Wealth Flow Model: Online Portfolio Selection Based on Learning Wealth Flow Matrices

JIANFEI YIN, College of Computer Science and Software Engineering and National Engineering Laboratory for Big Data System Computing Technology, Shenzhen University
RUILI WANG, School of Natural and Computational Sciences, Massey University
YEQING GUO, Tisson Regaltc Communication Technology
YIZHE BAI, SHUNDA JU, WEILI LIU, and JOSHUA ZHEXUE HUANG, Shenzhen University

30

This article proposes a deep learning solution to the online portfolio selection problem based on learning a latent structure directly from a price time series. It introduces a novel wealth flow matrix for representing a latent structure that has special regular conditions to encode the knowledge about the relative strengths of assets in portfolios. Therefore, a wealth flow model (WFM) is proposed to learn wealth flow matrices and maximize portfolio wealth simultaneously. Compared with existing approaches, our work has several distinctive benefits: (1) the learning of wealth flow matrices makes our model more generalizable than models that only predict wealth proportion vectors, and (2) the exploitation of wealth flow matrices and the exploration of wealth growth are integrated into our deep reinforcement algorithm for the WFM. These benefits, in combination, lead to a highly-effective approach for generating reasonable investment behavior, including short-term trend following, the following of a few losers, no self-investment, and sparse portfolios. Extensive experiments on five benchmark datasets from real-world stock markets confirm the theoretical advantage of the WFM, which achieves the Pareto improvements in terms of multiple performance indicators and the steady growth of wealth over the state-of-the-art algorithms.

CCS Concepts: • **Theory of computation** → **Online learning algorithms**; • **Computing methodologies** → **Sequential decision making**; • **Applied computing** → *Economics;*

Additional Key Words and Phrases: Online portfolio selection, wealth flow matrix, deep reinforcement learning, regret bound

We thank the editors and reviewers for their expert comments and constructive suggestions, and the support of the Shenzhen Basic Research Fund (Grant No: JCYJ20200813091134001) and National Natural Science Foundation of China (Grant No: 61972261).

Authors' addresses: J. Yin, College of Computer Science and Software Engineering and National Engineering Laboratory for Big Data System Computing Technology, Shenzhen University, Nanshan District, Shenzhen 518060, China; email: yjf@szu.edu.cn; R. Wang (corresponding author), School of Natural and Computational Sciences, Massey University, Auckland 102904, New Zealand; email: ruili.wang@massey.ac.nz; Y. Guo, Tisson Regaltc Communication Technology, Guangzhou 510623, China; email: guoyeqing@tisson.cn; Y. Bai, S. Ju, W. Liu, and J. Z. Huang, Shenzhen University, Shenzhen 518060, China; emails: {baiyizhe2017, jushunda2017}@email.szu.edu.cn, {liuwl, zx.huang}@szu.edu.cn.

1556-4681/2021/08-ART30 $15.00
https://doi.org/10.1145/3464308

ACM Reference format:
Jianfei Yin, Ruili Wang, Yeqing Guo, Yizhe Bai, Shunda Ju, Weili Liu, and Joshua Zhexue Huang. 2021. Wealth Flow Model: Online Portfolio Selection Based on Learning Wealth Flow Matrices. *ACM Trans. Knowl. Discov. Data.* 16, 2, Article 30 (August 2021), 27 pages.
https://doi.org/10.1145/3464308

1 INTRODUCTION

In this article, we propose a deep learning solution to the online portfolio selection problem based on learning a latent structure directly from price time series. Online portfolio selection [2, 7, 10, 13, 15, 46] is about how to rebalance wealth proportion vectors $\boldsymbol{x}_t$ among N assets of a portfolio during each trading period $t \in [1, T]$ to maximize the wealth of the portfolio at the end of period T. Many studies have demonstrated that markets have short-term trends, such as the mean reversion [56, 67] and trend following [26]. Therefore, when we make rebalancing decisions, the short-term market trends need to be naturally embedded in a latent structure that defines different capital preferences for the various assets of portfolios.

Since learning latent structures is an effective way to capture structural knowledge [18, 24], we propose a wealth flow matrix $\boldsymbol{Y}_t$ for representing the latent structure, and it encodes knowledge about the relative strengths of assets in portfolios. Wealth flow matrices $\boldsymbol{Y}_t$ can be learned directly from the price time series by our algorithm. The ith row of $\boldsymbol{Y}_t$ represents the output wealth proportions of asset i relative to other assets $j \neq i$. The sum of each row vector of $\boldsymbol{Y}_t$ is required to be one to meet the self-financing property of portfolios [37]. Given a wealth flow matrix Y_t, we can obtain the wealth proportion vector $\boldsymbol{x}_t$ from $\boldsymbol{Y}_t$ by using simple linear operations, but not vice versa. As such, a portfolio selection model that can predict wealth flow matrices is more generalizable than models that can only predict wealth proportion vectors.

Although, there are exist portfolio selection algorithms that can represent the latent structures of portfolios, such as the stability and sparsity of wealth proportions [17, 34, 67], directly from price data, they only treat these latent structures as regular conditions in their objective functions for limiting the searching space of candidate solutions (i.e., wealth proportions). These latent structures are implicitly presented in the solutions. That is, their solutions cannot learn these latent structures from the dataset D_1 and then reuse them in another dataset D_2 without searching D_2 again. Also, these latent structures cannot represent the relative strength relationship between any pair of assets in portfolios. There exist a few studies on learning useful signals from out-of-band sources, such as the use of news [60, 61] to generate wealth proportions from neural networks. The signals found by these methods have longer delays than the signals provided by our wealth flow matrices $\boldsymbol{Y}_t$, since $\boldsymbol{Y}_t$ are learned directly from price time series.

With the concept of the wealth matrix in mind, we build a **wealth flow model** (**WFM**) to learn wealth flow matrices and maximize portfolio wealth simultaneously. First, we define a linear function h to bridge the wealth flow matrices $\boldsymbol{Y}_t$ and wealth proportion vectors $\boldsymbol{x}_t$. Since all $\boldsymbol{Y}_t$ represent probabilistic latent structures of portfolios, based on the variational Bayesian method [6], we use the Kullback–Leibler divergence [54] to measure the similarity between the predictive and posterior distributions of $\boldsymbol{Y}_t$ so that $\boldsymbol{Y}_t$ can be learned online from samples. Second, to address both the exploitation of wealth flow matrices and the exploration of wealth growth, we investigate the regular conditions for updating $\boldsymbol{x}_t$ when updating directions $\boldsymbol{Z}_t$ for $\boldsymbol{Y}_t$ are given. We find that the regular conditions are $\phi_{t+1} = 1/(Ht)$ and $(\nabla_t - \boldsymbol{m}_t)^\intercal(\boldsymbol{x}_t - \boldsymbol{x}^*) \geq 0$, where ϕ_{t+1} are the learning rates for $\boldsymbol{x}_t$, $\boldsymbol{m}_t$ are the updating directions for $\boldsymbol{x}_t$ given $\boldsymbol{Z}_t$, and H is some scalar constant. The regular conditions are derived by letting $\boldsymbol{x}_t$ chase a regret bound under the online convex optimization framework [21]. The regular conditions are later integrated into our deep reinforcement learning algorithm for the WFM.

To learn wealth flow matrices effectively from high dimensional and massive price time series, we design a neural network to implement the WFM. The neural network of the WFM is a recursive neural network that uses long short-term memory [23] neural cells and runs on graphics processing units. We strengthen the neural cells by designing an attention mechanism based on the work [57, 59]. Therefore, the correlations of hidden features can be learned automatically to improve the prediction accuracy with respect to wealth flow matrices. Furthermore, we design a deep reinforcement learning algorithm to train the neural network of the WFM online. The algorithm estimates future losses by using a reply buffer [32, 48] without using the recursive estimation method used by Q-learning [20, 28] and the policy gradient [42, 43], because we have a precise definition of loss at each time step, instead of the uncertain rewards defined by the Q-value and action-value functions [28, 48].

In summary, we make the following main contributions in this article:

— We propose a novel wealth flow matrix to represent latent structures of portfolios. The wealth flow matrix has special regular conditions for encoding knowledge about the relative strengths of assets in portfolios. A model that can predict wealth flow matrices is more generalizable than models that can only predict wealth proportion vectors. The wealth flow matrix can be considered as side information [16, 65], but it can be learned directly from the given price time series without using out-of-band sources.
— We propose a novel portfolio selection model called the WFM, which learns wealth flow matrices and maximizes portfolio wealth simultaneously. The learning of wealth flow matrices is based on the principle of the variational Bayesian method [6]. The Kullback–Leibler divergence [54] is used to measure the similarities between the predictive and posterior distributions of wealth flow matrices. We investigate the conditions for updating the wealth proportion vectors when the updating directions for the wealth flow matrices are given. These conditions are later integrated into our deep reinforcement learning algorithm for the WFM so that the exploitation of wealth flow matrices and the exploration of wealth growth are integrated together in our training algorithm.
— We design a neural network to implement the WFM and a deep reinforcement learning algorithm to train the neural network. The neural network of the WFM is a recursive neural network in which cells are built with an attention mechanism. The deep reinforcement learning algorithm uses a replay buffer [32, 48] to estimate the loss at every time step without recursively estimating the Q-value [20, 28] and action-value [42, 43] functions.

The rest of this article is organized as follows. Section 2 reviews some related work regarding online portfolio selection. Section 3 gives the preliminaries and notations used throughout this article. Section 4 defines the proposed WFM for online portfolio selection. Section 5 presents the neural network for the WFM and the deep reinforcement learning algorithm. Section 6 presents a behavior analysis of the WFM neural network and some performance comparison experiments. Finally, we present our conclusion and future work in Section 7.

2 RELATED WORK

Online portfolio selection is a special case of an online optimization problem [4, 14, 63], in which an agent needs to make periodic decisions based on partially observable streaming data, such as financial time series. Cover [15] proposed the universal portfolio algorithm, which has a good sublinear regret bound $O(N \log T)$ with respect to the best constant rebalanced portfolio in hindsight. The universal portfolio algorithm belongs to a class of sampling-based algorithms [4]. Blum et al. [7] provided a proof of the regret bound in the universal portfolio algorithm based on α-parameterized integration. Following the logic of the proof given by Blum et al. [7], one can show that if an algorithm $\mathcal{A}$ gives any proportion of its wealth to its sub-algorithms that mimic the policy of the universal portfolio algorithm, then algorithm $\mathcal{A}$ obtains a sublinear regret bound [9, 65]. As such, sampling-based algorithms provide reasonable explorations of wealth growth in the space of wealth proportion vectors.

To improve the time complexity of universal portfolio algorithm while keeping sublinear regret bounds, some other algorithms were proposed. The exponentiated gradient algorithm [22] is a non-sampling-based algorithm that has a regret bound $O(\sqrt{N/T})$. The algorithm can be formalized as a special case of the multiplicative weight update method [4]. According to their method, if an algorithm $\mathcal{A}$ computes the probability of choosing portfolio decisions by the product of weights, algorithm $\mathcal{A}$ can have a sublinear regret bound. Agarwal et al. [2] proposed the online Newton step algorithm that has a bound $O(GDN \log T)$ by computing the inverse of the sum of matrices. Thus, the ONS algorithm exhibits poor scalability when the number of stocks N is large. Luo et al. [46] designed an efficient portfolio selection algorithm based on the online mirror descent framework [51].

The regret bounds of many algorithms [2, 15, 22, 46] are estimated in the sense of asymptotic behavior. These algorithms are distribution independent and do not consider the short-term patterns of inputs [31], such as the mean reversion [56] and the sparse weights of assets [17, 34, 67]. As tested on several benchmark datasets [9, 15, 38], these algorithms perform poorly,[1] compared with some other practical approaches [9, 36]. These practical algorithms are not proven to have sublinear regret bounds but perform well on benchmark datasets. For example, Borodin et al. [9] proposed the ANTICOR algorithm, which predicts wealth proportions by calculating the sample correlation coefficients between each pair of stocks in two adjacent time windows. Li et al. [36] proposed the online portfolio selection with the moving average reversion algorithm, which predicts wealth proportions based on a moving average estimation for the relative price vector $\boldsymbol{r}_{t+1}$ at the next trading period. Many online portfolio selection algorithms [10, 13, 17, 34, 67] have been proposed using different prior structures, and these have achieved good performances on benchmark datasets.

If users are most concerned about the intermediate losses in the contract life of a portfolio, they can adopt appropriate risk control strategies, such as introducing the indicator maximum drawdown into the optimization target [49], executing long-and-short-term risk control [5], optimizing for transaction costs [40], and conducting market sentiment analysis [60, 62]. If trading costs are so high as to restrict high frequency trading, timing of trading is one of the most effective trading methods [25, 66]. The basic idea of the method is to separate the given market sequence into sub-sequences and run an online portfolio selection algorithm $\mathcal{A}$ for each sub-sequence. Kozat et al. [33] proved that this method still has the property of possessing a sublinear regret bound if the algorithm $\mathcal{A}$ has a sublinear regret bound. The mean variance analysis approach [12] uses the covariance of rates of returns to measure the risk of the given portfolio. Xing et al. [61] proposed an algorithm using market sentiment views [11, 47] to estimate the mean vector and covariance matrix in the Black-Litterman model so that the wealth proportions from the first-order condition of the mean variance analysis can be computed.

