Meta-Learning the Optimal Mixture of Strategies for Online Portfolio Selection

Jiayu Shen, Jia liu*, Zhiping Chen

*a*School of Mathematics and Statistics, Xi'an Jiaotong University, Xianning West Road 28, Xi'an, 710049, Shaanxi, P. R. China

---

Abstract

This paper presents an innovative online portfolio selection model, situated within a meta-learning framework, that leverages a mixture policies strategy. The core idea is to simulate a fund that employs multiple fund managers, each skilled in handling different market environments, and dynamically allocate our funding to these fund managers for investment. To address the non-stationary nature of financial markets, we divide the long-term process into multiple short-term processes to adapt to changing environments. We use a clustering method to identify a set of historically high-performing policies, characterized by low similarity, as candidate policies. Additionally, we employ a meta-learning method to search for initial parameters that can quickly adapt to upcoming target investment tasks, effectively providing a set of well-suited initial strategies. Subsequently, we update the initial parameters using the target tasks and determine the optimal mixture weights for these candidate policies. Empirical tests show that our algorithm excels in terms of training time and data requirements, making it particularly suitable for high-frequency algorithmic trading. To validate the effectiveness of our method, we conduct numerical tests on cross-training datasets, demonstrating its excellent transferability and robustness.

Keywords: Online Portfolio Selection, Meta-Learning, Mixture Policies

---

*Corresponding author. Email: jialiu@xjtu.edu.cn

1. Introduction

Long-term portfolio management is a fundamental problem in finance, involving the periodic reallocation of assets. Online portfolio selection (OLPS) is an important and challenging problem in long-term portfolio management, which aims to maximize the expected returns while controlling risks through reallocating portfolio weights.

According to the survey [21], existing OLPS algorithms can be broadly categorized into the following five classes: benchmark algorithms, follow-the-winner (FTW) algorithms, follow-the-loser (FTL) algorithms, pattern-matching algorithms, and mixture policies. Traditional online portfolio algorithms mostly rely on artificially designed financial features. For example, FTL algorithms assume that the market follows the mean reversion principle. Building on this idea, Passive Aggressive Mean Reversion (PAMR) introduces an innovative design for its loss function [24]: if the expected return of the price in the previous period exceeds a specified threshold, the loss gradually increases; conversely, if the expected return is below or equal to the threshold, the loss is zero. In contrast, the FTW algorithm assumes that stocks currently performing well will continue to perform well. For instance, the Exponentiated Gradient (EG) strategy seeks to track the best-performing stock from the previous period while maintaining the portfolio as close as possible to its previous allocation [15]. Due to these assumptions, *most traditional investment strategies perform well only in specific datasets and can hardly adapt to the change of market or the market regime.* Unlike the other four algorithms, mixture policies do not focus solely on a single strategy but maintain a pool of candidate policies and allocate funds among them. This paper adopts this idea by maintaining a pool of candidate policies and determining the allocation ratios of these policies based on current market conditions.

More recently, deep learning has been developed to address the portfolio management problem[29, 31, 37, 32]. It provides new insights into online portfolio management, avoids various assumptions about the market, and achieves objective profitability. *However, deep learning encounters two challenges in online portfolio selection problems: long training time and high sample complexity.* Financial markets are highly dynamic, requiring investors to make timely decisions, particularly real time decisions for online portfolio selection problems. Additionally, due to the high instability of financial data sequences and quick variation of the corresponding distribution over time, it

is difficult to make a precise prediction. Meanwhile, the historical data contains a vast amount of irrelevant information, with only recent data being of significance. Training and maintaining a end-to-end neural network demands extensive datasets and high time costs, which are challenging to satisfy the real time requirement in online portfolio selection problems.

Meta-learning, an emerging machine learning method, focuses on rapidly learning, generalizing, and adapting across different tasks [38, 16, 26, 34]. The characteristics of meta-learning make it naturally suitable for online portfolio management problems. By learning patterns and rules from historical data, meta-learning enables models to better adapt to unknown environments and new investment scenarios. Its core strength lies in rapidly learning and adapting to new market features and conditions, thereby enhancing the robustness and generalization of investment portfolios. In recent years, an increasing number of papers have applied meta-learning to time series forecasting and stock trading problems [39, 28, 35, 13]. *However, due to differences in data structures across various markets, these methods lack the ability to transfer effectively between markets.* This limitation results in meta-learning models being able to learn from only a single market, which restricts the model's generalization ability.

