Intelligent Automation & Soft Computing

Tech Science Press

DOI: 10.32604/iasc.2024.042762

ARTICLE

Trading in Fast-Changing Markets with Meta-Reinforcement Learning

Yutong Tian[1], Minghan Gao[2], Qiang Gao[1,*] and Xiao-Hong Peng[3]

[1]School of Electronic and Information Engineering, Beihang University, Beijing, 100191, China

[2]School of Artificial Intelligence, Beijing University of Posts and Telecommunications, Beijing, 100191, China

[3]Faculty of Computing, Engineering and the Built Environment, Birmingham City University, Birmingham, B5 5JU, UK

*Corresponding Author: Qiang Gao. Email: gaoqiang@buaa.edu.cn

Received: 11 June 2023 Accepted: 21 December 2023 Published: 21 May 2024

ABSTRACT

How to find an effective trading policy is still an open question mainly due to the nonlinear and non-stationary dynamics in a financial market. Deep reinforcement learning, which has recently been used to develop trading strategies by automatically extracting complex features from a large amount of data, is struggling to deal with fast-changing markets due to sample inefficiency. This paper applies the meta-reinforcement learning method to tackle the trading challenges faced by conventional reinforcement learning (RL) approaches in non-stationary markets for the first time. In our work, the history trading data is divided into multiple task data and for each of these data the market condition is relatively stationary. Then a model agnostic meta-learning (MAML)-based trading method involving a meta-learner and a normal learner is proposed. A trading policy is learned by the meta-learner across multiple task data, which is then fine-tuned by the normal learner through a small amount of data from a new market task before trading in it. To improve the adaptability of the MAML-based method, an ordered multiple-step updating mechanism is also proposed to explore the changing dynamic within a task market. The simulation results demonstrate that the proposed MAML-based trading methods can increase the annualized return rate by approximately 180%, 200%, and 160%, increase the Sharpe ratio by 180%, 90%, and 170%, and decrease the maximum drawdown by 30%, 20%, and 40%, compared to the traditional RL approach in three stock index future markets, respectively.

KEYWORDS

Algorithmic trading; reinforcement learning; fast-changing market; meta-reinforcement learning

1 Introduction

Algorithmic trading, using mathematical models and computers to automate the buying and selling of financial securities, has created new opportunities as well as new challenges for the financial industry over the last few decades [1]. A variety of trading strategies, such as statistical arbitrage [2], momentum trading [3] and mean regression trading [4], have been proposed by applying mathematical and statistical methods. Due to the low signal-to-noise ratio of financial data and the non-linear nature of markets, the design of trading strategies for algorithmic trading systems is non-trivial [5].

The goal of trading strategies is to maximize the wealth accumulation during a trading period. The final wealth depends upon the sequences of interdependent trading actions in which optimal trading decisions do not just decide immediate trade returns but also affect subsequent future returns. The trading problem fits exactly with the framework of reinforcement learning (RL). Deep reinforcement learning (DRL), which effectively combines deep learning and reinforcement learning, has recently witnessed significant advances in various domains. Peters et al. [6] applied vanilla policy gradient (VPG) to robotics and evaluated its performance in hitting a baseball with an anthropomorphic arm. Schulman et al. [7] proposed a new family of policy gradient methods called proximal policy optimization (PPO) for robotic locomotion and Atari game playing.

Deep learning can automatically extract complex and implicit features from large amounts of data [8]. DRL combines the feature extraction capabilities of deep learning and the decision-making abilities of reinforcement learning, hence adapting for developing trading strategies in a highly noisy financial market with nonlinear dynamics. Andrew et al. [9] developed a paired trading strategy in the stock market that can be effectively learned and executed using a double deep Q-network. Lee et al. [10] presented a trading strategy that leverages wavelet transform and deep reinforcement learning, through decomposing input data within wavelet transformation and utilizing high-frequency data to perform deep reinforcement learning with a long short-term memory network.

Challenges remain in developing effective trading policies using DRL due to a combination of nonlinear and non-stationary dynamic behaviors exhibited within financial markets. The non-stationary behavior refers to the time-varying nature of the underlying distributions, which is marked by variations in the first, second, or higher moments shown in stochastic processes over time, and this is especially true in financial markets [11,12]. However, it is difficult for DRL to deal with non-stationary financial markets since it is notably sample inefficient. An optimal trading model can only be learned slowly through many iterative steps over many samples. Therefore, it is hard for DRL to keep pace with the time-varying nature of financial markets.

Meta-reinforcement learning (Meta-RL) can quickly adapt to new environment tasks by parameter updates using a small number of samples [13–16]. It has been used to tackle the control problem in a dynamic environment. Finn et al. [13] developed a model agnostic meta-learning (MAML) method and applied it in simulated continuous control environments. Maruan et al. [14] proposed a Meta-RL method for adaption in non-stationary locomotion and competitive multi-agent settings. Liaw et al. [15] put forward a meta-policy learning method which can realize efficient task executions in an autonomous driving simulator. Huang et al. [16] applied a meta-reinforcement learning method to grid emergency control, demonstrating its capability to quickly adapt to new power flow conditions and system dynamic parameters. To the best of our knowledge, there is no work that has investigated the realistic trading conditions and studied the possibility for trading in fast-changing financial markets using Meta-RL.

