Investor-Imitator: A Framework for Trading Knowledge Extraction

Yi Ding*
Nanjing University of Aeronautics and Astronautics
Nanjing, China
yiding.cn@outlook.com

Weiqing Liu
Microsoft Research
Beijing, China
Weiqing.Liu@microsoft.com

Jiang Bian
Microsoft Research
Beijing, China
Jiang.Bian@microsoft.com

Daoqiang Zhang
Nanjing University of Aeronautics and Astronautics
Nanjing, China
dqzhang@nuaa.edu.cn

Tie-Yan Liu
Microsoft Research
Beijing, China
tie-yan.liu@microsoft.com

ABSTRACT

Stock trading is a popular investment approach in real world. However, since lacking enough domain knowledge and experience, it is very difficult for common investors to analyze thousands of stocks manually. Algorithmic investment provides another rational way to formulate human knowledge as a trading agent. However, it still requires well-built knowledge and experience to design effective trading algorithms in such a volatile market. Fortunately, various kinds of historical trading records are easy to obtain in this big-data era, it is invaluable of us to extract the trading knowledge hidden in the data to help people make better decisions. In this paper, we propose a reinforcement learning driven Investor-Imitator framework to formalize the trading knowledge, by imitating an investor's behavior with a set of logic descriptors. In particular, to instantiate specific logic descriptors, we introduce the Rank-Invest model that can keep the diversity of logic descriptors by learning to optimize different evaluation metrics. In the experiment, we first simulate three types of investors, representing different degrees of information disclosure we may meet in real market. By learning towards these investors, we can tell the inherent trading logic of the target investor with the Investor-Imitator empirically, and the extracted interpretable knowledge can help us better understand and construct trading portfolios. Experimental results in this paper sufficiently demonstrate the designed purpose of Investor-Imitator, it makes the Investor-Imitator an applicable and meaningful intelligent trading framework in financial investment research.

CCS CONCEPTS

• **Computing methodologies** → *Machine learning*; *Reinforcement learning*; *Sequential decision making*;

KEYWORDS

Finance, Financial Investment, Data Mining, Reinforcement Learning

ACM Reference Format:
Yi Ding, Weiqing Liu, Jiang Bian, Daoqiang Zhang, and Tie-Yan Liu. 2018. Investor-Imitator: A Framework for Trading Knowledge Extraction. In *KDD '18: The 24th ACM SIGKDD International Conference on Knowledge Discovery Data Mining, August 19–23, 2018, London, United Kingdom.* ACM, New York, NY, USA, Article 4, 10 pages. https://doi.org/10.1145/3219819.3220113

1 INTRODUCTION

Compared with depositing money into banks, investing on stock market is considered as a potentially more profitable investment approach in the long run. A successful stock investor, in the meantime, usually spend huge efforts in collecting market information, analyzing company fundamentals, understanding the pattern of price changing, capturing relevant knowledge, etc. As a well-recognized weapon for maximizing long-term profit, successful investors are used to extensively reviewing the investment history and enhancing the future investment return in an iterative way. Therefore, by deeply diving into the trading history of a successful investor, we may understand the inherent logic and hidden patterns of his/her investment behaviors so as to construct portfolios following similar disciplines to gain profits. Moreover, if it is possible to simulate the investment behaviors of sufficient portion of investors on the market, we can foresee the future trend of the whole market, which will undoubtedly lead to tremendous profits.

Researchers have put efforts on formalizing the investment behaviors with data mining approaches. Some people are devoted in finding patterns which indicate how the market may develop [11]. Li and Kuo [27] invested a novel trajectory analysis with a modified Self-Organization Map (SOM) neural network [12, 22] and Discrete Wavelet Transform (DWS) method. By means of the proposed method, they concluded several interpretable patterns from the market, and incorporated the extracted knowledge with

*This work was done when the author was an intern at Microsoft Research Asia.

KDD '18, August 19–23, 2018, London, United Kingdom

ACM ISBN 978-1-4503-5552-0/18/08…$15.00
https://doi.org/10.1145/3219819.3220113

fixed investor behaviors to trade in the market. In addition to extracting market trend knowledge, there are also many previous works related to extracting static investor's trading preference with data analysis or data mining techniques [14, 21, 34], for example revealing the relationship between investors characteristics and stock characteristics. Although previous works can actually derive valuable trading knowledge, they overlooked the essential dynamics of an active investor's behaviors through the time [28]. For instance, during a bear market period, a majority of investors prefer to seek investment strategy which can succeed in avoiding the high variance of the asset value; on the other hand, in a bull market period, most of investors turn to strategies aiming at maximizing the absolute return. Obviously, the dynamic nature of investment behaviors raises a big challenge for analyzing inherent logic and mining hidden patterns of the investment process.

In order to model dynamic investment behaviors, a critical prerequisite is to identify stable measurements for the investment performance. Among previous research and practices in financial investment, people have designed many metrics [36] for assessing an investment strategy from different aspects, e.g. profit and risk. Meanwhile, these metrics have also been used as the objective to optimize investment strategies. Therefore, it enables us to discover inherent logic and hidden patterns behind a sequence of historical investment behaviors, by investigating their dynamic performance in terms of those metrics.

In this paper, we propose a reinforcement learning framework, called Investor-Imitator, to model the inherent logic of an investor's trading behaviors with a set of logic descriptors. Particularly, we introduce a Rank-Invest model to instantiate the logic descriptor, and the Rank-Invest model is obtained by optimizing different evaluation metrics and modeling the dynamism of metric preference through the time. Essentially, our Investor-Imitator framework is designed by jointly following two principle objectives:

- One objective is to mine knowledge in terms of behavior patterns of individual investors with respect to different market states. More importantly, such mined knowledge can shed more light on designing more sophisticated and profitable investment strategies. In the long run, these investment behavior knowledge can also enlighten us with more valuable criteria for designing healthy and efficient financial environment.
- Another objective is to simulate the certain individual investor's trading behaviors with respect to different market states, which empowers us to construct a series of portfolios following similar investing patterns behind this investor's trading behaviors. Inspired by this objective, it can open another path to high profit by simulating trading behaviors of legendary investors.

Driven by these vital objectives, our Investor-Imitator framework aims at modeling the hidden temporal-dynamic preference of an investor among various evaluation metrics. To this end, this Investor-Imitator framework can be generalized into three major parts:

• **Metric-specific investing models:** In this part, we first select a set of evaluation metrics to reflect various aspects of investment performance. Then, we can learn respective metric-specific models towards generating portfolios through optimizing each of selected metrics.

• **Measurement of consistency with the target investor:** Our framework requires an effective way to measure the consistency of a series of investing behaviors compared with the target investor. In real investing scenarios, we can leverage a couple of methods, such as portfolio similarity, correlation between revenue curves, coefficient in terms of various metrics, etc.

