Xi'an Jiaotong-Liverpool University

Final Year Project

Specification and Design Report

Deep Reinforcement Learning in On-Line Portfolio Selection

Dixing Xu (ID 1509014)

Department of Computer Science and Software Engineering

Supervised by

Dr. Fei Cheng

Department of Computer Science and Software Engineering

December 2, 2019

Contents

1 Specification

1.1 Project Description

On-line portfolio selection (OLPS) is a fundamental research problem in computational finance and an practical engineering task in financial engineering, where the goal is to achieve desired targets such as maximum cumulative return during a continuous period of time by continuously distributing the portfolio of various assets [1]. Many machine learning techniques have been applied to predict the financial price movements such as reinforcement learning [2], [3], decision trees [4], neural networks (NN) [5], support vector machines (SVM) [6], genetic algorithms [7]. However, they mainly concentrate on the forecasting of future trend/price movements and trade a single asset via signals [8]. On the other hand, this work considers multiple assets to trade and model the problem as an optimization of asset allocation with the aim of maximizing the final Accumulated Portfolio Return (APV).

Our previous works [9], [10] have established the basic framework for OLPS with deep reinforcement learning, more specifically deterministic gradient policy (DPG). In this work, we aim to propose and implement an improved deep reinforcement learning framework for OLPS. To summarize, major contributions of this project will include:

1. Broaden the scope of the previous work to more financial markets including foreign exchange (forex), Exchange-traded Fund (ETF) and stock markets.
2. Add short and leverage positions to the framework.
3. Reduce transaction cost by optimizing bid-ask spread.
4. Explore other value functions such as Deep Q-Network (DQN) and Deep Deterministic Policy Gradient (DDPG), policy, reward, model and planning.
5. Integrate sentimental analysis such as online tweets to the framework.

1.2 Statement of Deliverable

We have released our previous works on Github and they are freely available on-line [1]. After that, part of the framwork has also been released as an Python package **mercurius**[2] with additional features such as basic User Interface (UI) and plotting functionalities. An example usage of UCRP (described at Section 1.3.1) is shown as following:

```
# Uniform Constant Rebalanced Portfolio

from mercurius.strategy import ucrp

up = ucrp()
# custom input data with transaction cost 2.5%
up.trade(input_data, tc=0.025)
# re: for returning the result
# plot: for plotting the result
result = up.finish(re=True, plot=True)
# A list of portfolio values will be printed out
print(result['portfolio'])
```

Listing 1: Example Usage of Mercurius

However, the current python module only supports traditional machine learning algorithms. Therefore, it is necessary for this work to add more functionalities, documentation and unit test to the framework. The anticipated documentation will include a user manual with detailed examples on how to use the module, a walk-through for all the functionalities such as plotting graphs and tables. Although this module mainly servers as an engine for algorithmic trading with state-of-the-art machine learning techniques, it also contains a minimum viable product (MVP) UI powered by Flask Web Development Framework [11]. The MVP UI monitors the backtest result and on-line trading. Since this project tries to apply Deep RL methods to different markets, multiple experiments will be conducted

[1] https://github.com/ZhengyaoJiang/PGPortfolio
[2] https://github.com/dexhunter/mercurius

and an automated result analysis part is vital for shorten the time span of tests. Besides, as an old proverb goes,

> "A picture is worth a thousand words."

when trading in real time, graph shows more information than mere numbers. It is also anticipated that our package can better pre-process multiple datasets into a high dimensional data for financial engineers which is more suitable in the context of complex neural networks. Another important feature is to automatically download financial market data from broker. Currently, only cryptocurrency and forex data are avaiable for downloading with *mercurius* package, we will try to integrate stock from NASDAQ and future contracts market information into the framework.

Previously, Jiang et al. [9] performed only on cryptocurrency market because the cryptocurrency market operates 24 hours per day nonstop and without holidays. The price movements of large volume cryptocurrencies are also volatile which provide space for portfolio managers to capture the profits [10]. While this work aims to transfer the deterministic gradient policy (DPG) to more tradition markets, including forex, ETF funds and future contracts. Most experiments will be carried out on forex dataset with selected coin pairs. Since the publication of previous works, there are over 40 citations and in this work, we will try to compare some of the follow-up works to our methods.

