjoint subsets of the training sample space, in this training scheme the data points within a batch have to be in their time-order. In addition, since data sets are time series, mini-batches starting with different periods are considered valid and distinctive, even if they have a significantly overlapping interval. For example, if the uniform batch size is n_{b}, data sets covering $[t_{\mathrm{b}}, t_{\mathrm{b}} + n_{\mathrm{b}})$ and $[t_{\mathrm{b}} + 1, t_{\mathrm{b}} + n_{\mathrm{b}} + 1)$ are two validly different batches.

The ever-ongoing nature of financial markets means new data keeps pouring into the agent, and as a consequence the size of the of training sample explodes indefinitely. Fortunately, it is believed that the correlation between two market price events decades exponentially with the temporal distance between them (Holt, 2004; Charles et al., 2006). With this belief, here an Online Stochastic Batch Learning (OSBL) scheme is proposed.

At the end of the tth period, the price movement of this period will be added to the training set. After the agent has completed its orders for period $t + 1$, the policy network will be trained against N_{b} randomly chosen mini-batches from this set. A batch starting with period $t_{\mathrm{b}} \leqslant t - n_{\mathrm{b}}$ is picked with a geometrically distributed probability $P_{\beta}(t_{\mathrm{b}})$,

$$P_{\beta}(t_{\mathrm{b}}) = \beta(1 - \beta)^{t - t_{\mathrm{b}} - n_{\mathrm{b}}}, \tag{26}$$

where $\beta \in (0, 1)$ is the probability-decaying rate determining the shape of the probability distribution and how important are recent market events, and n_{b} is the number of periods in a mini-batch.

6. Experiments

The tools has been developed to this point of the article are examined in three back-test experiments of different time frames with all three policy networks on the crypto-currency exchange Poloniex. Results are compared with many well-established and recently published portfolio-selection strategies. The main compared financial metric is the portfolio value as well as maximum drawdown and the Sharpe ratio.

6.1 Test Ranges

Details of the time-ranges for the back-test experiments and their corresponding training sets are presented in Table 1. A cross validation set is used for determination of the hyperparameters, whose range is also listed. All time in the table are in Coordinated Universal Time (UTC). All training sets start at 0 o'clock. For example, the training set for Back-Test 1 is from 00:00 on November 1st 2014. All price data is accessed with Poloniex's official Application Programming Interface (API)[4].

---

4. https://poloniex.com/support/api/

Data Purpose	Data Range	Training Data Set
CV	2016-05-07 04:00 to 2016-06-27 08:00	2014-07-01 to 2016-05-07 04:00
Back-Test 1	2016-09-07 04:00 to 2016-10-28 08:00	2014-11-01 to 2016-09-07 04:00
Back-Test 2	2016-12-08 04:00 to 2017-01-28 08:00	2015-02-01 to 2016-12-08 04:00
Back-Test 3	2017-03-07 04:00 to 2017-04-27 08:00	2015-05-01 to 2017-03-07 04:00

Table 1: Price data ranges for hyperparameter-selection (cross-validation, CV) and back-test experiments. Prices are accessed in periods of 30 minutes. Closing prices are used for cross validation and back-tests, while highest, lowest, and closing prices in the periods are used for training. The hours of the starting points for the training sets are not given, since they begin at midnight of the days. All times are in UTC.

6.2 Performance Measures

Different metrics are used to measure the performance of a particular portfolio selection strategy. The most direct measurement of how successful is a portfolio management over a timespan is the accumulative portfolio value (APV), p_t. It is unfair, however, to compare the PVs of two management starting of different initial values. Therefore, APVs here are measured in the unit of their initial values, or equivalently $p_0 = 1$ and thus

$$p_t = p_t/p_0. \tag{27}$$

In this unit, APV is then closely related to the accumulated return, and in fact it only differs from the latter by 1. Under the same unit, the final APV (fAPV) is the APV at the end of a back-test experiment, $p_f = p_f/p_0 = p_{t_f+1}/p_0$.

A major disadvantage of APV is that it does not measure the risk factors, since it merely sums up all the periodic returns without considering fluctuation in these returns. A second metric, the Sharpe ratio (SR) (Sharpe, 1964, 1994), is used to take risk into account. The ratio is a risk adjusted mean return, defined as the average of the risk-free return by its deviation,

$$S = \frac{\mathbb{E}_t[\rho_t - \rho_{\mathrm{F}}]}{\sqrt{\underset{t}{\mathrm{var}}\,(\rho_t - \rho_{\mathrm{F}})}}, \tag{28}$$

where ρ_t are periodic returns defined in (9), and ρ_{F} is the rate of return of a risk-free asset. In these experiments the risk-free asset is Bitcoin. Because the quoted currency is also Bitcoin, the risk-free return is zero, $\rho_{\mathrm{F}} = 0$, here.

Although the SR considers volatility of the portfolio values, but it equally treats upwards and downwards movements. In reality upwards volatility contributes to positive returns, but downwards to loss. In order to highlight the downwards deviation, Maximum Drawdown (MDD) (Magdon-Ismail and Atiya, 2004) is also considered. MDD is the biggest loss from a peak to a trough, and mathematically