As aforementioned, RL methods may fail to adapt rapidly enough in highly dynamic financial environments because only a limited number of interactions are permitted before the market starts changing. This paper proposes a MAML-based trading method, which consists of a PPO meta-learner and a VPG learner, to deal with trading challenges encountered by traditional RL in a non-stationary financial market. Firstly, the training data is divided into multiple task data for fixed periods, so the market represented by each task data is relatively stationary during such a small period. Secondly, an easily adaptable trading policy is learned across the multiple task data by the PPO meta-learner. Finally, when trading in a new market task, the learned policy is updated by the VPG learner through the data available from the new task, and then the updated policy is used to trade in the rest part

of the market task. Moreover, to improve the adaptability of the MAML-based trading method in fast-changing market conditions, an ordered multiple-step updating mechanism is introduced in the MAML-based trading method.

The rest of this article is organized as follows. In Section 2, the proposed trading method based on MAML and its improvement are described in detail. The experimental results of different trading methods on IF300, CSI500, and Dow Jones index and the investigation on the effectiveness of mechanism updating are presented in Section 3. Section 4 delivers the conclusion of this study.

2 Methodology

In this section, we model the trading problem as a Markov decision process, introduce a traditional RL-based trading method, and propose the Meta-RL-based trading method and its improved version.

2.1 Markov Decision Process Formalization for Trading

The trading problem can be modeled as a Markov decision process, where a trading agent interacts with the market environment at discrete time steps. At each time step t, the agent receives a state s_t as a representation of the market, then chooses a trading action a_t and executes it on the market, and finally calculates the immediate profit as a reward R_{t+1} according to the new market state s_{t+1} at the next time step.

State space: Since the market state is partially observed, this paper uses the past price spreads of securities, technical indicators including MACD and RSI, and the last trading action to represent the market state.

The MACD index can be calculated as follows (Baz et al. [17]):

$$\begin{aligned} MACD_t &= DEA_t - DIF_t \\ DIF_t &= m\left(p_{t-12:t}\right) - m\left(p_{t-26:t}\right) \\ DEA_t &= m\left(DIF_{t-9:t}\right) \end{aligned} \tag{1}$$

$m\left(p_{t-n:t}\right)$ denotes the exponentially weighted moving average of prices over a time horizon n.

The RSI index (Wilder [18]) is

$$RSI = \frac{AUL}{AUL + ADL}. \tag{2}$$

AUL and ADL represent the average upward price movements and downward price movements over a period of time, respectively.

Action space: A discrete action space $\{-1, 0, +1\}$ is used for trading, where -1 corresponds to a maximal short position, 0 to no holdings, and $+1$ to a maximal long position.

Reward function: The value of the reward function at time t can be evaluated as follows, where bp represents the transaction cost rate.

$$R_t = a_t\left(p_t - p_{t-1}\right) - bp \cdot p_t \left|a_t - a_{t-1}\right| \tag{3}$$

The trading goal is to maximize the expected cumulative discounted rewards, where γ is the discounting factor.

$$G = \sum_{t=1}^{H} r^{t-1} R_t \tag{4}$$

2.2 Trading Method Based on VPG

For simplicity of calculation, the Vanilla Policy Gradients (VPG)-based method is taken as a baseline in this paper. In VPG, the trading policy π_θ, which takes the market state as the input and outputs the trading action, is parameterized using a neural network with parameters θ. The VPG learns an optimal policy by the stochastic gradient descent algorithm to minimize the loss, which is a negative expected cumulative reward defined in (4).

$$\theta = \theta - \alpha \nabla_\theta L(\theta) \tag{5}$$

According to the policy gradient theorem [19], the gradient can be calculated with the surrogate loss

$$L(\theta) = -E_{\tau \sim \pi_\theta} \left[\sum_{t=1}^{H} log \pi_\theta (a_t | s_t) \hat{A}_t \right] \tag{6}$$

$\tau = [s_0, a_0, r_1, s_1, a_1, r_2, \ldots]$ is a trajectory sampled by policy π_θ. $\hat{A}_t$ represents an advantage value over the trajectory assessing the good of action a_t at state s_t, which can be estimated by the generalized advantage function

$$\hat{A}_t = \delta_t + \gamma \lambda \delta_{t+1} + \ldots + (\gamma \lambda)^{H-t-1} \delta_{H-1} \tag{7}$$

λ is the hyperparameter. δ_t is the temporal difference error, which can be calculated based on the parameterized value function as follows:

$$\delta_t = R_{t+1} + \gamma V(s_{t+1}, w) - V(s_t, w) \tag{8}$$

The least squares method is used to estimate the value function

$$w^* = arg \min_w E_{\pi_\theta} \left[(G - V(s, w))^2 \right] \tag{9}$$

where the linear value function estimation is used to directly obtain the least squares solution of the value function weights.

An optimal trading policy is learned using VPG on the history trading data of a financial market, then the policy is used to trade in the market in the future. The learned trading policy will degenerate over time if the market is non-stationary and the market dynamic will go differently from that of the training market.

2.3 Trading Method Based on MAML

To deal with the trading challenges encountered by traditional RL, such as VPG, in a non-stationary financial market, this paper applies meta-reinforcement learning to seek trading policies with quick adaption to new environment tasks. Specifically, this paper proposes a MAML-based trading method consisting of a PPO meta-learner and a VPG learner.

The history trading data is firstly divided into a number of episodes with a fixed length H. A task data contains $N_s + 1$ consecutive episodes, where the first N_s episodes compose the support part and the last episode is the query part.

