Neurocomputing 498 (2022) 14–27

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

GPM: A graph convolutional network based reinforcement learning framework for portfolio management

Si Shi [a], Jianjun Li [a,*], Guohui Li [b], Peng Pan [a], Qi Chen [a], Qing Sun [c]

[a] School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China
[b] School of Software Engineering, Huazhong University of Science and Technology, Wuhan 430074, China
[c] Harbin Branch, China Merchants Bank, China

ARTICLE INFO

Article history:
Received 26 July 2021
Revised 26 February 2022
Accepted 23 April 2022
Available online 28 April 2022
Communicated by Zidong Wang

Keywords:
Portfolio management
Reinforcement learning
Graph convolutional network
Company relationship

ABSTRACT

Portfolio management is a decision-making process of periodically reallocating a certain amount of funds into a portfolio of assets, with the objective of maximizing the profits constrained to a given risk level. Due to its nature of learning from dynamic interactions and planning for long-run performance, reinforcement learning (RL) recently has received much attention in portfolio management. However, most of existing RL-based PM approaches in general only consider price changes of portfolio assets and the implicit correlations between price changes, while ignoring the rich relations between companies in the market, such as two assets in the same sector or two companies have a supplier-customer relation, which are very important for trading decision making. To address these limitations, in this paper, we propose GPM, a novel graph convolutional network-based reinforcement learning framework for portfolio management, which first employs Relational Graph Convolutional Network (R-GCN) to extract asset relational features, and then combines relational features with multi-scale temporal features to make trading decisions for better performance. We also introduce softmax with temperature to increase portfolio diversity, which leads to further increase in profits. Experimental results on two real-world datasets: NASDAQ and NYSE, validate the effectiveness of GPM over state-of-the-art PM methods. Further, more experiments are conducted to evaluate GPM with different graph neural networks and different number of network layers, to explore proper settings for different markets.

1. Introduction

A portfolio is an investor's collection of financial assets, and portfolio management involves building and overseeing these assets to meet the long-term financial objectives. There are two main portfolio management strategies: *passive* and *active*. Passive investors construct a portfolio of stocks or financial products to replicate the performance of an index, while active investors select stocks more flexibly and trade frequently to obtain high profits and beat the market. In this paper, we explore active portfolio management, with the objective of finding out the profitable potential of machine learning methods in portfolio management. Traditional active portfolio management methods [1,2] basically rely on predefined trend assumption. They either assume that the market will keep the current price trends and focus on more successful assets [3], or suppose that asset prices eventually will revert to the average level (called mean reversion) and transfer the wealth from winners to losers [4], which are not adaptable to the complex and unpredictable real-world financial markets. In other words, these methods can only obtain profits in specific markets, once the market does not meet the expected situation, their returns will be significantly impacted.

Recently, with the development of artificial intelligence, machine learning techniques are gaining traction in the financial domain. Reinforcement Learning (RL) can dynamically adapt to market changes and is well suited to PM. There are some good attempts on applying RL for solving the PM problem, such as iCNN [5], EIIE [6], EI^3 [7], SARL [8] and AlphaStock [9]. These methods first leverage historical asset prices or (and) some external information, e.g., financial news, to extract the price movement features, and then obtain the rebalanced portfolio weights based on the derived features to make decision. Though having achieved significant progress, there are still several challenges remain unsolved in PM:

* Corresponding author.
E-mail addresses: shisi_cc@hust.edu.cn (S. Shi), jianjunli@hust.edu.cn (J. Li), guohuili@hust.edu.cn (G. Li), panpeng@hust.edu.cn (P. Pan), chenqijason@hust.edu.cn (Q. Chen), sunqing@cmbchina.com (Q. Sun).

https://doi.org/10.1016/j.neucom.2022.04.105
0925-2312/

1) **Temporal Features Extraction**. Temporal features are essential features for financial time series. Through the analysis on recent price changes, trading agent can judge the future trend of asset prices and make trading decisions. Deep learning methods use LSTMs and CNNs to extract the information of asset prices in the portfolio. One of the shortcomings is that they cannot extract short, medium and long term price changes. These different scales of temporal information are necessary for trading decisions. In addition, a portfolio is composed of multiple assets, it is important to efficiently extract multi-scale temporal features of multiple assets.
2) **Relational Features Extraction**. Almost all existing RL-based PM methods extract asset features only according to their recent price movements or technical indicators [6]. They focus on price movement features but without considering the relations between companies in the market. However, the relations between assets and their corresponding companies may contain valuable clues for portfolio management. For example, as shown in Fig. 1, stocks under the same sector or industry like MSFT (Microsoft Corp.) and GOOG (Google Inc.) might have similar trends, the stock price of a supplier company (like Intel Corp.) might also impact the stock price of its consumer company (like Microsoft Corp.). Moreover, different types of relationships also need to be considered in portfolio management, because different relationships tend to have different influences. For example, the correlations between companies' prices vary across industries.
3) **Portfolio diversity**. Most RL-based PM methods use the softmax function to generate portfolio weights. However, the softmax function tends to be "The Winner Takes It All" and causes portfolio to shrink to only one or two assets, which is unfavorable to portfolio diversity and will lead to high risk in trading process. In finance domain, there is a famous saying that "do not put all your eggs in one basket", indicating that putting all the funds into a single asset tends to suffer serious losses once this asset performs badly. Therefore, in order to improve the ability of the portfolio to resist potential risks, it is necessary to increase the diversity of the portfolio so as to spread the risks. However, few machine learning methods take portfolio diversity into account. Hence, while these methods can be very profitable, they are also potentially risky.

To address the aforementioned challenges, in this paper, we propose GPM, a novel graph convolutional network based reinforcement learning framework for portfolio management. GPM can effectively extract the temporal and relational features of the portfolio, while taking into account the diversity of the portfolio. Specifically, we first utilize a convolutional network with multiple filter sizes to extract multi-scale temporal features. Then, by modeling the relations between companies in a trading market as a heterogeneous graph, we utilize Relational Graph Convolutional Network (R-GCN) [10], which is designed for heterogeneous graph and adept at handling multi-relational data, for relational feature extraction. Finally, the extracted relational features and price movement features are concatenated together to help make trading decisions. Moreover, we use softmax with temperature to produce softer distribution and hence increase portfolio diversity. The main contributions of this work can be summarized as follows:

- We propose a novel multi-scale Graph-based reinforcement learning framework for Portfolio Management, named GPM. In this framework, we propose a novel feature extraction method that combines multi-scale convolutions and relational graph convolutions to extract temporal and relational features of portfolio assets, so as to utilize the information of different temporal granularity and various types of intercompany relationships simultaneously for better PM performance.
- We further propose a weight allocation method that introduces temperature in softmax when generating portfolio weights. By choosing a proper high temperature in policy network, probability distribution over the assets becomes more soft, which reduces the risk and leads to more increase in profits.
- Experimental results on two real-world datasets: National Association of Securities Dealers Automated Quotations (NASDAQ) and New York Stock Exchange (NYSE), show that our method outperforms the state-of-the-art PM methods in terms of final accumulated profit.

The remainder of this paper is organized as follows. Section 2 discusses related work; Section 3 defines the problem to be addressed in this work; Section 4 details our proposed model; Section 5 presents and analyzes the experimental results; Finally, Section 6 draws a conclusion.

2. Related work

There are two types of portfolio management approaches: passive and active. For passive methods, investors construct a portfolio of financial products to replicate the performance of an index. For example, [11] uses a reparametrisation method for index tracking with cardinality constraints. [12] uses Deep Neural Network with Fixed noise (Deep NNF) to optimize the index-tracking portfolio. For active methods, investors make as much profit as possible without limiting in an index. Traditional activate portfolio management strategies can be roughly divided into four categories: benchmarks that allocates funds into assets equally, follow-the-winner approaches that increase the weights of more successful assets, follow-the-loser approaches that transfer the wealth from winners to losers, and pattern-matching approaches that build a portfolio based on some sampled similar historical patterns [2]. Most existing traditional strategies rely on predefined trend assumption and hence lack adaptivity to the real-world financial market.

Recently, deep learning, especially deep reinforcement learning, has been introduced in financial domain to address the PM problem. Several RL-based methods have been proposed to perform single stock trading, including critic-only methods [13,14] that learn a value function to improve trading policy, actor-only methods [15,16] that directly optimize trading policy, and actor-critic methods [17] that learn an actor to produce policy and a critic to evaluate the actor's performance. As for portfolio management involving multiple assets, integrated CNN (iCNN) [5] first uses CNN to extract portfolio features, and then applies fully connected layer to allocate asset weights. Ensemble of Identical Independent Evaluators (EIIE) [6] processes price data of each asset independently while sharing the network parameters. Based on EIIE CNN, Ensemble of Identical Independent Inception (EI^3) [7] is further proposed to consider price movement in multiple scales by extracting multi-scale features. State Augmented Reinforcement Learning (SARL) [8] utilizes price movement prediction from asset prices or financial news to augment states in reinforcement learning. Note that all these methods only consider price changes of assets, but without considering the relations between companies.

There are also some researches that consider stock price movement relations in the financial domain. The Markowitz mean-variance model [18] selects the most efficient portfolio according to their expected returns (mean) and the standard deviation (variance). Based on the Markowitz theory, a number of portfolio selection methods using variance (or covariance) to model stock

Fig. 1. Different relations between three companies.

relations have been proposed [19,20]. In statistics, correlation coefficient, such as Pearson coefficient [21], Spearman coefficient [22] and Kendall coefficient [23], and causality, such as Granger causality [24,25], are also widely used to represent relations between stocks. However, these methods are difficult to represent the nonlinear and indirect relationships between assets. AlphaStock [9] adopts the buy-winners-and-sell-losers strategy and uses Cross-Asset Attention Network (CAAN) to describe the interrelationships among stocks. Portfolio Policy Network (PPN) [26] uses a two-stream portfolio policy network that extracts price series patterns via LSTM and asset correlations via Temporal Correlational Convolution Block (TCCB). Though CAAN in AlphaStock and TCCB in PPN can extract the interrelationships between stocks within portfolio, they cannot model influence from stocks outside the portfolio. Moreover, indirect influence from other stocks is also not considered in these frameworks.