To evaluate the effectiveness of algorithm, multiple metrics will be used, including the Accumulated Portfolio Value (APV), Maximum Drawdown (MDD) and the Sharpe ratio (SR). The details of metrics is described at Section 2.2. In addition, the core deep RL algorithm will be compared against different machine learning methods as illustrated at Section 2.1.

1.3 Conduct of Project and Plan

1.3.1 Background and related works

Advancements of deep reinforcement learning (deep RL) have been applied to many fields, including modern computer games [12], ancient game of Go [13],

etc. and deep RL has always achieved superior performances than its human counterparts. Therefore, it is natural for us to consider applying deep RL to computational finance field as well. We consider OLPS as a challenging problem in finance and try to solve it with the state-of-the-art deep RL algorithms. There are many existing machine learning techniques in online portfolio selection [1], however, to our best knowledge, no one before us have worked on model-free deep RL method for this problem. Traditionally, asset management can be divided into two schools, Mean Variance Theory derived from financial studies [14] and Capital Growth Theory studied by information theorists [15], [16]. While Mean Variance Theory balances between mean (return) and variance (risk) for a single-period of an asset, Capital Growth Theory maximizes the expected growth rate of multiple period asset selection. This work, therefore, follows the latter theory since the scenarios of OLPS consists of multiple periods and assets naturally. There are four main divisions for algorithms with the aim of maximizing cumulative wealth:

- Follow-the-Winner (invest in price rising assets).
- Follow-the-Loser (invest in price dropping assets).
- Pattern-Matching Approaches (predict the trends).
- Meta-Learning Algorithms (algorithm of algorithms).

As the name indicates, "Follow-the-Winner" is set of algorithms that invest on the assets which have positive rates of return, including Universal Portfolios [17], Exponential Gradient [18], Online Newton Step [19] and Switching Portfolios [20]. Anti Correlation [21], Passive Aggressive Mean Reversion [22], Confidence Weighted Mean Reversion [23], Online Moving Average Reversion [24], Robust Median Reversion [25] are under the category "Follow-the-Loser", which believes the current losing assets will become winning again or recovering to averages of certain properties. "Pattern-Matching" predicts the future price movements based on historical financial market data points. Nonparametric Kernel-based Log-optimal Strategy [26], Nonparametric Nearest Neighbor Log-optimal Strategy [27] and Correlation-driven Nonparametric Learning Strategy [28] are some algorithms from this category that are selected for later experiments. For meta-learning, there are

Online Gradient Updates [29] and Online Newton Updates [29], where multiple algorithms are combined. In this work, most algorithms on OLPS toolbox [30] are compared with deep reinforcement learning algorithm as an evaluation method for the performance of designed algorithm.