Motivated by the aforementioned insights, *this paper integrates meta-learning approach with a mixture policies learning framework, resulting the Meta-LMPS-online model, to make online portfolio selection.* We decompose the long-term investment process into multiple short-term time segments, generating several small-sample tasks to address the challenge of evolving data distributions in financial markets over time. During training, we employ clustering techniques to identify historically high-performing and diverse policies as candidate policies, and utilize meta-learning approach to derive initial parameters capable of rapidly adapting to new tasks. This approach enables the model to swiftly and dynamically adjust investment strategies in response to new tasks, while demonstrating robust transferability and generalization across various market conditions.

Our main contributions are summarized in three points.

- Unlike traditional deep learning or meta-learning frameworks that directly learn the portfolio, we adopt a mixture policy learning framework, which focuses on learning the mixture weights of multiple online portfolio policies independent of the number of stocks. This approach effectively addresses the transferability and generalization challenges

3

across diverse markets with varying stock pool sizes and datasets.

- We address the challenge of non-stationarity in financial markets by decomposing the long-term investment process into multiple short-term tasks.

- Numerical results valid the superiority of the proposed algorithm over traditional OLPS strategies, when transferring between different time periods or markets.

In Section 2, we introduce the background of online portfolio selection and related work. In Section 3, we introduce the model-agnostic meta-learning for online portfolio selection. In Section 4, we carry out a series of numerical tests. Section 5 concludes. Table 1 introduce the mathematical notations in the paper.

Table 1: Notations

Symbol	Definition or Explanation
N	Number of assets
T	Number of trading periods
$\boldsymbol{p}_t = (p_{t,1}, \ldots, p_{t,N})^\top$	Closing price at the end of period t
$\boldsymbol{x}_t = \boldsymbol{p}_t / \boldsymbol{p}_{t-1}$	Price relative vector of period t
$\boldsymbol{x}_{[t_1, t_2]}$	Price changes over period t_1 to period t_2
$\boldsymbol{b}_t = (b_{t,1}, \ldots, b_{t,N})^\top \in \Delta_N^+$	Portfolio vector/Capital allocation of assets at the start of period t
$\boldsymbol{b}_{[t_1, t_2]}$	Portfolio from period t_1 to period t_2
$\widetilde{\boldsymbol{b}}_t = \frac{\boldsymbol{b}_t \odot \boldsymbol{x}_t}{\boldsymbol{b}_t^\top \boldsymbol{x}_t}$	Capital allocation of assets at the end of period t
S_t	Cumulative wealth at the end of period(S_0 denote the initial wealth)
μ_{t-1}	Transaction remainder factor(TRF)
δ	Unit transaction cost for buying/selling assets
M	Number of candidate policies
$\boldsymbol{r}_t$	Return rate of candidate policies in period t
w	Historical window size within each sub-data pair in the task
$\boldsymbol{\omega}_t = (\omega_t^1, \ldots, \omega_t^M) \in \Delta_M^+$	Mixture weight in period t
f_θ	Neural network model
$\mathcal{T}_l$	Sub-task l
$\mathcal{T}_l^{sup}$	Support Set of $\mathcal{T}_l$
$\mathcal{T}_l^{que}$	Query Set of $\mathcal{T}_l$
K	Number samples in $\mathcal{T}_l^{sup}$
Q	Number samples in $\mathcal{T}_l^{que}$
H	Number of batch training tasks
α	Inner loop learning rate
β	Outer loop learning rate

2. Online portfolio selection and related work

Problem setting. We consider the problem that an investor plans to allocate funds across a finite number of assets over a series of time periods. Specifically, we consider $N \geq 2$ assets and $T \geq 1$ periods. Let $\boldsymbol{p}_t = (p_{t,1}, p_{t,2}, \ldots, p_{t,N}) \in \boldsymbol{R}_+^N$ denote the closing prices of the N assets at the end of period t. These closing prices serve as the opening prices for the assets at the start of period $t+1$. In period t, we consider the price relative vector $\boldsymbol{x}_t = (x_{t,1}, x_{t,2}, \ldots, x_{t,N}) \in \boldsymbol{R}_+^N$, where $x_{t,i} = \frac{p_{t,i}}{p_{t-1,i}}$ representing the growth factor of asset i during period t. The market price changes from period t_1 through period t_2 are represented by $\boldsymbol{x}_{[t_1,t_2]} = \{\boldsymbol{x}_{t_1}, \ldots, \boldsymbol{x}_{t_2}\}$. At the beginning of period t, the investment is allocated according to the portfolio vector $\boldsymbol{b}_t = (b_{t,1}, b_{t,2}, \ldots, b_{t,N})^\top$, where $b_{t,i}$ denotes the proportion of capital allocated to asset i. Assuming a self-financing investment process without allowing short-selling positions, the portfolio vector adheres to the constraints $\boldsymbol{b}_t \in \Delta_N^+$, where $\Delta_N^+ = \{\boldsymbol{b} \mid \boldsymbol{b} \geq \boldsymbol{0}, \boldsymbol{b}^\top \boldsymbol{1} = 1\}$. The portfolio strategy from period t_1 through period t_2 is denoted by $\boldsymbol{b}_{[t_1,t_2]} = \{\boldsymbol{b}_{t_1}, \ldots, \boldsymbol{b}_{t_2}\}$.