Graph is a proper way to represent global relationships. In recent years, there has been increasing interest in the graphical expansion of deep learning methods. Graph Neural Networks (GNNs), including Graph Convolutional Network (GCN) [27], Graph ATtention network (GAT) [28], R-GCN [10] and Graph AutoEncoder (GAE) [29], have been widely studied. In financial domain, there are also several researches that represent stock relations as a graph and use graph convolutional networks to extract stock relations. For example, [30] incorporates company shareholding relationships via GCN to improve stock prediction accuracy. Relational Stock Ranking (RSR) [31] proposes Temporal Graph Convolution (TGC) that jointly considers temporal evolution and relations of stocks. Hierarchical ATtention network for Stock prediction (HATS) [32] first aggregates information from neighbors in each relation type, and then aggregates information from different edge types via hierarchical attention. Multi-GCGRU [33] uses GCN and GRU to combine shareholding, industry and topicality information for stock movement prediction. Note all these methods are designed for stock prediction, and hence cannot be applied to address the PM problem directly. These methods either divide the future trend into several categories that are difficult for trading agent to accurately assign portfolio weights, or forecast future prices that cannot be used to make trade decision directly. Recently, [34] proposes a graph convolutional reinforcement learning framework called DeepPocket to exploit the interrelations between assets. However, DeepPocket uses GCN to extract the relationship between financial instruments, which makes it can only deal with a single type of relations. In other words, DeepPocket fails to take advantage of the rich relation types, which play an important role in investment decision making.

3. Problem definition

In this section, we introduce the research problem of portfolio management. To facilitate description, Table 1 summarizes the mathematical notations used in this paper.

Portfolio management is a decision-making process of periodically redeploying funds into different assets. Suppose a trading agent is in a financial exchange market $\mathscr{V}$ that consists of a collection of M assets, whose prices are changing over time. The closing, highest, and lowest prices of asset $i (1 \leqslant i \leqslant M)$ at period t, denoted as $v_{i,t}$, $v^{\mathrm{H}}_{i,t}$ and $v^{\mathrm{L}}_{i,t}$, respectively, are arriving at a fixed period. Moreover, let $v_{0,t}$ denote the cash price at period t. We define $v_{0,t} = 1$ for all t, which means the cash is the benchmark price and is also a risk-free asset.

The relations between assets in the market can be defined as a directed relation graph $\mathscr{G} = (\mathscr{V}, \mathscr{E}, \mathscr{R})$, where each node $i \in \mathscr{V}$ represents an asset (and its corresponding company), and each labeled edge $(i, r, j) \in \mathscr{E}$ represents that there is a relation from asset i to asset j. Edge type $r \in \mathscr{R}$ represents what type the relation of edge $i \to j$ is, such as i is the owner of j, or both i and j are in the same industry. The edges can be constructed by using information from different sources, such as financial websites and wikidata. For example, in NASDAQ, both GOOGL (Alphabet Inc.) and FB (Facebook Inc.) are in computer software industry. So, node GOOGL and node FB are bidirectionally connected with edge type "computer software". Similarly, suppose BA (Boeing Co.) provides aircrafts to UAL (United Airlines Holdings, Inc.), then there exist a directed edge with type "Produce and Operate" from node BA to node UAL, as well as a directed edge with type "Operate and Produce" from node UAL to node BA.

A portfolio is a collection of investments held by the agent. The portfolio $\mathscr{P}$ is usually a subset of the market, i.e., $\mathscr{P} \subset \mathscr{V}$. Without loss of generality, we assume that the cash and m assets $k_1, k_2, \ldots, k_m$ $(1 \leqslant k_x \leqslant M)$ constitute the portfolio $\mathscr{P}$. With m assets and an initial cash, the agent considers how to re-balance the portfolio periodically, aiming at maximizing the total profit in a given time interval. Specifically, at the beginning of each period, the agent observes current price movement and allocates funds into

Table 1
Notations and Descriptions.

Notation	Description
$\mathscr{G}$	market graph, including assets and their relations
$\mathscr{E}$	the set of relation edges
$\mathscr{R}$	the set of relation types
$\mathscr{V}$	the set of assets in the market
$\mathscr{P}$	the set of assets in the portfolio
$\mathbf{w}^t$	weight vector of assets in the portfolio at period t
$\mathbf{X}^t$	the stacking of $\mathbf{V}^t$, $\overset{\mathrm{H}}{\mathbf{V}}_t$ and $\overset{\mathrm{L}}{\mathbf{V}}_t$
$\mathbf{V}^t$, $\overset{\mathrm{H}}{\mathbf{V}}_t$, $\overset{\mathrm{L}}{\mathbf{V}}_t$	normalized closing, highest, and lowest price matrix at period t
$v^{i,t}$, $\overset{\mathrm{H}}{v}_{i,t}$, $\overset{\mathrm{L}}{v}_{i,t}$	closing, highest, and lowest prices of asset i at period t
p^t	portfolio value at period t
r^t	profit of period t
k^x	the x-th asset in the portfolio
M	the number of assets in the market
m	the number of assets in the portfolio

these assets. Then, price of each asset changes during the period and the portfolio value changes accordingly. At the end of the period, the agent receives the profit and reallocates funds to the assets. The price change of $\mathscr{P}$ during period t can be defined as the price relative vector,

$$\mathbf{y}_t = \left[1, \frac{v_{k_1,t}}{v_{k_1,t-1}}, \frac{v_{k_2,t}}{v_{k_2,t-1}}, \cdots, \frac{v_{k_m,t}}{v_{k_m,t-1}}\right]^{\mathsf{T}} \tag{1}$$

For each period t, the trading agent will adjust the weights of each asset according to the portfolio vector, represented as,

$$\mathbf{w}_t = [w_{0,t}, w_{k_1,t}, \ldots, w_{k_m,t}]^{\mathsf{T}} \tag{2}$$

where $w_{0,t}$ is the weight of cash and $w_{k_x,t}$ is the weight of asset k_x. The weights of all the assets (including cash) in $\mathbf{w}_t$ sum up to 1, i.e., $w_{0,t} + \sum_{x=1}^{m} w_{k_x,t} = 1$. Because in some markets or countries, such as China, short-selling is not allowed, we adopt long-only investment strategies in this work, i.e., $w_{k_x,t} \geqslant 0$. We let $\mathbf{w}_0 = [1, 0, \ldots, 0]^{\mathsf{T}}$, indicating that we have only cash before investment. After paying the commission fee, the value of the portfolio will shrink, then the portfolio value at period t can be computed as,

$$p_t = \mu_t p_{t-1} \mathbf{y}_t \cdot \mathbf{w}_{t-1} \tag{3}$$

where μ_t is the transaction remainder factor [5]. In order to avoid the impact of initial funds, we let $p_0 = 1$.

To derive the final accumulated profit, the profit of period t is modeled as the logarithmic rate of return r_t, which is computed by,

$$r_t = \ln \frac{p_t}{p_{t-1}} \tag{4}$$

Subsequently, the final portfolio value can be computed as,

$$p_T = p_0 \exp\left(\sum_{t=1}^{T} r_t\right) \tag{5}$$

where $T-1$ is the final trading period and r_T is the return after the last allocation.

Based on the above description, our goal of portfolio management is to find a series of $\mathbf{w}_t$, i.e., $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_T$, to maximize the final portfolio value p_T.

4. GPM: a graph convolutional network based RL framework for PM

4.1. Framework

The portfolio management problem can be modeled as a Markov Decision Process (MDP) in the form of a 4-tuple $< S, A, P, R >$, where S is the state space, A is the action space, $P : S \times A \times S \to [0, 1]$ is the transition probability distribution, and $R : S \times A \to \mathbb{R}$ is the reward function. Detailed descriptions are as follows:

- State. When making trading decisions at period t, the agent will take the current market price vector $\mathbf{X}_t$, the asset relation graph $\mathscr{G}$, and the previous portfolio vector $\mathbf{w}_{t-1}$ into consideration. So,
$$s_t = (\mathbf{X}_t, \mathscr{G}, \mathbf{w}_{t-1}) \tag{6}$$
Note $\mathbf{X}_t$ is the stacking of normalized closing price matrix $\mathbf{V}_t$, normalized highest price matrix $\mathbf{V}_t^{\mathrm{H}}$, and normalized lowest price matrix $\mathbf{V}_t^{\mathrm{L}}$ within time window n (user defined). Consequently, the normalized closing price matrix $\mathbf{V}_t$can be obtained as,
$$\mathbf{V}_t = \begin{bmatrix} \frac{v_{1,t-n+1}}{v_{1,t}}, & \frac{v_{1,t-n+2}}{v_{1,t}}, & \cdots, & \frac{v_{1,t-1}}{v_{1,t}}, & 1 \\ \frac{v_{2,t-n+1}}{v_{2,t}}, & \frac{v_{2,t-n+2}}{v_{2,t}}, & \cdots, & \frac{v_{2,t-1}}{v_{2,t}}, & 1 \\ \cdots, & \cdots, & \cdots, & \cdots & \\ \frac{v_{M,t-n+1}}{v_{M,t}}, & \frac{v_{M,t-n+2}}{v_{M,t}}, & \cdots, & \frac{v_{M,t-1}}{v_{M,t}}, & 1 \end{bmatrix},$$
and $\mathbf{V}_t^{\mathrm{H}}$ and $\mathbf{V}_t^{\mathrm{L}}$ can be derived similarly.
- Action. The action is to determine the portfolio vector $\mathbf{w}_t$, i.e., $a_t = \mathbf{w}_t$.
- Reward. The reward at period t is the trading return r_t.

The agent makes decisions according to the deterministic policy $\pi : S \to A$ with parameter θ. The goal of our RL framework is to find the optimal policy π^* that can maximize the accumulated reward in an interval. In this work, we define the objective function $J_{[t_1,t_2]}(\pi_\theta)$ as the average logarithmic accumulated return within $[t_1, t_2]$, i.e.,

$$J_{[t_1,t_2]}(\pi_\theta) = \frac{1}{t_2 - t_1 + 1} \sum_{t=t_1}^{t_2} r_t \tag{7}$$

We use deterministic policy gradient [35] to solve the reinforcement learning problem. The parameters are updated in a mini-batch within $[t_1, t_2]$ via gradient ascent by the following formula,

$$\theta \longleftarrow \theta + \lambda \nabla_\theta J_{[t_1,t_2]}(\pi_\theta) \tag{8}$$

where λ represents the learning rate.