$$D = \max_{\underset{t}{\tau>t}} \frac{p_t - p_\tau}{p_t}. \tag{29}$$

Algorithm	2016-09-07 to 2016-10-28			2016-12-08 to 2017-01-28			2017-03-07 to 2017-04-27		
	MDD	fAPV	SR	MDD	fAPV	SR	MDD	fAPV	SR
CNN	0.224	**29.695**	**0.087**	0.216	**8.026**	**0.059**	0.406	31.747	0.076
bRNN	0.241	13.348	0.074	0.262	4.623	0.043	0.393	**47.148**	**0.082**
LSTM	0.280	6.692	0.053	0.319	4.073	0.038	0.487	21.173	0.060
iCNN	**0.221**	4.542	0.053	0.265	1.573	0.022	0.204	3.958	0.044
Best Stock	0.654	1.223	0.012	0.236	1.401	0.018	0.668	4.594	0.033
UCRP	0.265	0.867	-0.014	**0.185**	1.101	0.010	**0.162**	2.412	0.049
UBAH	0.324	0.821	-0.015	0.224	1.029	0.004	0.274	2.230	0.036
Anticor	0.265	0.867	-0.014	0.185	1.101	0.010	0.162	2.412	0.049
OLMAR	0.913	0.142	-0.039	0.897	0.123	-0.038	0.733	4.582	0.034
PAMR	0.997	0.003	-0.137	0.998	0.003	-0.121	0.981	0.021	-0.055
WMAMR	0.682	0.742	-0.0008	0.519	0.895	0.005	0.673	6.692	0.042
CWMR	0.999	0.001	-0.148	0.999	0.002	-0.127	0.987	0.013	-0.061
RMR	0.900	0.127	-0.043	0.929	0.090	-0.045	0.698	7.008	0.041
ONS	0.233	0.923	-0.006	0.295	1.188	0.012	0.170	1.609	0.027
UP	0.269	0.864	-0.014	0.188	1.094	0.009	0.165	2.407	0.049
EG	0.268	0.865	-0.014	0.187	1.097	0.010	0.163	2.412	0.049
B^K	0.436	0.758	-0.013	0.336	0.770	-0.012	0.390	2.070	0.027
CORN	0.999	0.001	-0.129	1.000	0.0001	-0.179	0.999	0.001	-0.125
M0	0.335	0.933	-0.001	0.308	1.106	0.008	0.180	2.729	0.044

Table 2: Performances of the three EIIE (Ensemble of Identical Independent Evaluators) neural networks, an integrated network, and some traditional portfolio selection strategies in three different back-test experiments (in UTC, detailed time-ranges listed in Table 1) on the cryptocurrency exchange Poloniex. The performance metrics are Maximum Drawdown (MDD), the final Accumulated Portfolio Value (fAPV) in the unit of initial portfolio amount (p_f/p_0), and the Sharpe ratio (SR). The bold algorithms are the EIIE networks introduced in this paper, named after the underlining structures of their IIEs. For example, bRNN is the EIIE of Figure 3 using basic RNN evaluators. Three benchmarks (italic), the integrated CNN (iCNN) previous proposed by the authors (Jiang and Liang, 2017), and some recently reviewed[a] strategies (Li et al., 2015a; Li and Hoi, 2014) are also tested. The algorithms in the table are divided into five categories, the model-free neural network, the benchmarks, follow-the-loser strategies, follow-the-winner strategies, and pattern-matching or other strategies. The best performance in each column is highlighted with boldface. All three EIIEs significantly outperform all other algorithms in the fAPV and SR columns, showing the profitability and reliability of the EIIE machine-learning solution to the portfolio management problem.

a. The exceptions are RMR of Huang et al. (2013) and WMAMR of Gao and Zhang (2013).

6.3 Results

The performances of all three EIIE policy networks proposed in the current paper will be compared to that of the integrated CNN (iCNN) (Jiang and Liang, 2017), several well-known or recently published model-based strategies, and three benchmarks.

The three benchmarks are the Best Stock, the asset with the most fAPV over the back-test interval, the Uniform Buy and Hold (UBAH), a portfolio management approach simply equally spreading the total fund into the preselected assets and holding them without making any purchases or selling until the end (Li and Hoi, 2014), and Uniform Constant Rebalanced Portfolios (UCRP) (Kelly, 1956; Cover, 1991).

Most of the strategies to be compared in this work were surveyed by Li and Hoi (2014), including Aniticor (Borodin et al., 2004), Online Moving Average Reversion (OLMAR) (Li et al., 2015b), Passive Aggressive Mean Reversion (PAMR) (Li et al., 2012), Confidence Weighted Mean Reversion (CWMR) (Li et al., 2013), Online Newton Step (ONS) (Agarwal et al., 2006), Universal Portfolios (UP) (Cover, 1991), Exponential Gradient (EG) (Helmbold et al., 1998), Nonparametric Kernel Based Log Optimal Strategy (B^K) (Györfi et al., 2006), Correlation-driven Nonparametric Learning Strategy (CORN) (Li et al., 2011), and M0 (Borodin et al., 2000), except Weighted Moving Average Mean Reversion (WMAMR) (Gao and Zhang, 2013) and Robust Median Reversion (RMR) (Huang et al., 2013).

Table 2 shows the performance scores fAPV, SR, and MDD of the EIIE policy networks as well as of the compared strategies for the three back-test intervals listed in Table 1. In term of fAPV or SR, the best performing algorithm in Back-Test 1 and 2 is the CNN EIIE whose final wealth is more than twice of the runner-up in the first experiment. Top three winners in these two measures in all back-tests are occupied by the three EIIE networks, losing only the MDD measure. This result demonstrates the powerful profitability and consistency of the current EIIE machine-learning framework.

When only considering fAPV, all three EIIEs outperform the best assets in all three back-tests, while the only model-based algorithm does that is RMR on the only occasion of Back-Test 3. Because of the high commission rate of 0.25% and the relatively high half-hourly trading frequency, many traditional strategies have bad performances. Especially in Back-Test 1, all model-based strategies have negative returns, with fAPV less than 1 or equivalently negative SRs. On the other hand, the EIIEs are able to achieve at least 4-fold returns in 20 days in different market conditions.

Figures 5, 6 and 7 plot the APV against time in the three back-tests respectively for the CNN and bRNN EIIE networks, two selected benchmarks and two model-based strategies. The benchmarks Best Stock and UCRP are two good representatives of the market. In all three experiments, both CNN and bRNN EIIEs beat the market throughout the entirety of the back-tests, while traditional strategies are only able to achieve that in the second half of Back-Test 3 and very briefly elsewhere.