Many regret bound-based online algorithms [2, 15, 22, 46] and other high-performance algorithms [9, 34, 36] design handcrafted features [50] for making decisions and use a minimal amount of historical data, usually only one to several data points, so that algorithms with limited computing power can make quick decisions when a new batch of data arrives. If the reality is that we only

[1]The test can be verified by using the open source codes at https://github.com/Marigold/universal-portfolios.

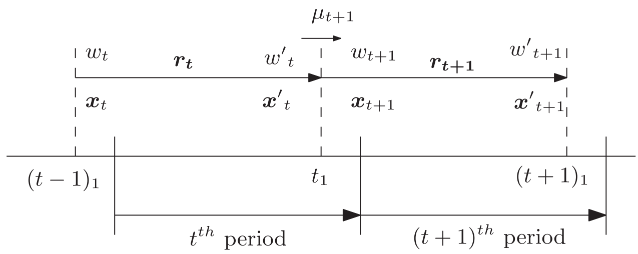


Fig. 1. Temporal relationship of variables in online portfolio selection. Trading time t_1 is generally the closing time in period t. The portfolio wealth w_{t+1} is evaluated immediately after trading assets at time t_1. The ith entry of relative price vector $\boldsymbol{r_{t+1}}$ is the ratio of the ith asset's price at time $(t+1)_1$ to the price at time t_1.

need to rebalance assets on a day-to-day basis (since relative price vectors are given daily); we need to manage a large number of assets, for example, $N > 1000$; the market has years of long-term memory [29]; the short-term behavior of the market do not meet the assumptions of the Markov process; or we do have sufficient computing power (e.g., in terms of graphics processing units), then, the performances of these algorithms will suffer from using too little data [3].

Online portfolio selection can be considered as an online game. Some **deep reinforcement learning** (**DRL**)- based methods [27, 28, 30, 42, 52] have been proposed for online trading and online portfolio selection. DRL is an integrated framework that combines high-dimensional feature learning and decision making. DRL uses graphics processing units to optimize complex functions built by neural networks, and it has been successful in many related areas, such as online video games [48] and the complex game Go [58]. To successfully apply DRL to solve the online portfolio selection problem, we need to carefully design reward functions that match the problem definitions to make the regrets of the obtained solutions bounded [27, 31, 52]. Q-learning [48] and policy gradient-based methods [42, 43] may be helpful for designing reward functions.

3 PRELIMINARIES AND NOTATIONS

In this section, we present the basic definition of the rebalancing process for a portfolio so that we can understand the objective functions defined in the following sections. The notations used in this article are also given in this section for reference.

3.1 Wealth Dynamic Model of Online Portfolio Selection

A portfolio has $N \geq 1$ tradable assets, which can be risk-free or risky assets. There are T periods for trading assets. At the end of each period t, $t \in [1, T)$, we can trade N assets so that the portfolio has a new wealth proportion $\boldsymbol{x_{t+1}} \in \triangle^N$, where $\triangle^N \stackrel{\text{def}}{=} \{\boldsymbol{x} : \boldsymbol{x} \in \mathbb{R}_+^N, \boldsymbol{x}_t^\intercal \mathbf{1} = 1\}$. For example, we sell and purchase assets at time t_1 during the period t to make the wealth of the ith asset equal to $w_{t+1}x_{t+1,i}$ for $i \in [1, N]$, where w_{t+1} is the portfolio wealth just after trading. The temporal relationship of portfolio selection is illustrated in Figure 1. Since the prices of the assets have changed from t_1 to $(t+1)_1$, the portfolio wealth w_{t+1} and its wealth proportion $\boldsymbol{x}_{t+1}$ will be changed implicitly by the market to

$$\begin{aligned} w'_{t+1} &= w_{t+1}\boldsymbol{x}_{t+1}^\intercal \boldsymbol{r}_{t+1}, \\ \boldsymbol{x}'_{t+1} &= \frac{\boldsymbol{x}_{t+1} \circ \boldsymbol{r}_{t+1}}{\boldsymbol{x}_{t+1}^\intercal \boldsymbol{r}_{t+1}}, \end{aligned} \tag{1}$$

where $\circ$ denotes the entry-wise product of two vectors. Note that rebalancing the portfolio at time t_1 will shrink the wealth portfolio from w'_t to

$$w_{t+1} = \mu_{t+1} w'_t, \tag{2}$$

by a factor $\mu_{t+1} \in (0, 1)$ if trading costs are considered. The factor μ_{t+1} satisfies a balance equation under the self-financing assumption [19]:

$$\sum_{i=1}^{N}(1 - cs_i)(x'_{t,i} - \mu_{t+1}x_{t+1,i})^+ = \sum_{i=1}^{N}(1 + cp_i)(\mu_{t+1}x_{t+1,i} - x'_{t,i})^+, \tag{3}$$

where cp_i and cs_i are the commission ratios of purchasing and selling assets, respectively. Typically, we set $cp_i = cs_i = 0$ for trading risk-free assets and $cp_i = cs_i = c$ for trading risky assets. $(y)^+ = y$ if $y \geq 0$, otherwise, $(y)^+ = 0$. The factor μ_t can be solved iteratively from the following iterative equation:

$$\mu_{t+1} = 1 - c_s \sum_{i=2}^{N}(\boldsymbol{x}'_{t,i} - \mu_{t+1}\boldsymbol{x}_{t+1,i})^+ - c_p \sum_{i=2}^{N}(\mu_{t+1}\boldsymbol{x}_{t+1,i} - \boldsymbol{x}'_{t,i})^+,$$

where the initial value of μ_t is $(1-c_s)/(1+c_p)$. By applying Equation (1) and Equation (2) recursively on w'_T, we know that the cumulative wealth of a portfolio at the end of period T will be $w'_T = w'_1 \prod_{t=2}^{T} \mu_t \boldsymbol{x}_t^\top \boldsymbol{r}_t$, where w'_1 is the initial wealth. The goal of portfolio selection is to maximize the cumulative wealth w'_T at the end of period T under random relative price vectors r_i.

Hereafter, we assume that the relative price vector r_i of any single asset i in a portfolio is bounded by $0 < L_r \leq r_i \leq U_r$, where L_r and U_r are some scalar constants. For the sake of readability, unless necessary, we suppress the suffix t for the time index when the meaning of time-dependent variables, e.g., $\boldsymbol{x}, \boldsymbol{r}$, can be understood from the context. We use the notation $\boldsymbol{y}$ to represent the vectorization of the wealth flow matrix $\boldsymbol{Y}$. The notations used in this article are summarized in Table 1.

4 OPTIMIZATION MODEL WITH WEALTH FLOW

This section presents an optimization model, i.e., the WFM for online portfolio selection. The model is based on the proposed prior structure of the wealth flow matrix. Several gradient-based properties of the WFM are investigated to enable the learning of wealth flow matrices and the maximization of portfolio wealth simultaneously.

4.1 Wealth Flow Matrix

Numerous studies support the existence of different short-term market trends [26, 56]. That is, before we make rebalancing decisions, the short-term market trends need to be naturally embedded in a latent structure that defines different capital preferences for the assets of portfolios. Based on this observation, we propose a wealth flow matrix to represent such latent structures. Each wealth flow matrix $\boldsymbol{Y}$ encodes knowledge about the relative strengths of the assets in the context of a portfolio. $\boldsymbol{Y}$ is an asymmetric and a nonnegative matrix. An example of $\boldsymbol{Y}$ is given as follows:

$$\begin{pmatrix} 0 & 0.5 & 0.5 \\ 0.2 & 0 & 0.8 \\ 0.3 & 0.7 & 0 \end{pmatrix}.$$

The example has $N = 3$ assets. The ith row vector represents the proportions of wealth that flow from asset i to the other assets $j \neq i$. For example, $y_{1,2}$ and $y_{1,3}$ are the proportions of wealth that flow from asset $i = 1$ to asset $j = 2$ and asset $j = 3$, respectively. We assume that there is no cache of wealth remaining in asset i's virtual account when asset i's wealth flows out to the virtual accounts of other assets. Thus, the diagonal elements of a wealth matrix are zeros, and the sum of each row is one.

Table 1. Glossary of Notations

Notation	Representation
N, T	N assets of a portfolio are traded in T trading periods
$\triangle^N$	N-dimensional simplex space, $\triangle^N = \{\boldsymbol{x} : \boldsymbol{x} \in \mathbb{R}_+^N, \boldsymbol{x}_t^\top \mathbf{1} = 1\}$
$\boldsymbol{e}_i, \boldsymbol{e}_w$	vectors, in which the ith or wth component is 1 and the others are zeros
$\mathbf{1}, \mathbf{I}$	$\mathbf{1}$ denotes a vector whose components are all ones, $\mathbf{I}$ denotes an unit matrix
$\boldsymbol{x}_t, \boldsymbol{x}'_t$	wealth proportion vectors at the beginning and end of the trading period t, respectively
w'_t, w_{t+1}	portfolio wealth before and after the rebalancing at the end of the trading period t, respectively
c_p, c_s	commission ratios for purchasing and selling assets
μ_{t+1}	discount factor for calculating the wealth w_{t+1}
$\boldsymbol{r}_t$	relative price vector at the end of the trading period t
$Y_t, \boldsymbol{y}_t$	Y_t denotes the wealth flow matrix at the beginning of the trading period t, $\boldsymbol{y}_t$ denotes the vectorization of Y_t, i.e., $\boldsymbol{y}_t = \text{vec}(Y_t)$
h	function that maps a Y_t to a $\boldsymbol{x}$
$\boldsymbol{Z}_t$	updating direction matrix for Y_t
$\boldsymbol{m}_t$	updating direction vector for $\boldsymbol{x}_t$ when updating Y_t
ρ, φ, ϕ	$\rho = \varphi/\phi$, where φ and ϕ are the learning rates for Y and $\boldsymbol{x}$, respectively
$\epsilon, \lambda, \Psi_1, \Psi_2$	penalty weights used in objective functions
$f_{t+1}(Y)$	objective function for Y
$g_t(\boldsymbol{x})$	evaluation function for $\boldsymbol{x}$ at the trading period t
$\theta, J_{t+1}(\theta)$	weights and objective functions for the WFM neural network, respectively
$\mathcal{R}$	replay buffer
L, R	loss function and regular function used by $J_{t+1}(\theta)$, respectively
p, q	density functions
H, G	constants used for estimating regret bounds
u	dimension of hidden features
v	number of convolutional kernels
w	window length of a sequence of $\boldsymbol{r}_t$
k	next k trading periods for computing the loss function L

Note that we can use the elements of a wealth flow matrix Y to compute a wealth proportion $\boldsymbol{x}$, but we cannot do the reverse. This means that a wealth flow matrix Y provides more information than yielded by a wealth proportion $\boldsymbol{x}$. It is supposed that if a model can predict wealth flow matrices Y and the wealth proportion $\boldsymbol{x}$ simultaneously, then the performance of the model will be improved.

4.2 Optimization Model

Since wealth flow information can be used as a prior to derive decisions regarding wealth proportions, we can represent the joint probability of wealth proportion $\boldsymbol{x}$ and wealth flow matrix Y as $\text{p}(\boldsymbol{x}, Y) = \text{p}(\boldsymbol{x} | Y)\text{p}(Y)$. Inspired by the method of maximum a posteriori estimation [53, 64], we design a WFM to maximize the logarithm of the joint probability $\text{p}(\boldsymbol{x}, Y)$. That is,

$$\max_{Y} f_{t+1}(Y) = \underbrace{\log(\boldsymbol{r}_t^\top h_{t+1}(Y))}_{\text{single-period goal}} - \underbrace{\frac{\epsilon}{2} \|h_{t+1}(Y)\|^2}_{\text{long-term goal}} - \underbrace{\lambda \sum_{i=1}^{N} \sum_{j \neq i} y_{i,j} \log\left(\frac{y_{i,j}}{\hat{y}_{i,j}}\right)}_{\text{short-term goal}} \tag{4}$$

$$\begin{aligned} s.t. \quad & \\ & Y \in \mathbb{R}_+^{N \times N}, \\ & \mathrm{Tr}(Y) = 0, \\ & Y\mathbf{1} = \mathbf{1}, \end{aligned}$$

where ϵ and λ are small positive numbers, $\mathrm{Tr}(Y)$ denotes the trace of matrix Y, $\|.\|_2$ denotes the l^2 norm function, and the linear function $h_{t+1} : \mathbb{R}_+^{N \times N} \times \mathbb{R}_+^N \longrightarrow \mathbb{R}_+^N$ is defined by

$$h_{t+1}(Y) = \boldsymbol{x}_t + Y^\top \boldsymbol{x}_t - \boldsymbol{x}_t \circ (Y\mathbf{1}). \tag{5}$$

For a future period $t+1$, h_{t+1} computes the net incomes (may be negative) for all assets by the vector $Y^\top \boldsymbol{x}_t - \boldsymbol{x}_t \circ (Y\mathbf{1})$. This vector is then added to the previous wealth proportion $\boldsymbol{x}_t$ to output the new proportion $\boldsymbol{x}_{t+1}$. The symbol $\circ$ denotes the entry-wise product. Since we do not allow assets to store wealth flows in themselves when rebalancing a portfolio, the diagonal elements $y_{i,i}$ of Y are constrained to zeros by $\mathrm{Tr}(Y) = 0$, and the sum of each row is normalized to 1 by the constraint $Y\mathbf{1} = \mathbf{1}$.