• **A learning approach for portfolio construction:** To model inherent logic of the target investor's trading behaviors, our framework applies a reinforcement learning approach to simulate its portfolio by a stochastic or deterministic selection of portfolios corresponding to each of selected metric-specific investing models, respectively. In this reinforcement learning approach, the selection of these investing models is viewed as action, to learn which we use previous actions and recent market status to define states and leverage consistency of a series of investing behaviors compared with the target investor as the reward.

To elaborate our Investor-Imitator framework, in this paper, we demonstrate three scenarios, differing in the information disclosure level of an accessible investor. The three target investors as described below:

• **Oracle Investor:** Imagine that there exists an investor who can foresee the future market. In this scenario, this oracle investor can precisely tell the future performance of any specific portfolio, in terms of a specific evaluation metric. By modeling this oracle perspective investor based on foretold future performance, we can easily generate a profitable portfolios which gains high scores on corresponding evaluation metric.

• **Collaborator Investor:** Individual investors are used to extensively reviewing their own investment history and iteratively enhancing the future investment return. Moreover, for those professional financial investment organizations, investors and researchers collaborate with each other to obtain higher profits for the organization. In this kind of scenario, the detailed historical portfolio series can be obtained for modeling the trading behavior of the target investor.

• **Public Investor:** In real investment market, there are many public investors such as Mutual Funds, which are comprised of a group of experts on financial investment. Hence, such public investors can be viewed as one typical and valuable class of the target investors. For common market participants, it is easier for them to access the profitability information of these public investors. By precisely modeling and simulating important public investors, the overall market trend can be predicted more easily, which enables us to obtain tremendous profit, since the public investors are the main part of the whole market[1].

We design several experiments to show the approximation ability of Investor-Imitator to the pre-set investor roles. Our experiment shows that by learning towards the Oracle Investor, the Investor-Imitator can learn to balance the profit/risk preference and generate more reliable performance. When the target of Investor-Imitator is his Collaborator, the Investor-Imitator can imitate the Collaborator's trading logic well. When given the Public Investor's trading performance history, the Investor-Imitator can also reproduce the

[1] https://en.wikipedia.org/wiki/Mutual_fund

Public Investor's performance in terms of the revenue curve. To conclude, when given more precise information, the Investor-Imitator can imitate better to the target investors.

The main contribution of this paper lies in that we switch the financial investment research from simply making more profit in a volatile stock market to abstracting trading knowledge by dynamically modelling investors' behaviors, because we believe the trading knowledge is a much more reliable information for people than giving the End-to-End trading decision. Also, different from traditional researches which put efforts in abstracting empirical knowledge from noisy price information or concluding static trading preferences from historical information, we find a way to represent the dynamism of trading knowledge by explicitly considering the market fluctuation, and this feature makes the proposed Investor-Imitator framework a more applicable model for trading knowledge extraction in stock market.

2 PRELIMINARIES

In this section, we will provide some necessary preliminaries for readers to better understand this paper. The work proposed in this paper is set up in the background of algorithmic financial investment. The basic trading strategy in this paper is to generate a portfolio each day and we will trade stocks according to these portfolios. People have also designed several evaluation metrics to have a comprehensive evaluation of the algorithm, these metrics can basically tell us the profit/risk aspects of our trading strategy.

2.1 Portfolio Investment

Portfolio refers to a combination of financial assets such as stocks, bonds and cash[2]. Investors including individual/professional investors, Mutual Funds, Hedge Funds, banks or other financial institutions, adjust their portfolio according to their understandings of the security market or an established trading logic. Due to the volatility of the market, dynamic portfolio is more practical than fixed holdings. Dynamic portfolio selection [28] is a sequential decision-making process which continuously reallocates an amount of fund into a number of different financial assets, aiming at maximizing the return while restraining the risk [15, 29]. Researches about dynamic portfolio have been well surveyed by [24]. Capital Growth Theory (CRT) [19] and Mean Reversion (MR) [5, 35] are two basic rules for dynamic portfolio construction. **Follow-the-Winner** is a typical category following *CRT* [1, 8, 13, 16] and **Follow-the-Loser** which typically follows *MR* has also showed its effectiveness [6, 17, 23, 25, 26].

2.2 Evaluation Metrics

We select three important evaluation metrics to evaluate the model performance on a standard back-test platform. These three main metrics are:

- **Annualized Return, AR** Annualized return is a common profit indicator in finance. It is calculated by re-scaling the Rate of Return($rate\_of\_return$) in a l-days period to one year.

$$AR = rate\_of\_return \times \frac{365}{l} \times 100\% \tag{1}$$

- **Max Drawdown, MD** Max drawdown is a popular trading risk indicator in finance. It measures the maximum loss of wealth in a trading period from a peak to a trough of a given portfolio strategy.

$$MD(t) = \max_{\tau \in (0,t)} \left[\max_{t \in (0,\tau)} X_t - X_\tau \right]$$

where t denotes the date and X_t is the total wealth at the t-th day.

- **Sharpe Ratio, SR** Sharpe ratio [36] is a risk-adjusted profit measure. SR measures the return per unit of deviation.

$$SR = \frac{E[R_a - R_b]}{\sqrt{var[R_a - R_b]}}$$

where R_a is the return of given portfolio and R_b denotes the benchmark return.

We also use the **Exceed Return** (ER) and **Win Rate** (WR) in this paper. Exceed Return is the excess return our strategy earns over the benchmark return and Win Rate evaluates how many trading days we earn more profit than benchmark during the whole trading period. In this paper, we set the benchmark model by investing all stocks equally in each trading day.

3 THE INVESTOR-IMITATOR FRAMEWORK

For training an Investor-Imitator, we should provide it with a set of logic descriptors which is formulated by several metric specific trading models. In this section, we first propose the Rank-Invest model which can output portfolio directly. The Rank-Invest model is trained to approximate different evaluation metrics and then we can choose some of them as a logic descriptor pool of the Investor-Imitator. With a logic descriptor pool, an Investor-Imitator module is built under the REINFORCE [43] framework. The action is defined as the selection or preference to these logic descriptors, and we also give the state definition specifically according to our Investor-Imitator framework. When imitating different target investors, the corresponding reward definition is also given in this section.

3.1 Market/Stock Descriptor

We can find many sources of information to represent how the Market or Stock performs including the historical price, news, fundamental information. In this paper, we use some popular technical factors [33] to describe the stock state and market state. These technical factors which is mathematically computed with stocks' historical price and volume can help foresee some aspects of stocks' future performance [2, 7, 20].

We use these factors to describe the stock's state at day t, these statistical factors can help reduce the noises in stock market and abstract higher level information of one stock. The same technique can also be applied to describe the market state. To formalize the model, given a set of technical factors, the stock state and market state at the t-th trading day is formulated as $D_t^k \in \Re^d$, D_t^{Market} where $k \in K$ denotes stock k in the stock pool K.

3.2 Metrics-specific Rank-Invest model

The Rank-Invest model in this paper is trained by learning towards the metric related rank information to generate the final portfolio.