7. Conclusion

This article proposed an extensible reinforcement-learning framework solving the general financial portfolio management problem. Being invented to cope with multi-channel market inputs and directly output portfolio weights as the market actions, the framework can be fit in with different deep neural networks, and is linearly scalable with the portfolio size. This scalability and extensibility are the results of the EIIE meta topology, which is able to accommodate many types of weight-sharing neural-net structures in the lower level. To take transaction cost into account when training the policy networks, the framework includes a portfolio-weight memory, the PVM, allowing the portfolio-management agent to
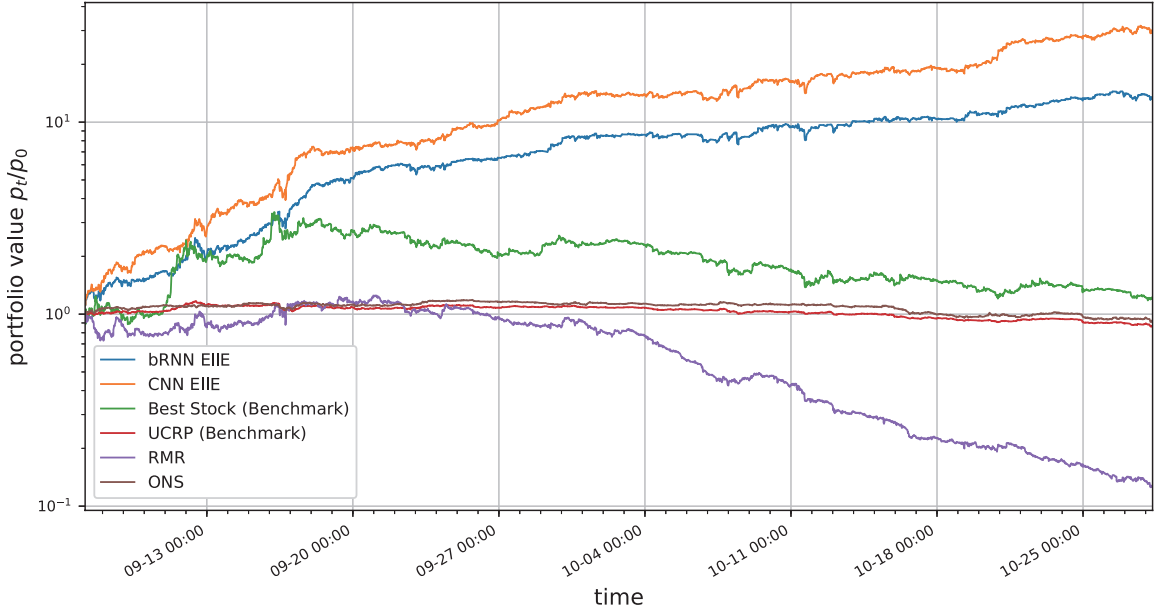
Figure 5: Back-Test 1: 2016-09-07-4:00 to 2016-10-28-8:00 (UTC). Accumulated portfolio values (APV, p_t/p_0) over the interval of Back-Test 1 for the CNN and basic RNN EIIEs, the Best Stock, the UCRP, RMR, and the ONS are plotted in log-10 scale here. The two EIIEs are leading throughout the entire time-span, growing consistently only with a few drawdown incidents.

learn restraining from oversized adjustments between consecutive actions, while avoiding the gradient vanishing problem faced by many recurrent networks. The PVM also allow parallel training within batching, beating recurrent approaches in learning efficiency to the transaction cost problem. Moreover, the OSBL scheme governs the online learning process, so that the agent can continuously digest constant incoming market information while trading. Finally, the agent was trained using a fully exploiting deterministic policy gradient method, aiming to maximize the accumulated wealth as the reinforcement reward function.

The profitability of the framework surpasses all surveyed traditional portfolio-selection methods, as demonstrated in the paper by the outcomes of three back-test experiments over different periods in a cryptocurrency market. In these experiments, the framework was realized using three different underlining networks, a CNN, a basic RNN and a LSTM. All three versions better performed in final accumulated portfolio value than other trading algorithms in comparison. The EIIE networks also monopolized the top three positions in the risk-adjusted score in all three tests, indicating the consistency of the framework in its performances. Another deep reinforcement learning solution, previously introduced by the authors, was assessed and compared as well under the same settings, losing too to the EIIE networks, proving that the EIIE framework is a major improvement over its more primitive cousin.
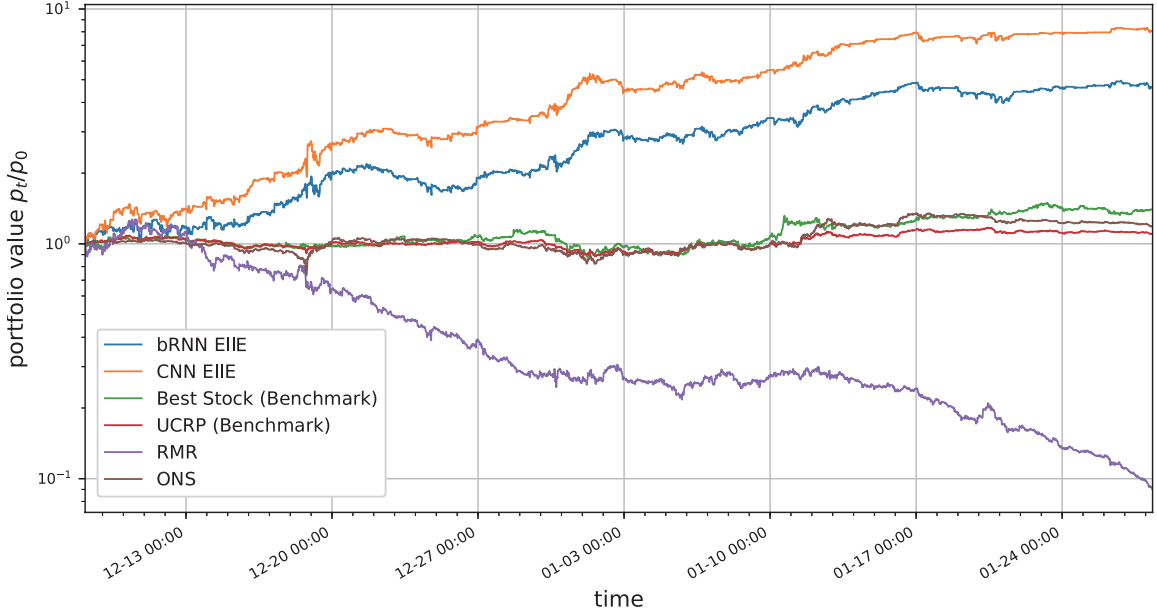
Figure 6: Back-Test 2: 2016-12-08-4:00 to 2017-01-28-8:00 (UTC), log-scale accumulated weath. This is the worst experiment amung the three back-tests for the EIIEs. However, they are able to steadily climb up till the end of the test.