The short-term goal is defined in Equation (4) is to make profits by following the short-term market trends provided by the sample matrices $\hat{Y}_t$, i.e., samples of the wealth flow matrices. We use the Kullback–Leibler divergence [54] to measure the similarity between the empirical distribution $\mathrm{q}(Y)$ and the posterior distribution $\mathrm{p}(Y \mid \hat{Y}_t)$, as shown in Equation (4). The Kullback–Leibler divergence expression is derived by applying the variational Bayesian method [6]:

$$\begin{aligned} & min_{\mathrm{q}}\, \mathrm{KL}\left(\mathrm{q}(Y)\,||\,\mathrm{p}(Y\,|\hat{Y}_t)\right) \\ & \quad = \mathbb{E}_Y \left[\mathrm{q}(Y)\log(\mathrm{q}(Y)) - \mathrm{q}(Y)\log(\mathrm{p}(Y \mid \hat{Y}_t)\right]. \end{aligned} \tag{6}$$

The sample matrices $\hat{Y}_t$ can be learned by the neural networks designed in Section 5.

The single-period goal is to make profits by assuming that the relative price vector $\boldsymbol{r}_{t+1}$ in the near future will not change much from the current vector $\boldsymbol{r}_t$. The long-term goal in Equation (4) is to encourage the WFM to output an increasingly-balanced wealth proportion $\boldsymbol{x}_{t+1}$, which may cope with risks from market reversals. The long-term goal is based on the fact that the uniform vector $\frac{1}{N}\mathbf{1}$ is the solution for minimizing the l^2-norm of the vectors in $\triangle^N$. The uniform vector $\frac{1}{N}\mathbf{1}$ can be thought of as a bias for the predicted $\boldsymbol{x}_{t+1}$. Since the expression of the single-period goal alone is concave but not strictly concave, the long-term goal makes the expression $\log(\boldsymbol{r}_t^\top h_{t+1}(Y)) - \frac{\epsilon}{2}\|h_{t+1}(Y)\|_2^2$ a H-strong concave function by setting a tiny number for ϵ, as proved by the following lemma.

Lemma 4.1. *Let $g_t(\boldsymbol{x}) = \log(\boldsymbol{r}_t^\top \boldsymbol{x}) - \frac{\epsilon}{2}\|\boldsymbol{x}\|^2$; then, when $\epsilon \geq H > 0$, g_t is a H-strong concave function, that is,*

$$\forall \boldsymbol{x}_t \in \mathbb{R}^N : \nabla^2 g_t(\boldsymbol{x}_t) \preceq -H\,\mathbf{I}.$$

Here, we use the notation $\boldsymbol{A} \preceq \boldsymbol{B}$ if $\boldsymbol{B} - \boldsymbol{A}$ is a positive semidefinite matrix.

Proof. By the definitions of log and the l^2-norm, we have

$$\begin{aligned} \nabla^2 \log(\boldsymbol{r}_t^\top \boldsymbol{x}) &= \frac{-\boldsymbol{r}_t \boldsymbol{r}_t^\top}{(\boldsymbol{r}_t^\top \boldsymbol{x})^2}, \\ \nabla^2 \frac{-\epsilon}{2}\|\boldsymbol{x}\|^2 &= -\epsilon \mathbf{I}. \end{aligned}$$

Thus, let $\boldsymbol{y} \in \mathbb{R}^N$; then, we have

$$\boldsymbol{y}^\top(-H\,\mathbf{I} - \nabla^2 g_t)\boldsymbol{y} = (\epsilon - H)\,\|\boldsymbol{y}\|^2 + \frac{(\boldsymbol{r}_t^\top \boldsymbol{y})^2}{(\boldsymbol{r}_t^\top \boldsymbol{x})^2},$$

which means that $\epsilon \geq H > 0$ is a sufficient condition for making the above equation nonnegative. □

4.3 Gradient Information about Wealth Proportions and Flows

In this section, we analyze the gradient information related to the wealth proportion vector $\boldsymbol{x}$ and the wealth flow matrix Y to understand the optimization process of the WFM and the dynamics of the wealth flows between each pair of assets in a portfolio.

In the WFM's Definition (4), the wealth proportion $\boldsymbol{x} = h(Y)$ is, in fact, a dependent variable, which is driven by the independent variable, i.e., the wealth flow matrix Y. In the following lemma, we apply differential calculus to analyze how the dependent variable $\boldsymbol{x}$ changes with the independent variable Y.

Lemma 4.2. *Let $Y \leftarrow Y + \varphi Z$ be the updating rule for the independent variable Y of the model (4) and $\boldsymbol{x} \leftarrow \boldsymbol{x} + \phi\boldsymbol{m}$ be the corresponding change in the dependent variable $\boldsymbol{x} = h(Y)$, where the learning rates are $\varphi > 0$ and $\phi > 0$. Let $\rho = \frac{\varphi}{\phi}$; then, we have $m_i = \rho\sum_{j\neq i}(z_{j,i}x_j - z_{i,j}x_i), i,j \in [1,N]$ and $\|\boldsymbol{m}\|_2 \leq 2\sqrt{N}\rho$.*

Proof. Let the symbol d denote the forward difference operator; then, we have

$$\begin{aligned} dY &= \varphi Z, \\ d\boldsymbol{x} &= \phi\boldsymbol{m}. \end{aligned} \tag{7}$$

Recall that by Equation (5), the function h is used to obtain a new $\boldsymbol{x}$, that is,

$$h(Y) = \boldsymbol{x} + Y^\top\boldsymbol{x} - \boldsymbol{x} \circ (Y\mathbf{1}). \tag{8}$$

If we fix $\boldsymbol{x}$ and make a very small change to Y by dY in Equation (8), then dh can be used to approximate $d\boldsymbol{x}$, that is,

$$d\boldsymbol{x} = dY^\top\boldsymbol{x} - \boldsymbol{x} \circ (dY\,\mathbf{1}). \tag{9}$$

By the constraints of Y defined in Equation (4) and Equation (7), the elements $dy_{i,j}$ of the difference matrix dY are

$$dy_{i,j} = \begin{cases} \varphi z_{i,j} & \text{if } i \neq j, \\ 0 & \text{otherwise} \end{cases}, \text{ where } z_{i,j} \in [-1, 1]. \tag{10}$$

Let $\boldsymbol{a} = dY^\top\boldsymbol{x}$ and $\boldsymbol{b} = \boldsymbol{x} \circ (dY\mathbf{1})$. After a simple calculation, we know that the elements of $\boldsymbol{a}$ and $\boldsymbol{b}$ are

$$\begin{aligned} a_i &= \sum_{j\neq i}(x_j dy_{j,i}), \\ b_i &= x_i \sum_{j\neq i} dy_{i,j}. \end{aligned} \tag{11}$$

Combining Equations (9) and (11), we have

$$dx_i = \sum_{j\neq i}(x_j dy_{j,i}) - x_i \sum_{j\neq i} dy_{i,j}.$$

Furthermore, using Equations (7) and (10), we arrive at

$$m_i = \frac{\varphi}{\phi}\sum_{j\neq i}\left(z_{j,i}x_j - z_{i,j}x_i\right). \tag{12}$$

Let $\rho = \frac{\varphi}{\phi}$; then, the square l^2-norm of $\boldsymbol{m}$ is

$$\begin{aligned}\|\boldsymbol{m}\|_2^2 &= \rho^2 \sum_{i=1}^{N} \left(\sum_{j \neq i} \left(z_{j,i} x_j - z_{i,j} x_i \right) \right)^2 \\ &\leq 4N\rho^2,\end{aligned}$$

by $z_{i,j} \in [-1, 1]$ and $x_i \in \triangle^N$. □

Equation (12) gives the net-income rate m_i for asset i by using the gradient matrix $\boldsymbol{Z}$. The net-income rate m_i is then scaled by the learning rate ϕ and added to the previous wealth proportion x_i. This updating logic demonstrates the validity of using differential calculus for the definition of the function h, as shown in Equation (9). The bound $\|m\|_2 \leq 2\sqrt{N}\rho$ established by Lemma 4.2 shows that the amplitude of the search direction for finding the optimal wealth proportion $\boldsymbol{x}^*$ can be estimated by the relative learning ratio ρ. The parameter ρ will be further used in the proof of the regular conditions for the wealth proportions $\boldsymbol{x}_t$ in Section 4.4.

The gradient information related to wealth flow matrix Y demonstrates an interesting property about the directions of the wealth flows between pairs of assets in a portfolio. Note that the gradient for the single-period goal defined in Equation (4) is as follows:

$$\begin{aligned}&\nabla_Y \log\left(\boldsymbol{r}^\top h(Y)\right) \\ &= \nabla_Y \log\left(\boldsymbol{r}^\top(\boldsymbol{x} + Y^\top \boldsymbol{x} - \boldsymbol{x} \circ (Y\mathbf{1})\right) \\ &= \frac{\boldsymbol{r}\boldsymbol{x}^\top - (\boldsymbol{r} \circ \boldsymbol{x}\mathbf{1}^\top)^\top}{\boldsymbol{r}^\top(\boldsymbol{x} + Y^\top \boldsymbol{x} - \boldsymbol{x} \circ (Y\mathbf{1})}.\end{aligned} \tag{13}$$

An example of the numerator in Equation (13) is the following 3 by 3 matrix:

$$\begin{pmatrix} 0 & (r_1 - r_2)x_2 & (r_1 - r_3)x_3 & \longrightarrow \text{output of asset}_1 \\ (r_2 - r_1)x_1 & 0 & (r_2 - r_3)x_3 & \\ (r_3 - r_1)x_1 & (r_3 - r_2)x_2 & 0 & \\ \uparrow \text{input of asset}_1 & & & \end{pmatrix},$$

in which the element $(r_1 - r_2)x_2$ in row 1 and column 2 means that if the relative price vector r_1 is greater than r_2, then the first asset will output a wealth rate $(r_1 - r_2)$, scaled by x_2, to the second asset, where the weight x_2 is the previous wealth proportion of the second asset. This behavior of wealth flow is consistent with that of the mean reversion market. Note that the gradient matrix in Equation (13) is obtained by using the numerator layout of the scale-by-matrix derivative [8]. If the denominator layout [8] is used, then the gradient matrix must be the transpose of the matrix in Equation (13), and trend-following behavior will be discovered.

4.4 Conditions for Updating Wealth Proportion Vectors

To integrate both the exploitation of wealth flow matrices and the exploration of wealth growth in our model, in this section, we investigate the regular conditions for updating the wealth proportion vectors $\boldsymbol{x}_t$ when the updating directions $\boldsymbol{Z}_t$ for the wealth flow matrices $\boldsymbol{Y}_t$ are given. The regular conditions are derived by using an online convex optimization framework [21], where the regret bound defined for algorithm $\mathcal{A}$ can be written as follows.

Definition 1.

$$\text{Regret}(\mathcal{A}, [g_1, \ldots, g_T]) \stackrel{\text{def}}{=} \max_{\boldsymbol{x} \in \triangle^N} \sum_{t=1}^{T} g_t(\boldsymbol{x}) - \mathbb{E}_{\{\boldsymbol{x}_t \sim \mathcal{A}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{t-1})\}} \left[\sum_{t=1}^{T} g_t(\boldsymbol{x}_t) \right]. \tag{14}$$

The real valued functions g_t are evaluation functions for computing regret bounds. The first item of Equation (14) is the profit obtained by algorithm $\mathcal{A}$'s adversary, who can see all the market data, i.e., the relative price vectors $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_T$, before making only one decision regarding the wealth proportion. The second item is the expected profit obtained by algorithm $\mathcal{A}$. The game rules defined for $\mathcal{A}$ are the same as those defined in Section 3.1. Note that $\mathcal{A}$ can output a probabilistic wealth proportion $\boldsymbol{x}_t$ given the previous proportions $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{t-1}$.

Based on the definition of a regret bound in Equation (14), we have the following result.