M task data are randomly sampled for each iteration during training. For task i, policy $\pi_{\theta_i'}$ can be achieved from an initial policy π_θ by the VPG learner with one step update on its support part.

$$\theta_i' = \theta - \alpha \nabla_\theta \frac{1}{N_s} \sum_{j=1}^{N_s} L_{i,j}\left(\tau_\theta^{1:K}\right) \tag{10}$$

where α is the step size. $L_{i,j}\left(\tau_\theta^{1:K}\right)$ indicates the surrogate loss on K trajectories produced by policy π_θ from the j-th episode of the support part of task i, which is calculated according to (6). The obtained policy can be used on the query part of the same task market since the market condition represented by one task data is relatively stationary for a short period.

A good initial policy π_θ is learned by the PPO meta-learner across multiple tasks from a random policy, which can quickly adapt to a new task market with one or a few update steps on a small number of samples. The parameter update of the initial policy on the query part of randomly sampled M tasks is

$$\theta = \theta - \beta \nabla_\theta \frac{1}{M} \sum_{i=1}^{M} L_{i;q}\left(\pi_{\theta_i'}\right) \tag{11}$$

where β is the step size for meta updates. The loss on the query part of task i, $L_{i;q}\left(\pi_{\theta_i'}\right)$, to be minimized is defined as

$$L_{i;q}\left(\pi_{\theta_i'}\right) = -\boldsymbol{E}_{s_t, a_t \sim \pi_{\theta_i'}}\left[min\left(r_t\left(\theta_i'\right)\hat{A}_t, \sim clip\left(r_t\left(\theta_i'\right), \varepsilon\right)\hat{A}_t\right)\right] \tag{12}$$

with

$$r_t\left(\theta_i'\right) = \frac{\pi_{\theta_i'}\left(a_t \middle| s_t\right)}{\pi_{\theta'_{old,i}}\left(a_t \middle| s_t\right)} \tag{13}$$

For each iteration of the meta-update process, $\pi_{\theta_{old}}$ is the initial policy without being optimized and π_θ is the recently optimized initial policy. The meta loss is computed in (12) and optimized with respect to θ for T steps, with each step being updated according to (11). $\pi_{\theta'_{old,i}}$ and $\pi_{\theta_i'}$ are achieved from $\pi_{\theta_{old}}$ and π_θ by the VPG learner with one step update on the support part of task i as given in (10), respectively.

The PPO is used as a meta-learner here since it can reuse the generated trajectories and prevent destructively large policy updates. Therefore, a good initial policy can be learned stably by the PPO meta-learner with a small number of tasks. The framework of the MAML-based trading method is shown in Fig. 1.

2.4 MAML-Based Trading Method with Ordered Multiple-Step Updating

To improve the adaptability of the MAML-based trading method in fast-changing markets, an ordered multiple-step updating mechanism is proposed to couple with the MAML framework. The dynamically changing market within a task is explored by updating the learned initial trading policy through sequential updates on the episodes of the support part data before being applied to trade in the query part of the task market. The ordered multiple-step updating mechanism is formulated as

$$\theta_i^{j+1} = \theta_i^j - \alpha \nabla_{\theta_i^j} L_{i,j}\left(\tau_{\theta_i^j}^{1:K}\right), j = 1, 2, \ldots, N_s \tag{14}$$

where $\theta_i^1 = \theta$ are the parameters of the common initial policy for every task market learned by the PPO meta-learner. The equation describes that the latest policy parameters θ_i^j after j-1 steps update from initial policy parameters are updated to θ_i^{j+1} with K trajectories on the j-th episode of the support part produced by $\pi_{\theta_i^j}$, which will in turn be updated on the (j+1)-th episode. The final policy $\pi_{\theta_i^{Ns+1}} = \pi_{\theta_i'}$ updated on the last episode of the support part will be used to trade in the query part of the same task market.

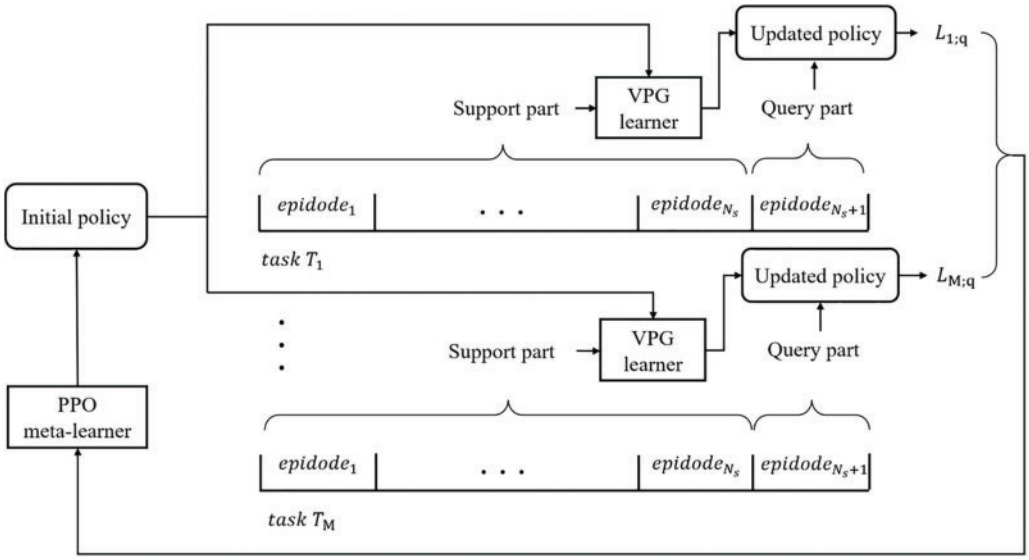


Figure 1: Framework of the MAML-based trading method

3 Experimental Results and Evaluation

The experimental results of different trading methods used on IF300, CSI500, and DJI are presented and analyzed in this section. The results of different updating mechanisms applied in the MAML method are also provided and evaluated.