[2] https://en.wikipedia.org/wiki/Portfolio_(finance)

The main purpose of Rank-Invest model is to provide metric-specific trading models as the trading logic descriptors, each logic descriptor can represent the investors' preference in trading. A natural way of training a model preferring a certain evaluation metric is to design a supervised model with hand-made labels related to the metric performance. However, because of much noise and randomness of stocks, designing a label-based model for each stock is too explicit to resist the market noise. In this paper, we give a fuzzy learning approach which utilizes the second-order information represented by the stocks' metric rank. To our knowledge, the stocks' performance rank information is relatively stable compared to single stock's price performance.

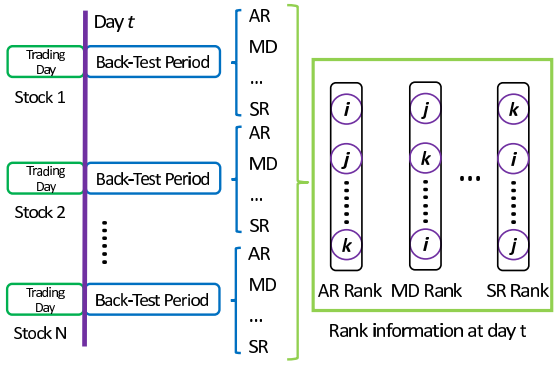


Figure 1: Rank Information Extraction

Firstly, we derive the metric-specific rank by trading each stock in future T days like Fig. 1 shows, and compute the target metric scores like AR, MD and SR. We denote the rank of i-th metric at t-th day as $\hat{G}_{i,t}$. A metric-specific Rank-Invest model can map Market/Stock Descriptor D_t^k of each stock k to stock rank $G_{i,t}$ for a given target metric i.

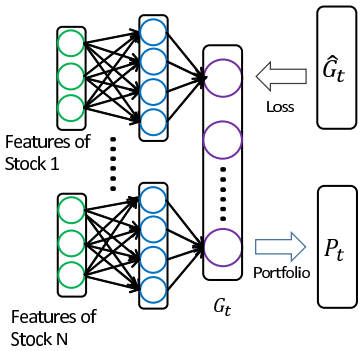


Figure 2: Rank-Invest Model

Each element in $G_{i,t}$ is mapped from stock feature described by Market/Stock Descriptor D_t^k with a three-layer neural network $f_{\theta_{rk}}(\cdot)$ which is shared across all stocks in Fig. 2. For one certain evaluation metric, the rank information is represented by $G_{i,t} = \left[f_{\theta_{rk}}\left(D_t^1\right), \cdots, f_{\theta_{rk}}\left(D_t^{|K|}\right)\right]$. We can compute the Pearson Correlation of $G_{i,t}$ and the ground truth $\hat{G}_{i,t}$, and we maximize the Pearson Correlation score in Eq. 2 to optimize the ranking model by Gradient Ascent $\nabla\theta_{rk} = \frac{\partial Pear_{i,t}}{\partial\theta_{rk}}$.

$$Pear_{i,t} = \frac{Cov\left(G_{i,t}, \hat{G}_{i,t}\right)}{\sqrt{Var\left(G_{i,t}\right)}\sqrt{Var\left(\hat{G}_{i,t}\right)}} \tag{2}$$

The Rank-Invest model can be trained towards any evaluation metrics such as AR, MD and SR. Except for the advantage that the Rank-Invest model can help filter the fluctuation in stock market by approximating ground truth rank, the other superiority of the proposed Rank-Invest model to traditional learning method is that it can directly give the final trading portfolio by normalizing the learned rank information $G_{i,t}$. For simplicity, we use the soft-max portfolio strategy Eq. 3 in this paper.

$$\mathbf{P}_t = \left[\frac{exp^{G_{i,t}^1}}{\sum_{k=1}^{|K|} exp\left(G_{i,t}^k\right)}, \frac{exp^{G_{i,t}^2}}{\sum_{k=1}^{|K|} exp\left(G_{i,t}^k\right)}, \cdots, \frac{exp^{G_{i,t}^{|K|}}}{\sum_{k=1}^{|K|} exp\left(G_{i,t}^k\right)}\right] \tag{3}$$

where K denotes the stock pool we can trade with, and $\mathbf{P}_t$ is the portfolio of next trading day related to the optimized rank information $G_{i,t}$. Readers should be noted that the framework proposed in this paper is not limited with this kind of logic descriptor, any kinds of trading methods or hybrid methods having certain preference signals, can be handled in the proposed framework.

3.3 The Investor-Imitator Module

In this section, we will give the detailed design of Investor-Imitator. When we have a set of logic descriptors and each of them has a clear preference to certain evaluation metric empirically, we can train an Investor-Imitator to extract the background trading logic of the target investor by choosing one logic descriptor to trade or combining some of these logic descriptors. In this paper, we adopt the reinforcement learning framework to construct the Investor-Imitator model which can learn from multi-types information by designing different rewards.

3.3.1 ***Direct Reinforcement Framework.*** We take the direct reinforcement method or policy search method to teach the Investor-Imitator with immediate reinforcement. Although delay reinforcement receives much attention, it is impracticable for us to approximate long-term reward in the stock market. In the standard reinforcement learning framework, where we build an agent interacting with a Markov decision process(MDP), the action, state and reward at each time t are denoted $s_t \in S$, $a_t \in A$, and $R_t \in \Re$. During the trading process, the action a_t we take is the selection of logic descriptor under certain market state and the reward R_t is from imitating the target investor. The trading dynamics can be represented by transition probabilities $P_{s,s'}^a = Pr\{s_{t+1} = s' | s_t = s, a_t = a\}$, the expectation of reward is $R_s^a = E\{R_{t+1} | s_t = s, a_t = a\}$, $\forall s, s' \in S, a \in A$. The policy is characterized by $\pi(s, a, \theta) = Pr\{a_t = a | s_t = s, \theta\}$, $\forall s \in S, a \in A$, where $\theta \in \Re^l, l \ll |S|$. With the immediate reinforcement when imitating the target investor, we build a direct reinforcement model which is optimized by policy gradient. Let our policy function $\pi(s, a)$ be differentiable with respect to the parameter θ that $\frac{\partial\pi(s,a)}{\partial\theta}$ exists. In this paper, we set a soft-max

policy to generate the trading decision.

$$\pi_\theta(s,a) \propto e^{f_\theta(\phi(s,a))} \tag{4}$$

where $\phi(s,a)$ denotes the feature vector. The corresponding probability of actions in our problem can be seen as the preference to certain model in the logic descriptor pool which is expected to have a better performance in the future, we name the probability $P^a_{s,s'}$ as **Trading Preference** or **Trading Knowledge** in the following paper. The Investor-Imitator is running in the stock market, and it is optimized by maximizing the reward expectation.

$$J(\theta) = \sum_s d^{\pi_\theta}(s) \sum_a \pi_\theta(s,a) R_{s,a} \tag{5}$$

where $d^{\pi_\theta}(s)$ is stationary distribution of Markov chain for π_θ and the reward $R_{s,a}$ is the advantages of action a to other action $a' \in \{A, a' \neq a\}$.