Let S_t denote the cumulative wealth of the investor at the beginning of period t. After the market fluctuations during period t, the change in the investor's wealth is $S_t \boldsymbol{b}_t^\top \boldsymbol{x}_t$. The portfolio changes are represented by $\widetilde{\boldsymbol{b}}_t = (\widetilde{b}_{t,1}, \ldots, \widetilde{b}_{t,N})^\top$, where $\widetilde{b}_{t,i} = \frac{b_{t,i} x_{t,i}}{\boldsymbol{b}_t^\top \boldsymbol{x}_t}$. At the beginning of period $t+1$, the investment strategy is adjusted to $\boldsymbol{b}_{t+1}$, resulting in transaction costs. Let μ_t denote the transaction remainder factor [30, 6, 25], which represents the remaining proportion of the capital after the deduction of transaction cost at the beginning of period t. Consequently, the change in the investor's wealth at this time is $\mu_t S_t \boldsymbol{b}_t^\top \boldsymbol{x}_t$. Therefore, after T periods, the investor's cumulative wealth is computed as

$$S_T(\boldsymbol{b}_{[1,T]}) = S_0 \prod_{t=1}^{T} \mu_{t-1} \boldsymbol{b}_t^\top \boldsymbol{x}_t. \tag{1}$$

Assuming the transaction cost rate is δ, we approximate Equation (1) as follows:

$$S_T(\boldsymbol{b}_{[1,T]}) \approx S_0 \prod_{t=1}^{T} \left[\sum_{i=1}^{N} b_{t,i} x_{t,i} \left(1 - \delta \sum_{i=1}^{N} \left| b_{t,i} - \widetilde{b}_{t-1,i} \right| \right) \right].$$

This approximation technique is proposed by Guo et al. [11].

Traditional OLPS methods. Table 2 lists classical online portfolio selection policies, including benchmarks, FTW, FTL, and pattern-matching categories. We use these algorithms to build our **candidate policy pool**. For mixture policies, Akcoglu et al. [2] allocates assets evenly a mong experts who operate autonomously, then aggregates their wealth. Hazan and Seshadhri [14] maintains a limited set of experts, dynamically adds or removes them based on performance, and adjusts weights among the active experts. Lin et al. [27] and Yang et al. [40] apply the online gradient update algorithm or exponentially weighted average algorithm to integrate a pool of expert strategies. These approach smooths overall performance by balancing various policies and enhances adaptability by combining universal strategies with heuristic algorithms.

Deep learning in portfolio optimization. The application of deep learning in OLPS can be divided into two categories. The first category involves using deep learning to predict asset returns and subsequently constructing an optimization model to formulate portfolios based on the predictions. The second category adopts an end-to-end approach, where deep neural networks are used to directly generate portfolios. Lee and Yoo [19], Ma et al. [29], Martínez-Berbero et al. [31] use recurrent neural networks or convolutional networks to predict stock prices and employ optimization methods to determine investment strategies. Uysal et al. [37], Gu et al. [10], Zhang et al. [41] combine prediction and optimization tasks into an end-to-end neural network, considering various metrics including investment returns and volatility as input features for the network. Niu et al. [32] explored the application of mixture policy in deep learning, integrating expert policies and environmental interactions to adapt to dynamic markets, providing a new perspective for online portfolio optimization.

Meta learning in time series forecasting and portfolio management. Grazzi et al. [9] and Woo et al. [39] applied meta-learning to historical value models and time index models respectively to tackle the problem of non-stationary forecasting, which verified the improvement of model generalization by meta-learning. Sorensen et al. [35] applied a quantum meta-learning algorithm to the field of stock trading, proposing a quantum neural network model. They demonstrated through simulations on quantum computers that it can achieve efficient learning and adapt trading strategies even with a significant reduction in the number of parameters. Ha et al. [13] applied meta reinforcement learning to the online portfolio optimization problem, by breaking down long-term investment goals into a series of short-term tasks