Among the three EIIE networks, LSTM had much lower scores than the CNN and the basic RNN. The significant gap in performance between the two RNN species under the same framework might be an indicator to the well-known secret in financial markets, that history repeats itself. Not being designed to forget its input history, a vanilla RNN is more able than a LSTM to exploit repetitive patterns in price movement for higher yields. The gap might also be due to lack of fine-tuning in hyper-parameters for the LSTM. In the experiments, same set of structural hyper-parameters were used for both basic RNN and LSTM.

Despite the success of the EIIE framework in the back-tests, there is room for improvement in future works. The main weakness of the current work is the assumptions of zero market impact and zero slippage. In order to consider market impact and slippage, large amount of well-documented real-world trading examples will be needed as training data. Some protocol will have to be invented for documenting trade actions and market reactions. If that is accomplished, live trading experiments of the auto-trading agent in its current version can be recorded, for its future version to learn the principles behind market impacts and slippages from this recorded history. Another shortcoming of the work is that the framework has only been tested in one market. To test its adaptability, the current and later versions will need to be examined in back-tests and live trading in a more traditional financial market. In addition, the current award function will have to be amended, if not abandoned, for the reinforcement-learning agent to include awareness of longer-term market reactions. This may be achieved by a critic network. However, the backbone of the
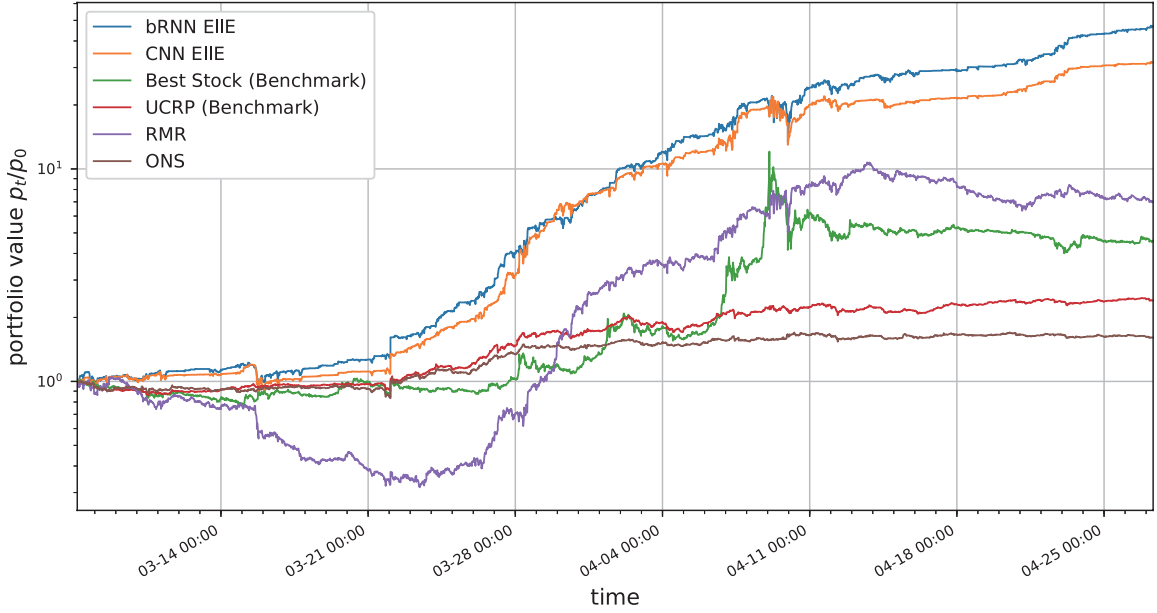
Figure 7: Back Test 3: 2017-03-07-4:00 to 2017-04-27-8:00 (UTC), log-scale accumulated weath. All algorithms struggle and consolidate at the beginning of this experiment, and both of the EIIEs experience two major dips on March 15 and April 9. This diving contributes to their high Maximum Drawdown in the text (Table 2). Nevertheless, this is the best month for both EIIEs in term of final wealth.

current framework, including the EIIE meta topology, the PVM, and the OSBL scheme, will continue to take important roles in future versions.

Appendix A. Proof of Theorem 1

In order to prove Theorem 1, it is handy to have the following five lemmas.

Lemma A.1 *The function $f(\mu)$ in Theorem 1 is monotonically increasing. In other words, $f(\mu_2) \geqslant f(\mu_1)$ if $\mu_2 > \mu_1$.*

Proof Recall that from Section 2.3,

$$f(\mu) := \frac{1}{1 - c_{\mathrm{p}} w_{t,0}} \left[1 - c_{\mathrm{p}} w'_{t,0} - (c_{\mathrm{s}} + c_{\mathrm{p}} - c_{\mathrm{s}} c_{\mathrm{p}}) \sum_{i=1}^{m} (w'_{t,i} - \mu w_{t,i})^{+} \right],$$

The fact that the linear rectifier $(x)^{+}$ $\big((x)^{+} = x$ if $x > 0$, $(x)^{+} = 0$ otherwise$\big)$ is monotonically increasing readily implies that $f(\mu)$ is also monotonically increasing. ∎