4.2. Network

The architecture of our proposed network is shown in Fig. 2. The input of the network is the state of MDP for PM, which consists of the market price tensor $\mathbf{X}_t$, an asset relation graph $\mathscr{G}$ and previous portfolio vector $\mathbf{w}_{t-1}$. To facilitate distinction, assets in portfolio $\mathscr{P}$ are represented in color, while other stocks not in $\mathscr{P}$ are in white.

We first perform multi-scale convolution on $\mathbf{X}_t$ to derive a multi-scale temporal feature tensor $\mathbf{X}_{\text{temporal}}$. The relation graph $\mathscr{G}$, combining with $\mathbf{X}_{\text{temporal}}$, will be fed into the R-GCN network to get the relational feature tensor $\mathbf{X}_{\text{relational}}$. Next, $\mathbf{X}_{\text{temporal}}$ and $\mathbf{X}_{\text{relational}}$ are concatenated to get a more comprehensive market feature tensor $\mathbf{X}_{\text{market}}$. We then select features of assets in $\mathscr{P}$ from $\mathbf{X}_{\text{market}}$ to derive $\mathbf{X}_{\text{port}}$, and concatenate it with the previous portfolio tensor $\mathbf{W}_{t-1}$ (reshaped from $\mathbf{w}_{t-1}$). Afterwards, a convolution with filter size 1×1 is applied to compress $[\mathbf{X}_{\text{port}} | \mathbf{W}_{t-1}]$ to get

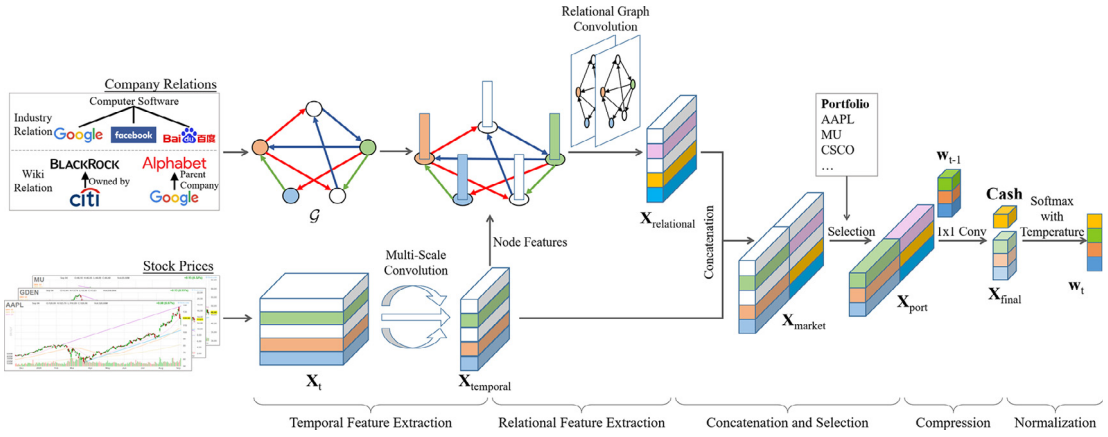

Fig. 2. Architecture of our proposed network.

$\mathbf{X}_{\text{final}}$ with 1 channel. Finally, we combine $\mathbf{X}_{\text{final}}$ with the cash tensor and normalize them to get the portfolio weight vector $\mathbf{w}_t$. After a period, the portfolio vector will be fed into the network recurrently to help make the next trading decision.

4.2.1. Multi-scale temporal feature extraction

For temporal feature extraction, we perform multi-scale convolution on the price tensor $\mathbf{X}_t$ with three branches (as shown in Fig. 3). Specifically, we use three branches to extract short-term price movement feature, mid-term price movement feature and long-term price movement feature with multiple convolution filter sizes respectively. The result is the temporal feature tensor, denoted as $\mathbf{X}_{\text{temporal}}$. The first dimensions of all the convolution filters are set to 1 to keep the assets independent with each other.

The first branch with two layers performs short-term price movement feature extraction. The first layer applies convolution with small filter size $[1 \times k_{\text{short}}]$ and the second layer compresses the tensor into short-term price movement feature $\mathbf{X}_{\text{short}}$ with shape $(f_{\text{short}}, M, 1)$,

$$\mathbf{X}_{\text{short}} = \sigma\left(B^{(2)}_{\text{short}} + W^{(2)}_{\text{short}} * \left(\sigma\left(B^{(1)}_{\text{short}} + W^{(1)}_{\text{short}} * \mathbf{X}_t\right)\right)\right) \tag{9}$$

where σ is the activation function, $*$ denotes the convolution operator, and $B^{(l)}_{\text{short}}$ and $W^{(l)}_{\text{short}}$ are bias and weights of the l-th convolution layer. Note we use ReLU as the activation function throughout this paper except of graph convolution.

The second branch with two layers performs mid-term price movement feature extraction. The first layer applies convolution with mid filter size $[1 \times k_{\text{mid}}]$ and the second layer compresses the tensor into mid-term price movement feature $\mathbf{X}_{\text{mid}}$ with shape $(f_{\text{mid}}, M, 1)$,

$$\mathbf{X}_{\text{mid}} = \sigma\left(B^{(2)}_{\text{mid}} + W^{(2)}_{\text{mid}} * \left(\sigma\left(B^{(1)}_{\text{mid}} + W^{(1)}_{\text{mid}} * \mathbf{X}_t\right)\right)\right) \tag{10}$$

The third branch with two layers performs long-term price movement feature extraction. The first layer applies convolution with large filter size $[1 \times k_{\text{long}}]$ and the second layer compresses the tensor into long-term price movement feature $\mathbf{X}_{\text{long}}$ with shape $(f_{\text{long}}, M, 1)$,

$$\mathbf{X}_{\text{long}} = \sigma\left(B^{(2)}_{\text{long}} + W^{(2)}_{\text{long}} * \left(\sigma\left(B^{(1)}_{\text{long}} + W^{(1)}_{\text{long}} * \mathbf{X}_t\right)\right)\right) \tag{11}$$

Finally, the short-term price movement feature $\mathbf{X}_{\text{short}}$, mid-term price movement feature $\mathbf{X}_{\text{mid}}$ and long-term price movement feature $\mathbf{X}_{\text{long}}$ are concatenated as the comprehensive market price feature $\mathbf{X}_{\text{price}}$ with shape $(f_{\text{short}} + f_{\text{mid}} + f_{\text{long}}, M, 1)$.

$$\mathbf{X}_{\text{price}} = \left[\mathbf{X}_{\text{short}} | \mathbf{X}_{\text{medium}} | \mathbf{X}_{\text{long}}\right] \tag{12}$$

where $|$ denotes the concatenation operation.

4.2.2. Relational feature extraction

In order to incorporate asset relations, we utilize R-GCN [10] to extract nodes' features from their neighbors. Our implementation is a little different from the original R-GCN in that we only consider ingoing edges. The diagram for computing the forward-pass update of a node is shown in Fig. 4. Activations from neighboring nodes first perform a linear transformation, and then are averaged for each relation type individually. Afterwards, activations from different edge types are added up to get the representation of the node.

Since asset relation graph $\mathcal{G}$ is a heterogeneous graph, we feed the market temporal features $\mathbf{X}_{\text{temporal}}$ and $\mathcal{G}$ into R-GCN that can handle multi-relational data. The feature of asset i from $\mathbf{X}_{\text{temporal}}$, denoted as $\mathbf{x}_i$, is used as the initial feature of node i in $\mathcal{G}$, i.e., $\mathbf{h}_i^{(1)} = \mathbf{x}_i$. For each node in the graph, the feature is the normalized sum of weighted features from its neighbors. The feature of asset i after l-th layers feature propagation can be computed by,

$$\mathbf{h}_i^{(l+1)} = \varphi\left(\sum_{r\in\mathcal{R}}\sum_{j\in\mathcal{N}_i^r}\frac{1}{|\mathcal{N}_i^r|}W_r^{(l)}\mathbf{h}_j^{(l)} + W_0^{(l)}\mathbf{h}_i^{(l)}\right) \tag{13}$$

where φ is the activation function, $\mathcal{N}_i^r$ is the neighbor set containing assets that have edge type r with asset i, $W_r^{(l)}$ is the weight matrix of edge type r in the l-th layer, and $W_0^{(l)}$ denotes the self-loop weight matrix. We use LeakyReLU as the activation function. After L layers feature propagation, we can get the relational feature vector for asset i, i.e., $\mathbf{h}_i^{(L)}$. All the relational feature vectors in the market form the relational feature tensor $\mathbf{X}_{\text{relational}}$.

After obtaining $\mathbf{X}_{\text{relational}}$, we concatenate it with $\mathbf{X}_{\text{temporal}}$ to derive a more comprehensive market feature tensor $\mathbf{X}_{\text{market}}$.

4.2.3. Selection and compression

Since our goal is to decide the weights for each asset in the portfolio, we first select assets that are in $\mathcal{P}$ and construct the portfolio feature $\mathbf{X}_{\text{port}}$. For further normalization, we concatenate $\mathbf{X}_{\text{port}}$ with previous portfolio tensor $\mathbf{W}_{t-1}$ and then compress them into tensor with 1 channel via a 1×1 convolution,

$$\mathbf{X}_{\text{final}} = \sigma\left(B_{\text{final}} + W_{\text{final}} * \left[\mathbf{X}_{\text{port}} | \mathbf{W}_{t-1}\right]\right) \tag{14}$$

Note that $\mathbf{X}_{\text{final}}$ dose not consider cash, which is also an important part of portfolio. Therefore, we further concatenate the cash tensor (always 0 as the benchmark) with $\mathbf{X}_{\text{final}}$, and compress them to get a trend vector,

$$\mathbf{v}_{\text{trend}} = \left[v_{\text{trend}}(k_0), v_{\text{trend}}(k_1), \ldots, v_{\text{trend}}(k_m)\right]^{\mathrm{T}} \tag{15}$$

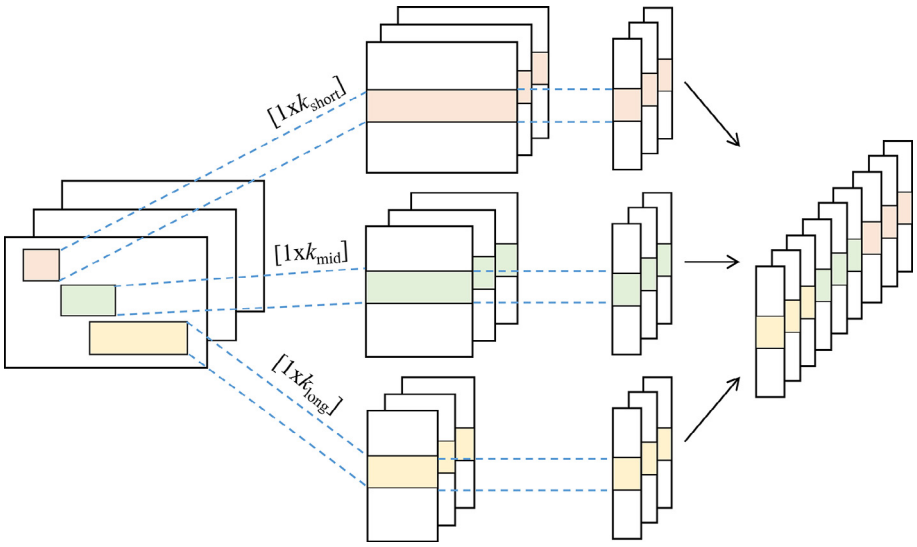

Fig. 3. Multi-scale temporal feature extraction.