6

Table 2: Classical Online Portfolio Strategies

Classifications	Number	Strategies	Updating rule of $\boldsymbol{b}_{t+1}$
Benchmarks	1	BAH[21]	$\frac{\boldsymbol{b}_t \bigotimes \boldsymbol{x}_t}{\boldsymbol{b}_t \cdot \boldsymbol{x}_t}$
	2	CRP[18]	$\boldsymbol{b}_t$
FTW	3	UP[36]	$\frac{\int_{\Delta_N^+} \boldsymbol{b} S_t(\boldsymbol{b}) du(\boldsymbol{b})}{\int_{\Delta_N^+} S_t(\boldsymbol{b}) du(\boldsymbol{b})}$
	4	Best-so-far	$\mathrm{argmax}_{\boldsymbol{b} \in \Delta_N^+} \sum_{i=0}^{w-1} \frac{1}{w} \log(\boldsymbol{b} \cdot \boldsymbol{x}_{t-w})$
	5	EG[15]	$\mathrm{argmax}_{\boldsymbol{b} \in \Delta_N^+} \eta \log(\boldsymbol{b} \cdot \boldsymbol{x}_t) - R(\boldsymbol{b}, \boldsymbol{b}_t)$
	6	ONS[1]	$\mathrm{argmax}_{\boldsymbol{b} \in \Delta_N^+} \sum_{i=1}^{t} \log(\boldsymbol{b} \cdot \boldsymbol{x}_i) - \lambda R(\boldsymbol{b})$
FTL	7	ANTICOR[4]	If assets i increases more than asset j and they are positively correlated, Anticor transfers from i to j by $M_{cor}(i,j) - \min\{0, M_{cor}(i,i)\} - \min\{0, M_{cor}(j,j)\}$ (where M_{cor} is the cross correlation value)
	8	PAMR[24]	$\mathrm{argmin}_{b \in \Delta_N^+} R(b, b_t)$ s.t. $\boldsymbol{b} \cdot \boldsymbol{x}_t \leq \epsilon$
	9	CWMR[23]	$(\boldsymbol{\mu}_{t+1}, \sum_{t+1}) =$ $\mathrm{argmin}_{\boldsymbol{\mu}, \sum} D_{KL}(\mathcal{N}(\boldsymbol{\mu}, \sum) \| \mathcal{N}(\boldsymbol{\mu}_t, \sum_t)$ s.t. $Pr(\boldsymbol{\mu} \cdot \boldsymbol{x}_t \leq \epsilon) \geq \theta$ $\boldsymbol{\mu} \in \Delta_N, \boldsymbol{b}_{t+1} \in \mathcal{N}(\boldsymbol{\mu}_{t+1}, \sum_{t+1})$
	10	OLMAR[20]	$\mathrm{argmin}_{\boldsymbol{b} \in \Delta_N^+} R(\boldsymbol{b}, \boldsymbol{b}_t)$ s.t. $\boldsymbol{b} \cdot \hat{\boldsymbol{x}}_{t+1} \geq \epsilon$ $\hat{\boldsymbol{x}}_{t+1}(w) = \frac{1}{w}\left(1 + \frac{1}{\boldsymbol{x}_t} + \cdots + \frac{1}{\prod_{i=0}^{w-2} \boldsymbol{x}_{t-i}}\right)$
	11	WMAMR[8]	$\mathrm{argmin}_{b \in \nabla_N^+} \frac{1}{2}\|\boldsymbol{b} - \boldsymbol{b}_t\|$ s.t. $\boldsymbol{b} \cdot \widetilde{\boldsymbol{x}}_{t,w} \leq \epsilon$, $\widetilde{\boldsymbol{x}}_{t,w} = \frac{1}{w}\left(1 + \frac{1}{\boldsymbol{x}_t} + \cdots + \frac{1}{\prod_{i=0}^{w-2} \boldsymbol{x}_{t-i}}\right)$
	12	RMR[17]	$\mathrm{argmin}_{\boldsymbol{b} \in \Delta_N^+} \boldsymbol{b} \cdot \widehat{\boldsymbol{x}}_t + \lambda R(\boldsymbol{b}, \boldsymbol{b}_t)$ $\widehat{\boldsymbol{x}}_t = \frac{\boldsymbol{u}_{t+1}}{\boldsymbol{p}_t}, \boldsymbol{u}_{t+1} = \min_u \sum_{i=0}^{\omega-1} \|\boldsymbol{p}_{t-i} - \boldsymbol{u}\|$
Pattern Matching	13	B$^{\mathrm{NN}}$[12]	$\mathrm{argmax}_{\boldsymbol{b} \in \Delta_N^+} \sum_{i \in C_t} \frac{1}{|C_t|} \log \boldsymbol{b} \cdot \boldsymbol{x_i}$ (C_t is the set of similar windows; pattern matching methods use different criteria)
	14	CORN[22]	$\mathrm{argmax}_{\boldsymbol{b} \in \Delta_N^+} \sum_{i \in C_t} \frac{1}{|C_t|} \log \boldsymbol{b} \cdot \boldsymbol{x_i}$

and utilizing convolutional networks to match and evaluate market data patterns. However, these methods have not yet fully leveraged the potential of meta-learning in the field of online portfolio management, and they lack the ability to transfer and generalize across multiple markets.