Lemma A.2
$$f(0) > 0.$$

Proof Using the fact that $w_{t,0}, w'_{t,0} \in [0,1]$,

$$f(0) = \frac{1}{1 - c_{\mathrm{p}} w_{t,0}} \left[1 - c_{\mathrm{p}} w'_{t,0} - (c_{\mathrm{s}} + c_{\mathrm{p}} - c_{\mathrm{s}} c_{\mathrm{p}}) \sum_{i=1}^{m} (w'_{t,i})^{+} \right]$$

$$= \frac{1}{1 - c_{\mathrm{p}} w_{t,0}} \left[1 - c_{\mathrm{p}} w'_{t,0} - (c_{\mathrm{s}} + c_{\mathrm{p}} - c_{\mathrm{s}} c_{\mathrm{p}})(1 - w'_{t,0}) \right]$$

$$\geqslant 1 - 2c_{\mathrm{p}} - c_{\mathrm{s}} + c_{\mathrm{s}} c_{\mathrm{p}} > 0,$$

for $c_{\mathrm{p}}, c_{\mathrm{s}} < 38\%$. For a commission rate, 38% is impractically high. Therefore, $f(0) > 0$ always holds. ∎

Lemma A.3

$$f(1) \leqslant 1.$$

Proof The proof is split into two cases. The fact that $c_{\mathrm{p}}, c_{\mathrm{p}} \in [0,1)$ implies $(c_{\mathrm{s}} + c_{\mathrm{p}} - c_{\mathrm{s}} c_{\mathrm{p}}) \geqslant 0$ will be used in Case 1.

Case 1: $w'_{t,0} \geqslant w_{t,0}$. Since $1 - c_{\mathrm{p}} w_{t,0} > 0$,

$$f(1) = \frac{1 - c_{\mathrm{p}} w'_{t,0}}{1 - c_{\mathrm{p}} w_{t,0}} - \frac{1}{1 - c_{\mathrm{p}} w_{t,0}} (c_{\mathrm{s}} + c_{\mathrm{p}} - c_{\mathrm{s}} c_{\mathrm{p}}) \sum_{i=1}^{m} (w'_{t,i} - w_{t,i})^{+}$$

$$\leqslant 1 - \frac{1}{1 - c_{\mathrm{p}} w_{t,0}} (c_{\mathrm{s}} + c_{\mathrm{p}} - c_{\mathrm{s}} c_{\mathrm{p}}) \sum_{i=1}^{m} (w'_{t,i} - w_{t,i})^{+}$$

$$\leqslant 1$$

Case 2: $w'_{t,0} < w_{t,0}$. This will be proved by contradiction. By assuming $f(1) > 1$,

$$1 - c_{\mathrm{p}} w'_{t,0} - (c_{\mathrm{s}} + c_{\mathrm{p}} - c_{\mathrm{s}} c_{\mathrm{p}}) \sum_{i=1}^{m} (w'_{t,i} - w_{t,i})^{+} > 1 - c_{\mathrm{p}} w_{t,0}.$$

Bringing the two w_0's together,

$$c_{\mathrm{p}}(w_{t,0} - w'_{t,0}) > (c_{\mathrm{s}} + c_{\mathrm{p}} - c_{\mathrm{s}} c_{\mathrm{p}}) \sum_{i=1}^{m} (w'_{t,i} - w_{t,i})^{+}. \tag{A.1}$$

Noting that $w'_{t,0} + \sum_{i=1}^{m} w'_{t,i} = 1 = w_{t,0} + \sum_{i=1}^{m} w_{t,i}$,

$$w_{t,0} - w'_{t,0} = 1 - \sum_{i=1}^{m} w_{t,i} - \left(1 - \sum_{i=1}^{m} w'_{t,i} \right) = \sum_{i=1}^{m} \left(w'_{t,i} - w_{t,i} \right).$$

Using identity $(a - b)^{+} - (b - a)^{+} = a - b$, (A.1) becomes

$$c_{\mathrm{p}} \left[\sum_{i=1}^{m} (w'_{t,i} - w_{t,i})^{+} - \sum_{i=1}^{m} (w_{t,i} - w'_{t,i})^{+} \right] > (c_{\mathrm{s}} + c_{\mathrm{p}} - c_{\mathrm{s}} c_{\mathrm{p}}) \sum_{i=1}^{m} (w'_{t,i} - w_{t,i})^{+}.$$

Moving the $(w'_{t,i} - w_{t,i})^+$ terms to the right-hand side,

$$-c_{\mathrm{p}} \sum_{i=1}^{m} (w_{t,i} - w'_{t,i})^+ > c_{\mathrm{s}} (1 - c_{\mathrm{p}}) \sum_{i=1}^{m} (w'_{t,i} - w_{t,i})^+. \tag{A.2}$$

The left-hand side of (A.2) is a non-positive number, and the right-hand side is a non-negative number. The former is greater than the latter, arriving at a contradiction.

Therefore, $f(1) \leqslant 1$ in both cases. ∎

Lemma A.4 *the sequence $\left\{ \tilde{\mu}_t^{(k)} \right\}$, defined as*

$$\left\{ \tilde{\mu}_t^{(k)} \,\middle|\, \tilde{\mu}_t^{(0)} = 0 \text{ and } \tilde{\mu}_t^{(k)} = f\left(\tilde{\mu}_t^{(k-1)} \right), \ k \in \mathbb{N}_0 \right\}$$

converges to μ_t.