$$R_{s,a} = r_{s,a} - \frac{1}{|A|-1} \sum_{a' \in A, a' \neq a} r_{s,a'} \tag{6}$$

Where $r_{s,a}$ is the action-state feedback from the target investor. When optimizing towards different investor, we will revise the feedback according to the investor's knowledge. The policy function $\pi_\theta(s,a)$ is optimized by policy gradient.

$$\nabla \pi_\theta(s,a) = E_{s' \in S, a' \in A}\left[\nabla \pi_\theta(s',a') R_{s',a'}\right] \tag{7}$$

3.3.2 ***Action & State Design in Investor-Imitator.*** The action a_t determines which logic descriptor to choose in current market, or known as the **Preference** or **Knowledge**. In our framework, we can choose an action deterministically or stochastically according to $\pi_\theta(s,a)$. Different from traditional reinforcement learning problem, the most influential state in our problem is the market state like rising or falling, while our action has no direct impact on the market. However, trading in the stock market is a sequential process, and people always refer to their previous knowledge and experience to derive a reasonable decision. Moreover, the Investor-Imitator will examine the recent performance of the logic descriptor pool to drop these poorest models. So the state s_t in our problem is composed of action-irrelevant market state s_t^{Market}, the trading memory s_t^{Mem} and the state of the logic descriptor pool s_t^{Pool}, $s_t = \left[s_t^{Market}, s_t^{Mem}, s_t^{Pool}\right]$. We use the technical factors to describe the market state s_t^{Market} and we assume the market state s_t^{Market} will not be influenced by our trading decision. We evaluate each of the logic descriptor m in the logic descriptor pool and test the model in the most recent market with a set of evaluation metrics $\mathbf{em}_t^m$ including ER, AR, MD, SR, WR, we can generate the market-pool state vector $s_t^{Pool} = \{\mathbf{em}_t^1, \mathbf{em}_t^2, \cdots, \mathbf{em}_t^M\}$ by concatenating evaluation vector of these logic descriptors. The descriptor pool state vector can provide a strong indication of how these model perform recently to Investor-Imitator. For building the action-state transition and making a smoother decision, we interact with the **Preference Memory** in Fig. 3 for the Investor-Imitator to make sequential decision and also, from the perspective of learning, the memory can help the Investor-Imitator better converges to a satisfactory state. Finally we get the state vector $s_t = \left[s_t^{Market}, s_t^{Mem}, s_t^{Pool}\right]$ provided for the Investor-Imitator.

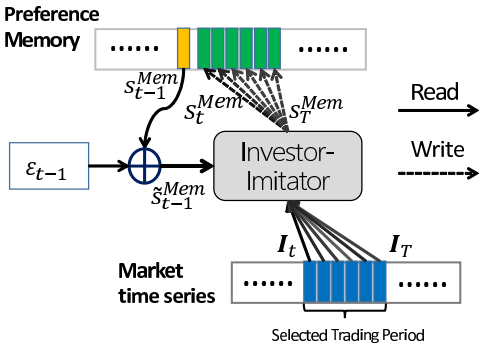


Figure 3: The Investor-Imitator Memory

3.3.3 ***Reward Design in Investor-Imitator.*** Investor-Imitator is designed to learn towards different types of investors such as the Oracle Investor, the Collaborator Investor and the Public Investor. We design the investor-specific rewards for these investors respectively.

• **Oracle Investor:** The Oracle Investor can directly tell us the Investor-Imitator's performance with respect to a certain evaluation metric such as AR, MD and SR. We can directly take the evaluation score as the reward and maximize the historical reward expectation. Thus the Oracle Investor's feedback is defined by:

$$r_{s,a} = OI_{score},\ OI_{score} \in \{AR, MD, \cdots, SR\} \tag{8}$$

• **Collaborator Investor:** The Collaborator Investor shares his historical portfolio, and the Investor-Imitator can learn to approximate him by extracting his trading knowledge. Thus given a series of portfolio, the feedback from the Collaborator Investor can be evaluated by defining a similarity score.

$$r_{s,a} = \frac{\frac{Cov\left(\mathbf{P}_{s,a}, \hat{\mathbf{P}}_s\right)}{\sqrt{Var\left(\mathbf{P}_{s,a}\right)}\sqrt{Var\left(\hat{\mathbf{P}}_s\right)}}}{\sqrt{\frac{1}{|K|}\sum_{k \in K}\left(\hat{\mathbf{P}}_{s,k} - \mathbf{P}_{s,a,k}\right)^2}} \tag{9}$$

where $\hat{\mathbf{P}}_{s,k}$ is the portfolio score of stock k of the target Collaborator Investor at state s and $\mathbf{P}_{s,a}$ is the portfolio of Investor-Imitator by taking action a at state s. Eq. 9 evaluates the correlation score and portfolio value similarity of learned portfolio to target portfolio, the numerator measures the rank similarity and the denominator measures the scale divergence.

• **Public Investor:** In the open trading environment, we can easily obtain the revenue curve of public investors such as Mutual Funds. With the revenue curve, the Investor-Imitator can learn to trade by approximating the Public Investor's revenue curve and try to formulate the trading logic of the Public Investor. Thus, in each trading day, the Investor-Imitator runs it's portfolio $\mathbf{P}_{s,a}$ and obtain the corresponding revenue curve $c_{s,a}$ of the following T days. Given the Public Investor's revenue curve as $\hat{c}_s$, the Investor-Imitator can

compute the similarity score similar to Eq. 9:

$$r_{s,a} = \frac{\frac{Cov(c_{s,a}, \hat{c}_s)}{\sqrt{Var(c_{s,a})}\sqrt{Var(\hat{c}_s)}}}{\sqrt{\frac{1}{|T|}\sum_{t=1}^{T}\left(\hat{c}_{s,t} - c_{s,a,t}\right)^2}} \tag{10}$$

where $\hat{c}_{s,t}$ is the revenue score of target investor at day t.

3.3.4 ***Exploration & Exploitation***. Exploration and Exploitation are two essential techniques in reinforcement learning [18] to learn a model with high generalization performance. In the Investor-Imitator, we select the action with probability $P(a|a \in A, s) = \pi_\theta(s, a)$. We can decide to choose the logic descriptor with the highest output probability deterministically or stochastically, however the model might be biased by the output probability because of the **Preference Memory**. In this paper, we utilize the idea of ϵ-Greedy by randomly selecting an action with a given probability during training. It's important for the Investor-Imitator model to explore more in the action space because the logic descriptor pool might be dominated by one logic descriptor in some market period. And, when referring the Investor-Imitator Preference Memory, each time we query the memory for historical information, we add a small Gaussian disturbance to avoid over-fitting of the memory and do more exploration.

$$\tilde{s}_{t-1}^{Mem} = s_{t-1}^{Mem} + \varepsilon_{t-1} \text{ with } \varepsilon_{t-1} \sim N(\mu, \Sigma) \tag{11}$$

where the noise ε is generated from a M dimensional Gaussian distribution. Moreover, the back-test period length when we receive the target investor's feedback and generate the reward, is randomly selected from 5 days to 30 days. This strategy is necessary for the Investor-Imitator to exploit more in the volatile market.