3. Model-agnostic meta-learning for online portfolio selection

The meta-learning framework employed in this paper is model-agnostic meta-learning (MAML) proposed by Finn et al. [7]. The core concept involves learning an initial set of parameters from historical tasks that can quickly adapt to new tasks. The background introduction of MAML is provided in Appendix A. The process of learning from historical tasks is known as meta-training and is performed offline, whereas the adaptation to new tasks is known as meta-testing and is performed online (ref. Figure 1).

Figure 1: Meta learning

3.1. Single Task in Online Portfolio Selection

Portfolio selection task. Considering the high instability of financial data sequences and quick variation of the corresponding distribution over time, we consider the investment process in a short time period in one single market as a single task. Then the investment tasks over several segments in one single market or in different markets can be viewed as a meta task.

8

Suppose in a given market, we utilize the historical relative price data of stocks over the previous L periods, $\boldsymbol{x}_{[1,L]} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_L\}$, to construct the investment portfolio vector for the subsequent Q periods. From the policy pool (Table 2), we select M candidate policies that demonstrate strong performance and diverse decision-making under varying market conditions. Specifically, we apply k-means clustering to the return sequences of policies from the policy pool over the first L periods. The policy with the highest cumulative return in each cluster is selected as a candidate.

Each candidate policy generates an investment portfolio vector for every period based on the historical relative price sequence. Assuming that the investment portfolios of M candidate policies at the start of period t are $\boldsymbol{b}_t^1, \ldots, \boldsymbol{b}_t^M$, and their return rate at period t are $\boldsymbol{r}_t = (r_t^1, \ldots, r_t^M)$. Different from traditional deep learning or meta-learning frameworks, we learn the mixture weights of the candidate portfolio policies $\boldsymbol{\omega}_t$ rather than the portfolio $\boldsymbol{b}_t$ directly. We set the historical window size as w. A single task of online portfolio selection is to determine the optimal mixture weights $\boldsymbol{\omega}_t = (\omega_t^1, \ldots, \omega_t^M)$ among M candidate policies, for the following period based on the historical return data $[\boldsymbol{r}_{t-w}, \ldots, \boldsymbol{r}_{t-1}]$, $t = L+1, \ldots, L+Q$.

We then denote the portfolio selection model in a task as

$$f_\theta : [\boldsymbol{r}_{t-w}, \ldots, \boldsymbol{r}_{t-1}] \to \boldsymbol{\omega}_t, \ t = L+1, \ldots, L+Q, \tag{2}$$

where θ is the parameter of the training model. After obtaining the mixture weights through the neural network, we can ultimately derive the investment portfolio vector as

$$\boldsymbol{b}_t = \sum_{j=1}^{M} \omega_t^j \boldsymbol{b}_t^j, \quad t = L+1, \ldots, L+Q.$$

Neural Network. We then design the neural network. As shown in Equation (2), the input of the network consists of the historical return rates of the candidate policies $[\boldsymbol{r}_{t-w}, \ldots, \boldsymbol{r}_{t-1}]$ over the previous w periods. The output of the network is the mixture weights $\boldsymbol{\omega}_t$, and its parameters are denoted as θ.

The network architecture is shown in Figure 2 and comprises three components: a two-layer LSTM, an attention mechanism, and fully connected layers [5, 33, 3]. The two-layer LSTM is employed to extract features from the time series data and capture sequential dependencies. The self-attention

mechanism is utilized to discern correlations among the vectors. The fully connected layers are responsible for mapping the extracted features to the final output space, with the Softmax layer providing the probability distribution over the selected candidate policies.
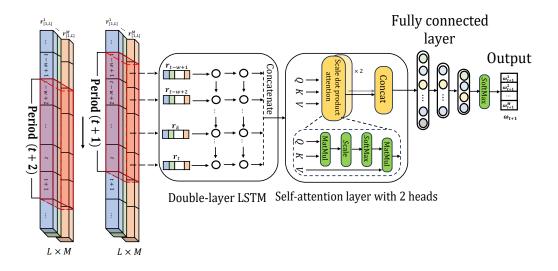

Figure 2: Neural network structure

Data and labels. To train the neural network, we need $L - w$ pairs of labeled data:

$$\{[\boldsymbol{r}_{t-w}, \ldots, \boldsymbol{r}_{t-1}]; \hat{\boldsymbol{\omega}}_t\}_{t=w+1}^L.$$