3.1 Experimental Setup

This paper conducts experiments on the stock index futures of China IF300, CSI500, and the US Dow Jones stock index (DJI). The daily closing price data are used as shown in Fig. 2. The training period is from January 04, 2013 to November 26, 2020, and the testing period is from November 27, 2020 to November 25, 2022.

A Multi-Layer Perceptron with two hidden layers is used as the policy network, which takes the market state as input. The first and second hidden layers have 128 and 32 neurons, respectively. Both use a leaky ReLu activation function. The policy network outputs logits for each trading action, which are then input to the categorical distribution function to obtain the softmax-normalized trading action probabilities. During trading policy training the discount factor is set to 0.99 and the GAE parameter 0.9. The horizon of the episode is 20 days. Other specific hyperparameter settings for MAML and VPG are shown in Table 1. The MAML model and the VPG model are trained for 12000 iterations and 1200 iterations, respectively, when they converge.

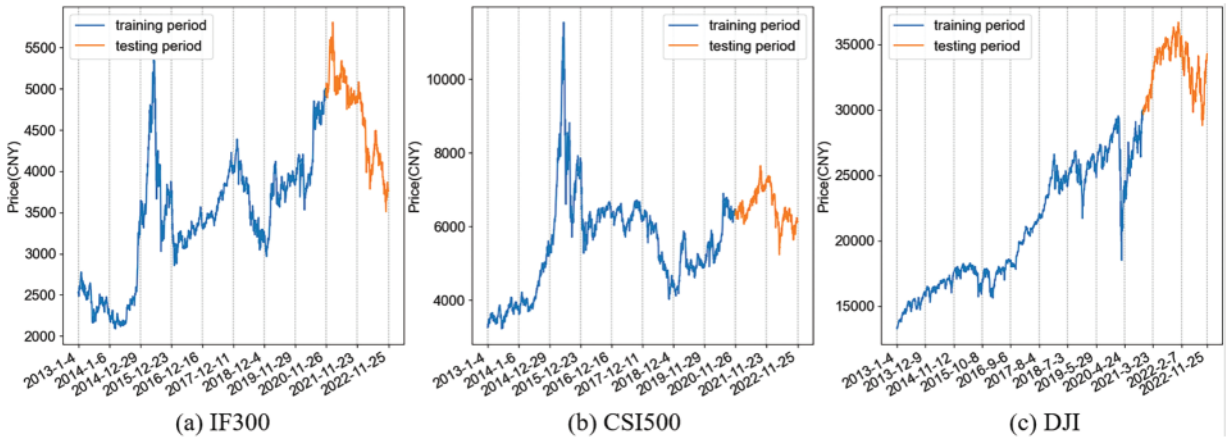


Figure 2: Daily closing price of IF300, CSI500, and DJI

Table 1: Hyperparameters of MAML and VPG

Trading method	Hyperparameter	Value	Description
MAML	N_s	3	Episodes in support part
	N_q	1	Episodes in query part
	K	10	Number of trajectories sampled per episode
	M	30	Meta batch size
	α	0.1	Fine-tuning learning rate
	β	0.01	Meta-update learning rate
	T	15	Number of updates to initial policy parameters per iteration
	ε	0.1	Clipping parameter for PPO surrogate loss
VPG	K	30	Number of trajectories sampled per episode
	α	0.1	Learning rate

The proposed model is built and run on Pytorch 1.8.1, a machine learning platform. The programming language is Python 3.6.3. Both the MAML-based trading model and the VPG-based trading model are trained and evaluated on a server with two Intel Golden 6240 CPUs, two NVIDIA RTX 2080 Ti GPUs and 128 GB RAM.

3.2 Evaluation Metrics

This paper quantitatively assesses the trading models' performance by employing the following metrics:

ER (Annualized Expected Return Rate): This metric gauges the annualized rate of expected return.

SR (Annualized Sharpe Ratio): The Sharpe ratio, calculated as the ratio of the annualized expected return (ER) to the annualized standard deviation (STD) of trade returns, serves as an indicator of risk-adjusted performance.

MDD (Maximum Drawdown): MDD is defined as the most significant loss experienced during a trading period, measured from the highest peak to the subsequent lowest trough in the profit curve.

These quantitative metrics provide a comprehensive evaluation of the models' effectiveness in terms of returns, risk management, and potential downturns in the trading strategies.

3.3 Performance Comparison of MAML and VPG for Different Markets and Different Market Phases

The cumulative profit curves of MAML, VPG, and Buy and Hold methods in the testing period of IF300, CSI 500, and DJI are shown in Fig. 3. The curve of MAML without fine-tuning is also represented in the figure, in which the initial policy trained by the meta-learner of MAML is applied for trading directly without fine-tuning. Buy and Hold builds a long position at the beginning of the testing period and holds it until the end, reflecting the market itself. From the figure we can see that the curves of the MAML model are above for most time period in different markets. This demonstrates that the proposed Meta-RL-based trading method can perform better than the RL-based method. We can also see that MAML without fine-tuning and VPG have similar performances.