Theorem 4.3. *Let $g_t(\boldsymbol{x}) = \log(\boldsymbol{r}_t^\intercal \boldsymbol{x}) - \frac{\epsilon}{2} \|\boldsymbol{x}\|^2$ be the evaluation functions required by the regret bound definition (14). Let $\boldsymbol{x}^* = \arg\max_{\boldsymbol{x} \in \triangle^N} \sum_{t=1}^N g_t(\boldsymbol{x})$ be the solution found by the adversary in hindsight, and let $\boldsymbol{x}_t$ and $\boldsymbol{m}_t$ be the wealth proportions and gradients given by Lemma 4.2, respectively. If conditions $\epsilon \geq H > 0$ and $(\nabla_t - \boldsymbol{m}_t)^\intercal (\boldsymbol{x}_t - \boldsymbol{x}^*) \geq 0, \forall t$ are satisfied, then the wealth proportions $\boldsymbol{x}_t$ have regrets bounded by $2N\rho^2(1 + \log T)/H$.*

Proof. Let $\nabla_t = \nabla g_t(\boldsymbol{x}_t)$, $\nabla_t^2 = \nabla^2 g_t(\boldsymbol{x}_t)$ be the gradient and Hessian of g_t at point $\boldsymbol{x}$. Since each g_t is twice differentiable, we can apply a Taylor series, up to second-orders, around our point $\boldsymbol{x}_t$ to represent the adversary's value $g_t(\boldsymbol{x}^*)$. Let $\boldsymbol{\xi} = \alpha \boldsymbol{x}_t + (1-\alpha)\boldsymbol{x}^*$, $\alpha \in [0, 1]$ and $\nabla_t^2(\boldsymbol{\xi})$ be the Hessian at $\boldsymbol{\xi}$; thus, we have

$$g_t(\boldsymbol{x}^*) = g_t(\boldsymbol{x}_t) + \nabla_t^\intercal (\boldsymbol{x}^* - \boldsymbol{x}_t) + \frac{1}{2}(\boldsymbol{x}^* - \boldsymbol{x}_t)^\intercal \nabla_t^2(\boldsymbol{\xi})(\boldsymbol{x}^* - \boldsymbol{x}_t). \tag{15}$$

By Lemma 4.1, if $\epsilon \geq H > 0$, then $\nabla_t^2(\boldsymbol{x}) \preceq -H\,\mathbf{I}, \forall \boldsymbol{x} \in \mathbb{R}^N$; thus,

$$2\left[g_t(\boldsymbol{x}^*) - g_t(\boldsymbol{x}_t)\right] \leq -2\nabla_t^\intercal (\boldsymbol{x}_t - \boldsymbol{x}^*) - H \left\|(\boldsymbol{x}_t - \boldsymbol{x}^*)\right\|_2^2. \tag{16}$$

We define two variables $\boldsymbol{d}_t = \boldsymbol{x}_t - \boldsymbol{x}^*$ and $d_t^2 = \|\boldsymbol{x}_t - \boldsymbol{x}^*\|^2$. Given $(\nabla_t - \boldsymbol{m}_t)^\intercal \boldsymbol{d}_t \geq 0$, the inequality (16) becomes

$$\begin{aligned} 2\left[g_t(\boldsymbol{x}^*) - g_t(\boldsymbol{x}_t)\right] &\leq -2\nabla_t^\intercal \boldsymbol{d}_t - H d_t^2 \\ &\leq -2\boldsymbol{m}_t^\intercal \boldsymbol{d}_t - H d_t^2. \end{aligned} \tag{17}$$

We need to find a good upper bound for the last expression in inequality (17).

By Lemma 4.2, we know $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \phi_{t+1}\boldsymbol{m}_t$; thus,

$$\begin{aligned} d_{t+1}^2 &= \left\|\mathrm{Proj}_{\triangle^N}(\boldsymbol{x}_t + \phi_{t+1}\boldsymbol{m}_t) - \boldsymbol{x}^*\right\|^2 \\ &\leq \|\boldsymbol{d}_t + \phi_{t+1}\boldsymbol{m}_t\|^2 \\ &= d_t^2 + 2\phi_{t+1}\boldsymbol{m}_t^T \boldsymbol{d}_t + \phi_{t+1}^2 \|\boldsymbol{m}_t\|^2, \end{aligned} \tag{18}$$

where $\mathrm{Proj}_{\triangle^N}(.)$ denotes the projection operation that maps its inputs into the space $\triangle^N$. Let a constant G be an upper bound of $\|\boldsymbol{m}_t\|_2$, i.e., $\|\boldsymbol{m}_t\|_2 \leq G$, so by the inequality (18), the last expression in inequality (17) is upper bounded by

$$-2\boldsymbol{m}_t^T \boldsymbol{d}_t - H d_t^2 \leq \left\{\frac{1}{\phi_{t+1}}(d_t^2 - d_{t+1}^2) - H d_t^2\right\} + \phi_{t+1} G^2, \tag{19}$$

where the bracketed part of the inequality (19) must not depend on d_t to make the sum of the right-hand side of the inequality (19) sublinearly bounded. Thus, the repeated terms in the sum have to be zeros by making ϕ_t and H solve the following equation:

$$\begin{aligned} &\frac{1}{\phi_{t+1}} d_t^2 - \frac{1}{\phi_t} d_t^2 - H d_t^2 = 0, \\ &\Leftrightarrow \frac{1}{\phi_{t+1}} - \frac{1}{\phi_t} - H = 0. \end{aligned} \tag{20}$$

If we treat difference equations as differential equations, the equation above means that $\frac{1}{\phi_t}$ is a linear function of t with a gradient H. A simple solution is

$$\frac{1}{\phi_{t+1}} = Ht;$$

thus, the sum of the inequality (17) is

$$\begin{aligned}\sum_{t=1}^{T}[g_t(\boldsymbol{x}^*) - g_t(\boldsymbol{x}_t)] &\leq \frac{G^2}{2}\sum_{t=1}^{T}\phi_{t+1} \\ &\leq \frac{G^2}{2H}\sum_{t=1}^{T}\frac{1}{t} \\ &\leq \frac{G^2}{2H}(1+\log T).\end{aligned} \tag{21}$$

By Lemma 4.2, the l^2-norm of $\boldsymbol{m}_t$ has an upper bound $2\sqrt{N}\rho$, which can be a choice for G. □

The condition $\epsilon \geq H > 0$ leads to a relatively high constant factor $1/H$ in the regret bound $2N\rho^2(1+\log T)/H$ according to Theorem 4.3. However, it enables us to regulate the wealth proportions $\boldsymbol{x}_t$ by using only first-order gradients ∇_t and $\boldsymbol{m}_t$, which can be easily obtained from neural network optimizers. If the condition $\epsilon \geq H > 0$ is relaxed, a better regret bound [21] can be obtained by using the second-order gradient of the evaluation function $g_t(\boldsymbol{x}) = \log(\boldsymbol{r}_t^\top \boldsymbol{x})$, where the computation of the second-order gradients has higher time complexity and numerical instability than the first-order gradients.

5 NEURAL NETWORK BASED ALGORITHM

In this section, we design a neural network for the WFM by using deep neural network architectures [23, 59]. The WFM neural network can learn high-dimensional features [45] for the prediction of wealth flow matrices directly from massive financial data, and it can also implement the goals of the WFM by using automatic differentiation frameworks [1] and graphics processing units. Based on the DRL paradigm [63], the main aspects of the WFM neural network are as follows:

- The negative of the objective function f_{t+1} defined in Equation (4) is used as the loss function J_{t+1} of the WFM neural network. J_{t+1} is different from the general value functions used in DRL; that is, the loss is recalculated for every trading period and not accumulated during the execution of our reinforcement learning algorithm. Using single-period losses can enable us to avoid the problem of unbounded value functions in Q-learning [20, 28] and to remove the dependencies between the Q-values [48] and action-values [42, 43] for different states.
- The wealth flow matrices $\boldsymbol{Y}$ are parameterized and vectorized as $\boldsymbol{y}(\boldsymbol{\theta})$. The parameter $\boldsymbol{\theta}$ is predicted by our reinforcement learning algorithm. We add a new loss term to the loss function J_{t+1}; that is, the difference between the profit obtained by using the best constant rebalanced portfolio strategy in the next k trading periods and the profit obtained by using $\boldsymbol{y}_{t+1}^{t+k}(\boldsymbol{\theta})$. The constraint $\mathrm{Tr}(\boldsymbol{Y})$ is approximated by adding a penalty item with a large weight to J_{t+1}. We also add a sparsity penalty item to J_{t+1}, since using the wealth proportions obtained from sparse wealth flow matrices can yield higher returns with lower risks.
- The WFM neural network is based on the recursive neural network architecture [23] for effectively predicting wealth flow matrices by learning high-dimensional hidden features [45] since short-term trends are stateful and relative price vectors are high-dimensional data. In the WFM neural network, we use the long short-term memory [23] neural cells optimized with an attention mechanism [57, 59] to capture the correlations of hidden feature vectors.

Because hidden feature vectors represent the relative price vectors of the hidden layers of the WFM neural network, the predictions of wealth flow matrices can be improved by using this attention optimization approach.

For the above reasons, the improved objective function $J_{t+1}(\boldsymbol{\theta})$ is

$$\min_{\boldsymbol{\theta}} J_{t+1}(\boldsymbol{\theta}) = -\log\left(\boldsymbol{r}_t^\top h_{t+1}\left(\boldsymbol{y}_{t+1}(\boldsymbol{\theta})\right)\right) + \frac{\epsilon}{2}\left\|h_{t+1}\left(\boldsymbol{y}_{t+1}(\boldsymbol{\theta})\right)\right\|^2 + L\left(\boldsymbol{y}_{t+1}^{t+k}(\boldsymbol{\theta})\right) + R\left(\boldsymbol{y}_{t+1}(\boldsymbol{\theta})\right), \tag{22}$$

where $\boldsymbol{y}_{t+1}(\boldsymbol{\theta}) = \mathrm{vec}(Y_{t+1}), \boldsymbol{y}_{t+1}(\boldsymbol{\theta}) \in \mathbb{R}^{N^2}$ is a wealth flow vector predicted by the WFM neural network for trading period $t+1$. The vectorization operator vec can rearrange the elements of a matrix $\boldsymbol{A}$ into a column vector $\boldsymbol{a}$ by the column-wise order of $\boldsymbol{A}$. The loss function $L(\boldsymbol{y}_{t+1}^{t+k}(\boldsymbol{\theta}))$ is defined by

$$L\left(\boldsymbol{y}_{t+1}^{t+k}(\boldsymbol{\theta})\right) = \mathbb{E}_{\{\boldsymbol{r}_\tau \sim \mathrm{p}(\boldsymbol{r}_\tau)\}}\left[max_{\boldsymbol{x}} \sum_{\tau=t+1}^{t+k} \log\left(\boldsymbol{r}_\tau^\top \boldsymbol{x}\right) - \sum_{\tau=t+1}^{t+k} \log\left(\boldsymbol{r}_\tau^\top h_\tau(\boldsymbol{y}_\tau(\boldsymbol{\theta}))\right)\right], \tag{23}$$

which is the expected difference between the forecasts generated by the WFM and those generated by using the best constant rebalanced portfolio strategy in k future trading periods. Since the relative price vectors $\boldsymbol{r}_\tau$ follow some unknown distribution $\mathrm{p}(\boldsymbol{r}_\tau)$, we use the historic relative vectors, which are stored in replay buffer, as an empirical distribution of $\mathrm{p}(\boldsymbol{r}_\tau)$ so that the loss function $L(\boldsymbol{y}_{t+1}^{t+k}(\boldsymbol{\theta}))$ can be estimated.

The regular item $R\left(\boldsymbol{y}_{t+1}(\boldsymbol{\theta})\right)$ in the new objective function $J_{t+1}(\boldsymbol{\theta})$ is utilized for the short-term goal and the constraints defined in the objective function $f_{t+1}(Y)$. That is,

$$\begin{aligned} R\left(\boldsymbol{y}_{t+1}(\boldsymbol{\theta})\right) = & \lambda \sum_{i=1,\ldots,N^2} [i \bmod (N+1) \neq 1] y_{t+1,i}(\boldsymbol{\theta}) \log\left(\frac{y_{t+1,i}(\boldsymbol{\theta})}{\hat{y}_{t,i}}\right) \\ & + \Psi_1 \sum_{i=1,\ldots,N^2} [i \bmod (N+1) = 1] y_{t+1,i}(\boldsymbol{\theta}) \\ & + \Psi_2 \left\|\boldsymbol{y}_{t+1}\right\|_1 \end{aligned}, \tag{24}$$

where the first part is the Kullback–Leibler divergence between the current estimation $\boldsymbol{y}_{t+1}$ and the sample $\hat{\boldsymbol{y}}_t$; $[a]$ is an indicator function: it is 1 if its parameter a is true; otherwise, it is zero; $\Psi_1, \Psi_2 \gg 0$ are large numbers for enforcing the constraint $\mathrm{Tr}(Y_{t+1}) = 0$ defined in Equation (4) and the sparsity of $\boldsymbol{y}_{t+1}$, respectively. The l^1-norm of $\boldsymbol{y}_{t+1}$ is used to measure the sparsity of $\boldsymbol{y}_{t+1}$. If the constraint $\mathrm{Tr}(Y_{t+1}) = 0$ is approximately satisfied by the second part of Equation (24), the constraint $\boldsymbol{Y1} = 1$ can be satisfied by applying the softmax function on the output $\boldsymbol{y}_{t+1}(\theta)$.