Here, $\hat{\boldsymbol{\omega}}_t = (\hat{\omega}_t^1, \ldots, \hat{\omega}_t^M)$ represents a label vector composed of the relative weights of M candidate policies. The weights are determined by the actual returns of each policy in the period t. The specific calculation for $\hat{\boldsymbol{\omega}}_t$ is as follow:

$$\hat{\omega}_t^j = \frac{r_t^j - \min\{r_t^1, \ldots, r_t^M\}}{\sum_{j=1}^M r_t^j - M \times \min\{r_t^1, \ldots, r_t^M\}},$$
$$j = 1, \ldots, M, \quad t = w + 1, \ldots, L.$$

Loss function. Given the label $\hat{\boldsymbol{\omega}}_t$ at period t, we define the loss function

as:

$$\mathcal{L}(\boldsymbol{\omega}_t, \hat{\boldsymbol{\omega}}_t) = \|\boldsymbol{\omega}_t - \hat{\boldsymbol{\omega}}_t\|_2^2 + \eta\|\boldsymbol{\omega}_t\|_2^2$$

$$+\lambda\delta\|\sum_{j=1}^{M}(\omega_t^j \boldsymbol{b}_t^j - \omega_{t-1}^j \widetilde{\boldsymbol{b}}_{t-1}^j)\|_1. \tag{3}$$

The loss function consists of three terms. The first term is used to control the gap between the predicted values and the actual values, the second term is for encouraging a diversified allocation of weights among each strategy, and the third term is used to control transaction costs. Here, η and λ are both positive constants.

3.2. Meta-Learning Framework

We then consider the meta task with many sub-investment tasks over several time segments in one single market or in several different markets.

Meta Training. Assuming that we have a set of T^{train} training tasks, each consisting of a support set with K samples and a query set with Q samples, which may come from the same market or different markets. The training tasks are generated from the original historical return rate data of the M candidate policies by using a rolling window approach. That is, we generate a task in a time window with $K + Q + w$ continue trading days and rolling the time window forward to generate several successive training tasks with overlapping time periods. Please refer to Figure 3 for generation of the training tasks.

We represent the training tasks as:

$$\mathcal{C} = \left\{\mathcal{T}_l = \mathcal{T}_l^{\text{sup}} \cup \mathcal{T}_l^{\text{que}}, \quad l = 1, \ldots, T^{train}\right\}, \tag{4}$$

where $\mathcal{T}_l^{\text{sup}} = \{[\boldsymbol{r}_{t-w}, \ldots, \boldsymbol{r}_{t-1}]; \hat{\boldsymbol{\omega}}_t\}_{t=l-K}^{l-1}$, and $\mathcal{T}_l^{\text{que}} = \{[\boldsymbol{r}_{t-w}, \ldots, \boldsymbol{r}_{t-1}]; \hat{\boldsymbol{\omega}}_t\}_{t=l}^{l+Q-1}$.

We randomly select H tasks $\mathcal{B} = \{\mathcal{T}_{l_1}, \ldots, \mathcal{T}_{l_H}\}$ from $\mathcal{C}$ for the meta-training. For each selected task $\mathcal{T}_{l_k}$, the support set $\mathcal{T}_{l_k}^{\text{sup}}$ is used to update the base parameters θ_k' for each task, serving as the adaptation process. The query set $\mathcal{T}_{l_k}^{\text{que}}$ is used to update the meta-parameter θ^*, helping to find an effective initial parameter that enable the model to quickly adapt to new tasks. The process is described as follows:

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \sum_{k=1}^{H} \mathcal{L}_k(f_{\theta_k'}(\mathcal{T}_{l_k}^{que})) \tag{5}$$

$$\text{s.t.} \quad \theta_k' = \underset{\theta}{\operatorname{argmin}} \mathcal{L}_k(f_\theta(\mathcal{T}_{l_k}^{sup})) \tag{6}$$
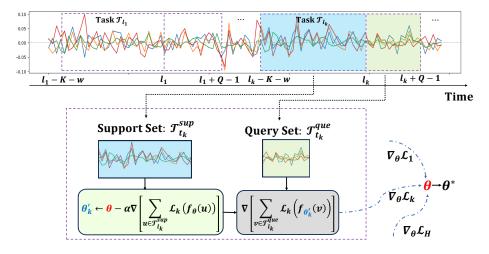
11