3.3.5 ***Working Flow of Investor-Imitator Module***. We give the working flow of the Investor-Imitator module in Fig. 4. In the process of trading, we use the State Descriptor with technical factors to derive the market state s_t^{Market} and load the recent trading history s_t^{Mem} shown in Fig. 3. Then we run the logic descriptor pool in recent market to collect the model pool state vector $s_t^{Pool} = \{\mathbf{em}_t^1, \mathbf{em}_t^2, \cdots, \mathbf{em}_t^M\}$ for indicating how these model perform recently. Finally the Investor-Imitator can map the state feature $s_t = \left[s_t^{Market}, s_t^{Mem}, s_t^{Pool}\right]$ to give the final action probability distribution. We select the action stochastically and get the corresponding reward in back-test period. During sampling from the market with exploration and exploitation, we can optimize the Investor-Imitator to approximate any target investor.

4 EXPERIMENTS AND ANALYSIS

In this section, we set adequate experiments and give solid analysis to show the effectiveness of the Investor-Imitator. The Investor-Imitator is built with a logic descriptor pool which plays the role of logic elements, the Investor-Imitator is expected to interpret the trading logic of target investors with these logic elements. We'll model target investors including the Oracle Investor, the Collaborate Investor and the Public Investor in a totally simulating environment, for the simplicity of quantifying the Investor-Imitator's performance. In the following of this section, we first introduce our experiment environment. We'll firstly test the proposed Rank-Invest

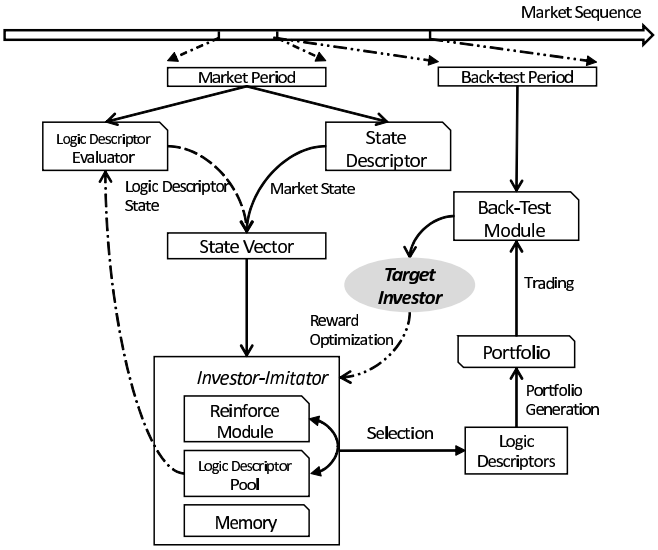


Figure 4: Working Flow of Investor-Imitator Module

model particularly on the Profit (AR) and the Risk (MD) aspect, because of the more intuitive analysis when they are accepted as the logic descriptors . Finally, we'll train Investor-Imitators to approximate three different types of investors to show its capability.

4.1 Data Description

In our experiment, we concentrate on researching Chinese Stock Market and focus on a set of barometer stock with enough historical record. We select 167 barometer stocks from HS300 Index in total, these stocks are selected from different industries including Oil, Steel, Retail, Bank, Electronic, etc. Thus the selected stocks have an overview of the whole market. The data we use ranges from 1/1/2005 to 12/31/2016, including some most representative bull market and bear market period (In Fig. 5). We use data from 1/1/2005 to 12/31/2013 for training, and test in the following 2014, 2015 and 2016 (Fig. 6). Fig. 6 shows the test period includes three most representative markets such as bull market in 2014, oscillating market in 2016 and even meet a market crash in 2015, it helps us have a clearer assessment of the Investor-Imitator.

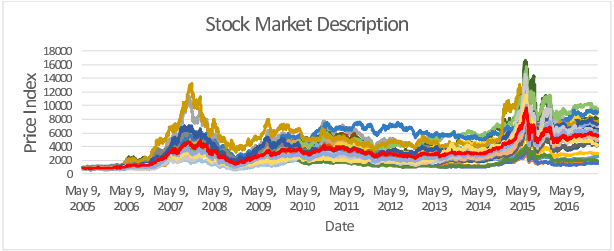


Figure 5: Stock Market Description. Lines with different colors represents the price index of stock corresponding industries, the red line stands for the Market price which is the weighted average price of all industries.

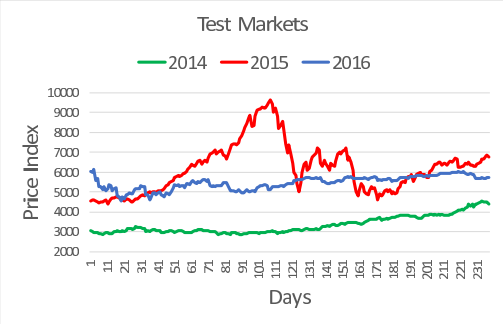


Figure 6: 2014, 2015, 2016 Market for Test

4.2 Evaluation of Rank-Invest Model

In real stock trading, the most essential aspects that people care are profit (AR) and risk (MD) performance. Thus, we can apply a profit model and risk model basically to help better interpret the trading logic. In this part, we set some experiments to demonstrate whether the Rank-Invest model itself can actually reflect the trading preference when it is trained to optimize certain evaluation metric.

In Table. 1, we can find that the profit AR model is more profitable in 2014 and 2016 with higher risk, and risk MD model has a dominating risk performance especially in the extremely oscillating market like 2015 compared to other models. Thus, with the experimental results of Rank-Invest models, we know that optimizing towards profit (AR) and risk (MD) is practical and the Rank-Invest models can actually improve the corresponding evaluation metrics performance under certain market state. So, it is a good choice of us to apply the AR model and MD model as the logic descriptors for Investor-Imitator. We also show the SR model performance for comparison, according to the result that the SR model has a neutral performance and shows no obvious advantages in any of these evaluation metrics. It may result from its complexity in considering the contradictory profit and risk simultaneously.

4.3 Imitate the Oracle Investor

We train the Investor-Imitator with AR model and MD model as the logic descriptors to approximate the Oracle Investor, who can foresee the future performance of the Investor-Imitator. We take SR model as the example for its explicit relation to profit and risk. In each trading day, the Oracle Investor tells us the Sharpe Ratio of our trading decisions in the near future.