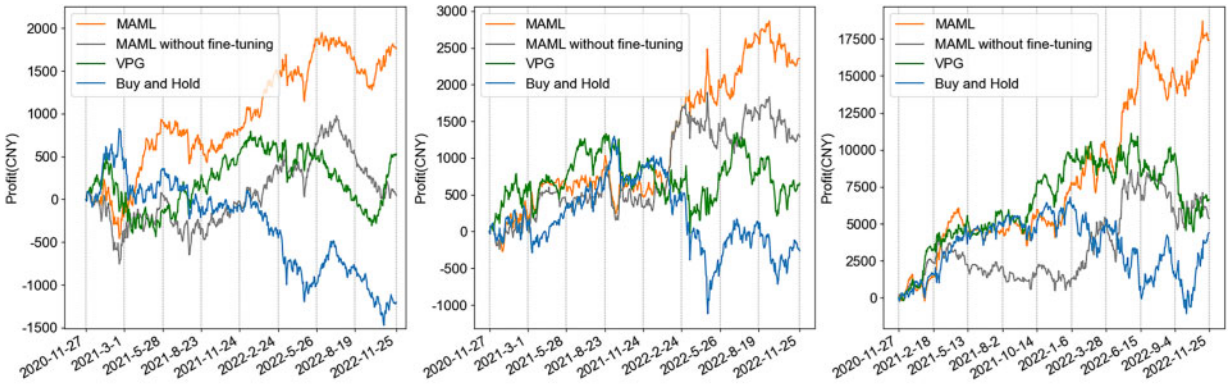


Figure 3: The cumulative profit of MAML, MAML without fine-tuning, VPG, and buy and hold

The quantitative performance comparisons of different trading methods are shown in Table 2. We can see that MAML has the highest annualized return and Sharpe ratio and the lowest maximum drawdown. Therefore, the MAML-based trading method can achieve higher profit and lower risk than other methods.

This paper also investigates the performance of trading policies learned by MAML and VPG across market phases with different changing speeds. The price standard deviation is used to characterize the changing speed. The market with a higher standard deviation fluctuates faster over time. We divide the testing period into multiple market phases. Each phase has a fixed length of three months and the division operates with a step length of one month.

Fig. 4 represents the profit of the VPG model and the profit gap between the MAML model and the VPG model on the market phases with different standard deviations. We can see from the figure that the profit of the VPG model decreases with the increment of the standard deviation. The trading method based on the traditional RL method is hard to keep pace with the time-varying financial market. We can also see from the figure that the profit gap between the MAML model and the VPG model increases with the increment of the standard deviation. This again demonstrates that

the proposed meta-RL-based trading method can effectively tackle the challenges encountered by the traditional RL-based method in a fast-changing financial market.

Table 2: The experimental results of MAML, MAML without fine-tuning, VPG, and buy and hold

Trading method	ER	SR	MDD
(a) IF300			
MAML	**0.193**	**1.033**	**0.162**
MAML without fine-tuning	0.005	0.030	0.239
VPG	0.068	0.359	0.258
Buy and Hold	−0.118	−0.627	0.473
(b) CSI500			
MAML	**0.169**	**0.975**	**0.141**
MAML without fine-tuning	0.085	0.495	0.149
VPG	0.056	0.301	0.200
Buy and Hold	−0.003	−0.018	0.365
(c) DJI			
MAML	**0.237**	**1.632**	**0.124**
MAML without fine-tuning	0.076	0.556	0.134
VPG	0.089	0.586	0.213
Buy and Hold	0.052	0.037	0.227

3.4 Effectiveness of Fine-Tuning Mechanisms in MAML

When testing the MAML model on part of a testing period (the query part of a task), a good initial policy learned from the training period is first fine-tuned on the adjacent prior part (the corresponding support part), and the tuned policy is fixed and applied to the query part. The fine-tuning mechanism plays an important role for MAML to adapt to market changes.

This paper presents three metrics to measure the market change between a query part and the corresponding support part.

The difference of mean prices (MD), which is the first-order moment change, is defined by

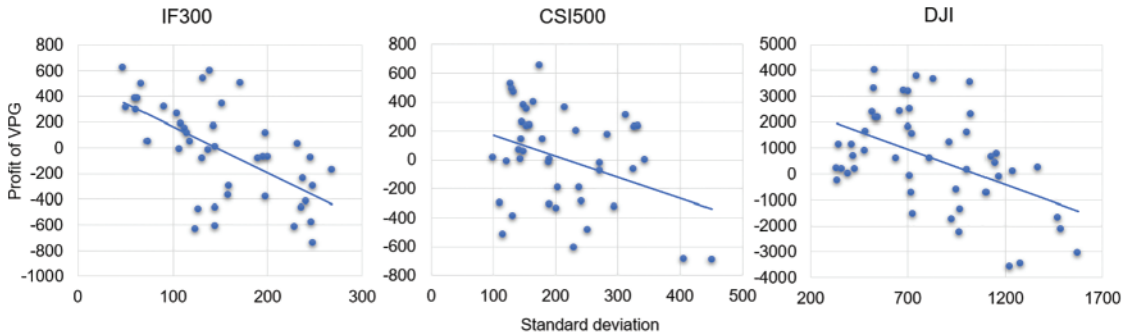
$$MD = |Mean\,price\,of\,query\,part\ -\ Mean\,price\,of\,corresponding\,support\,part|$$