Figure 3: Meta-training framework for online portfolio selection

where

$$\mathcal{L}_k(f_{\theta'_k}(\mathcal{T}^{que}_{l_k})) = \sum_{(\boldsymbol{r},\hat{\boldsymbol{\omega}})\in\mathcal{T}^{que}_{l_k}} \mathcal{L}(f_{\theta'_k}(\boldsymbol{r}),\hat{\boldsymbol{\omega}}), \qquad (7)$$

$$\mathcal{L}_k(f_{\theta}(\mathcal{T}^{sup}_{l_k})) = \sum_{(\boldsymbol{r},\hat{\boldsymbol{\omega}})\in\mathcal{T}^{sup}_{l_k}} \mathcal{L}(f_{\theta}(\boldsymbol{r}),\hat{\boldsymbol{\omega}}). \qquad (8)$$

The optimization of base parameters for each sub-task (6) is performed individually for each task $\mathcal{T}_{l_k}$. It involves updating the base parameters through gradient descent (GD) on the support set $\mathcal{T}^{sup}_{l_k}$:

$$\theta'_k \leftarrow \theta - \alpha\nabla\mathcal{L}_k(f_{\theta}(\mathcal{T}^{sup}_{l_k})),$$

where α is the sub-task learning rate. The meta-optimization (5) updates the meta-parameter θ by measuring the overall performance of the base parameters θ'_k based on θ on the query set $\mathcal{T}^{que}_{l_k}$. The meta-parameters is update by the stochastic gradient descent (SGD) on the H randomly selected tasks:

$$\theta^* \leftarrow \theta - \beta\nabla\left[\sum_{k=1}^{H}\mathcal{L}_k(f_{\theta'_k}(\mathcal{T}^{que}_{l_k}))\right],$$

where β is the meta-learning rate. We illustrate this process in Figure 3.

In the proposed meta-learning framework, we decompose a long-term portfolio selection problem into multiple short-term tasks, which allows us

12

to treat the global non-stationary time series as locally stationary. As a result, meta-learning helps mitigate the bias caused by changes in temporal distributions and enhances the model's generalizability.

Meta Testing. After completing the meta-training process, we obtain the initial model parameters θ^*. We then use the support set of the test task $\{\mathcal{T}_l^{sup}\}$ to perform gradient descent (GD) on θ^*, rapidly acquiring model parameters tailored to this specific task, denoted as θ_l^{new}. Next, by feeding the features into the model $f_{\theta_l^{new}}$, we obtain the weight allocation for the M candidate policies, which allows us to determine the final investment vector.

refWe refer to this investment portfolio management algorithm, which is based on meta-learning and mixture policies learning, as the Meta-LMPS-Online algorithm. The algorithm flow is presented in Algorithm 1.

Algorithm 1 Meta-LMPS-Online

Input: candidate policies $\{\pi^j(\boldsymbol{b}|\boldsymbol{x})\}_{j=1}^M$, historical price vector sequence $\boldsymbol{x}$
Output: Model parameters θ_l^{new} suitable for test task $\mathcal{T}_l$
 1: Construct the set of training tasks set $\mathcal{C} = \{\mathcal{T}\}$ using candidate policies and historical price vector sequence
 2: **for** each epoch **do**
 3:    Randomly select H tasks $\mathcal{B} : \{\mathcal{T}_{l_1}, \ldots, \mathcal{T}_{l_H}\}$ from $\mathcal{C}$
 4:    **for** $\mathcal{T}_{l_k}$ **do**
 5:       Evaluate $\nabla_\theta \mathcal{L}_k(f_\theta(\mathcal{T}_{l_k}^{sup}))$ using $\mathcal{T}_{l_k}^{sup}$ and $\mathcal{L}_k$ in equation (8)
 6:       Compute base parameters with gradient descent: $\theta_k' \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}_k(f_\theta(\mathcal{T}_{l_k}^{sup}))$
 7:    **end for**
 8:    Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{k=1}^H \mathcal{L}_k(f_{\theta_k'}(\mathcal{T}_{l_k}^{que}))$
 9: **end for**
10: Acquire optimized meta-parameter $\theta^* \leftarrow \theta$
11: **for** trading period l **do**
12:    Set the initial parameter $\theta_0 = \theta^*$
13:    Construct the support set $\mathcal{T}_l^{sup}$ using price vectors
14:    Perform gradient descent on $\mathcal{T}_l^{sup}$ to quickly obtain $\theta_l^{new} = \theta_0 - \alpha \nabla_\theta \mathcal{L}_l(f_{\theta_0}(\mathcal{T}_l^{sup}))$
15: **end for**

Single-market and Cross-market algorithm. The Meta-LMPS-Online model features two sub-case algorithms for two types of task data: the Single-market Adapted Meta-LMPS-Online Algorithm (LMPS-SMO) and

the Cross-Market Learning Meta-LMPS-Online Algorithm (LMPS-CMO). LMPS-SMO uses training data and test data from the same market, allowing for swithing of market regimes. While LMPS-CMO utilizes training data and test data from multiple different markets, demonstrating broader generalization. Unlike other methods that are limited to a single market due to the data-consistency issue, our approach facilitates the transfer of learning experiences across different markets, offering greater flexibility and adaptability. The detailed algorithms and illustrations for LMPS-SMO and LMPS-CMO are provided in Algorithm 2 and Figure 4 for the former, and in Algorithm 3 and Figure 5 for the latter.