Proof This is a special case of the final goal Theorem 1 when $\mu_\odot = 0$. This convergence is proved by the Monotone Convergence Theorem (MCT) (Rudin, 1976, Chapter 5). The monotonicity of f by Lemma A.1 with Mathematical Induction establishes an upper bound for $\left\{ \tilde{\mu}_t^{(k)} \right\}$.

$$\left. \begin{array}{c} \tilde{\mu}_t^{(0)} = 0 < \mu_t, \\ \text{If } \tilde{\mu}_t^{(k-1)} \leqslant \mu_t, \ \tilde{\mu}_t^{(k)} = f\left(\tilde{\mu}_t^{(k-1)} \right) \leqslant f(\mu_t) = \mu_t \end{array} \right\} \implies \tilde{\mu}_t^{(k)} \leqslant \mu_t, \ \forall k.$$

Note that by definition μ_t is the transaction remainder factor, and $0 < \mu_t \leqslant 1$. The monotonicity of sequence $\left\{ \tilde{\mu}_t^{(k)} \right\}$ itself can be also proved by Mathematical Induction and Lemma A.2.

$$\left. \begin{array}{c} \tilde{\mu}_t^{(1)} = f(0) > 0 = \tilde{\mu}_t^{(0)}, \\ \text{If } \tilde{\mu}_t^{(k-1)} \geqslant \tilde{\mu}_t^{(k-2)}, \ \tilde{\mu}_t^{(k)} = f\left(\tilde{\mu}_t^{(k-1)} \right) \geqslant f\left(\tilde{\mu}_t^{(k-2)} \right) = \tilde{\mu}_t^{(k-1)} \end{array} \right\} \implies \tilde{\mu}_t^{(k)} \geqslant \tilde{\mu}_t^{(k-1)}, \ \forall k.$$

If $\tilde{\mu}_t^{(k)} = \tilde{\mu}_t^{(k-1)}$, then $\tilde{\mu}_t^{(k)}$ is the solution of Equation (14), and the proof ends here. Otherwise, the sequence $\left\{ \tilde{\mu}_t^{(k)} \right\}$ is strictly increasing and bounded above by μ_t. In that case, by MCT, $\lim_{k \to \infty} \tilde{\mu}_t^{(k)} = \mu^*$, where μ^* is the Least Upper Bound of $\left\{ \tilde{\mu}_t^{(k)} \right\}$. As a result, $0 = \lim_{k \to \infty} \left(\tilde{\mu}_t^{(k+1)} - \tilde{\mu}_t^{(k)} \right) = \lim_{k \to \infty} \left(f\left(\tilde{\mu}_t^{(k)} \right) - \tilde{\mu}_t^{(k-1)} \right) = f(\mu^*) - \mu^*$. Therefore, μ^* is the solution to Equation (14), and hence $\lim_{k \to \infty} \tilde{\mu}_t^{(k)} = \mu_t$. ∎

Lemma A.5 *the sequence $\left\{ \tilde{\mu}_t^{(k)} \right\}$, defined as*

$$\left\{ \tilde{\mu}_t^{(k)} \,\middle|\, \tilde{\mu}_t^{(0)} = 1 \text{ and } \tilde{\mu}_t^{(k)} = f\left(\tilde{\mu}_t^{(k-1)} \right), \ k \in \mathbb{N}_0 \right\}$$

converges to μ_t.

Proof The proof is similar to that of Lemma A.4 using Mathematical Induction and MCT. The sequence is monotonically decreasing and bounded below by μ_t. The monotonically is a result of Lemma A.3, and the boundedness is by Lemma A.1. ∎

With the previous two Lemmas, it is in a good position to prove the general convergence theorem. Recall Theorem 1 from Section 2.3:

Theorem 1 *Denoting*

$$f(\mu) := \frac{1}{1 - c_{\mathrm{p}} w_{t,0}} \left[1 - c_{\mathrm{p}} w_{t,0}' - (c_{\mathrm{s}} + c_{\mathrm{p}} - c_{\mathrm{s}} c_{\mathrm{p}}) \sum_{i=1}^{m} (w_{t,i}' - \mu w_{t,i})^+ \right],$$

the sequence $\left\{ \tilde{\mu}_t^{(k)} \right\}$, *defined as*

$$\left\{ \tilde{\mu}_t^{(k)} \left| \tilde{\mu}_t^{(0)} = \mu_\odot \text{ and } \tilde{\mu}_t^{(k)} = f\left(\tilde{\mu}_t^{(k-1)} \right), \; k \in \mathbb{N}_0 \right. \right\} \tag{15}$$

converges to μ_t, *the solution to Equation* (14), *for any* $\mu_\odot \in [0,1]$.

Proof It is proved in three cases:

Case 1: $\mu_\odot = \mu_t$, 0, or 1. This case is trivial, as when $\mu_\odot = \mu_t$ it is the solution of $\mu = f(\mu)$, and the sequence will obviously converge. The convergence to μ_t for the other two values of $\mu_\odot$ is guaranteed by Lemma A.4 and A.5

Case 2: $0 < \mu_\odot < \mu_t$. Constructing a sequence $\left\{ \widehat{\mu}_t^{(k)} \right\}$ using $\mu_\odot = 0$,

$$\left\{ \widehat{\mu}_t^{(k)} \left| \widehat{\mu}_t^{(0)} = 0 \text{ and } \widehat{\mu}_t^{(k)} = f\left(\widehat{\mu}_t^{(k-1)} \right), \; k \in \mathbb{N}_0 \right. \right\}.$$

By the proof of Lemma A.4, $\left\{ \widehat{\mu}_t^{(k)} \right\}$ is strictly increasing and bounded above by μ_t, so there is a $j \in \mathbb{N}_0$ such that

$$\widehat{\mu}^{(j)} \leqslant \mu_\odot \leqslant \widehat{\mu}^{(j+1)}.$$

If any of the above equality holds, the two sequences coincide from $j + 1$ onward, converging to μ_t, and the proof ends here. Otherwise,

$$\widehat{\mu}^{(j)} < \mu_\odot < \widehat{\mu}^{(j+1)}.$$

Using the monotonicity of $f(\mu)$ by Lemma A.1, these inequalities become

$$\widehat{\mu}^{(j+1)} = f(\widehat{\mu}^{(j)}) \leqslant \tilde{\mu}^{(1)} = \mu_\odot \leqslant f(\widehat{\mu}^{(j+1)}) = \widehat{\mu}^{(j+2)}.$$

Again, if one of the equality holds, the proof ends here. This chain can go on indefinitely if no equality holds,

$$\widehat{\mu}^{(j+k+1)} < \tilde{\mu}^{(k)} < \widehat{\mu}^{(j+k+2)}.$$

By the Squeeze Theorem (Leithold, 1996),

$$\lim_{k \to \infty} \tilde{\mu}^{(k)} = \lim_{k \to \infty} \widehat{\mu}^{(k)} = \mu_t.$$

Case 3: $1 > \mu_\odot > \mu_t$. This case is proved in a similar way to Case 2, by constructing a sequence with $\mu_\odot = 1$ and making use of Lemma A.5.