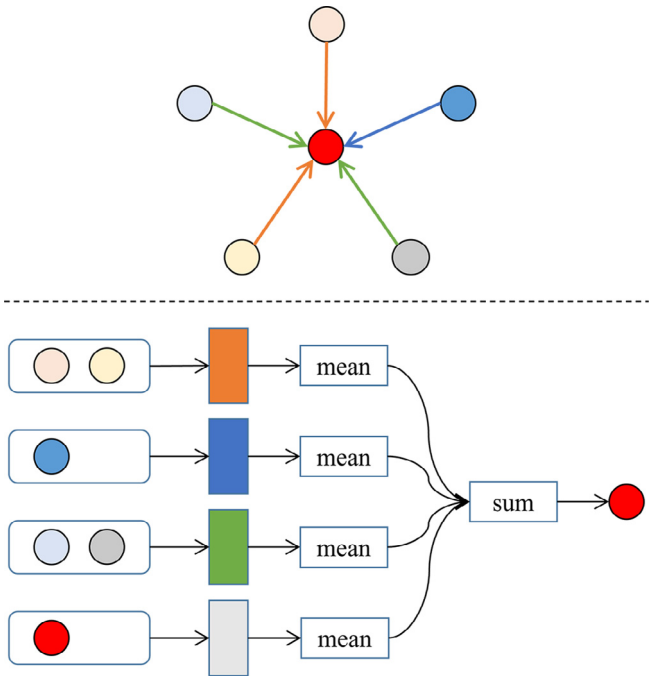

Fig. 4. Propagation of a node in relational graph.

where $v_{\text{trend}}(k_x)$ is the trend scalar of asset k_x, and $k_0 = 0$.

4.2.4. Normalization

The final step of portfolio management is to assign weights to different assets via softmax function. In this way, we can normalize $\mathbf{v}_{\text{trend}}$ to get the portfolio vector $\mathbf{w}_t = [w_{0,t}, w_{k_1,t}, \ldots, w_{k_m,t}]^\top$.

However, the softmax function may cause portfolio to shrink to only one or two assets, and hence lead to excessive concentration of risk. In order to increase diversity, we introduce a hyper-parameter, temperature [36], to scale the distance between asset trends before applying softmax. Specifically,

$$w_{k_x,t} = \frac{\exp\left(v_{\text{trend}}(k_x)/T\right)}{\sum_{y=0}^{m} \exp\left(v_{\text{trend}}(k_y)/T\right)} \tag{16}$$

where T denotes the temperature. Note when $T = 1$, it degenerates to a standard softmax.

By applying a higher temperature, the network will produce a softer probability distribution over the assets. In other words, it increases the weight of low-weight assets and decrease the weight of high-weight assets, and leads to more diversity. When probabilities get soft, low-probability assets get more attention and the trading agent will focus on more assets that are also profitable. In this way, risk is no longer concentrated in one or two assets. When the temperature becomes too high, however, the agent focuses not only on profitable assets but also on non-profitable assets, resulting in lower profits. Finally, extreme temperature forces the probability distribution uniform, making agents deploy funds equally over assets. Based on the above analysis, it is clear that a proper T should be set to make a balance between risk and profit.

The trading agent will adjust the weights of each asset according to the portfolio vector $\mathbf{w}_t$ and wait for rewards and price information from exchange after a period.

4.3. Training

The training process of our network consists of offline training that learns from historical price data, and online training that adapts itself to the highly unstable financial market. Besides historical datasets and asset relation information, following [6], we also use portfolio vector memory (PVM) to store portfolio vector $\mathbf{w}_t$ in chronological order.

Before entering the market, the agent will train the network to learn the optimal policy that can achieve the highest profits based on historical price data. Specifically, for each training step, the agent will load $\mathscr{G}$, select a mini-batch of price data from historical price database to construct $\mathbf{X}_t$, and select $\mathbf{w}_{t-1}$ from PVM. These tensors will be fed into the network, which outputs $\mathbf{w}_t$. Then, the agent will compute the average logarithmic accumulated return and update the network parameters. The new derived portfolio vector $\mathbf{w}_t$ will be stored in PVM for later training.

After trading starts, for each period, the agent will receive the latest market price data from the exchange. Then, the agent uses the latest price data in the time window to construct market price tensor $\mathbf{X}_t$, which will be fed into the network and receive an output $\mathbf{w}_t$ to be stored in PVM. The agent submits buying or selling orders to the exchange to adjust the weights of each asset according to $\mathbf{w}_t$.

After receiving the profit or loss from the exchange, the agent will perform online training to fine-tune the network, as offline training does.

5. Experiments

5.1. Experimental setup

5.1.1. Datasets

We evaluate on two real-world datasets, NASDAQ and NYSE, both are from [31] and publicly available.[1] We use historical daily price data,[2] sector-industry relation data[3] and wiki company-based relation data[4] of NASDAQ and NYSE markets in this paper. Detailed descriptions of the two relations are summarized in Table 2. Note that for GPM, we present the better performance between the two relations if not otherwise specified. The historical price data is from 2013-01-02 to 2017-12-08 (details are shown in Table 3). In our experiments, the period is divided into 80% training period for offline training and 20% back-testing period for online learning. Since portfolio selection is beyond the scope of this paper, following several previous research [6,26], we choose portfolio assets in order of transaction volume. Specifically, we select the top m stocks with the highest trading volume in the last 30 days of the training period.

Figs. 5 and 6 present the price variations of NASDAQ and NYSE, respectively, where AAPL, MU, etc., in the figures represent stocks in the portfolio. It can be observed that NASDAQ is more volatile while NYSE is relatively stable [37]. In both figures, we present only the top 11 volumed stocks, which are also the components of the small portfolio used in our paper. The training period is on the left side of the black line and the test period is on the right side. Most stocks in the portfolio in NASDAQ keep rising while most stocks except for BAC and C in NYSE exhibit irregular fluctuations.

5.1.2. Compared methods

To evaluate the performance, we compare the proposed **GPM** method and GPM with $T = 1$ (**GPM-T1**) with state-of-the-art RL-based methods and some well-known traditional portfolio trading strategies, including,

- Reinforcement learning based approaches.[5] **PPN** [26] uses a two-stream portfolio policy network that extracts price series patterns and asset correlations respectively.**EI^3** [7] uses CNN with multiple filter sizes to extract multi-scale temporal features.**EIIE CNN**, **RNN** and **LSTM** [6] extract price features independently while sharing network parameters.Integrated CNN (**iCNN**) [5] extracts the portfolio feature via convolutional neural network and allocates weights via fully connected layer.
- Benchmarks.Uniform Buy and Hold (**UBAH**) [2] deploys the funds into assets equally in the beginning and holds them without buying and selling.Uniform Constant Rebalanced Portfolios (**UCRP**) [38] reallocates the portfolio weights equally per period.
- Follow-the-winner approaches.Methods in which assets that make higher profits are given more weight, including Universal Portfolios (**UP**) [38], Exponential Gradient (**EG**) [39], Online Newton Step (**ONS**) [3].
- Follow-the-loser approaches.Methods in which poor-performing assets are more heavily weighted based on the mean reversion theory, including **Anticor** [40], Passive Aggressive Mean Reversion(**PAMR**) [41], Online Moving Average Reversion (**OLMAR**) [42], Confidence Weighted Mean Reversion (**CWMR**) [43], Weighted Moving Average Mean Reversion (**WMAMR**) [44], Robust Median Reversion (**RMR**) [4].
- Pattern-matching based approaches. Methods that first select the similar historical price relatives and then learn an optimal portfolio based on the similarity set, including **M0** [1], Nonparametric Kernel Based Log Optimal Strategy (**B^K**) [45], and Correlation-driven Nonparametric Learning (**CORN**) [46].

5.1.3. Performance metrics

We evaluate the performance of the above methods via the following three commonly used metrics,

- final Accumulative Portfolio Value (fAPV) that measures the accumulated return compared with the initial capital,

$$\text{fAPV} = \frac{p_T}{p_0} \tag{17}$$

- Sharpe Ratio (SR) that measures the profitability of an investment when considering risk,

$$\text{SR} = \frac{\mathbb{E}_t[\rho_t - \rho_F]}{\sqrt{\text{var}_t(\rho_t - \rho_F)}} \tag{18}$$

where $\rho_t = p_t/p_{t-1} - 1$ is portfolio's rate of return, ρ_F is the rate of return of risk-free asset (cash in this paper) and it is always 0.

- Maximum DrawDown (MDD) that measures the maximum of decline from a peak in accumulative portfolio value,

$$\text{MDD} = \max_{\tau\in(0,T)}\left(\max_{t\in(0,\tau)}\frac{p_t - p_\tau}{p_t}\right) \tag{19}$$

The three metrics measure different aspects of performance. Higher fAPV indicates better profitability, higher SR represents higher profits considering risk, and lower MDD means a lower risk. Note it is difficult to excel in all metrics, especially in a volatile market, since it usually takes more risks to reap the benefits.