The difference of standard deviations (SD), which is the second-order moment change is defined by

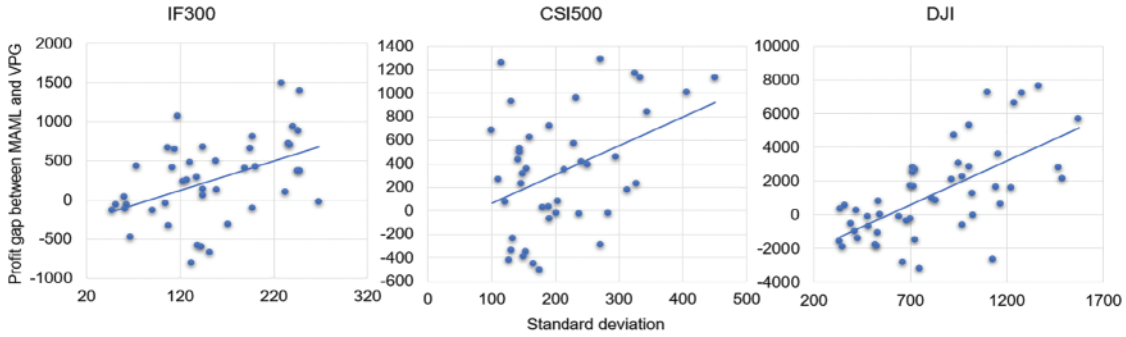
$$SD = |Standard\,deviation\,of\,query\,part\ -\ Standard\,deviation\,of\,corresponding\,support\,part|$$

The overlap ratio of price range (RO), which is the proportion of a query part contained in the corresponding support part is given by

$$RO = \frac{price\,range\,ofquery\,part \cap price\,range\,of\,corresponding\,support\,part}{price\,range\,of\,query\,part}$$



(a) Profit of VPG



(b) Profit gap between MAML and VPG

Figure 4: The profit of the VPG model and the profit gap between the MAML model and the VPG model on the market phases with different standard deviations

Fig. 5 represents the profit gap of the MAML models with and without fine-tuning for different market changes between the query part and the support part in a testing period on IF300, CSI500, and DJI. We can see from the figure that the profit gap between different MAML models increases with the increment of MD and SD, and decreases as RO increases. It means that the fine-tuning mechanism is more effective for MAML in a faster-changing market. We can also see that when RO is close to 1 or MD and SD are close to 0, the performances of fine-tuned and non-fine-tuned trading policies converge. The fine-tuning mechanism no longer works in a stable market environment.

3.5 Performance Comparison of MAML with Different Updating Mechanisms

In this subsection, we compare the profit and risk performances of the MAML method with different updating mechanisms, i.e.: ordered multiple-step updating, random multiple-step updating and one-step updating for IF300, CSI500, and DJI. The one-step updating mechanism introduced in Subsection 2.3 and the ordered multiple-step updating mechanism in Subsection 2.4 are proposed to deal with the dynamic changes within a task. In the random multiple-step updating mechanism, the episodes of the support part in a task data are selected randomly to update the initial policy.

From the results as shown in Fig. 6, we can see that the profit generated by the ordered multiple-step updating mechanism is greater than the others for most of the testing periods in all different markets examined. This demonstrates that the ordered multiple-step updating can better adapt to fast-changing market than other updating mechanisms. We can also see from Table 3 that even though the one-step updating mechanism of MAML has the smallest MDD, the ordered multiple-step updating

mechanism achieves the largest ER and SR. In other words, the proposed MAML-based trading method with ordered multiple-step updating can earn the highest total profit or profit per risk.