The data processing mechanism of the WFM neural network is given by the following series of tensor operations:

$$\boldsymbol{h}_t^1 = \boldsymbol{W}_1 \boldsymbol{r}_t + \boldsymbol{b}_1, \tag{25a}$$

$$\boldsymbol{H}_t^{j+1} = \mathrm{lstm}_a(\boldsymbol{H}_t^j), \tag{25b}$$

$$\boldsymbol{H}_t^{j+1} = \mathrm{BatchNorm} \circ \mathrm{Dropout}(\boldsymbol{H}_t^{j+1}),\ j = 1, \ldots, 2, \tag{25c}$$

$$\boldsymbol{y}_t = \boldsymbol{W}_y \mathrm{vec}\left([\boldsymbol{H}_t^1 \boldsymbol{e}_w, \boldsymbol{H}_t^2 \boldsymbol{e}_w, \boldsymbol{H}_t^3 \boldsymbol{e}_w]\right), \tag{25d}$$

where the jth matrix $\boldsymbol{H}^j = [\boldsymbol{h}_{t-w+1}^j, \ldots, \boldsymbol{h}_t^j]$ is a set of the last w feature vectors $\boldsymbol{h}_t^j \in \mathbb{R}^u$ sorted by their time-steps t. Each feature vector $\boldsymbol{h}_t$ is a high-dimensional representation of each relative price vector $\boldsymbol{r}_t$, as shown in Equation (25 a). There are three hidden layers for the three feature matrices $\boldsymbol{H}^j$. Equations (25a) to (25c) are used for nonlinear processing, which maps a relative price vector $\boldsymbol{r}_t$ to a feature vector $\boldsymbol{h}_t^j$. Equation (25d) is a linear transformation that outputs a wealth flow vector $\boldsymbol{y}_t$ from the three feature vectors $\boldsymbol{h}_t^j$ at the three hidden layers. Equations (25b) and (25c) indicate that neural cell lstm_a in the jth layer receives feature matrix $\boldsymbol{H}^j$ and outputs a new feature matrix $\boldsymbol{H}^{j+1}$ for the $(j+1)$th layer.

Inside a long short-term memory neural cell lstm_a, an attention mechanism [57, 59] is implemented to capture the correlations between the current feature vector $\boldsymbol{h}_t$ and the previous vectors $\boldsymbol{h}_{t-w+1}, \ldots, \boldsymbol{h}_{t-1}$, as follows:

$$b_{i,j} = (\boldsymbol{e}_i^\top \boldsymbol{H}) * \text{Kernel}_j,\ i = 1, \ldots, u, j = 1, \ldots, v, \tag{26a}$$

$$\alpha_i = \text{sigmoid}\left(\boldsymbol{e}_i^\top \boldsymbol{B} \boldsymbol{W}_\alpha \boldsymbol{h}_t\right), \tag{26b}$$

$$\boldsymbol{v}_t^\top = \boldsymbol{\alpha}^\top \boldsymbol{B}, \tag{26c}$$

$$\boldsymbol{h}_t = \boldsymbol{W}_s(\boldsymbol{W}_h \boldsymbol{h}_t + \boldsymbol{W}_v \boldsymbol{v}_t), \tag{26d}$$

where Kernel_j denotes the jth 1-D convolutional neural network filter. Equation (26) establishes a dynamic basis $\boldsymbol{B} \in \mathbb{R}^{u \times v}$ based on v kernels [57]. That is, the ith row of $\boldsymbol{B}$ contains the weights of k kernels for representing the ith row vector of $\boldsymbol{H}$. By using the dynamic basis $\boldsymbol{B}$, the correlations between the current feature vector $\boldsymbol{h}_t$ and the recent $w-1$ feature vectors $\boldsymbol{h}_{t-w+1}, \ldots, \boldsymbol{h}_{t-1}$ can be computed by Equation (26c). Different from the general attention mechanism [59], here, the correlation is measured on the feature vectors in a component-wise manner. This can enrich the information of the new feature vector $\boldsymbol{h}_t$ defined in Equation (26d).

As a summary of the above design concerns, we present a deep reinforcement learning algorithm, shown in Algorithm 1, for the WFM neural network defined by Equations (25) and (26). Steps (7–12) check the conditions defined by Theorem 4.3 and scale the gradient $\boldsymbol{m}_t$ to update $\boldsymbol{x}_{t+1}$ if necessary. Steps (14) and (18) actively update the WFM neural network, since a predictor deviates from its goals if it does not receive correct feedback in a timely manner. We give an estimation of the time complexity of Algorithm 1. Most of the time costs are incurred in Steps (3) and (17). If we model the operations of the neural network as a chain of matrix multiplications, then the forward-pass and back-propagation processes have the same time complexity scale. Thus, we compute the time complexity for the forward-pass process of the WFM neural network $\boldsymbol{y}(\boldsymbol{\theta}; u, v, w)$. For each epoch, according to Equations (25a), (25c), and (25d), we know that the complexity is $O(u(N + 3w + 3N^2) + 3X)$, where X is the complexity of the long short-term memory cell. According to Equation (26), we know that $X = O(uvw + uvd_\alpha^c + 2uv + uv + 2u^2 d_s^c)$, where d_α^c and d_s^c are the column dimensions of $\boldsymbol{W}_\alpha$ and $\boldsymbol{W}_s$, respectively. Therefore, the time complexity of the WFM neural network is $O\left(n_t l u \left(N + 3w + 3N^2 + 3(vw + vd_\alpha^c + 3v + 2ud_s^c)\right)\right)$, where n_t is the number of training samples of relative price vectors and l is the number of epochs.

6 EXPERIMENTS

This section presents two kinds of experiments to analyze the performance of the WFM from two aspects. The first is the analysis of the internal behavior of the WFM based on the hidden-layer outputs of the WFM neural network, namely, the wealth flow matrices. The second involves the performance comparisons between the WFM and other algorithms based on the final-layer outputs of the WFM neural network, namely, the wealth proportions.

6.1 Behavioral Experiment

The behavioral experiment studies the behavioral characteristics [55] of the WFM neural network. We need to see whether the WFM neural network has learned the latent structures of the given portfolios via wealth flow matrices. Based on the learned latent structures, the WFM neural network can generate reasonable investment behavior through wealth proportions.

ALGORITHM 1: A deep reinforcement learning algorithm for the WFM neural network

Input:

—the WFM neural network $\boldsymbol{y}(\boldsymbol{\theta}; u, v, w)$, where $\boldsymbol{\theta}$ are the weights of the WFM neural network, u is the dimension of hidden features, v is the number of convolutional kernels and w is the window length of input relative price vectors

—window length k used by loss function L

—learning rate ϕ for $\boldsymbol{Y}$ and $\boldsymbol{x}$

—penalty coefficient λ for stable prediction

—regular parameter ϵ for uniform allocation

—penalty coefficients Ψ_1, Ψ_2 for the diagonal-zeros and sparsity constaints, respectively

—number of epochs l, and minibatch size n

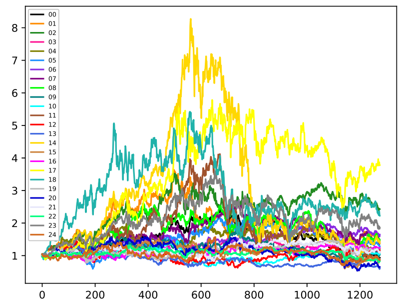
—replay buffer $\mathcal{R}$

Output: wealth proportion vectors $\boldsymbol{x}_{t+1}$

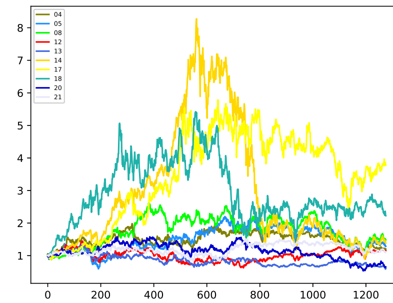
begin
  while $t < T$ **do**
    make prediction $\boldsymbol{y}_{t+1} = \boldsymbol{y}(\boldsymbol{\theta}; u, v, w)$
    compute $\nabla_t = \nabla g_t(\boldsymbol{x}_t)$ and $\boldsymbol{m}_t$ by Equation (12)
    estimate $\hat{\boldsymbol{x}}^*$ from $\mathcal{R}$
    let $\boldsymbol{d}_t = \boldsymbol{x}_t - \hat{\boldsymbol{x}}^*$
    if $(\nabla_t - \boldsymbol{m}_t)^\top \boldsymbol{d}_t \geq 0$ **then**
      output $\boldsymbol{x}_{t+1}$ by Equation (5)
    else
      let $\boldsymbol{m}_t = \nabla_t$ and update ϕ_{t+1} by $\phi_t(t-1)/t$
      output $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \phi_{t+1}\boldsymbol{m}_t$
    end
    put $(\boldsymbol{r}_t, \boldsymbol{y}_t)$ in $\mathcal{R}$
    if $|\mathcal{R}| >= k$ **then**
      for 1 **to** l **do**
        sample a sample of $\{\boldsymbol{r}_i, \boldsymbol{y}_i\}$ of length k from $\mathcal{R}$
        update $\boldsymbol{\theta}$ by Equation (22), (23) and (24)
      end
    end
    $t = t + 1$
  end
end

We randomly choose a subset of the SP500 dataset defined in Table 2 as inputs and capture the wealth flow matrices $\boldsymbol{y}_t$ located at the hidden layer of the WFM neural network. Some of $\boldsymbol{y}_t$ are shown in Figure 2. Figure 2(a) contains the price time series of the 25 stocks of the SP500 dataset. Figure 2(b) is a subset of the 25 stocks. The ten stocks are selected according to the columns of the four wealth flow matrices visualized from Figure 2(c)–2(f). These ten stocks receive considerable income flows from other assets.

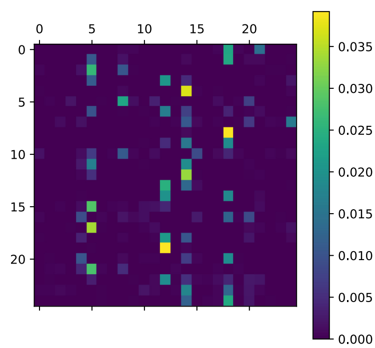
— **Short-term trend following.** Four examples of wealth flow matrices show that the prior structure of a wealth flow matrix can capture meaningful short-term market trends. On day 200, the 14th and 18th stocks were the top 2 stocks that received the most income. In particular, the fourth and eighth stocks moved much of their money to the 14th and 18th stocks, since the fourth and eighth stocks were relatively weak stocks, and the 14th and 18th were the two strongest stocks, as shown in Figure 2(b). This is a trend-following behavior.



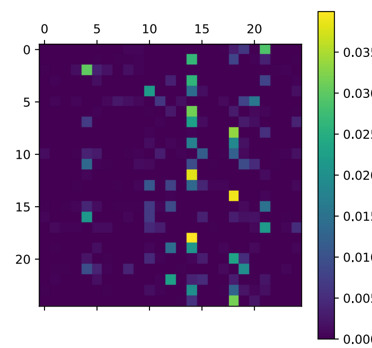
(a) SP500's 25 stocks



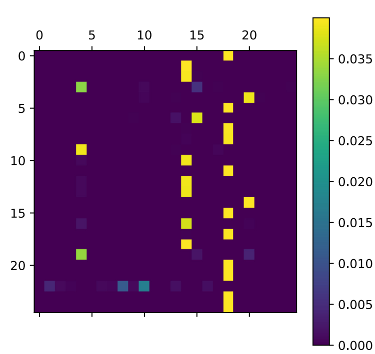
(b) Ten of the 25 stocks



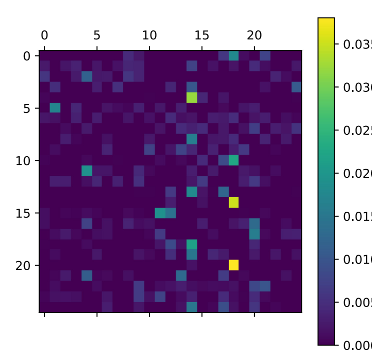
(c) day 200



(d) day 201



(e) day 202



(f) day 203

Fig. 2. The evolution of the wealth flow matrix during days 200–203.

As time progressed, additional weak stocks transferred their money into the 14th and 18th stocks in the short-term future. This group behavior can be observed from the evolution of the wealth flow matrix from days 200–203. On day 202, the 14th and 18th stocks received almost all the money held by other stocks. Note that we need to sum the 14th and 18th columns of the matrices for days 200 to 203 to calculate the cumulative money given to the 14th and 18th stocks. From days 200–203, the amount of money flowing into the 18th stock was more than that of the 14th stock. In particular, there were two continuously enhanced output flows from the 14th stock to the 18th stock (since the 18th stock was stronger than the 14th stock), as shown in Figure 2(b). This trend-following behavior reached a temporary steady state on day 203, and this can be seen in Figure 2(f). There are almost no active flows (very few yellow patches) in Figure 2(f).

— **Following a few losers.** There are a few exceptions that are not trend-following behaviors. For example, the fifth stock was a relatively weak stock but received some money from other stocks, such as the 2nd, 3rd, and 17th stocks, where the 17th was a relatively strong stock, as shown in Figure 2(b). An explanation for this is that on day 200, the price of the fifth stock reached at a local minimum, and the WFM neural network used its policy of following a few losers. From day 200 onwards, the fifth stock began to grow steadily, although not very strongly. We can also see the loser-following behaviors of element $y_{14,20}$ of the wealth flow matrix on day 202 and the 20th column of the wealth flow matrix on day 202. The loser-following behavior may be caused by our gradient Equation (13) for updating the elements of $\boldsymbol{Y}$. The gradient equation can favor a mean reversion investment behavior, if the numerator layout of a scale-by-matrix derivative is used for computing the gradient.