5.1.4. Training settings

The filter size of the first layer of short-term price movement feature extraction branch is 1×3 and therefore the second layer is 1×14. The filter size of the first layer of mid-term price movement feature extraction branch is 1×8 and therefore the second layer is 1×9. The filter size of the first layer of long-term price movement feature extraction branch is 1×14 and therefore the second layer is 1×3. We train 3000 steps with batch size 30 in training period and perform rolling training 10 steps after taking an action in back-testing period. To evaluate the adaptability and scalability of the evaluated methods, we construct a small portfolio and a large portfolio respectively for each market. Specifically, we select the top $m = 11$ stocks with the most volume in the last 30 days of the training period as the small portfolio, and choose the top $m = 300$ stocks as the large portfolio. The learning rate is 0.001 and decays 0.9 per 1000 steps. The Adam algorithm is utilized to optimize the learning rate. We tune temperature T within $\{1, 500, 1000, 1500, 2000, 3000, 4000\}$.

5.2. Overall performance

Table 4 reports the results of the evaluated PM methods on NASDAQ and NYSE datasets with both small and large portfolios, where the best performance is in boldface. As can be observed, RL-based methods in general can obtain better results as compared to traditional portfolio trading strategies, which reflects the excellent ability of RL in finding policies on interactive long-running PM tasks. Within all the RL-based PM methods, GPM achieves the

[1] https://github.com/hennande/Temporal_Relational_Stock_Ranking.
[2] https://www.google.com/finance.
[3] https://www.nasdaq.com/screening/industries.aspx.
[4] https://www.mediawiki.org/wiki/Wikibase/DataModel/JSON.
[5] Since the source code of AlphaStock is not available and hard to reproduce, we do not include it as a competitor.

Table 2
Statistics of Relations in NASDAQ and NYSE.

Datasets	#Stocks	Sector-Industry		Wikidata	
		#Relations	#Edges	#Relations	#Edges
NASDAQ	1026	96	53596	42	2060
NYSE	1737	107	284113	32	8766

Table 3
Statistics of Train and Test Datasets.

	Period	#Examples
Train	2013-01-02 to 2016-12-31	979
Test	2017-01-01 to 2017-12-08	232

highest fAPV in both NASDAQ and NYSE within both small and large portfolios. This demonstrates the effectiveness of the design philosophy of GPM, i.e., incorporating asset relations for better feature extraction, exploiting temperature for better diversity and eventually better PM performance. GPM also achieves the highest SR with both portfolio sizes in both markets. This demonstrates that GPM still exhibits good performance even when considering risks. For MDD, GPM achieves high MDD in NASDAQ and low MDD in NYSE. The reason is that MDD depends on the price volatility of the financial market, and NASDAQ is relatively volatile while NYSE is more stable. It is difficult to guarantee high return and low risk at the same time in a volatile market. Therefore, a high MDD is acceptable considering its high profitability achieved by GPM in NASDAQ. In NYSE, GPM's MDD is relatively low (though not the lowest when $m = 300$). This indicates that GPM can guarantee both high return and low risk in a stable market. Though the performance of GPM decreases a lot with $T = 1$ (GPM-T1), it still outperforms other state-of-the-art methods in terms of profit. This demonstrates the effectiveness of extracting asset relational features for better portfolio management performance. Moreover, the performance gap between GPM and GPM-T1 further reveals the great potential of diversity in portfolio management. Among the remaining four categories of methods, Pattern-matching and Benchmarks perform the best, followed by Follow-the-winner and Follow-the-loser.

In NASDAQ, The performance of portfolio with $m = 300$ is generally much worse than that with $m = 11$, while in NYSE, the two perform similarly. This demonstrates that large portfolios are more difficult to manage as compared to small assets in volatile markets, mainly due to the huge differences in price variances among different stocks, which makes the agent difficult to pick the proper stocks to deploy appropriate weights. However, GPM is still able to maintain high profitability even with large portfolio in volatile market, which verifies GPM's robustness under different portfolio sizes.

5.3. Performance over back-testing period

We also compare the performance of GPM over back-testing period with PPN, EI[3] and other four methods that each achieves the best performance in the four traditional strategy categories.

Fig. 7 depicts the fluctuation of the evaluated PM methods' portfolio values along with time in NASDAQ with small portfolio ($m = 11$). GPM outperforms all the state-of-the-art PM methods. GPM-T1 and PPN exhibit outstanding performance and pattern-matching method M0 performs the fourth-best. GPM-T1 and PPN perform highly similarly, mainly because they deploy most weights into the same stocks. In the beginning, the portfolio values of GPM-T1 and PPN are lower than other methods. But after May 1, they become much higher than other methods. However, in the final stage, their portfolio values drop down rapidly. This illustrates why they can get the highest fAPV but high MDD, which indicates the coexistence of high returns and high risks.

Fig. 8 depicts the fluctuation of the evaluated PM methods' portfolio values along with time in NASDAQ with large portfolio ($m = 300$). GPM has put up winning margins so large that it is one of the few methods that can perform as well as within small portfolio. GPM exhibits the same change pattern as in the case of $m = 11$. As can be observed, it stands out after May 1, the reason lies in that it deploys more weights into the same stocks as GPM and PPN do in the small portfolio ($m = 11$). GPM-T1 also performs well most of the time except for the last few days.

Fig. 9 depicts the performance of the evaluated PM methods along with time in NYSE with small portfolio ($m = 11$). Note that GPM performs the best when $T = 1$, so GPM is equal to GPM-T1 in this experiment. As can be seen, The portfolio value of GPM is stable before June 10, and then exhibits a steady growth. Therefore, it has high fAPVs but low risk. Fig. 10 depicts the performance of the evaluated methods along with time in NYSE with large portfolio ($m = 300$). It can be observed that GPM and GPM-T1 outperform all the portfolio management methods, and GPM shows huge performance advantage.

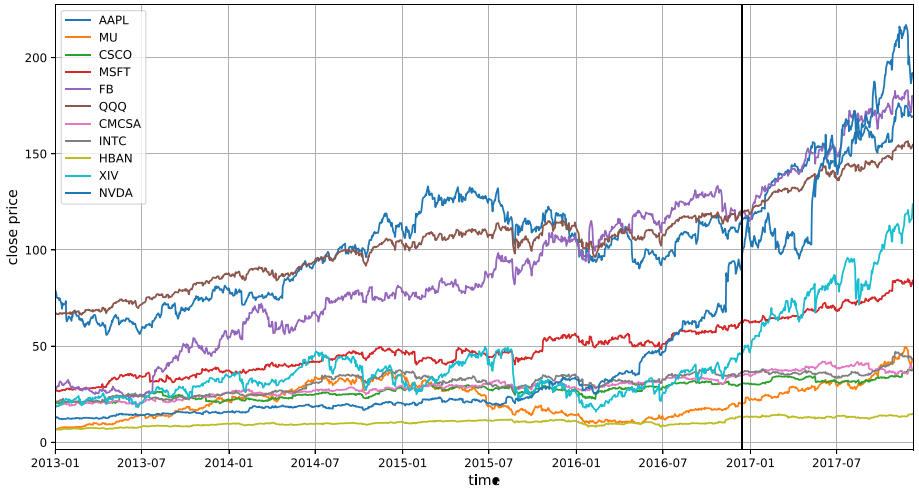

Fig. 5. Price variation of NASDAQ (#asset = 11).

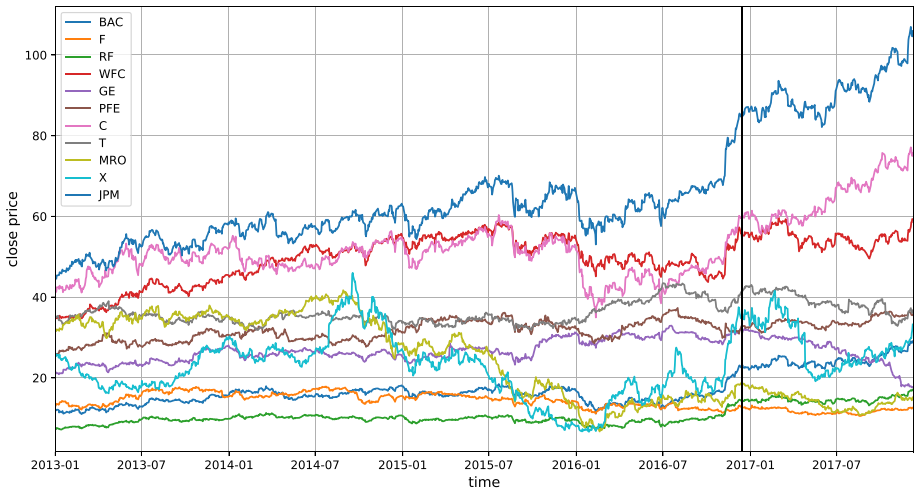

Fig. 6. Price variation of NYSE (#asset = 11).

Table 4
Performances of evaluated PM methods under NASDAQ and NYSE. Best performance is in boldface.