The Investor-Imitator is trained in the period from 1/1/2005 to 12/31/2013 and in each trading day we take the future Sharpe Ratio to compute the advantage reward in Eq. 6 by running the Investor-Imitator. After several training epochs, we test Investor-Imitator in years 2014, 2015, 2016. Referring to Table. 1 and Fig. 7, we can find in the rising market 2014 when the profit AR model performs better, the Investor-Imitator can give a high probability over 0.5 to take the AR model in this rising market at most time. Especially in Fig. 7(a), the descending trend of AR preference in 2014 is caused by the Sharpe Ratio itself because SR penalizes not only the downside variance but also the upside variance, if the profit keeps rising sharply, the model will control it under a stable state by considering more to risks. In 2015 [Fig. 7(b)], when a stock market crash happened, the market is really an ordeal to investors. The Investor-Imitator gives rational trading signals when the market is rising irrationally, it can also switch to the risk model when the market falling suddenly to avoid risks. In 2016 [Fig. 7(c)], the market experienced a violent falling in the beginning of the year and stays in oscillating but rising state the left days, the Investor-Imitator's overall preference matches the market trend well and continuously switches between the profit model and the risk model to achieve better trading performance with respect to Sharpe Ratio. Thus, compared to Table. 1, with a deterministic action selection strategy by selecting the AR or MD model with the highest preference during test time, we can further improve the trading performance especially the target Sharpe Ratio with Investor-Imitator, the result is given in Table. 2. Note that the Sharpe Ratio scores of our directly learned rank-invest model are *WORSE* than the result of our Investor-Imitator framework, it also enlightens us to optimize profit-risk mixed evaluation metrics by considering profit and risk separately in future researches.

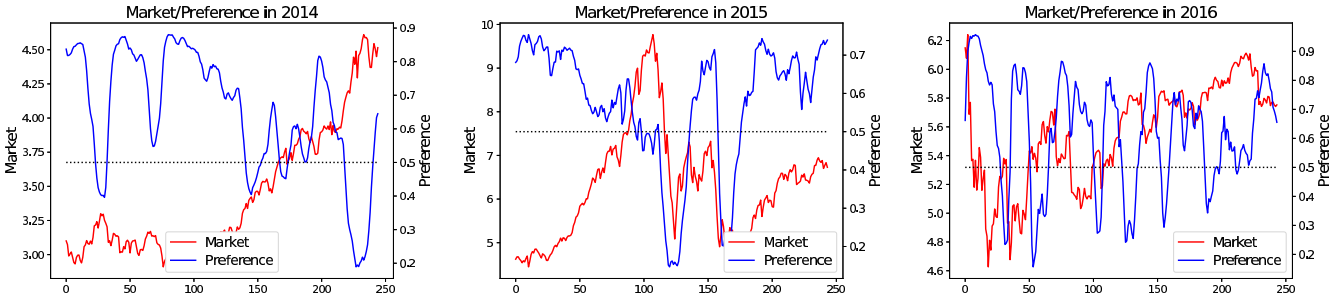
Apart from setting risk-adjusted return - Sharpe Ratio, as the target Oracle Investor, we can also set AR, MD as the target Oracle Investor representing profit preferred investor and risk preferred investor, for better analyzing the property of Investor-Imitator. We take the rising market 2014 for clearer comparison. The result is shown in Fig. 8, We can find in Fig. 8 that when we make the Investor-Imitator learn towards a profit targeting Oracle Investor who only cares the future AR, the Investor-Imitator will much prefer the AR model in the rising market, but if the Investor-Imitator is only told with the MD performance by the Oracle Investor, it will increase the importance of the MD model. Compared to the AR and MD targeting Investor-Imitator, when the Oracle Investor is risk-adjusted metric guided like Sharpe Ratio, the Investor-Imitator will be neutral between AR and MD. The result well demonstrates the designed property of the Investor-Imitator.

4.4 Imitate the Collaborator Investor

When collaborating with the Collaborator Investor, we can ask for the collaborator's historical portfolio series. However, the historical portfolio cannot provide any direct informative messages to our current trading decision. Thus, we can train an Investor-Imitator to simulate the Collaborator Investor's underlying trading logic, and we can consult our *Virtual Collaborator* for trading suggestions in return. In this scenario, we also use the AR model and MD model as the basic descriptor, and simulate a Collaborator Investor with the SR Rank-Invest model, the reward we take is defined in Eq. 9. In this experiment, we take oscillating market 2016 as a representative example. We plot the revenue curve of target investor and Investor-Imitator in Fig. 9. We also give the Pearson Correlation Curve between Investor-Imitator's daily portfolio and the Collaborator's portfolio given by the SR Rank-Invest model. We can find in Fig. 9(a) that by switching between AR model and MD model, the Investor-Imitator can have a positive correlation with the target portfolio and keep a high positive correlation in most trading days. The similarity scores (Eq. 10) of Investor-Imitator model to AR model, MD model and target SR model in their revenue curve are also given in Fig. 9(a), it is obvious that the Investor-Imitator model imitates

Table 1: Test Rank-Invest Models in Three Years

Year	2014					2015					2016				
Metrics	ER	AR	MD	SR	WR	ER	AR	MD	SR	WR	ER	AR	MD	SR	WR
AR model	**0.029**	**0.577**	0.111	0.151	0.486	0.090	0.373	0.446	0.055	0.500	**0.026**	**-0.028**	0.218	**0.002**	**0.566**
MD model	0.018	0.566	**0.084**	**0.173**	**0.518**	**0.203**	**0.485**	**0.351**	**0.084**	0.492	0.019	-0.035	**0.187**	-0.004	0.496
SR model	0.013	0.561	0.111	0.149	0.514	0.046	0.329	0.462	0.051	**0.525**	0.012	-0.042	0.221	-0.001	0.504



(a) 2014 (b) 2015 (c) 2016

Figure 7: Learning towards the Oracle[Sharpe Ratio]

Table 2: Test result with Investor-Imitator in three years

Year/Metrics	ER	AR	MD	SR	WR
2014	0.095	0.643	0.096	0.178	0.547
2015	0.266	0.548	0.414	0.074	0.496
2016	0.076	0.022	0.218	0.013	0.561

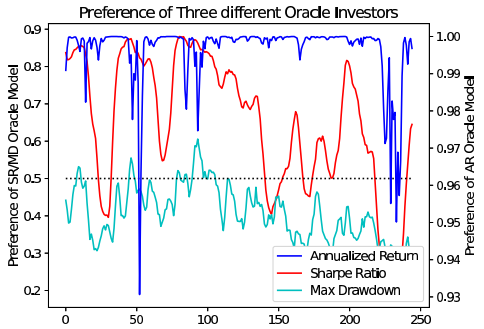
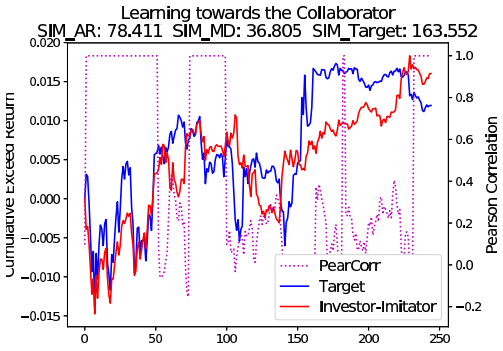


Figure 8: Test Investor-Imitator with 3 Targets in 2014

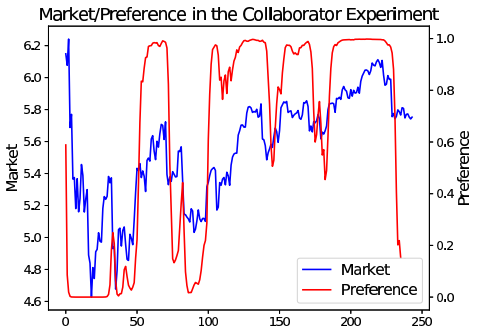
the Collaborator well. The profit preference of the learned Investor-Imitator is given in Fig. 9(b), the preference trend has a obvious positive correlation with the market trend, and this trading logic matches people's common sense well. We can conclude from Fig. 9 that the Investor-Imitator can learn to imitate the collaborator.