1.3.2 Implementation

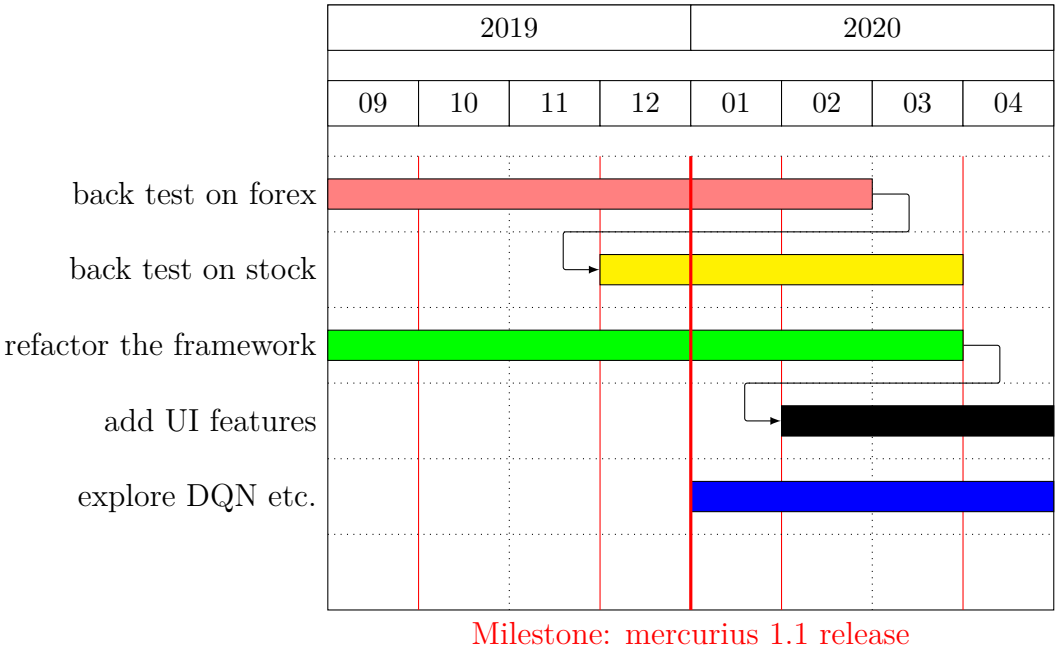
The implementation is mostly in Python. OLPS toolbox [30] is originally written in M scripts of MATLAB, while we have translated it to Python as a open-source Python module *mercurius*. A benchmark strategy UCRP is shown as below.

```
class ucrp(expert):
    """uniform constant rebalanced portfolio"""
    def __init__(self):
        super(ucrp, self).__init__()

    def get_b(self, data, last_b):
        n, m = data.shape
        return np.ones((m,1))/m
```

Listing 2: Example Strategy

1.3.3 Plan



We tested all the algorithms on a server with Ubuntu 18.04 Operating System and used 4 NVIDIA 2080TI GPU for training the neural networks. Python Version is 3.7.3, the virtual environment is managed by Anaconda [31].

1.3.4 Risk assessment

The major challenges for this project are following:

1. Reinforcement learning (RL) is hard to converge and thus requires many computational resources to finish the training [32].
2. Hyper-parameters are hard to tune. Combined with deep RL, even for one round of complete testing the time span will over one week [33].

From the past experience, an anticipated challenge will be the tuning of hyper-parameters such as the number of layers, neurons, window size. To address the challenges mentioned above, many new knowledge and corresponding modules are acquired. For example, cutting edge hyper-parameter optimization method such as Tree-structured Parzen Estimator Approach (TPE) is employed [33]. Besides, bet-

ter understanding of finance is also required for technical engineers. Although the most work are implementation, understanding of market is vital for this project.

2 Design

2.1 Design of system

2.1.1 Problem Setting

Online Portfolio Selection is the process that one distributes his/her wealth into m different financial market assets sequentially in a period of time t, aiming to achieve certain goals such as maximum return. The movements of an asset price usually includes open, high, low, close, volume (OHLCV) and closing price is mostly used in OLPS [1]. In this work, price movement at time t is defined in *price relative vector*, $\boldsymbol{X}_t$. For example, the ith element of $\boldsymbol{X}_t$, $\boldsymbol{X}_{i,t}$ is the closing price of the ith asset in the tth period. If the market is continuous, $\boldsymbol{X}_t$ is also the opening prices at Period $t+1$. Mathematically, the *price relative vector* is the ratio of tth closing price to previous $(t-1)$th closing price:

$$\boldsymbol{x}_t := \boldsymbol{x}_t \oslash \boldsymbol{x}_{t-1} = \left(\frac{x_{1,t}}{x_{1,t-1}}, \frac{x_{2,t}}{x_{2,t-1}}, ..., \frac{x_{m,t}}{x_{m,t-1}} \right)^\intercal. \tag{1}$$

The portfolio value P at the end of Period t without considering transaction cost is thus calculated as the product of market sequence X and portfolio weight vector $\boldsymbol{w}$,

$$P_t = \prod_{t-1}^{i=0} \boldsymbol{x}_{i+1} \cdot \boldsymbol{w}_i, \tag{2}$$

The sum of $\boldsymbol{w}_t$ is always 1 when leverage and short are not considered. $\forall t, \sum_i w_{t,i} = 1$. Then we can calculate the *rate of return*(RoR) as,

$$\rho_t := \frac{P_t}{P_{t-1}} = \boldsymbol{x}_t \cdot \boldsymbol{w}_{t-1} \tag{3}$$