Market	NASDAQ						NYSE					
#Asset	m = 11			m = 300			m = 11			m = 300		
Metrics	fAPV	SR	MDD	fAPV	SR	MDD	fAPV	SR	MDD	fAPV	SR	MDD
GPM-T1	1.8687	0.1182	0.1984	1.4347	0.0963	0.1394	**1.1939**	**0.0792**	**0.1223**	1.1805	0.0611	0.1853
GPM	**2.0665**	**0.1758**	0.1396	**1.9715**	**0.1370**	0.1396	**1.1939**	**0.0792**	**0.1223**	**1.3371**	**0.1433**	0.0540
PPN	1.8311	0.1142	0.1984	1.1517	0.0875	0.0725	1.0020	0.0054	0.1475	1.0650	0.0523	0.0525
EI3	1.4971	0.1257	0.1353	1.1541	0.1029	0.0573	1.0037	0.0062	0.1456	1.0694	0.0556	0.0524
iCNN	1.4560	0.1562	0.0632	0.8107	−0.0324	0.3146	0.9424	−0.0197	0.2074	1.1458	0.0715	0.0990
CNN	1.4892	0.1235	0.1369	1.1543	0.1030	0.0570	1.0001	0.0043	0.1413	1.0690	0.0554	0.0522
RNN	1.4464	0.1721	0.0522	1.1511	0.1011	0.0578	1.0091	0.0089	0.1486	1.0688	0.0550	0.0527
LSTM	1.4466	0.1724	0.0523	1.1507	0.1007	0.0576	1.0069	0.0078	0.1483	1.0690	0.0552	0.0522
UBAH	1.4243	0.1637	0.0503	1.1742	0.1132	**0.0527**	1.0122	0.0106	0.1349	1.0801	0.0663	**0.0448**
UCRP	1.4031	0.1719	0.0481	1.1482	0.0999	0.0572	1.0067	0.0076	0.1369	1.0659	0.0530	0.0535
UP	1.4066	0.1713	0.0485	1.1482	0.0999	0.0572	1.0068	0.0077	0.1371	1.0664	0.0534	0.0533
EG	1.4044	0.1715	0.0482	1.1496	0.1006	0.0569	1.0071	0.0078	0.1369	1.0666	0.0537	0.0527
ONS	1.2521	0.1615	**0.0366**	0.6696	−0.0530	0.5116	0.9463	−0.0161	0.1604	0.8622	−0.0143	0.3663
Anticor	1.2919	0.0933	0.0790	1.4602	0.1026	0.1370	0.6778	−0.1002	0.3565	0.9225	−0.0158	0.2370
PAMR	0.5215	−0.1212	0.5094	0.5186	−0.1154	0.5230	0.4302	−0.1749	0.6194	0.4179	−0.1641	0.6519
CWMR	0.5079	−0.1262	0.5246	0.5080	−0.1185	0.5341	0.4047	−0.1878	0.6428	0.4186	−0.1639	0.6543
OLMAR	0.7305	−0.0456	0.3343	0.6089	−0.0417	0.5436	0.4569	−0.1563	0.6073	0.3355	−0.1336	0.6992
WMAMR	0.8868	−0.0085	0.3515	0.8809	−0.0095	0.3125	0.6917	−0.0797	0.4570	0.8429	−0.0232	0.3579
RMR	0.7163	−0.0510	0.3590	0.8828	0.0013	0.3545	0.4530	−0.1597	0.5962	0.3750	−0.1344	0.6596
M0	1.5940	0.1536	0.0834	1.1954	0.1153	0.0602	0.9372	−0.0183	0.2352	1.0834	0.0557	0.0822
B^K	1.1366	0.0358	0.2396	1.1504	0.0569	0.1190	0.7670	−0.0771	0.3413	1.1711	0.0446	0.2201
CORN	0.8254	−0.0304	0.2990	0.9608	0.0012	0.2524	0.8559	−0.0614	0.2309	0.5050	−0.1474	0.5041

5.4. Evaluation on relations and graph neural networks

In this set of experiments, we evaluate GPM with different graph neural networks to see how different GNNs perform. We also evaluate the impact of the number of GNN layers on fAPVs. Take GPM but without graph neural network (noGNN) as a baseline, we apply different graph neural networks, including GCN [27], TGC [31] and R-GCN [10], on GPM. For GCN, we assume there is an edge between two nodes once if they have at least one type of relations. The fAPV performance of GPM with different GNNs is shown in Fig. 11. As can be observed, R-GCN exhibits the best performance in both datasets, demonstrating its excellent capability of handing multi-relational data. Moreover, the results also indicate that leveraging different relations may result in different performance in different markets.

Performance of GCN and TGC highly depends on the relations to which the graph corresponds. For GCN, wikidata relation is more suitable than sector-industry relation as base graph in both NASDAQ and NYSE markets. For TGC, trading with sector-industry relations in NASDAQ and wikidata relations in NYSE gets higher profits. As for R-GCN, trading with both sector-industry and wikidata gets high profits. This demonstrates that the influence of companies in the same industry and the relations in wididata helps the agent to make better trading decisions.

We compare different graph neural network with better performance within sector-industry and wikidata, i.e., GCN and R-GCN with wikidata and TGC with sector-industry in NASDAQ market, and GCN, TGC and R-GCN with wikidata in NYSE market. In general, the introduction of appropriate relations is of great help for profit. Because in GCN methods, we omit the edge types to transform the heterogeneous graph into a homogeneous one, edge type information is ignored. However, it still obtains higher profits than the baseline method noGNN, which reveals that relations even without types between companies are helpful for making trading

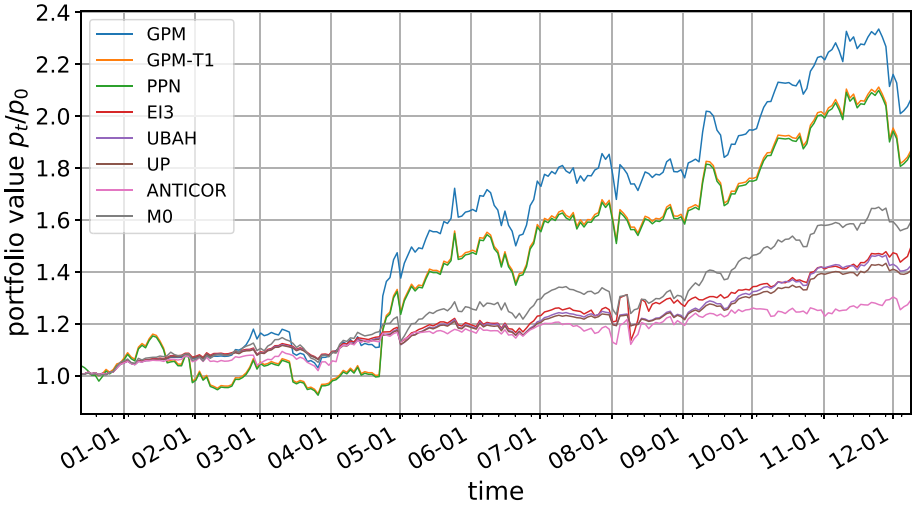

Fig. 7. Performance over back-testing period in NASDAQ (#asset = 11).

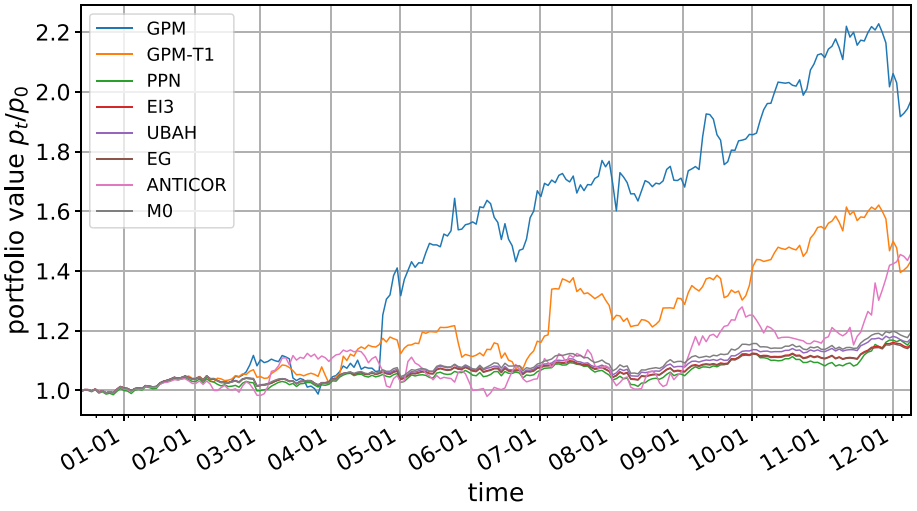

Fig. 8. Performance over back-testing period in NASDAQ (#asset = 300).

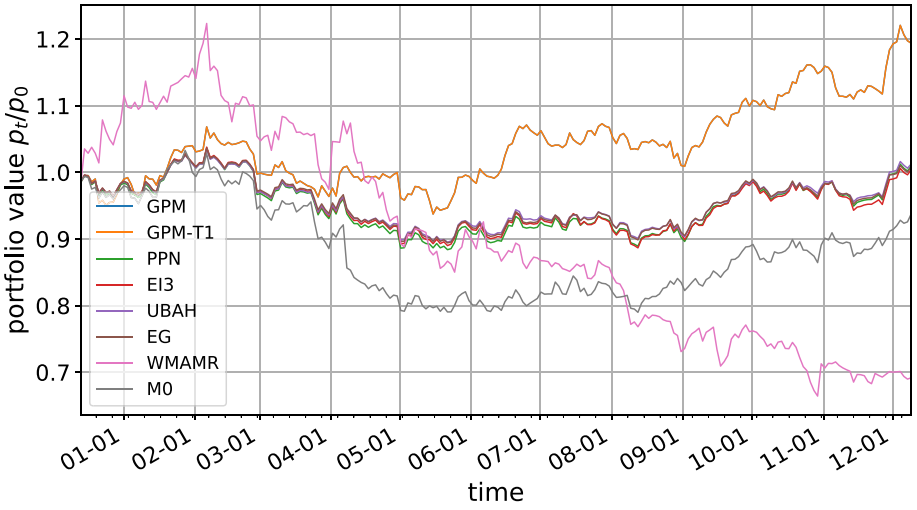

Fig. 9. Performance over back-testing period in NYSE (#asset = 11).

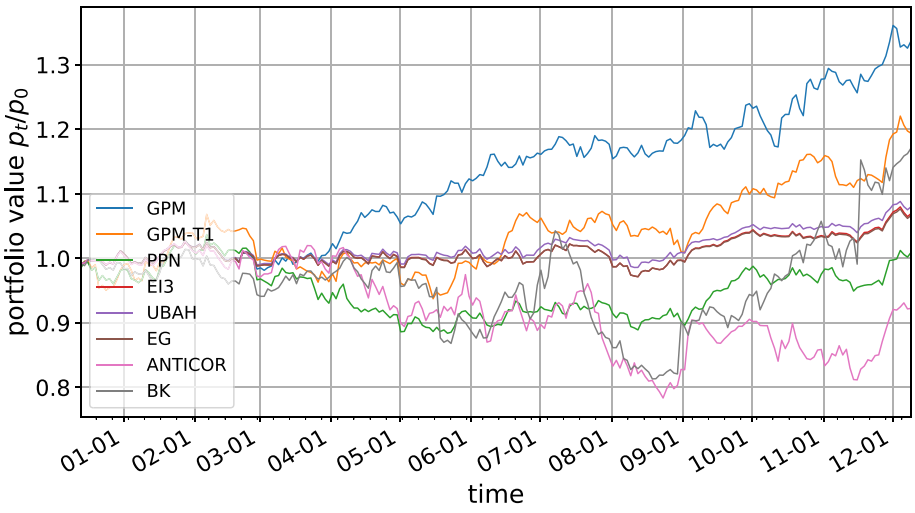

Fig. 10. Performance over back-testing period in NYSE (#asset = 300).