— **No self-investment.** Note that we cannot see any self-investment behavior in the four examples of the wealth flow matrix. This is the result of the constraint $\mathrm{Tr}(\boldsymbol{Y}) = 0$ defined in Equation (4). The constraint is also presented in the new objective function $J(\boldsymbol{\theta})$ in Equation (24). Since self-investment is forbidden by our algorithm, the wealth flows among assets are strengthened and quickly respond to changes in the values of the objective function $J(\boldsymbol{\theta})$. For example, element $y_{5,18}$ on day 202 shows that the fifth stock immediately transferred the money it received to the 18th stock.

— **Sparse portfolios.** Given randomly selected stocks in any dataset, the WFM neural network is supposed to choose a small number of stocks among them. For example, when the wealth matrix reached a steady state, as shown in Figure 2(e), only two stocks, namely, the 14th and 18th stocks, received more than 80% of the total wealth of the portfolio. This is a sparse portfolio investment behavior, which enables the WFM neural network to obtain high returns with low risks. Sparse portfolios are caused by the structural sparsity defined in Equation (24). Note that the regular condition $\Psi_2 \left\|\boldsymbol{y}_{t+1}\right\|_1$ alone is not enough to obtain sparse portfolios, as shown in Figure 2(e), since it is a specification, not the full implementation. We adopt the dropout function provided by the TensorFlow framework [1] to generate wealth flow matrices with explicit structural sparsity.

As mentioned above, we conclude that the WFM neural network can effectively learn the latent structure of portfolios, i.e., the relative strengths of the assets in portfolios. The reasonable investment behavior obtained from the learned latent structure include short-term trend following, the following of a few losers, no self-investment, and sparse portfolios.

6.2 Comparison Experiments

This section presents several performance comparisons between the WFM and other algorithms from different aspects, i.e., Pareto improvement, improvement of the support number, clustering effects, annualized returns under different transaction costs and wealth dynamics.

6.2.1 Indicators. We use four indicators to compare the performances of the various algorithms: the **annualized return (AR)**, **Sharpe ratio (SR)**, **maximum drawdown (MDD)**, and turnover.

Table 2. Summary of Datasets

Dataset	Stocks	Duration	Days
NYSE(O)	36	[1962.07.03, 1984.12.31]	5,651
NYSE(N)	23	[1985.01.01, 2010.06.30]	6,431
DJIA	30	[2001.01.14, 2003.01.14]	507
SP500	25	[1998.01.02, 2003.01.31]	1,276
TSE	88	[1994.01.04, 1998.12.31]	1,259

An AR, which is the geometric average amount of money earned by an investment each year over a given time period, is calculated as:

$$\text{AR} = \left(1 + CR^{\frac{365}{\text{Days Held}}}\right) - 1,$$

in which CR is the cumulative return defined by $CR = \frac{w_T - w_1}{w_1}$, where w_T is the final wealth and w_1 is the initial wealth of the portfolio.

The SR, which is a measure of the risk-adjusted return of a portfolio, is:

$$\text{SR} = \frac{R_p - R_f}{\sigma_p},$$

where R_p is the expected rate of return; R_f is the risk-free rate of return; and σ_p is the standard deviation of the portfolio. The default value for R_f here is 0.02, which is approximately one year's rate of return on the deposit of cash in a bank.

Another more direct measure of risk is the MDD indicator, which measures the maximum cumulative loss from a market peak to the following trough. It can be used to indicate the extent to which a person can sustain losses. It is defined by

$$\text{MDD} = \underset{t>\tau}{max} \frac{w_t - w_\tau}{w_t}.$$

Turnover is a measure of the cumulative change in wealth proportion vectors during the trading periods, and it is defined as follows:

$$\text{Turnover} = \sum_{i=2}^{T} \left\| \boldsymbol{x}_i - \boldsymbol{x}_{i-1} \right\|_1 .$$

A high turnover generates more commissions on trades placed by a broker.

It is very challenging for an algorithm to achieve high rankings on all these indicators simultaneously, since they are conflicting indicators. For instance, in a case with general price sequences, there is no algorithm that can obtain the maximum AR and the minimum turnover simultaneously.

6.2.2 Test Datasets. The performance experiments are conducted on five benchmark datasets[2] from real-world stock markets: NYSE(O) [15] and NYSE(N) [38] from the New York stock exchange (NYSE), DJIA [9] from the Dow Jones industrial average (DJIA), SP500 [9] from the standard and poor's 500 (S&P500) and TSE [9] from the Toronto stock exchange (TSE). The information about these datasets is listed in Table 2.

The records of each of these datasets are relative price vectors represented by the ratios of prices between two adjacent trading days. The precise definition of a relative price vector is given in Section 3. There are different trends exhibited by these datasets. For instance, by observing the BAH (buy-and-hold) curves in Figure 6, we know that the overall trends of the NYSE(O) and NYSE(N) datasets are bull markets, while the SP500 and DJIA datasets are bear markets. The time

[2]All datasets can be downloaded from https://github.com/OLPS/OLPS.

spans of these datasets are different; the NYSE(N) datasets has the longest time span. The numbers of stocks (features) in these datasets vary, with the TSE dataset having the largest number of stocks. These characteristics of datasets pose challenges to online portfolio selection algorithms.

6.2.3 Parameter Settings. The parameter settings of different algorithms are listed below, and they are used to run these algorithms on the five datasets described in Section 6.2.2.

- WFM: Our WFM neural network-based online training algorithm defined in Algorithm 1. The window length k is set to 5, the learning rate ϕ is set to 1E-06, the penalty coefficient λ for tracking a previous wealth flow matrix is set to 1E02 and the regular parameter ϵ is set to 1E-5. The penalty coefficients Ψ_1 and Ψ_2 for diagonal zeros and sparsity, respectively, are set to 1E07. The number of epochs l for repeating the training process is set to 1,000.
- BAH: The market strategy, i.e., the buy-and-hold (BAH) approach [37].
- BCRP: The **best constant rebalanced portfolio (BCRP)** in hindsight. A benchmark algorithm [7, 15].
- EG: The **exponential gradient (EG)** algorithm with the parameter η set to 0.05 as recommended in [22].
- ONS: The **online Newton step (ONS)** with the parameter setting $\eta = 0$, $\beta = 1$, and $\gamma = 1/8$ as recommended in [2].
- ANTICOR: **The anticorrelation (ANTICOR)** algorithm [9], window length is set to 30.
- OLMAR: The **online moving average reversion (OLMAR)** strategy [36] with the parameter setting $\epsilon = 10, \alpha = 0.5$ as recommended in [39].
- PAMR: The passive aggressive mean reversion (PAMR) strategy with the parameter setting $\epsilon = 0.5, C = 500$ as recommended in [41].

We build our WFM neural network by using Python 3.5 with the TensorFlow framework 1.5 and train it online by using the Nvidia GeForce RTX 2080Ti GPU. We use an open source portfolio benchmark tool[3] to run the WFM neural network and the other algorithms.

6.2.4 Summary of Indicators. The values of indicators of all algorithms running on different datasets are presented in Table 3, in which the best results are marked in bold.

The WFM algorithm takes first place on the AR, SR, and MDD indicators for all datasets, except for the SP500 dataset. For instance, the ARs achieved on the WFM algorithm are 2.00, 1.58, 2.27, and 1.37 times higher than those of the second place algorithms. On the SP500 dataset, the AR of the WFM algorithm is 0.89 times lower than that of ANTICOR. However, the SR of the WFM algorithm is 1.93 times higher than that of ANTICOR, and the MDD and turnover of the WFM algorithm are 0.60 and 0.14 times lower than those of ANTICOR, respectively. This means that the WFM algorithm has a much lower risk than ANTICOR when the returns of both are approximately equal.

6.2.5 Pareto Improvement. To investigate the relationship between the returns and costs of the tested algorithms, we use the ARs and turnovers of all algorithms as data to draw a two-dimensional view, as shown in Figure 3. With the indicators AR and turnover considered, algorithm $\mathcal{A}$ is a Pareto improvement [44] of algorithm $\mathcal{B}$, and this denoted by a generalized inequality $\mathcal{A} \geq \mathcal{B}$ if the following condition is met:

$$\mathcal{A}.\text{AR} \geq \mathcal{B}.\text{AR} \wedge \mathcal{A}.\text{Turnover} \leq \mathcal{B}.\text{Turnover} \\ \wedge (\mathcal{A}.\text{AR} \neq \mathcal{B}.\text{AR} \vee \mathcal{A}.\text{Turnover} \neq \mathcal{B}.\text{Turnover}).$$

The visual representation of the above conditions suggests that data point $\mathcal{A}$ must be above and left of data point $\mathcal{B}$. For instance, data point O_w (i.e., the WFM algorithm on the NYSE(O) dataset)

[3]The portfolio benchmark tool is at https://github.com/Marigold/universal-portfolios.

Table 3. Summary of Indicators AR, SR, MDD, and Turnover (Transaction Costs=0.1%)

Dataset	Indicators	BAH	BCRP	WFM	ANTICOR	EG	OLMAR	ONS	PAMR
NYSE(O)	AR(%)	12.56	27.15	**605.82**	43.76	15.44	303.05	21.72	227.55
	SR	0.79	0.78	**5.35**	0.87	1.06	2.55	1.08	2.41
	MDD(%)	41.72	68.29	**16.26**	80.5	37.05	49.54	27.82	48.03
	Turnover	66.1	0.0	2929.2	3046.7	**3.3**	7608.8	243.7	9534.7
NYSE(N)	AR(%)	12.04	20.30	**93.31**	33.49	14.11	59.07	14.20	13.65
	SR	0.63	0.74	**2.87**	0.65	0.69	0.83	0.32	0.27
	MDD(%)	53.59	48.58	**40.26**	80.35	64.01	96.46	88.63	90.74
	Turnover	67.6	0.0	1359.1	2863.5	**3.9**	8002.2	170.2	10786.2
DJIA	AR(%)	−12.59	11.45	**59.39**	25.06	−10.40	7.06	18.82	−45.41
	SR	−0.56	0.45	**1.73**	0.48	−0.43	0.13	0.51	−1.31
	MDD(%)	39.29	23.03	**19.59**	45.98	38.10	55.84	34.46	85.64
	Turnover	7.0	0.0	101.2	203.1	**0.4**	649.6	61.5	817.6
SP500	AR(%)	5.82	31.40	34.86	**39.29**	9.62	26.02	23.91	−8.75
	SR	0.23	0.64	**1.33**	0.69	0.42	0.44	0.89	−0.19
	MDD(%)	45.80	51.41	31.58	52.32	32.03	50.26	**24.43**	67.93
	Turnover	20.6	0.0	83.8	582.3	**1.0**	1646.2	92.7	2070.6
TSE	AR(%)	9.79	45.40	**145.99**	50.59	9.05	64.45	5.29	106.68
	SR	0.72	0.99	**4.39**	0.66	0.66	0.55	0.15	1.06
	MDD(%)	30.02	48.42	**22.49**	59.39	33.61	88.40	51.95	70.01
	Turnover	17.0	0.0	214.5	650.4	**0.9**	1587.8	92.5	1911.1

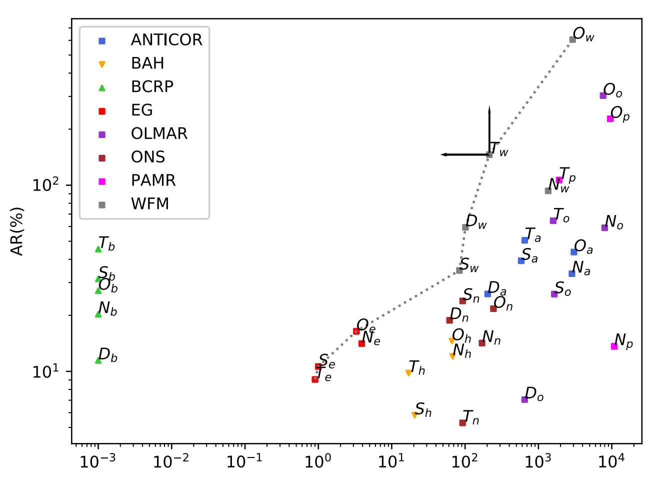


Fig. 3. AR-turnover data points. Each data point has a name in the form X_y, where $X \in \{O, N, D, S, T\}$ denotes the NYSE(O), NYSE(N), DJIA, SP500, and TSE datasets, respectively, and $y \in \{h, b, w, a, e, o, n, p\}$ denotes the BAH, BCRP, WFM, ANTICOR, EG, OLMAR, ONS, and PAMR algorithms, respectively. A small turnover of 0.001 is added to BCRP on each dataset so that data points of BCRP, e.g., T_b, S_b, can be shown in this logarithmic coordinates system.