The *gross logarithmic rate of return* is simply apply logarithmic function to RoR,

mathematically,

$$\log r_t := \ln \frac{P_t}{P_{t-1}} = \ln \boldsymbol{x}_t \cdot \boldsymbol{w}_{t-1}. \tag{4}$$

Therefore, the final portfolio value will be

$$P_{\mathrm{f}} = P_0 \exp\left(\sum_{t=1}^{t+1} r_t\right) = P_0 \prod_{t=1}^{t+1} \boldsymbol{x}_t \cdot \boldsymbol{w}_{t-1}, \tag{5}$$

where P_0 is the initial investment amount.

By maximizing rate of return of each period, the final accumulative return is expected to be maximized as well. The general OLPS framework is as following:

Algorithm 1: Online Portfolio Selection Framework Overview

Input : $\boldsymbol{x}_t^0$ Historic price data sequences

Output: p_{f} Final Portfolio Value

Initialize $\boldsymbol{w}_0 = (\frac{1}{m}, \ldots, \frac{1}{m})$

for $t = 1, 2, \ldots, n$ **do**

- Portfolio weight vector $\boldsymbol{w}_t$ is calculated by the algorithm.;
- Price data x_t is revealed.;
- Portfolio value vector at Period t is computed as Eq 2 indicates.;

end

There are three assumptions widely used in previous works [1], [10] and are also adopted in this work,

1. Transaction cost: it is assumed that no transaction cost/slippages incurred during online trading.
2. Market liquidity: it is assumed that market liquidity is high enough, that the agents are able to buy/sell assets according to their algorithms.
3. Market impact: it is assumed that the investments made by agents have zero influence on the price movement of the market.

2.1.2 Deep Reinforcement Learning Algorithm

This section presents a deep reinforcement learning algorithm to OLPS problem as Section 2.1.1 describes, specifically a deep deterministic policy gradient algorithm. At the context of reinforcement learning, an agent interacts with its environment. For OLPS, the agent is the software or trading robot and the corresponding environment is certain financial market [10]. The agent makes an action which is buying/selling one or more assets. Mathematically,

$$\boldsymbol{a}_t = \boldsymbol{w_t}. \tag{6}$$

where w_t is the *portfolio weight vector* at Period t. The action once made will influence the current situation. While in this work, $\boldsymbol{w}_{t-1}$ will include the influence as a part of policy making, the state therefore is represented as the pair of X_t and $\boldsymbol{w}_{t-1}$,

$$\boldsymbol{s}_t = (\boldsymbol{X}_t, \boldsymbol{w}_{t-1}), \tag{7}$$

where $\boldsymbol{w}_0$ is predetermined by the algorithm. A policy is a mapping from state to action, $\pi : \mathcal{S} \to \mathcal{A}$. With gradient ascent, the optimal policy is made from a set of parameters $\boldsymbol{\theta}$ and the action will be $\boldsymbol{a}_t = \pi_{\boldsymbol{\theta}}(\boldsymbol{s}_t)$. The optimal policy will maximize the reward function from time interval $[0, t_\mathrm{f}]$, and performance metrics J of $\pi_{\boldsymbol{\theta}}$ is the reward function from $t = 0$ to $t = \mathrm{f}$:

$$J(\pi_{\boldsymbol{\theta}}) = R\left(\boldsymbol{s}_1, \pi_{\boldsymbol{\theta}}(s_1), \cdots, \boldsymbol{s}_t, \pi_{\boldsymbol{\theta}}(s_t), \boldsymbol{s}_{t+1}\right). \tag{8}$$

Then the parameters are updated by gradient descent method, towards the optimal direction,

$$\boldsymbol{\theta} \longrightarrow \boldsymbol{\theta} + \lambda \nabla_{\boldsymbol{\theta}} J(\pi_{\boldsymbol{\theta}}). \tag{9}$$

where λ is the learning rate.