4.5 Imitate the Public Investor

In stock market, we can easily access the revenue curve of public investors such as the Mutual Funds. Obviously, the revenue curve contains narrow information for learners. Thus, we expect the Investor-Imitator can help extract valuable trading knowledge of these public investors. We simulate the Public Investor by SR Rank-Invest model, and the experiment is set in the oscillating market 2016. We firstly run the SR model in the market to collect its revenue curve. Then, We take AR model and MD model as the basic descriptor. The reward for learning towards the Public is given in Eq. 10. The experimental result is shown in Fig. 10. We plot the revenue curve of AR model, MD model, SR model and the Investor-Imitator. It is easy to find that the Investor-Imitator can obtain a satisfactory approximation to the target revenue curve. In Fig. 10(a), the similarity score computed with Eq. 10 shows that the Investor-Imitator can well imitate the target SR model, compared to the basic AR model and MD model in the revenue curve perspective. Comparing to imitating the Collaborator, although we imitating the same SR model with the same AR model and MD model, it is difficult of imitating the Public Investor to obtain a comparable performance to imitating the Collaborator. The cause is the noise hidden in the market makes the resultant revenue curse less informative. Thus, in this experiment, the output portfolio in test is obtained by the weighted sum of AR model portfolio and MD model portfolio according to the output preference because the limited information provided to Investor-Imitator makes it hard to output

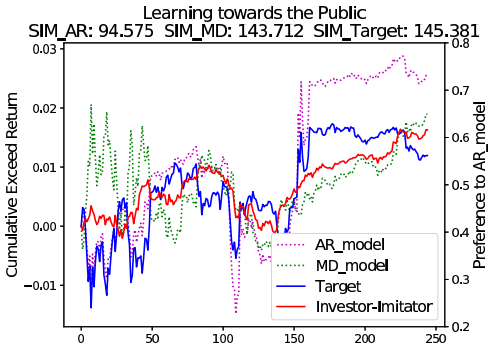


(a) the Collaborator/Investor-Imitator

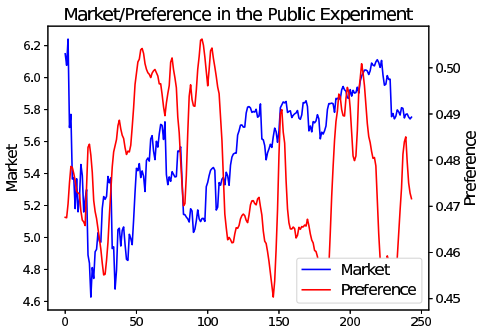


(b) Preference/Market

Figure 9: Learning towards the Collaborator



(a) the Public/Investor-Imitator



(b) Preference/Market

Figure 10: Learning towards the Public

an explicit preference like Fig. 10(b) shows - the output preference is around 0.5. Also in Fig. 10(b), a vague positive correlation of the preference curve to the market is given. The experimental results inspire us that if we can feed the Investor-Imitator more informative and explicit signals, the Investor-Imitator can approximate better to the target investor.

5 RELATED WORKS

Trading in a stock market aiming at making more profit is always a hot research topic. The trading strategy represented by dynamic portfolio also attracts the community of data mining and machine learning, researches try to invent new methods to mine informative knowledge to guide trading process in the noisy stock market. There are many representative research papers about data-driven dynamic portfolio strategy. [9] introduced group sparsity regularization to utilize the industry information. [37, 39] applied bandit learning framework to solve this problem. [38] was further inspired by ensemble learning and used re-sampling technique to promote portfolio research.

Due to the dynamism of portfolio selection, Reinforcement learning (RL) [41] gives an alternative to this problem. Reinforcement learning is an effective approach to automating goal-directed learning and has achieved great success in applications such as playing Atari-game [30] and even the complex Go game [40]. Different from traditional supervised learning with exemplary supervision, reinforcement learning receives direct supervision from interactions with the environment. There are two popular types of reinforcement learning methods including the value function method and policy search method in modern researches. Value function based methods such as Dynamic Programming [4], Q-Learning [42] works by learning to approximate the value function or Q-function and search the policy with respect to the value function. Policy search methods such as Williams' REINFORCE [43] map a finite-dimensional feature space directly to policy space with a differentiable function and optimize with policy gradients. Actor-Critic method [3] is another substantial RL method, this algorithm is intermediate between value function and policy search methods, which sets a "critic" learns a value function which is then used to update the parameters of the "actor".

Back to the stock trading problem, [32] proposed to use recurrent reinforcement learning (RRL) for direct reinforcement in portfolio selection, further [31] compared the direct reinforcement with Q-learning approach and proved the superiority of direct reinforcement to value function methods in trading problem. Recently, [10] researched the policy function incorporated in RRL, it took the advantages from a deep feature representation with a fuzzy feature transform layer, and tried to fully exploit the feature information.

Compared to this paper, previous RL related portfolio researches all tries to optimize a profitable portfolio function under the RL framework to dynamically trading in the market, while the proposed Investor-Imitator tries to combine the advantages of given portfolio models for more profit and better interpretation if these models have explicit meaning. It is our future work to take advantages of previous portfolio researches to generate more powerful logic descriptors for Investor-Imitator, to help train a wiser Investor-Imitator.

6 CONCLUSION

This paper proposed a reinforcement learning framework: the Investor-Imitator with Rank-Invest model. The purpose of the Investor-Imitator is to select or combine a set of interpretable logic descriptors in different market state to imitate a real investor and extract the investor's trading knowledge. The Investor-Imitator can be trained with rich trading histories from the market. For evaluating the capability of Investor-Imitator, we simulate 3 types of investor roles in the market such as the Oracle Investor, the Collaborator Investor and the Public Investor, representing different level of information disclosure we may meet in real market. Under the Investor-Imitator framework, we designed reward functions specifically for different investors and introduced the Rank-Invest model as the logic descriptor. Experimental results demonstrate that the Investor-Imitator can well utilize different types of information given by target investors. In general, the Investor-Imitator is not limited in learning to imitate these three roles or only with the Rank-Invest model as the logic descriptor. Under Investor-Imitator framework, people can apply their own trading models to learn to imitate their target investors. We assert that it is more meaningful in stock market to trade with knowledge than simply trading end-to-end.