In conclusion, for any initial value $\mu_\odot \in [0, 1]$, the sequence $\tilde{\mu}^{(k)}$ converges to μ_t. ∎

Appendix B. Hyper-Parameters

hyper-parameters	value	description
batch size	50	Size of mini-batch during training. (Section 3.1)
window size	50	Number of the columns (number of the trading periods) in each input price matrix. (Section 3.2)
number of assets	12	Total number of preselected assets (including the cash, Bitcoin). (Section 3.1)
trading period (second)	1800	Time interval between two portfolio redistributions. (Section 2.1)
total steps	2×10^6	Total number of steps for pre-training in the training set.
regularization coefficient	10^{-8}	The L2 regularization coefficient applied to network training.
learning rate	3×10^{-5}	Parameter α (i.e. the step size) of the Adam optimization (Kingma and Ba, 2014).
volume observation (day)	30	The length of time during which trading volumes are used to preselect the portfolio assets. (Section 3.1)
commission rate	0.25%	Rate of commission fee applied to each transaction. (Section 2.3)
rolling steps	30	Number of online training steps for each period during cross-validation and back-tests.
sample bias	5×10^{-5}	Parameter of geometric distribution when selecting online training sample batches. (The β in Equation 26 of Section 5.3)

Table B.1: Hyper-parameters of the reinforcement-learning framework. They are chosen based on the networks' scores in the cross-validation set described in Table 1 of Section 6.1. Although these are the values used in the experiments of the paper, they are all adjustable in the framework.

The hyper-parameters and their values used in the back-test experiments of the paper are listed in Table B.1. These numbers are selected to maximize the network scores in the cross-validation time-range (see Section 6.1). In order to avoid over-fitting, the cross-validation range and the back-tests do not overlap.

Different topologies of the IIEs were also tried in the cross-validation set, and it turned out that deeper network structures than those presented in Figure 2 and 3 did not improve scores on the set.

Bibliography

Amit Agarwal, Elad Hazan, Satyen Kale, and Robert E Schapire. Algorithms for portfolio management based on the newton method. In *Proceedings of the 23rd international conference on Machine learning*, pages 9–16. ACM, 2006.

Iddo Bentov, Charles Lee, Alex Mizrahi, and Meni Rosenfeld. Proof of activity: Extending bitcoin's proof of work via proof of stake [extended abstract] y. *ACM SIGMETRICS Performance Evaluation Review*, 42(3):34–37, 2014.

Joseph Bonneau, Andrew Miller, Jeremy Clark, Arvind Narayanan, Joshua A Kroll, and Edward W Felten. Sok: Research perspectives and challenges for bitcoin and cryptocurrencies. In *2015 IEEE Symposium on Security and Privacy*, pages 104–121. IEEE, 2015.

Allan Borodin, Ran El-Yaniv, and Vincent Gogan. On the competitive theory and practice of portfolio selection. In *Latin American Symposium on Theoretical Informatics*, pages 173–196. Springer, 2000.

Allan Borodin, Ran El-Yaniv, and Vincent Gogan. Can we learn to beat the best stock. *J. Artif. Intell. Res.(JAIR)*, 21:579–594, 2004.

D Charles, II Kirkpatrick, and Julie R Dahlquist. Technical analysis: The complete resource for financial market technician. *ISBN-13*, pages 978–0137059447, 2006.

Thomas M Cover. Universal portfolios. *Mathematical finance*, 1(1):1–29, 1991.

Thomas M Cover. Universal data compression and portfolio selection. In *Foundations of Computer Science, 1996. Proceedings., 37th Annual Symposium on*, pages 534–538. IEEE, 1996.

James Cumming. An investigation into the use of reinforcement learning techniques within the algorithmic trading domain. Master's thesis, Imperial College London, United Kiongdoms, 2015. URL `http://www.doc.ic.ac.uk/teaching/distinguished-projects/2015/j.cumming.pdf`.

Puja Das and Arindam Banerjee. Meta optimization and its application to portfolio selection. *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '11*, page 1163, 2011. doi: 10.1145/2020408.2020588. URL `http://dl.acm.org/citation.cfm?doid=2020408.2020588`.

M.A.H. Dempster and V. Leemans. An automated fx trading system using adaptive reinforcement learning. *Expert Systems with Applications*, 30(3):543 – 552, 2006. ISSN 0957-4174. doi: http://dx.doi.org/10.1016/j.eswa.2005.10.012. URL `http://www.sciencedirect.com/science/article/pii/S0957417405003015`. Intelligent Information Systems for Financial Engineering.

Yue Deng, Feng Bao, Youyong Kong, Zhiquan Ren, and Qionghai Dai. Deep direct reinforcement learning for financial signal representation and trading. *IEEE transactions on neural networks and learning systems*, 28(3):653–664, 2017.

Evan Duffield and Kyle Hagan. Darkcoin: Peer to peer cryptocurrency with anonymous blockchain transactions and an improved proofofwork system. *bitpaper.info*, 2014.

Fabio D Freitas, Alberto F De Souza, and Ailson R de Almeida. Prediction-based portfolio optimization model using neural networks. *Neurocomputing*, 72(10):2155–2170, 2009.

Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4): 193–202, 1980.

Li Gao and Weiguo Zhang. Weighted moving average passive aggressive algorithm for online portfolio selection. In *Intelligent Human-Machine Systems and Cybernetics (IHMSC), 2013 5th International Conference on*, volume 1, pages 327–330. IEEE, 2013.