The reward function R is the average logarithmic cumulated return for n periods,

expressly,

$$R = \frac{1}{n} \sum_{n} \ln \boldsymbol{x}_t \cdot \boldsymbol{w}_{t-1} \tag{10}$$

$$= \frac{1}{n} \sum_{n} \log r_t \tag{11}$$

2.1.3 Neural Networks and Policies

Different neural networks are used to construct the policy $\pi_{\boldsymbol{\theta}}$, including Convolutional Neural Networks [34], Long-Short term Memory Networks [35] and Recurrent Networks [36]. Besides, several novel designs such as Identical Independent Evaluators (IIE), Ensemble of IIE (EIIE), Portfolio-Vector Memory (PVM), Online Stochastic Batch Learning (OSBL) are applied inside the network [10].

EIIE topology means the neural networks output directly to the *portfolio weight vector* $\boldsymbol{w}_t$ via softmax function. PVM saves the previous portfolio weights and takes them as input for next period of training. OSBL selects the mini-batch by a geometric distribution, simply the more recent price data will be selected more frequently.

2.1.4 Design of the Python Package

With the basic understanding of the algorithm, this section describes some functionalities of our python package *mercurius*. The example is performed on futures contract market with data available online[3]. There are 10 assets for selection and we select several traditional algorithms to test. The input price movements is shown as Figure 1.

The performances of different algorithms is compared at Figure 2 where we select Buy-and-Hold (BAH), Constant Rebalanced Portfolio (CRP) as benchmark. Robust Median Reversion (RMR) performs the best in terms of final APV. The numeric table is demonstrated at Table 1.

[3]project url: https://github.com/dexhunter/eco301

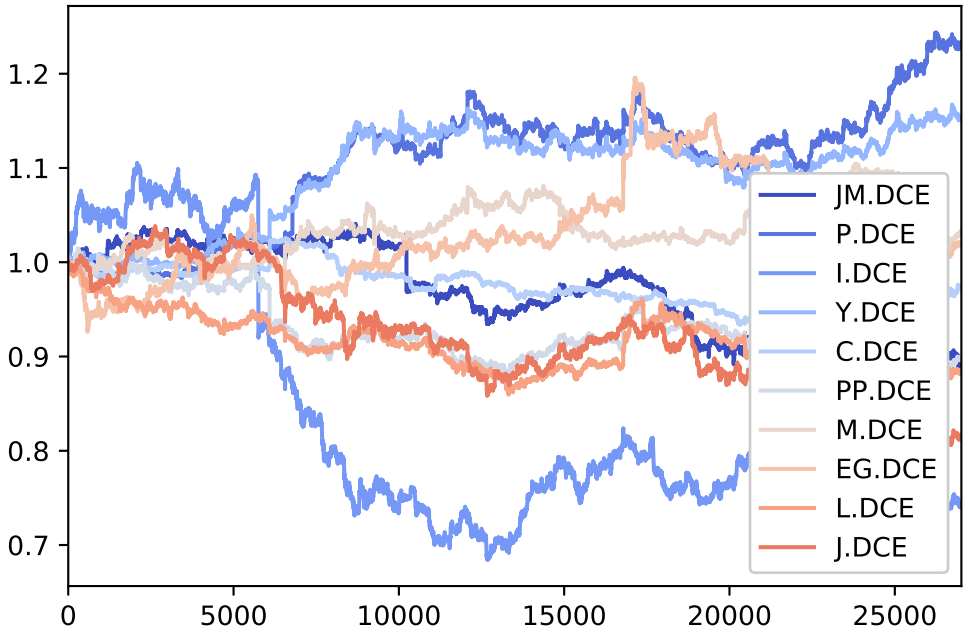


Figure 1: Input Sequence Data from Futures Contract Market, 10 assets are contracts for coking coal, palm oil, iron ore, soybean oil, corn, polypropylene, soybean meal, ethylene glycol, plastic and coke respectively

	profit	sharpe	APV	VaR(90%)
BAH	0.98876	-1.07573	-9.885370e+00	-0.00045
BCRP	1.03393	2.89564	7.971866e+01	-0.00094
CRP	0.98824	-1.14052	-1.063131e+01	-0.00046
CWMR	1.42910	4.71027	2.044639e+01	-0.00018
EG	0.98825	-1.13839	-1.060031e+01	-0.00046
OLMAR	1.54763	38.42997	3.673105e+06	-0.00114
ONS	0.98814	-1.18334	-1.485441e+01	-0.00064
PAMR	1.65176	44.91529	1.530392e+07	-0.00108
RMR	1.57637	39.98359	5.552418e+06	-0.00113