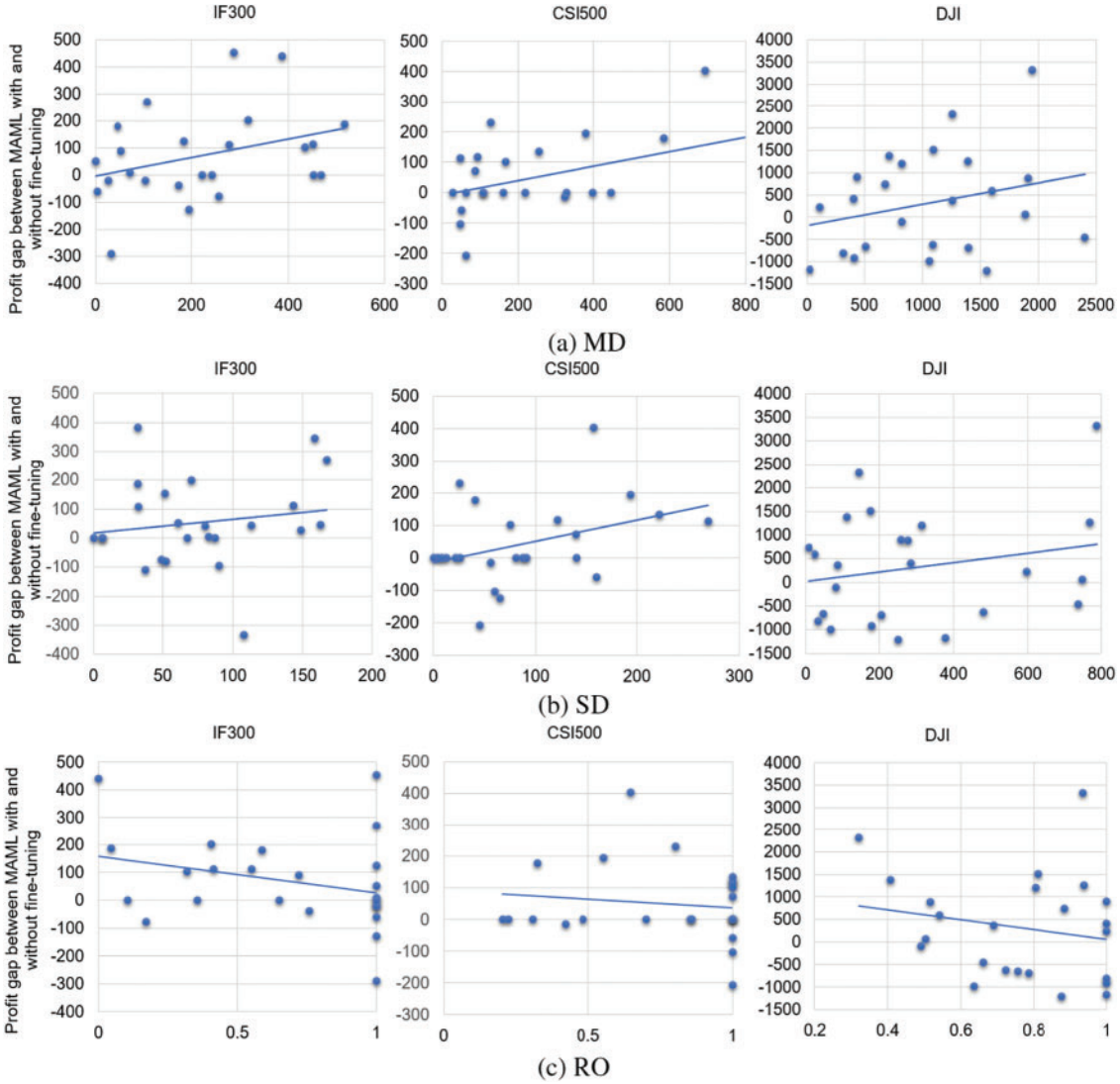


Figure 5: The profit gap of the MAML models with and without fine-tuning for different market changes between query part and support part

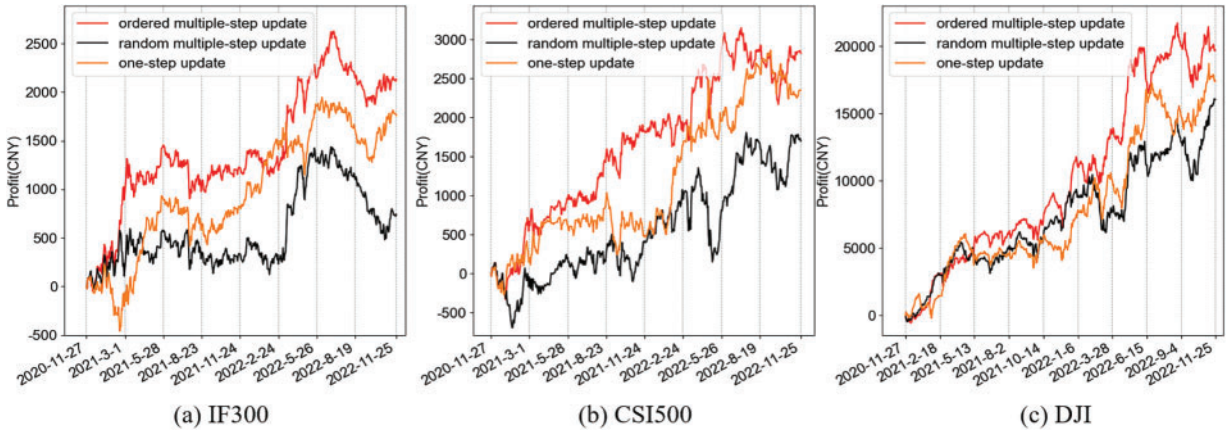


Figure 6: The cumulative profit of MAML-based trading methods with one-step update, ordered multiple-step update and random multiple-step update

Table 3: The experimental results of MAML-based trading methods with one-step update, ordered multiple-step update and random multiple-step update

Updating mechanism	ER	SR	MDD
(a) IF300			
Ordered multiple-step update	**0.249**	**1.356**	0.172
Random multiple-step update	0.088	0.476	0.237
One-step update	0.193	1.033	**0.162**
(b) CSI500			
Ordered multiple-step update	**0.223**	**1.275**	0.159
Random multiple-step update	0.134	0.751	0.204
One-step update	0.169	0.975	**0.141**
(c) DJI			
Ordered multiple-step update	**0.266**	**1.829**	0.152
Random multiple-step update	0.221	1.538	0.156
One-step update	0.237	1.632	**0.124**

4 Conclusions

This paper proposes a MAML-based trading method consisting of a PPO meta-learner and a VPG learner, to develop a trading policy capable of quickly adapting to market environments exhibiting non-stationary characteristics. In this method, training data is divided into multiple task data with each over an equal and small time period. This division ensures that each task data experiences a relatively stationary market environment. This new trading method can effectively address the challenges faced by traditional reinforcement learning-based trading methods. It achieves this by generating a good

initial policy through the PPO meta-learner and subsequently fine-tuning the initial policy using the VPG learner when trading in new market tasks.