ACKNOWLEDGEMENT

This work was supported in part by the National Natural Science Foundation of China (61422204, 61473149, and 61732006).

REFERENCES

[1] Amit Agarwal, Elad Hazan, Satyen Kale, and Robert E Schapire. 2006. Algorithms for portfolio management based on the newton method. In *Proceedings of the 23rd international conference on Machine learning*. ACM, 9–16.
[2] Gerald Appel. 2005. *Technical analysis: power tools for active investors*. FT Press.
[3] Andrew G Barto, Richard S Sutton, and Charles W Anderson. 1983. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE transactions on systems, man, and cybernetics* 5 (1983), 834–846.
[4] Richard Bellman. 2013. *Dynamic programming*. Courier Corporation.
[5] Werner FM Bondt and Richard Thaler. 1985. Does the stock market overreact? *The Journal of finance* 40, 3 (1985), 793–805.
[6] Allan Borodin, Ran El-Yaniv, and Vincent Gogan. 2004. Can we learn to beat the best stock. In *Advances in Neural Information Processing Systems*. 345–352.
[7] Robert W Colby. 2002. *The encyclopedia of technical market indicators*. McGraw Hill Professional.
[8] Thomas M Cover. 1991. Universal portfolios. *Mathematical finance* 1, 1 (1991), 1–29.
[9] Puja Das, Nicholas Johnson, and Arindam Banerjee. 2014. Online Portfolio Selection with Group Sparsity.. In *AAAI*. 1185–1191.
[10] Yue Deng, Feng Bao, Youyong Kong, Zhiquan Ren, and Qionghai Dai. 2017. Deep direct reinforcement learning for financial signal representation and trading. *IEEE transactions on neural networks and learning systems* 28, 3 (2017), 653–664.
[11] Robert D Edwards, John Magee, and WH Charles Bassetti. 2007. *Technical analysis of stock trends*. CRC press.
[12] Tomas Eklund, Barbro Back, Hannu Vanharanta, and Ari Visa. 2002. Assessing the feasibility of self organizing maps for data mining financial information. *ECIS 2002 Proceedings* (2002), 140.
[13] Alexei A Gaivoronski and Fabio Stella. 2000. Stochastic nonstationary optimization for finding universal portfolios. *Annals of Operations Research* 100, 1 (2000), 165–188.
[14] Paul A Gompers and Andrew Metrick. 2001. Institutional investors and equity prices. *The quarterly journal of Economics* 116, 1 (2001), 229–259.
[15] Robert A Haugen and Lemma W Senbet. 1986. Corporate finance and taxes: a review. *Financial Management* (1986), 5–21.
[16] David P Helmbold, Robert E Schapire, Yoram Singer, and Manfred K Warmuth. 1998. On-Line Portfolio Selection Using Multiplicative Updates. *Mathematical Finance* 8, 4 (1998), 325–347.
[17] Dingjiang Huang, Junlong Zhou, Bin Li, Steven CH Hoi, and Shuigeng Zhou. 2013. Robust Median Reversion Strategy for On-Line Portfolio Selection.. In *IJCAI*. 2006–2012.
[18] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. 1996. Reinforcement learning: A survey. *Journal of artificial intelligence research* 4 (1996), 237–285.
[19] John L Kelly. 1956. A new interpretation of information rate. *Bell Labs Technical Journal* 35, 4 (1956), 917–926.
[20] Charles D Kirkpatrick II and Julie A Dahlquist. 2010. *Technical analysis: the complete resource for financial market technicians*. FT press.
[21] Anju KJ and Anuradha PS. 2015. Determinants of Investor's Behaviour–An Analytical Review. (2015).
[22] Teuvo Kohonen. 1998. The self-organizing map. *Neurocomputing* 21, 1-3 (1998), 1–6.
[23] Bin Li and Steven CH Hoi. 2012. On-line portfolio selection with moving average reversion. *arXiv preprint arXiv:1206.4626* (2012).
[24] Bin Li and Steven CH Hoi. 2014. Online portfolio selection: A survey. *ACM Computing Surveys (CSUR)* 46, 3 (2014), 35.
[25] Bin Li, Steven CH Hoi, Peilin Zhao, and Vivekanand Gopalkrishnan. 2013. Confidence weighted mean reversion strategy for online portfolio selection. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 7, 1 (2013), 4.
[26] Bin Li, Peilin Zhao, Steven CH Hoi, and Vivekanand Gopalkrishnan. 2012. PAMR: Passive aggressive mean reversion strategy for portfolio selection. *Machine learning* 87, 2 (2012), 221–258.
[27] Sheng-Tun Li and Shu-Ching Kuo. 2008. Knowledge discovery in financial investment for forecasting and trading strategy through wavelet-based SOM networks. *Expert Systems with applications* 34, 2 (2008), 935–951.
[28] Harry Markowitz. 1952. Portfolio selection. *The journal of finance* 7, 1 (1952), 77–91.
[29] Harry M Markowitz. 1968. *Portfolio selection: efficient diversification of investments*. Vol. 16. Yale university press.
[30] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529.
[31] John Moody and Matthew Saffell. 2001. Learning to trade via direct reinforcement. *IEEE transactions on neural Networks* 12, 4 (2001), 875–889.
[32] John E Moody and Matthew Saffell. 1999. Reinforcement learning for trading. In *Advances in Neural Information Processing Systems*. 917–923.
[33] John J Murphy. 1999. *Technical analysis of the financial markets: A comprehensive guide to trading methods and applications*. Penguin.
[34] Lilian Ng and Fei Wu. 2006. Revealed stock preferences of individual investors: Evidence from Chinese equity markets. *Pacific-Basin Finance Journal* 14, 2 (2006), 175–192.
[35] James M Poterba and Lawrence H Summers. 1988. Mean reversion in stock prices: Evidence and implications. *Journal of financial economics* 22, 1 (1988), 27–59.
[36] William F Sharpe. 1966. Mutual fund performance. *The Journal of business* 39, 1 (1966), 119–138.
[37] Weiwei Shen and Jun Wang. 2016. Portfolio Blending via Thompson Sampling.. In *IJCAI*. 1983–1989.
[38] Weiwei Shen and Jun Wang. 2017. Portfolio Selection via Subset Resampling.. In *AAAI*. 1517–1523.
[39] Weiwei Shen, Jun Wang, Yu-Gang Jiang, and Hongyuan Zha. 2015. Portfolio Choices with Orthogonal Bandit Learning.. In *IJCAI*. 974.
[40] 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. 2016. Mastering the game of Go with deep neural networks and tree search. *nature* 529, 7587 (2016), 484–489.
[41] Richard S Sutton and Andrew G Barto. 1998. *Reinforcement learning: An introduction*. Vol. 1. MIT press Cambridge.
[42] Christopher John Cornish Hellaby Watkins. 1989. *Learning from delayed rewards*. Ph.D. Dissertation. King's College, Cambridge.
[43] Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Reinforcement Learning*. Springer, 5–32.