Table 1: Traditional Machine Learning Techniques on Futures Contract Market

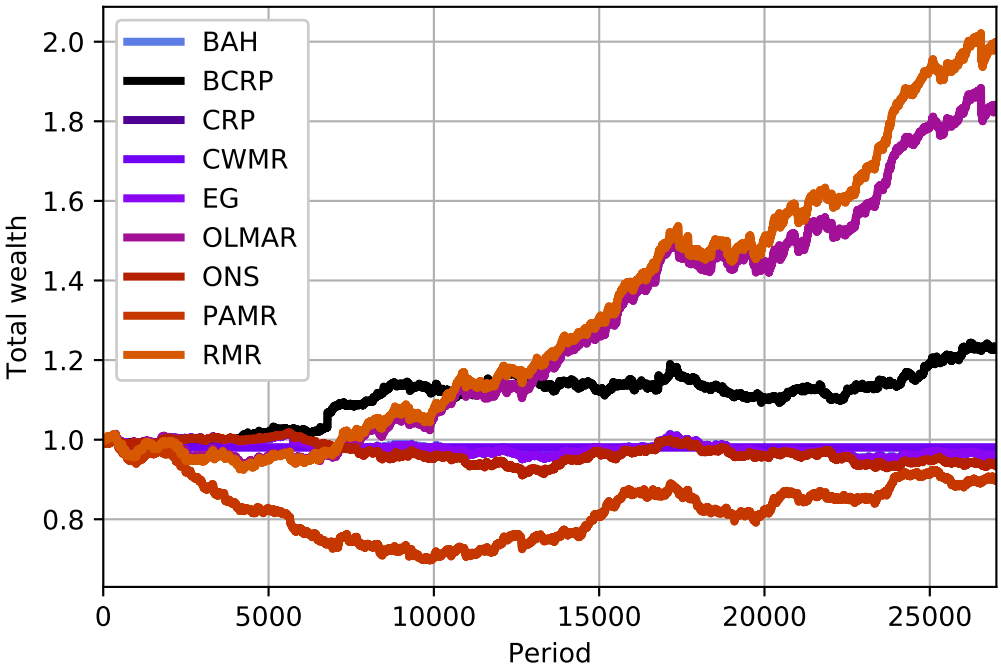


Figure 2: Comparison of traditional machine learning techniques on Futures Contract Market

2.2 Design of evaluation

All the algorithms mentioned at Section 1.3.1 will be backtested, in other words, historic data will be used to simulate a trading environment for algorithm to trade as if it sees the data points in real time. One caution there is to avoid look-ahead bias since the algorithm should receive the price data in a sequence. During backtesting, multiple metrics including accumulative portfolio value (APV), Sharpe ratio (SR), Maximum drawdown (MDD) are applied to measure the performance of different OLPS algorithms. To fairly compare portfolio values of different algorithms, APV is divided by the corresponding unit of the initial portfolio values:

$$p_t = p_t / p_0. \tag{12}$$

where p_t is the accumulative portfolio value at time t and p_0 is the initial portfolio value and will be set to 0 after this division. APV reflects the profitability of an algorithm, however, it lacks risk information for the algorithm. Sharpe ratio and maximum drawdown, therefore, are taken into consideration for risk measurements. SR [37], [38] considers the volatility of the portfolio values for both upwards and downwards movements, while MDD [39], [40] calculates the largest loss from a peak to a trough. Rigorously, SR is defined as the average of risk-free return by its deviation,

$$S = \frac{\overline{p}_t - p_f}{\sigma_p}, \tag{13}$$

where $\overline{p}_t$ is the expected portfolio return at time t, p_f is the risk free rate of return (0 in this case), and σ_p is the standard deviation of the portfolio value. For maximum drawdown, mathematically:

$$D = \max_{\substack{\tau > t \\ t}} \frac{p_t - p_\tau}{p_t}. \tag{14}$$

While in this work, more recent data from broker markets will be used. For example, forex data from October 2018 to October 2019 will be put into backtesting while training data will be from earliest available data points from broker[4].