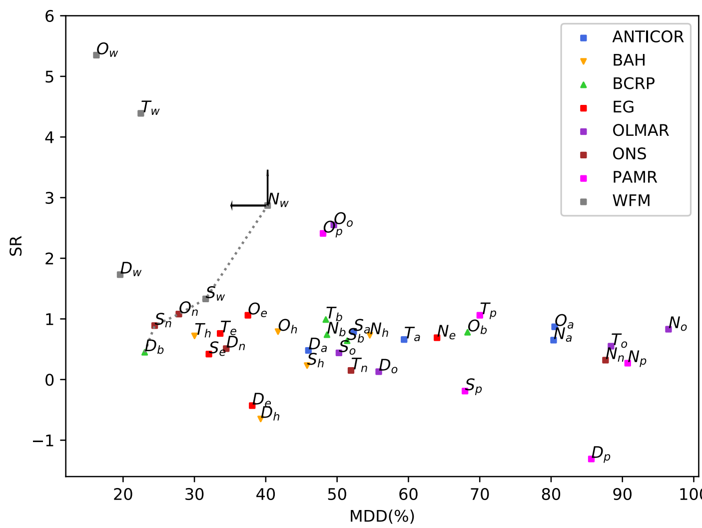


Fig. 4. SR-MDD data points.

is a Pareto improvement of data points O_o (i.e., the OLMAR algorithm on the NYSE(O) dataset), O_p (i.e., the PAMR algorithm on the NYSE(O) dataset), and O_a (i.e., the ANTICOR algorithm on the NYSE(O) dataset). Pareto improvements can also be found across different datasets. For instance, the relationships $T_w \geq O_a, T_w \geq N_o, T_w \geq D_o$ and $T_w \geq S_a$ indicate that data point T_w of the WFM algorithm on the TSE dataset is a Pareto improvement of data points O_a, N_o, D_o, and S_a.

The dotted line starting from data point T_e (i.e., the EG algorithm on the TSE dataset) to data point D_w (i.e., the WFM algorithm on the DJIA dataset) is a Pareto frontier of all points except data points O_b, N_b, D_b, S_b and T_b of the BCRP algorithm. The WFM algorithm has four data points S_w, D_w, T_w, and O_w on the Pareto frontier. The remaining data point N_w of the WFM algorithm is improved only by the data point T_w of itself, that is, $\forall X : X \geq N_w \implies X = T_w$.

Similarly, to investigate the relationship between the returns and risks of the tested algorithms, we use the SRs and MDDs of all algorithms as data to draw a two-dimensional view, as shown in Figure 4. The dotted line starting from data point D_b (i.e., the BCRP algorithm on the DJIA dataset) to data point N_w (i.e., the WFM algorithm on the NYSE(N) dataset) is a Pareto frontier of all points except data points D_w, T_w, and O_w of the WFM algorithm. Data points D_w, T_w, and O_w are superior to the other data points in terms of their SRs and MDDs. This means that the WFM algorithm achieves high returns while maintaining low risk.

6.2.6 Improvement of Support Numbers. We define an indicator called the support number $c_{\mathcal{A}}$ of the $\mathcal{A}$ algorithm as follows:

$$s_{\mathcal{A}} = \left|X_y : X_{\mathcal{A}} \geq X_y, X_y[0] > 0\right|,$$

where $X \in \{O, N, D, S, T\}$ denotes the NYSE(O), NYSE(N), DJIA, SP500, and TSE datasets, respectively, and $y \in \{h, b, w, a, e, o, n, p\}$ denotes the BAH, BCRP, WFM, ANTICOR, EG, OLMAR, ONS, and PAMR algorithms, respectively.

The support number of the $\mathcal{A}$ algorithm measures how many data points support the probability that $\mathcal{A}$ yields a Pareto improvement. Following the definition of the support number, we obtain the results presented in Table 4, in which the best results are marked in bold. For the AR and turnover indicators, the WFM algorithm achieves first place if the unreal algorithm BCRP is excluded. For the SR and MDD indicators, the WFM algorithm places first. Because the definition

Table 4. Support Numbers $s_{\mathcal{A}}$ of Different Algorithms

Indicator pairs	Support numbers	BAH	BCRP	WFM	ANTICOR	EG	OLMAR	ONS	PAMR
(AR, Turnover)	$s_{\mathcal{A}}$	4	25	18	6	9	6	7	4
	$s_{\mathcal{A}}/\max\{s_{\mathcal{A}}\}$	0.16	**1.00**	0.72	0.24	0.36	0.24	0.28	0.16
(SR, MDD)	$s_{\mathcal{A}}$	19	18	35	11	22	16	26	18
	$s_{\mathcal{A}}/\max\{s_{\mathcal{A}}\}$	0.54	0.51	**1.00**	0.31	0.63	0.46	0.74	0.51

Table 5. Clusters of AR-Turnover Data Points

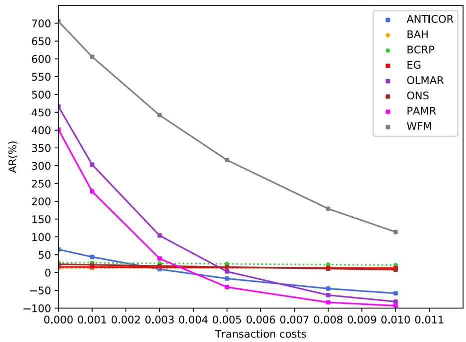
No. cluster	Members
1	$\{O_e, N_e, S_e, T_e\}$
2	$\{O_h, N_h, O_n, S_n, D_n, N_n, D_a, S_w\}$
3	$\{O_a, N_a, S_a, T_a, S_o, T_o, N_o, N_w, T_p\}$
4	$\{O_o, O_p, O_w\}$

of the support number considers the AR and turnover (or the SR and MDD) indicators equally, the risk-conservative algorithms EG and ONS obtain higher rankings than those of the risk-aggressive algorithms ANTICOR, OLMAR, and PAMR.

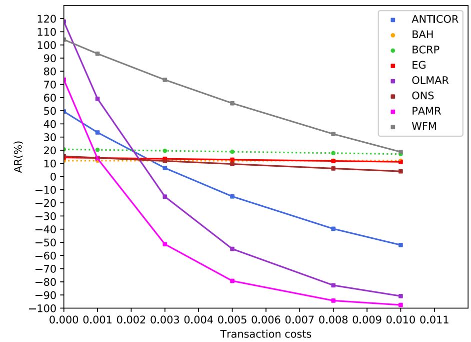
6.2.7 Clustering Effects. The AR-turnover data points of the ANTICOR, ONS, EG, and BAH algorithms demonstrate the resulting clustering effects in Figure 3. For instance, data points with the same suffix a (i.e., the ANTICOR algorithm) are close to each other. The data points in Figure 3 can be roughly divided into four clusters, which are presented in Table 5. It can be inferred that the EG and ONS algorithms are similar to the BAH algorithm regarding their turnover behaviors. The ONS and EG algorithms are not sensitive to changes in market trends, and experience few changes in their turnovers, so their earnings tend to be conservative and fluctuate little. The ANTICOR and OLMAR algorithms behave similarly, but OLMAR is more aggressive on the turnover indicator. Thus, the AR of the OLMAR algorithm is not as stable as that of the ANTICOR algorithm in bear markets, as shown in the DJIA dataset. The WFM algorithm is different from the other algorithms in that the AR-turnover data points of the WFM algorithm are outliers in Figure 3. This means that the WFM algorithm is effective in obtaining higher ARs with lower turnover costs. By a similar analysis as that done with Figure 4, the WFM algorithm can obtain high returns in terms of SRs with low risks of MDDs.

6.2.8 Annualized Returns Under Different Transaction Costs. If we use different transaction costs as weights to scale the turnover indicator, it is possible to transform two noncomparable data points, such as those in Figure 3, into comparable ones. Thus, we measure the annualized returns of all algorithms under different transaction costs $c \in [0\%, 1\%]$, where $c = 1\%$ is a rather high cost rate for stock trading. The results shown in Figure 5 indicate that the WFM algorithm outperforms the ANTICOR, EG, OLMAR, ONS, and PAMR algorithms when the transaction costs are above 0.1%. When the transaction costs are lower than 0.1%, the ARs of the WFM algorithm are slightly less than those of the OLMAR, ANTICOR, and PAMR algorithms on the NYSE(N), SP500, and TSE datasets. However, as the transaction costs increase, the ARs of the OLMAR, ANTICOR, and PAMR algorithms decline approximately exponentially and quickly fall below zero, whereas the ARs of the WFM algorithm decline approximately linearly and remain positive. Since 0.1% is a reasonable transaction cost [35], the WFM algorithm is applicable for real-world financial transactions.

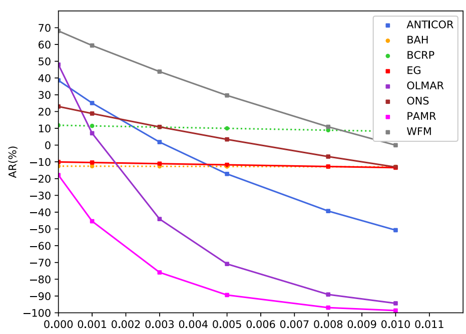
6.2.9 Wealth Dynamics. The wealth dynamics of all algorithms are shown in Figure 6. The wealth of the WFM algorithm exhibits steady growth on the NYSE(O), NYSE(N), TSE, and DJIA datasets. When the markets show bear trends, the wealth of the WFM algorithm is less affected.



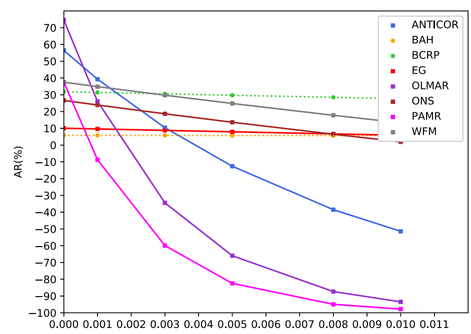
(a) NYSE(O)



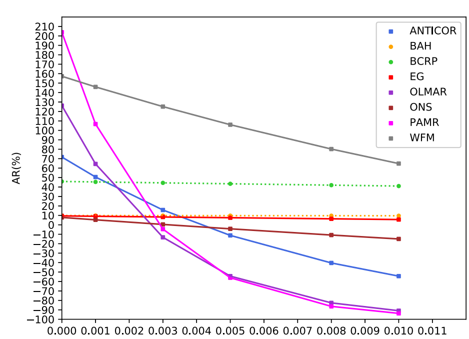
(b) NYSE(N)



(c) DJIA



(d) SP500



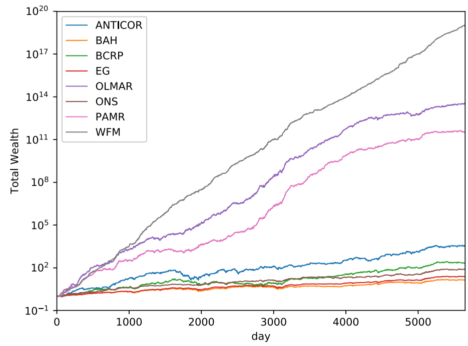
(e) TSE

Fig. 5. Annualized returns (ARs) under different transaction costs.

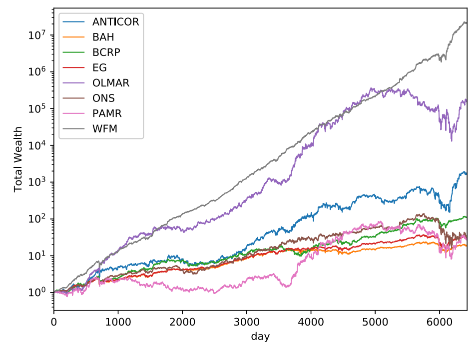
This means that the WFM neural network has learned the effective latent structure for representing the relative strengths of the assets, as analyzed in Section 6.1. The wealth proportions derived from using the knowledge of the relative strengths of the assets are different from those found by searching for historical price data online and are able to better adapt to changes in market trends.

7 CONCLUSION

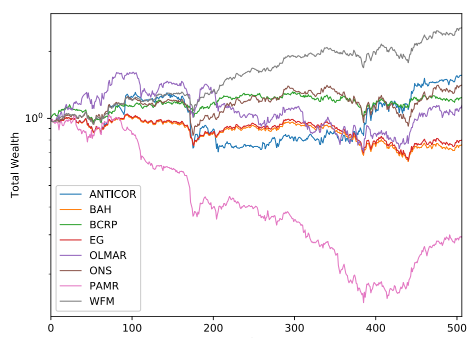
This article proposes a novel wealth flow matrix for representing a latent structure that encodes knowledge about the relative strengths of assets and a WFM for learning wealth flow matrices



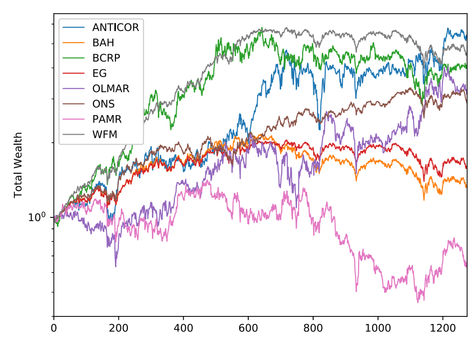
(a) NYSE(O)



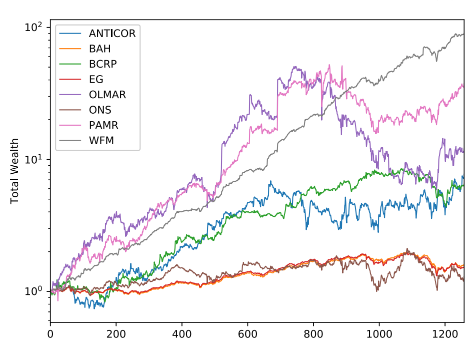
(b) NYSE(N)



(c) DJIA



(d) SP500



(e) TSE

Fig. 6. Total wealth of different algorithms.

and maximizing portfolio wealth simultaneously. In the WFM, wealth flow matrices are learned by using the principle of the variational Bayesian method; the exploitation of wealth flow matrices and the exploration of wealth growth are integrated by using regular conditions. The WFM is implemented by our recursive neural network and trained online with our algorithm based on deep reinforcement learning.