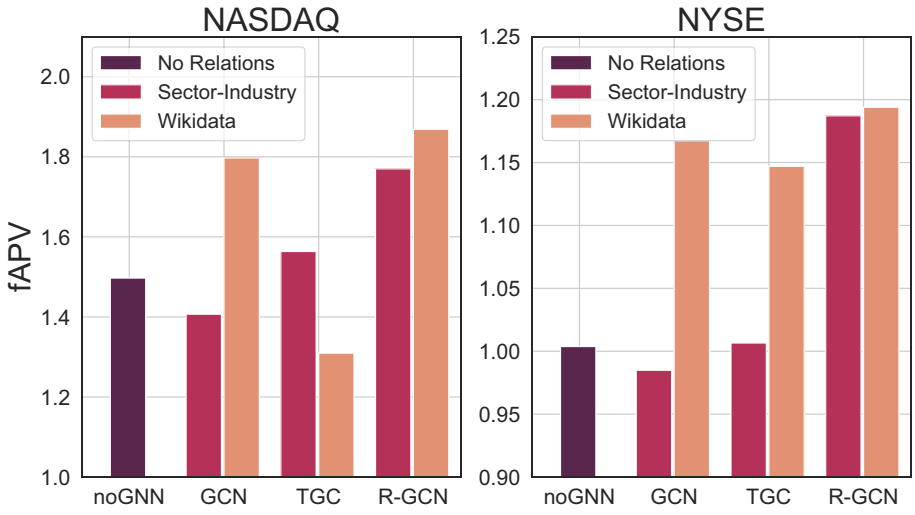

Fig. 11. Impact of different GNNs.

decisions. Though TGC does not perform as well as GCN and R-GCN, it still outperforms the benchmark noGNN. R-GCN methods considering edge types achieve even higher fAPV than GCN methods, demonstrating that relation types have positive impact on making profits.

We also evaluate the impact of the number of R-GCN layers, and the fAPV results are shown in Fig. 12. It can be observed that GNN with only 1 layer can obtain much higher fAPV than the method without GNN, demonstrating the importance of exploiting asset relations in PM. In general, the best performance can be achieved with 2 layers in both datasets, indicating that stacking more layers in R-GCN dose not necessary lead to better performance.

5.5. Evaluation on temperature

In this set of experiments, we study the impact of hyper-parameter temperature T. Fig. 13 depicts the variation process of the portfolio vector over back-testing period with temperature $T = [1, 3000, 6000]$ in NASDAQ with sector-industry relational graph. As can be observed, in the beginning, the agent tends to allocate weights equally. However, after a period of exploration, portfolio vectors with different temperature present different patterns. When $T = 1$, which represents a standard softmax function, trading agent devotes most of the weights into three assets. Since the stock prices are fluctuating, the agent adjusts the portfolio vector according to the price movement of assets. When $T = 3000$, the agent first assigns weights equally, and then devotes most of the weights into the same three assets for a short time. Finally, it allocates most of the weights to four assets. When $T = 6000$, the agent allocates funds uniformly and remains unchanged. In such a case, it actually becomes the UCRP trading strategy. On the whole, with the increase of the temperature, the diversity has also increased. However, too high temperature T makes trading agent re-balance portfolio weights uniformly and lost the adjustment ability.

We further study the impact of temperature T on profits. Fig. 14 shows the portfolio performance with different numbers of assets and temperatures in market NASDAQ and NYSE with wikidata relational graph. The best performance is colored in red. In general, standard softmax function with $T = 1$ cannot achieve the best performance, except for portfolio with 11 assets in NYSE. Moreover, it can be observed that in most cases, with the increase of temperature T, fAPV first increases and then drops down. When temperature $T = 1$, most of the funds are concentrated in a few assets and other assets that could also benefit are ignored. High temper-

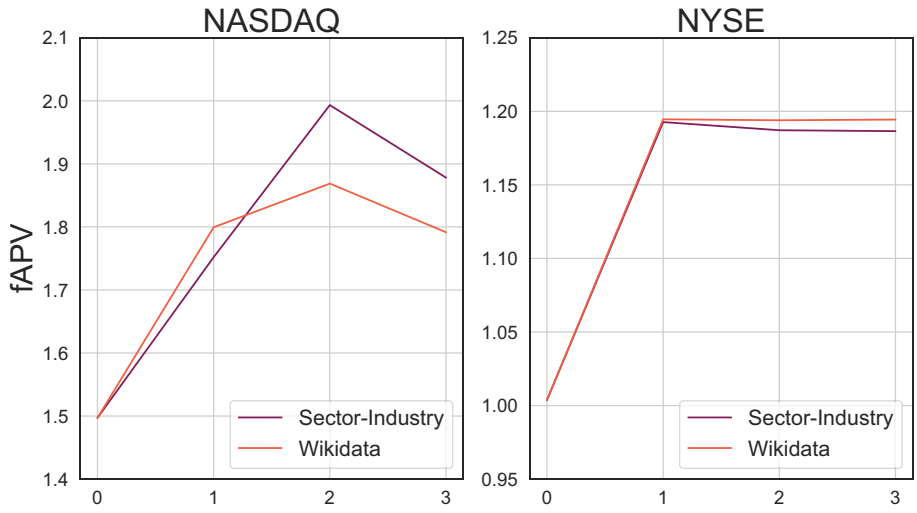

Fig. 12. Impact of number of graph neural network layers.

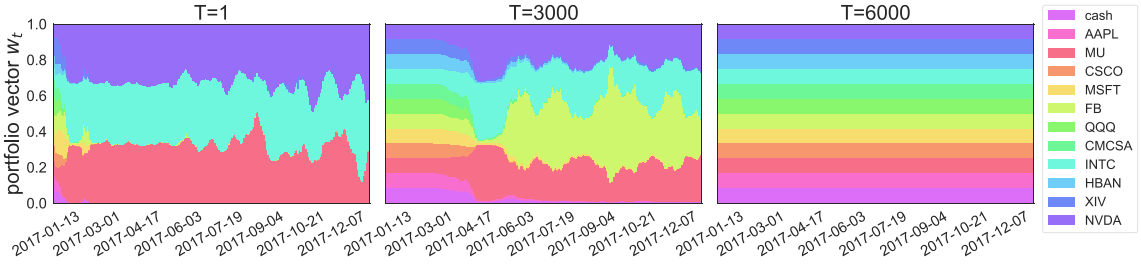

Fig. 13. Portfolio vectors with different temperature T over back-testing period.

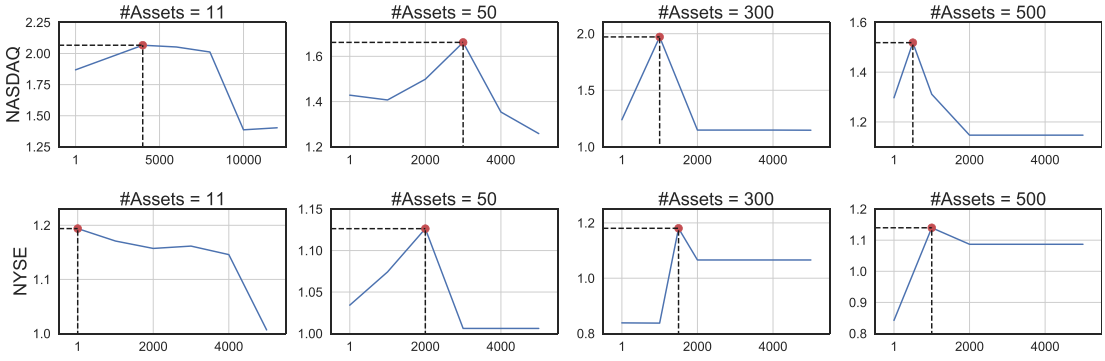

Fig. 14. The fAPVs of Portfolio with different numbers of assets and temperatures.

ature T produces a softer probability distribution over the assets, which makes assets with low probability get more attention. Hence, the agent will concentrate on more assets that are also profitable. However, high temperature does not always yield high profits. When the temperature becomes too high, the agent pays attention to not only profitable assets but also unprofitable assets, leading to low fAPV. Finally, the probability distribution becomes the uniform distribution, and the increase of the temperature will no longer affect the profits.

It is also shown in Fig. 14 that managing a large number of assets is difficult for a trading agent. For example, when temperature $T = 1$, the fAPV in NASDAQ decreases from 1.87 to 1.43, 1.24 and 1.30, with the increase of the asset number from 11 to 50, 300 and 500. A similar trend can also be observed in NYSE. The reason is that the agent with softmax function usually devotes most of its funds into a few assets and is difficult to choose the most profitable among a large number of assets. However, choosing a proper temperature can alleviate this problem. Raising the temperature softens probability distribution over the assets, which reduces the risk and enhances the robustness.

In summary, from the experimental results, we can conclude that the introduction of R-GCN for multiple asset relations and temperature for better diversity is beneficial for profit and profit at risk in portfolio management. By applying R-GCN, different types of company relationships can be utilized for trading decisions, which is of great help for profit. Applying a proper temperature can increase the diversity of a portfolio to make balance between profit and risk. Together with multi-scale temporal features, GPM exhibits good performance in portfolio management in both volatile and stable environments.

6. Conclusion

In this paper, we proposed GPM, a novel graph convolutional network based reinforcement learning framework for portfolio management. By first constructing an asset relation graph and then utilizing relational graph convolutional network to extract the relational features, we incorporate the asset relations into consideration when learning the asset features. Combining with the multi-scale temporal feature extracted by a network similar to EI^3, we have substantially enriched the feature representation of each asset, which in turn leads to better portfolio management performance. Moreover, we use softmax with temperature to produce softer distribution and hence increase portfolio diversity, which leads to further increase in profits. Extensive experiments on two real-world datasets, NASDAQ and NYSE, demonstrate the superiority of GPM over existing RL-based PM methods.