[4]This work uses forex data provided by OANDA APIs

3 Review

The current progress have been demonstrated at Section 4. It is planned for *mercurius* to release 1.1 version around the end of December, the improvements will include update for algorithms, RL integration in the framework. The rest of work will be completed before May, 2020.

4 References

[1] B. Li and S. C. Hoi, "Online portfolio selection: A survey", *ACM Computing Surveys (CSUR)*, vol. 46, no. 3, p. 35, 2014.

[2] J. Moody, L. Wu, Y. Liao, and M. Saffell, "Performance functions and reinforcement learning for trading systems and portfolios", *Journal of Forecasting*, vol. 17, no. 56, pp. 441–470, 1998.

[3] J. Moody and M. Saffell, "Learning to trade via direct reinforcement", *IEEE Transactions on Neural Networks*, vol. 12, no. 4, pp. 875–889, Jul. 2001, ISSN: 1045-9227. DOI: 10.1109/72.935097.

[4] E. Tsang, P. Yung, and J. Li, "Eddie-automation, a decision support tool for financial forecasting", *Decision Support Systems*, vol. 37, no. 4, pp. 559–565, 2004.

[5] T. Kimoto, K. Asakawa, M. Yoda, and M. Takeoka, "Stock market prediction system with modular neural networks", in *1990 IJCNN international joint conference on neural networks*, IEEE, 1990, pp. 1–6.

[6] F. E. Tay and L. Cao, "Modified support vector machines in financial time series forecasting", *Neurocomputing*, vol. 48, no. 1-4, pp. 847–861, 2002.

[7] S. Mahfoud and G. Mani, "Financial forecasting using genetic algorithms", *Applied artificial intelligence*, vol. 10, no. 6, pp. 543–566, 1996.

[8] A. Borodin, R. El-Yaniv, and V. Gogan, "On the competitive theory and practice of portfolio selection", in *Latin American symposium on theoretical informatics*, Springer, 2000, pp. 173–196.

[9] Z. Jiang and J. Liang, "Cryptocurrency portfolio management with deep reinforcement learning", in *2017 Intelligent Systems Conference (IntelliSys)*, IEEE, 2017, pp. 905–913.

[10] Z. Jiang, D. Xu, and J. Liang, "A deep reinforcement learning framework for the financial portfolio management problem", *arXiv preprint arXiv:1706.10059*, 2017.

[11] M. Grinberg, *Flask Web Development: Developing Web Applications with Python*, 1st. O'Reilly Media, Inc., 2014, ISBN: 1449372627, 9781449372620.

[12] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning", *Nature*, vol. 518, no. 7540, p. 529, 2015.

[13] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, *et al.*, "Mastering the game of go without human knowledge", *Nature*, vol. 550, no. 7676, p. 354, 2017.

[14] H. Markowitz, "Portfolio selection", *The journal of finance*, vol. 7, no. 1, pp. 77–91, 1952.

[15] J. Kelly jr, "A new interpretation of information rate", *the bell system technical journal*, 1956.

[16] J. L. Kelly Jr, "A new interpretation of information rate", in *The Kelly Capital Growth Investment Criterion: Theory and Practice*, World Scientific, 2011, pp. 25–34.

[17] T. M. Cover, "Universal portfolios", *Mathematical finance*, vol. 1, no. 1, pp. 1–29, 1991.

[18] D. P. Helmbold, R. E. Schapire, Y. Singer, and M. K. Warmuth, "On-line portfolio selection using multiplicative updates", *Mathematical Finance*, vol. 8, no. 4, pp. 325–347, 1998.