Through behavioral experiments, we find that the WFM neural network can effectively learn latent structures, and this leads to reasonable investment behavior including short-term trend following, the following of few losers, no self-investment, and sparse portfolios. Through extensive experiments on five benchmark datasets from real-world stock markets, we find that the WFM neural network achieves Pareto improvements on multiple performance indicators and the steady growth of wealth over the state-of-the-art algorithms.

In the future, we will further understand the behavior of the WFM neural network and improve its prediction ability by including more effective training algorithms and neural cells.

REFERENCES

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng. 2016. Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation.* 265–283.

[2] A. Agarwal, E. Hazan, S. Kale, and R. E. Schapire. 2006. Algorithms for portfolio management based on the newton method. In *Proceedings of the 23rd International Conference on Machine Learning.* 148, 9–16.

[3] I. Aldridge. 2019. Big data in portfolio allocation: A new approach to successful portfolio optimization. *The Journal of Financial Data Science* 1, 1 (2019), 45–63.

[4] S. Arora, E. Hazan, and S. Kale. 2012. The multiplicative weights update method: A meta-algorithm and applications. *Theory of Computing* 8, 1 (2012), 121–164.

[5] Y. Bai, J. Yin, S. Ju, Z. Chen, and J. Z. Huang. 2020. Long and short term risk control for online portfolio selection. In *Proceedings of the International Conference on Knowledge Science, Engineering and Management.* Springer, 472–480.

[6] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. 2017. Variational inference: A review for statisticians. *Journal of the American Statistical Association* 112, 518 (2017), 859–877.

[7] A. Blum and A. Kalai. 1999. Universal portfolios with and without transaction costs. *Machine Learning* 35, 3 (1999), 193–205.

[8] E. Bodewig. 2014. *Matrix Calculus.* Elsevier.

[9] A. Borodin, R. El-Yaniv, and V. Gogan. 2004. Can we learn to beat the best stock. *Journal of Artificial Intelligence Research* 21, 1 (2004), 579–594.

[10] X. Cai and Z. Ye. 2019. Gaussian weighting reversion strategy for accurate online portfolio selection. *IEEE Transactions on Signal Processing* 67, 21 (2019), 5558–5570.

[11] E. Cambria. 2016. Affective computing and sentiment analysis. *IEEE Intelligent Systems* 31, 2 (2016), 102–107.

[12] W. Chen, D. Li, S. Lu, and W. Liu. 2018. Multi-period mean–semivariance portfolio optimization based on uncertain measure. *Soft Computing* 23, 15 (2019), 6231–6247.

[13] G. Chu, W. Zhang, G. Sun, and X. Zhang. 2019. A new online portfolio selection algorithm based on Kalman Filter and anti-correlation. *Physica A: Statistical Mechanics and Its Applications* 536, 1 (2019), 120949.

[14] M. Colombino, E. Dall'Anese, and A. Bernstein. 2019. Online optimization as a feedback controller: Stability and tracking. *IEEE Transactions on Control of Network Systems* 7, 1 (2019), 422–432.

[15] T. M. Cover. 1991. Universal portfolios. *Mathematical Finance* 1, 1 (1991), 1–29.

[16] T. M. Cover and E. Ordentlich. 1996. Universal portfolios with side information. *IEEE Transactions on Information Theory* 42, 2 (1996), 348–363.

[17] P. Das, N. Johnson, and A. Banerjee. 2014. Online portfolio selection with group sparsity. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence.* 1185–1191.

[18] S. J. Gershman and Y. Niv. 2010. Learning latent structure: Carving nature at its joints. *Current Opinion in Neurobiology* 20, 2 (2010), 251–256.

[19] László Gyorfi and Harro Walk. 2012. Empirical portfolio selection strategies with proportional transaction costs. *IEEE Transactions on Information Theory* 58, 10 (2012), 6320–6331.

[20] Van Hasselt H., A. Guez, and D. Silver. 2016. Deep reinforcement learning with double q-learning. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence.* 2094–2100.

[21] E. Hazan, A. Agarwal, and S. Kale. 2007. Logarithmic regret algorithms for online convex optimization. *Machine Learning* 69, 2–3 (2007), 169–192.

[22] D. P. Helmbold, R. E. Schapire, Y. Singer, and M. K. Warmuth. 1998. On-line portfolio selection using multiplicative updates. *Mathematical Finance* 8, 4 (1998), 325–347.

[23] S. Hochreiter and J. Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9, 8 (1997), 1735–1780.

[24] N. Hu, G. Englebienne, Z. Lou, and B. Kröse. 2014. Learning latent structure for activity recognition. In *Proceedings of the 2014 IEEE International Conference on Robotics and Automation.* IEEE, 1048–1053.

[25] D. Huang, Y. Zhu, B. Li, S. Zhou, and S. C. Hoi. 2015. Semi-universal portfolios with transaction costs. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence.*

[26] B. Hurst, Y. H. Ooi, and L. H. Pedersen. 2017. A century of evidence on trend-following investing. *The Journal of Portfolio Management* 44, 1 (2017), 15–29.

[27] T. Jaksch, R. Ortner, and P. Auer. 2010. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research* 11, Apr (2010), 1563–1600.

[28] G. Jeong and H. Y. Kim. 2019. Improving financial trading decisions using deep Q-learning: Predicting the number of shares, action strategies, and transfer learning. *Expert Systems with Applications* 117, 1 (2019), 125–138.

[29] Y. Jiang, H. Nie, and W. Ruan. 2018. Time-varying long-term memory in Bitcoin market. *Finance Research Letters* 25, 1 (2018), 280–284.

[30] Z. Jiang and J. Liang. 2017. Cryptocurrency portfolio management with deep reinforcement learning. In *Proceedings of the 2017 Intelligent Systems Conference*. IEEE, 905–913.

[31] H. Kaplan, D. Naori, and D. Raz. 2020. Competitive analysis with a sample and the secretary problem. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2082–2095.

[32] Y. Keneshloo, T. Shi, N. Ramakrishnan, and C. K. Reddy. 2019. Deep reinforcement learning for sequence-to-sequence models. *IEEE Transactions on Neural Networks and Learning Systems* 31, 7 (2019), 2469–2489.

[33] S. S. Kozat and A. C. Singer. 2011. Universal semiconstant rebalanced portfolios. *Mathematical Finance: An International Journal of Mathematics, Statistics and Financial Economics* 21, 2 (2011), 293–311.

[34] Z. R. Lai, P. Y. Yang, L. Fang, and X. Wu. 2018. Short-term sparse portfolio optimization based on alternating direction method of multipliers. *The Journal of Machine Learning Research* 19, 1 (2018), 2547–2574.

[35] D. A. Lesmond, J. P. Ogden, and C. A. Trzcinka. 1999. A new estimate of transaction costs. *The Review of Financial Studies* 12, 5 (1999), 1113–1141.

[36] B. Li and S. C. Hoi. 2012. On-line portfolio selection with moving average reversion. In *Proceedings of the 29th International Conference on Machine Learning*.

[37] B. Li and S. C. Hoi. 2014. Online portfolio selection: A survey. *Computing Surveys* 46, 3 (2014), 1–36.

[38] B. Li, S. C. Hoi, P. Zhao, and V. Gopalkrishnan. 2013. Confidence weighted mean reversion strategy for online portfolio selection. *ACM Transactions on Knowledge Discovery from Data* 7, 1 (2013), 1–38.

[39] B. Li, D. Sahoo, and S. C. Hoi. 2016. OLPS: A toolbox for on-line portfolio selection. *Journal of Machine Learning Research* 17, 35 (2016), 1–5.

[40] B. Li, J. Wang, D. Huang, and S. C. Hoi. 2018. Transaction cost optimization for online portfolio selection. *Quantitative Finance* 18, 8 (2018), 1411–1424.

[41] B. Li, P. Zhao, S. C. Hoi, and V. Gopalkrishnan. 2012. PAMR: Passive aggressive mean reversion strategy for portfolio selection. *Machine Learning* 87, 2 (2012), 221–258.

[42] J. Li, R. Rao, and J. Shi. 2018. Learning to trade with deep actor critic methods. In *Proceedings of the 2018 11th International Symposium on Computational Intelligence and Design*, Vol. 02. 66–71.

[43] S. Li, Y. Wu, X. Cui, and H. Dong. 2019. Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 4213–4220.

[44] P. Lindroth, M. Patriksson, and A. B. Strömberg. 2010. Approximating the Pareto optimal set using a reduced set of objective functions. *European Journal of Operational Research* 207, 3 (2010), 1519–1534.

[45] W. Long, Z. Lu, and L. Cui. 2019. Deep learning-based feature engineering for stock price movement prediction. *Knowledge-Based Systems* 164, 1 (2019), 163–173.

[46] H. Luo, C. Y. Wei, and K. Zheng. 2018. Efficient online portfolio with logarithmic regret. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. 8235–8245.

[47] L. Malandri, F. Z. Xing, C. Orsenigo, C. Vercellis, and E. Cambria. 2018. Public mood–driven asset allocation: The importance of financial sentiment in portfolio management. *Cognitive Computation* 10, 6 (2018), 1167–1176.

[48] V. Mnih, K. Kavukcuoglu, D. Silver, and A. A. Rusu. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529–533.

[49] E. Mohr and R. Dochow. 2017. Risk management strategies for finding universal portfolios. *Annals of Operations Research* 256, 1 (2017), 129–147.

[50] L. Nanni, S. Ghidoni, and S. Brahnam. 2017. Handcrafted vs. non-handcrafted features for computer vision classification. *Pattern Recognition* 71, 1 (2017), 158–172.

[51] Francesco Orabona, Koby Crammer, and Nicolo Cesa-Bianchi. 2015. A generalized online mirror descent with applications to classification and regression. *Machine Learning* 99, 3 (2015), 411–435.

[52] R. Ortner, P. Gajane, and P. Auer. 2020. Variational regret bounds for reinforcement learning. In *Proceedings of the 35th Uncertainty in Artificial Intelligence*. PMLR, 81–90.

[53] M. Pereyra. 2019. Revisiting maximum-a-posteriori estimation in log-concave models. *SIAM Journal on Imaging Sciences* 12, 1 (2019), 650–670.

[54] F. Pérez-Cruz. 2008. Kullback-Leibler divergence estimation of continuous distributions. In *Proceedings of the 2008 IEEE International Symposium on Information Theory*. IEEE, 1666–1670.

[55] I. Rahwan, M. Cebrian, N. Obradovich, and J. Bongard. 2019. Machine behaviour. *Nature* 568, 7753 (2019), 477–486.

[56] A. Serletis and A. A. Rosenberg. 2009. Mean reversion in the US stock market. *Chaos, Solitons & Fractals* 40, 4 (2009), 2007–2015.

[57] S. Y. Shih, F. K. Sun, and H. Y. Lee. 2019. Temporal pattern attention for multivariate time series forecasting. *Machine Learning* 108, 8–9 (2019), 1421–1441.

[58] D. Silver, J. Schrittwieser, K. Simonyan, L. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. Driessche, T. Graepel, and D. Hassabis. 2017. Mastering the game of Go without human knowledge. *Nature* 550, 7676 (2017), 354–359.

[59] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. 2017. Attention is all you need. In *Proceedings of the Advances in Neural Information Processing Systems*. 5998–6008.

[60] F. Xing, L. Malandri, Y. Zhang, and E. Cambria. 2020. Financial sentiment analysis: An investigation into common mistakes and silver bullets. In *Proceedings of the 28th International Conference on Computational Linguistics*. 978–987.

[61] F. Z. Xing, E. Cambria, and R. E. Welsch. 2018. Intelligent asset allocation via market sentiment views. *IEEE Computational Intelligence Magazine* 13, 4 (2018), 25–34.

[62] F. Z. Xing, E. Cambria, and Y. Zhang. 2019. Sentiment-aware volatility forecasting. *Knowledge-Based Systems* 176, 1 (2019), 68–76.

[63] L. Yang and M. Wang. 2020. Reinforcement learning in feature space: Matrix bandit, kernels, and regret bound. In *Proceedings of the 37th International Conference on Machine Learning*. PMLR, 10746–10756.

[64] M. Yang, M. Pei, and Y. Jia. 2020. Online maximum a posteriori tracking of multiple objects using sequential trajectory prior. *Image and Vision Computing* 94, 9 (2020), 103867.

[65] X. Yang, J. Xian, H. Lin, J. He, and Y. Zhang. 2020. Aggregating expert advice strategy for online portfolio selection with side information. *Soft Computing* 24, 3 (2020), 2067–2081.

[66] J. Yin, R. Wang, S. Ju, Y. Bai, and J. Z. Huang. 2020. An asymptotic statistical learning algorithm for prediction of key trading events. *IEEE Intelligent Systems* 35, 2 (2020), 25–35.

[67] J. Zhang, T. Leung, and A. Aravkin. 2020. Sparse mean-reverting portfolios via penalized likelihood optimization. *Automatica* 111, 1 (2020), 108651.

Received June 2019; revised March 2021; accepted April 2021