CRediT authorship contribution statement

Si Shi: Conceptualization, Methodology, Software, Writing – original draft. **Jianjun Li:** Supervision, Resources, Writing – review & editing. **Guohui Li:** Supervision, Project administration. **Peng Pan:** Supervision. **Qi Chen:** Data curation. **Qing Sun:** Project administration.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] A. Borodin, R. El-Yaniv, V. Gogan, On the competitive theory and practice of portfolio selection, in: LATIN, vol. 1776 of Lecture Notes in Computer Science, Springer, 2000, pp. 173–196.
[2] B. Li, S.C.H. Hoi, Online portfolio selection: A survey, ACM Computing Survey 46 (3) (2014) 35:1–35:36.
[3] A. Agarwal, E. Hazan, S. Kale, R.E. Schapire, Algorithms for portfolio management based on the newton method, in: Proceedings of ICML, vol. 148, ACM, 2006, pp. 9–16.
[4] D. Huang, J. Zhou, B. Li, S.C.H. Hoi, S. Zhou, Robust median reversion strategy for online portfolio selection, IEEE Trans. Knowl. Data Eng. 28 (9) (2016) 2480–2493.
[5] Z. Jiang, J. Liang, Cryptocurrency portfolio management with deep reinforcement learning, in: IntelliSys, 2017, pp. 905–913.
[6] Z. Jiang, D. Xu, J. Liang, A deep reinforcement learning framework for the financial portfolio management problem, CoRR abs/1706.10059.
[7] S. Shi, J. Li, G. Li, P. Pan, A multi-scale temporal feature aggregation convolutional neural network for portfolio management, in: Proceedings of the Conference on Information and Knowledge Management (CIKM), ACM, 2019, pp. 1613–1622.
[8] Y. Ye, H. Pei, B. Wang, P. Chen, Y. Zhu, J. Xiao, B. Li, Reinforcement-learning based portfolio management with augmented asset movement prediction states, Proceedings of AAAI (2020) 1112–1119.
[9] J. Wang, Y. Zhang, K. Tang, J. Wu, Z. Xiong, Alphastock: A buying-winners-and-selling-losers investment strategy using interpretable deep reinforcement attention networks, in: KDD, ACM, 2019, pp. 1900–1908.
[10] M.S. Schlichtkrull, T.N. Kipf, P. Bloem, R. van den Berg, I. Titov, M. Welling, Modeling relational data with graph convolutional networks, in: ESWC, Vol. 10843 of Lecture Notes in Computer Science, Springer, 2018, pp. 593–607.
[11] Y. Zheng, B. Chen, T.M. Hospedales, Y. Yang, Index tracking with cardinality constraints: A stochastic neural networks approach, in: AAAI, AAAI Press, 2020, pp. 1242–1249.
[12] Y. Kwak, J. Song, H. Lee, Neural network with fixed noise for index-tracking portfolio optimization, Expert Syst. Appl. 183 (2021) 115298.
[13] G. Jeong, H.Y. Kim, Improving financial trading decisions using deep q-learning: Predicting the number of shares, action strategies, and transfer learning, Expert Syst. Appl. 117 (2019) 125–138.
[14] J. Chakole, M.P. Kurhekar, Trend following deep q-learning strategy for stock trading, Expert Syst. J. Knowl. Eng. 37 (4).
[15] Y. Deng, F. Bao, Y. Kong, Z. Ren, Q. Dai, Deep direct reinforcement learning for financial signal representation and trading, IEEE Trans. Neural Networks Learn. Syst. 28 (3) (2017) 653–664.
[16] X. Wu, H. Chen, J. Wang, L. Troiano, V. Loia, H. Fujita, Adaptive stock trading strategies with deep reinforcement learning methods, Inf. Sci. 538 (2020) 142–158.
[17] J. Li, R. Rao, J. Shi, Learning to trade with deep actor critic methods, in: ISCID (2), IEEE, 2018, pp. 66–71..
[18] H. Markowitz, Portfolio selection, J. Finance 7 (1) (1952) 77–91.
[19] X. Jin, Time-varying return-volatility relation in international stock markets, Int. Rev. Econ. Finance 51 (2017) 157–173.
[20] V. Bergen, M. Escobar, A. Rubtsov, R. Zagst, Robust multivariate portfolio choice with stochastic covariance in the presence of ambiguity, Quantitative Finance 18 (8) (2018) 1265–1294.
[21] G.-J. Wang, C. Xie, H.E. Stanley, Correlation structure and evolution of world stock markets: Evidence from pearson and partial correlation-based networks, Comput. Econ. 51 (3) (2018) 607–635.
[22] C. Boghean, M. State, The relation between foreign direct investments (fdi) and labour productivity in the european union countries, Proc. Econ. Finance 32 (2015) 278–285.
[23] W. Zhang, X. Li, D. Shen, A. Teglio, Daily happiness and stock returns: Some international evidence, Physica A 460 (2016) 201–209.
[24] Y. Tang, J.J. Xiong, Y. Luo, Y. Zhang, How do the global stock markets influence one another? evidence from finance big data and granger causality directed network, Int. J. Electron. Commer. 23 (1) (2019) 85–109.
[25] K.H. Al-Yahyaee, W. Mensi, I.M.W. Al-Jarrah, A.K. Tiwari, Testing for the granger-causality between returns in the us and gipsi stock markets, Physica A 531 (2019) 120950.
[26] Y. Zhang, P. Zhao, Q. Wu, B. Li, J. Huang, M. Tan, Cost-sensitive portfolio selection via deep reinforcement learning, IEEE Trans. Knowl. Data Eng. (2020) 1–18.
[27] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, Proceedings of ICLR (2017).
[28] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio, Graph attention networks, Vol. abs/1710.10903, 2017. arXiv:1710.10903. URL:http://arxiv.org/abs/1710.10903.
[29] T.N. Kipf, M. Welling, Variational graph auto-encoders, CoRR abs/1611.07308..
[30] Y. Chen, Z. Wei, X. Huang, Incorporating corporation relationship via graph convolutional neural networks for stock price prediction, in: Proceedings of the Conference on Information and Knowledge Management (CIKM), ACM, 2018, pp. 1655–1658.
[31] F. Feng, X. He, X. Wang, C. Luo, Y. Liu, T. Chua, Temporal relational ranking for stock prediction, ACM Trans. Inf. Syst. 37 (2) (2019) 27:1–27:30.
[32] R. Kim, C.H. So, M. Jeong, S. Lee, J. Kim, J. Kang, HATS: A hierarchical graph attention network for stock movement prediction, CoRR abs/1908.07999.
[33] J. Ye, J. Zhao, K. Ye, C. Xu, Multi-graph convolutional network for relationship-driven stock movement prediction, CoRR abs/2005.04955.
[34] F. Soleymani, E. Paquet, Deep graph convolutional reinforcement learning for financial portfolio management - deeppocket, Expert Syst. Appl. 182 (2021) 115127.
[35] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, M.A. Riedmiller, Deterministic policy gradient algorithms, in: Proceedings of ICML, vol. 32, 2014, pp. 387–395.
[36] G.E. Hinton, O. Vinyals, J. Dean, Distilling the knowledge in a neural network, CoRR abs/1503.02531.
[37] G.W. Schwert, Stock volatility in the new millennium: how wacky is nasdaq?, J Monetary Econ. 49 (1) (2002) 3–26.
[38] T.M. Cover, Universal portfolios, Math. Finance 1 (1) (1991) 1–29.
[39] D.P. Helmbold, R.E. Schapire, Y. Singer, M.K. Warmuth, On-line portfolio selection using multiplicative updates, in: Proceedings of ICML, Morgan Kaufmann, 1996, pp. 243–251.
[40] A. Borodin, R. El-Yaniv, V. Gogan, Can we learn to beat the best stock, Proceedings of NIPS (2003) 345–352.
[41] B. Li, P. Zhao, S.C.H. Hoi, V. Gopalkrishnan, PAMR: passive aggressive mean reversion strategy for portfolio selection, Mach. Learn. 87 (2) (2012) 221–258.
[42] B. Li, S.C.H. Hoi, On-line portfolio selection with moving average reversion, in: Proceedings of the 29th International Conference on Machine Learning (ICML), 2012, pp. 273–280.
[43] B. Li, S.C.H. Hoi, P. Zhao, V. Gopalkrishnan, Confidence weighted mean reversion strategy for online portfolio selection, ACM Trans. Knowl. Discov. Data 7 (1) (2013) 4:1–4:38.
[44] L. Gao, W. Zhang, Weighted moving average passive aggressive algorithm for online portfolio selection, in: IHMSC, vol. 1, IEEE, 2013, pp. 327–330.
[45] L. Györfi, G. Lugosi, F. Udina, Nonparametric kernel-based sequential investment strategies, Mathematical Finance: An International Journal of Mathematics, Statistics and Financial, Economics 16 (2) (2006) 337–357.
[46] B. Li, S.C.H. Hoi, V. Gopalkrishnan, CORN: correlation-driven nonparametric learning approach for portfolio selection, ACM Trans. Intell. Syst. Technol. 2 (3) (2011) 21:1–21:29.

Si Shi received the B.S. degree in information management (2014) and the M.S. degree in management information systems (2017) from Zhongnan University of Economics and Law, Wuhan, China. She is currently working toward the PhD degree in computer science and technology at Huazhong University of Science and Technology, Wuhan, China. Her research interests include fintech, reinforcement learning and portfolio management.

Jianjun Li received the PhD degree in computer science from Huazhong University

of Science and Technology (HUST), Wuhan, China in 2012. He was a Senior Research Associate with the Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong. He is currently an associate professor at the School of Computer Science and Technology in HUST. His research interests include fintech, recommender systems, natural language processing, real-time systems, and advanced data management.

Guohui Li received the PhD degree in computer science from Huazhong University of Science and Technology (HUST), China in 1999. He was promoted to a full professor in 2004, and currently acts as dean of the School of Software Engineering in HUST. His research interests mainly include deep learning, bid data analysis, real-time systems and advanced data management.

Peng Pan received the PhD degree in computer science from Huazhong University of Science and Technology (HUST), Wuhan, China in 2008. He is currently an associate professor at the School of Computer Science and Technology in HUST. His research interests include big data analysis, deep learning and advanced data management.

Qi Chen received the PhD degree in computer science from Huazhong University of Science and Technology (HUST), Wuhan, China in 2020. He is currently a post doctor at the School of Computer Science and Technology in HUST. His research interests include recommender systems and deep learning.

Qing Sun received the B.S. degree in information management (1990) from JiLin University of Finance and Economics, Dalian, China. He is currently a manager at China Merchants Bank, Harbin Branch. His research interests include portfolio management and stock investment.