[19] A. Agarwal, E. Hazan, S. Kale, and R. E. Schapire, "Algorithms for portfolio management based on the newton method", in *Proceedings of the 23rd international conference on Machine learning*, ACM, 2006, pp. 9–16.

[20] Y. Singer, "Switching portfolios", *International Journal of Neural Systems*, vol. 8, no. 04, pp. 445–455, 1997.

[21] A. Borodin, R. El-Yaniv, and V. Gogan, "Can we learn to beat the best stock.", *J. Artif. Intell. Res.(JAIR)*, vol. 21, pp. 579–594, 2004.

[22] B. Li, P. Zhao, S. C. H. Hoi, and V. Gopalkrishnan, "PAMR: Passive aggressive mean reversion strategy for portfolio selection", *Machine Learning*, vol. 87, no. 2, pp. 221–258, May 2012, ISSN: 08856125. DOI: 10.1007/s10994-012-5281-z. [Online]. Available: http://link.springer.com/10.1007/s10994-012-5281-z.

[23] B. Li, S. C. Hoi, P. Zhao, and V. Gopalkrishnan, "Confidence weighted mean reversion strategy for online portfolio selection", *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 7, no. 1, p. 4, 2013.

[24] B. Li, S. C. Hoi, D. Sahoo, and Z.-Y. Liu, "Moving average reversion strategy for on-line portfolio selection", *Artificial Intelligence*, vol. 222, pp. 104–123, 2015.

[25] D. Huang, J. Zhou, B. Li, S. C. Hoi, and S. Zhou, "Robust median reversion strategy for on-line portfolio selection.", in *IJCAI*, 2013, pp. 2006–2012.

[26] L. Györfi, G. Lugosi, and F. Udina, "Nonparametric kernel-based sequential investment strategies", *Mathematical Finance*, vol. 16, no. 2, pp. 337–357, 2006.

[27] L. Györfi, F. Udina, and H. Walk, "Nonparametric nearest neighbor based empirical portfolio selection strategies", *Statistics & Decisions International mathematical journal for stochastic methods and models*, vol. 26, no. 2, pp. 145–157, 2008.

[28] B. Li, S. C. Hoi, and V. Gopalkrishnan, "Corn: Correlation-driven nonparametric learning approach for portfolio selection", *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, p. 21, 2011.

[29] P. Das and A. Banerjee, "Meta optimization and its application to portfolio selection", in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 2011, pp. 1163–1171.

[30] B. Li, D. Sahoo, and S. C. Hoi, "Olps: A toolbox for on-line portfolio selection", *Journal of Machine Learning Research*, vol. 17, no. 35, pp. 1–5, 2016. [Online]. Available: http://jmlr.org/papers/v17/15-317.html.

[31] Anaconda, *Anaconda software distribution*, version 2-2.4.0, Nov. 1, 2016. [Online]. Available: https://anaconda.com.

[32] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. 2018.

[33] J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization", in *Advances in neural information processing systems*, 2011, pp. 2546–2554.

[34] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, *et al.*, "Gradient-based learning applied to document recognition", *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[35] S. Hochreiter and J. Schmidhuber, "Long short-term memory", *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[36] M. Jordan, "Attractor dynamics and parallelism in a connectionist sequential machine", in *Proc. of the Eighth Annual Conference of the Cognitive Science Society (Erlbaum, Hillsdale, NJ), 1986*, 1986.

[37] W. F. Sharpe, "Capital asset prices: A theory of market equilibrium under conditions of risk", *The journal of finance*, vol. 19, no. 3, pp. 425–442, 1964.

[38] ——, "The sharpe ratio", *The journal of portfolio management*, vol. 21, no. 1, pp. 49–58, 1994.

[39] O. J. Blanchard, "Speculative bubbles, crashes and rational expectations", *Economics letters*, vol. 3, no. 4, pp. 387–389, 1979.

[40] M. Magdon-Ismail, A. Atiya, A. Pratap, and Y. Abu-Mostafa, "The maximum drawdown of the brownian motion", in *Computational Intelligence for Financial Engineering, 2003. Proceedings. 2003 IEEE International Conference on*, IEEE, 2003, pp. 243–247.