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

László Györfi, Gábor Lugosi, and Frederic Udina. Nonparametric kernel-based sequential investment strategies. *Mathematical Finance*, 16(2):337–357, 2006.

Robert A Haugen. *Modern investment theory*. Prentice Hall, 1986.

J. B. Heaton, N. G. Polson, and Jan Hendrik Witte. Deep learning for finance: deep portfolios. *Applied Stochastic Models in Business and Industry*, 2016. ISSN 1526-4025. doi: 10.1002/ASMB.2209. URL http://www.ssrn.com/abstract=2838013.

David P Helmbold, Robert E Schapire, Yoram Singer, and Manfred K Warmuth. On-line portfolio selection using multiplicative updates. *Mathematical Finance*, 8(4):325–347, 1998.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Charles C Holt. Forecasting seasonals and trends by exponentially weighted moving averages. *International journal of forecasting*, 20(1):5–10, 2004.

Dingjiang Huang, Junlong Zhou, Bin Li, Steven CH Hoi, and Shuigeng Zhou. Robust median reversion strategy for on-line portfolio selection. In *IJCAI*, pages 2006–2012, 2013.

Zhengyao Jiang and Jinjun Liang. Cryptocurrency portfolio management with deep reinforcement learning. In *Proceedings of 2017 Intelligent Systems Conference*. SAI Conferences, 2017. Preprint: arXiv:1612.01277 [cs.LG].

J. L. Kelly. A new interpretation of information rate. *The Bell System Technical Journal*, 35(4):917–926, July 1956. ISSN 0005-8580. doi: 10.1002/j.1538-7305.1956.tb03809.x.

Diederik Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations*, pages 1–13, dec 2014. URL http://arxiv.org/abs/1412.6980.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

Louis Leithold. *The calculus 7*. HarperCollins College Publishing, 1996.

B Li, D Sahoo, and SCH Hoi. Olps: A toolbox for online portfolio selection. *Journal of Machine Learning Research (JMLR)*, 2015a.

Bin Li and Steven CH Hoi. Online portfolio selection: A survey. *ACM Computing Surveys (CSUR)*, 46(3):35, 2014.

Bin Li, Steven CH Hoi, and Vivekanand Gopalkrishnan. Corn: Correlation-driven non-parametric learning approach for portfolio selection. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):21, 2011.

Bin Li, Peilin Zhao, Steven C. H. Hoi, and Vivekanand Gopalkrishnan. PAMR: Passive aggressive mean reversion strategy for portfolio selection. *Machine Learning*, 87(2):221–258, may 2012. ISSN 08856125. doi: 10.1007/s10994-012-5281-z. URL http://link.springer.com/10.1007/s10994-012-5281-z.

Bin Li, Steven CH Hoi, Peilin Zhao, and Vivekanand Gopalkrishnan. Confidence weighted mean reversion strategy for online portfolio selection. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 7(1):4, 2013.

Bin Li, Steven CH Hoi, Doyen Sahoo, and Zhi-Yong Liu. Moving average reversion strategy for on-line portfolio selection. *Artificial Intelligence*, 222:104–123, 2015b.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous Control with Deep Reinforcement Learning. *arXiv*, 2016. ISSN 1935-8237. doi: 10.1561/2200000006.

Andrew W Lo, Harry Mamaysky, and Jiang Wang. Foundations of technical analysis: Computational algorithms, statistical inference, and empirical implementation. *The journal of finance*, 55(4):1705–1770, 2000.

Malik Magdon-Ismail and Amir F Atiya. Maximum drawdown. *Risk Magazine*, 17(1): 99–102, 2004.

Harry M Markowitz. *Portfolio selection: efficient diversification of investments*, volume 16. Yale university press, 1968.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518 (7540):529–533, feb 2015. ISSN 0028-0836. doi: 10.1038/nature14236. URL http://www.nature.com/doifinder/10.1038/nature14236http://dx.doi.org/10.1038/nature14236.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pages 1928–1937, 2016.

J. Moody and M. Saffell. Learning to trade via direct reinforcement. *IEEE Transactions on Neural Networks*, 12(4):875–889, Jul 2001. ISSN 1045-9227. doi: 10.1109/72.935097.

John Moody, Lizhong Wu, Yuansong Liao, and Matthew Saffell. Performance functions and reinforcement learning for trading systems and portfolios. *Journal of Forecasting*, 17(56): 441–470, 1998.

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

Seyed Taghi Akhavan Niaki and Saeid Hoseinzade. Forecasting S&P 500 index using artificial neural networks and design of experiments. *Journal of Industrial Engineering International*, 9(1):1, 2013. ISSN 2251-712X. doi: 10.1186/2251-712X-9-1. URL `http://www.jiei-tsb.com/content/9/1/1`.

Mihály Ormos and András Urbán. Performance analysis of log-optimal portfolio strategies with transaction costs. *Quantitative Finance*, 13(10):1587–1597, 2013.

L Christopher G Rogers and Stephen E Satchell. Estimating variance from high, low and closing prices. *The Annals of Applied Probability*, pages 504–512, 1991.

Walter Rudin. *Principles of mathematical analysis*. McGraw-Hill New York, 3 edition, 1976. ISBN 9780070856134.

Pierre Sermanet, Soumith Chintala, and Yann LeCun. Convolutional neural networks applied to house numbers digit classification. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 3288–3291. IEEE, 2012.

William F Sharpe. Capital asset prices: A theory of market equilibrium under conditions of risk. *The journal of finance*, 19(3):425–442, 1964.

William F Sharpe. The sharpe ratio. *The journal of portfolio management*, 21(1):49–58, 1994.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic Policy Gradient Algorithms. *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 387–395, 2014.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

Volodya Vovk and Chris Watkins. Universal portfolio selection. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 12–23. ACM, 1998.

Paul J Werbos. Generalization of backpropagation with application to a recurrent gas market model. *Neural networks*, 1(4):339–356, 1988.