The experimental results based on IF300, CSI500, and DJI demonstrate that the MAML-based trading method outperforms the traditional reinforcement learning-based method in terms of profit and risk in a fast-changing market. Especially, the fine-tuning mechanism plays an important role that makes MAML a promising candidate for supporting adaptive and sustainable trading in such an environment. Moreover, an ordered multiple-step updating mechanism is also proposed to further improve the adaptability of the MAML-based method by exploring the changing dynamic within a task market. Looking forward, utilizing the temporal relationship between tasks to enhance our model's adaptation to fast-changing markets is a direction for achieving further performance improvements.

Acknowledgement: We would like to extend our sincere gratitude to those who provided support throughout this work, as well as to the reviewers who offered valuable comments that significantly contributed to the improvement of this paper.

Funding Statement: The authors received no specific funding for this study.

Author Contributions: The authors confirm contribution to the paper as follows: study conception and design: Q.G., X.P. and Y.T.; data collection: M.G. and Y.T.; analysis and interpretation of results: M.G. and Y.T.; draft manuscript preparation: Y.T. and Q.G. supervision: Q.G. and X.P. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: Not applicable.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] A. A. Kirilenko and A. W. Lo, "Moore's law vs. Murphy's law: Algorithmic trading and its discontents," *Soc. Sci. Electron. Publ.*, vol. 27, no. 2, pp. 51–72, 2013. doi: 10.2139/ssrn.2235963.

[2] M. Avellaneda and J. H. Lee, "Statistical arbitrage in the U.S. equities market," *Quant. Financ.*, vol. 10, no. 7, pp. 761–782, 2008. doi: 10.1080/14697680903124632.

[3] T. A. Darden, M. E. Ferrenz, C. C. Klann, M. J. Ledwith, and G. M. Davis, "Modified momentum strategies in commodity futures markets," *Syst. Inf. Eng. Design Symp.*, vol. 31, pp. 218–222, 2009.

[4] T. Lubnau and N. Todorova, "Trading on mean-reversion in energy futures markets," *Energy Econ.*, vol. 51, pp. 312–319, 2015. doi: 10.1016/j.eneco.2015.06.018.

[5] Z. Zhang, S. Zohren, and S. Roberts, "Deep reinforcement learning for trading," *J. Finan. Data Sci.*, vol. 2, pp. 25–40, 2020.

[6] J. Peters and S. Schaal, "Policy gradient methods for robotics," in *Proc. IROS*, Beijing, China, 2006, pp. 2219–2225.

[7] J. Schulman, F. Wolski, and P. Dhariwal, "Proximal policy optimization algorithms," in *Proc. ICML*, Sydney, Australia, 2017, pp. 2931–2940.

[8] X. Xu *et al.*, "A group-wise feature enhancement-and-fusion network with dual-polarization feature enrichment for SAR ship detection," *Remote Sens.*, vol. 14, pp. 5276, 2022.

[9] B. Andrew, "Deep reinforcement learning pairs trading with a double deep Q-network," in *Proc. CCWC*, Las Vegas, USA, 2020, pp. 222–227.

[10] J. Lee, H. Koh, and H. J. Choe, "Learning to trade in financial time series using high-frequency through wavelet transformation and deep reinforcement learning," *Appl. Intell.*, vol. 51, pp. 6202–6223, 2021. doi: 10.1007/s10489-021-02218-4.
[11] A. Hong, M. Gao, and Q. Gao, "Non-stationary financial time series forecasting based on meta-learning," *Electron. Lett.*, vol. 59, no. 1, pp. e12681, 2023. doi: 10.1049/ell2.12681.
[12] N. E. Huang, M. L. Wu, W. Qu, S. R. Long, and S. Shen, "Applications of hilbert-huang transform to non-stationary financial time series analysis," *Appl. Stoch. Models Bus. Indust.*, vol. 19, pp. 245–268, 2003. doi: 10.1002/asmb.501.
[13] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proc. ICML*, Sydney, Australia, 2017, pp. 1126–1135.
[14] A. Maruan, B. Trapit, Y. Baram, I. Sutskever, I. Mordatch and P. Abbeel, "Continuous adaptation via meta-learning in non-stationary and competitive environments," in *Proc. ICLR*, Vancouver, Canada, 2018.
[15] R. Liaw, S. Krishnan, A. Garg, D. Crankshaw, J. E. Gonzalez and K. Goldberg, "Composing meta-policies for autonomous driving using hierarchical deep reinforcement learning," in *Proc. ICRA*, Singapore, 2017.
[16] R. Huang *et al.*, "Learning and fast adaptation for grid emergency control via deep meta reinforcement learning," *IEEE Trans. Power Syst.*, vol. 37, pp. 4168–4178, 2022. doi: 10.1109/tpwrs.2022.3155117.
[17] J. Baz, N. Granger, C. R. Harvey, N. L. Roux, and S. Rattray, "Dissecting investment strategies in the cross section and time series," *Soc. Sci. Electron. Publ.*, vol. 22, 2015.
[18] J. W. Wilder, "New concepts in technical trading systems," *Trend Res.*, pp. 63–70, 1978.
[19] R. Sutton, D. A. McAllester, and S. Singh, "Policy gradient methods for reinforcement learning with function approximation," in *Proc. NIPS*, Denver, USA, 1999, pp. 1057–1063.