Figure 4: The workflow of LMPS-SMO algorithm

Figure 5: The workflow of LMPS-CMO algorithm

Algorithm 2 Single-Market Adapted Meta-LMPS-Online Algorithm(LMPS-SMO)
___

Input: $\boldsymbol{x}_{[1,T_{train}]}$: Training portion of price relative vector sequences
Input: $\boldsymbol{x}_{[T_{train}+1,T]}$: Testing portion of price relative vector sequences
Input: M: Number of candidate policies
Input: K, Q: Number samples in support set, Number samples in query set
Input: H: Number of batch training tasks
Output: $S_{[T_{train}+1,T]}$: Cumulative wealth sequences in the testing portion

    //Data Preparation and Data Preprocessing
1: Based on $\boldsymbol{x}_{[1,T_{\text{train}}]}$, use a clustering method to identify M historically high-performing policies with low similarity as candidate strategies.
2: Obtain the return sequences of M candidate policies $\boldsymbol{r}^{j}_{[1,T_{train}]}, j = 1, \ldots, M$ on training portion $\boldsymbol{x}_{[1,T_{\text{train}}]}$, and construct training task set $\mathcal{C} = \{\mathcal{T}\}$

    //Meta Training
3: Randomly initialize θ
4: **for** each epoch **do**
5:     Randomly select H tasks $\mathcal{B} = \{\mathcal{T}_{l_k}\}_{k=1}^{H}$ from $\mathcal{C}$
6:     **for** $\mathcal{T}_{l_k}$ **do**
7:         Evaluate $\nabla_{\theta}\mathcal{L}_k(f_{\theta}(\mathcal{T}^{sup}_{l_k}))$ with respect to K examples
8:         Compute adapted parameters with gradient descent: $\theta'_k = \theta - \alpha\nabla_{\theta}\mathcal{L}_k(f_{\theta}(\mathcal{T}^{sup}_{l_k}))$
9:     **end for**
10:     Update $\theta \leftarrow \theta - \beta\nabla_{\theta}\sum_{k=1}^{H}\mathcal{L}_{\mathcal{T}_k}(f_{\theta'_k}(\mathcal{T}^{que}_{l_k}))$
11: **end for**
12: Update $\theta^* \leftarrow \theta$

    //Meta Testing
13: Initialization: $S_{T_{train}+1} = 1$
14: **for** $l = T_{train}+1, \ldots, T$ **do**
15:     Set the initial parameter $\theta_0 = \theta^*$
16:     Calculate the returns of M candidate policies in period $(t-1)$ to obtain the support set for the current meta-test task $\mathcal{T}^{sup}_l = \{[\boldsymbol{r}_{l+t-w}, \ldots, \boldsymbol{r}_{l+t-1}]; \hat{\boldsymbol{\omega}}_{l+t}\}_{i=-K}^{-1}$
17:     Perform gradient descent on $\mathcal{T}^{sup}_l$ to quickly obtain $\theta^{new} = \theta_0 - \alpha\nabla_{\theta}\mathcal{L}_t(f_{\theta_0}(\mathcal{T}^{sup}_t))$

18:     Obtain the mixture weights in period t: $\boldsymbol{\omega}_t = f_{\theta^{new}_t}(\boldsymbol{r}_{t-w}, \ldots, \boldsymbol{r}_{t-1})$
19:     Acquire the portfolios of M candidate policies in period l $(\boldsymbol{b}^1_l, \ldots, \boldsymbol{b}^M_l)$, and set the investment portfolio as $\boldsymbol{b}_l = \sum_{j=1}^{M}\omega^j_l\boldsymbol{b}^j_l$
20:     The market reveals actual price relative vectors $\boldsymbol{x}_l$
21:     Update the cumulative wealth as $S_l = S_{l-1} \cdot \boldsymbol{b}^{\top}_l\boldsymbol{x}_l(1 - \delta\|\boldsymbol{b}_l - \widetilde{\boldsymbol{b}}_{l-1}\|_1)$
22: **end for**
___

Algorithm 3 Cross-Market Learning LMPS-Online Algorithm(LMPS-CMO)
***

Input: $\boldsymbol{x_{1,[1,T_1]}}, \ldots, \boldsymbol{x_{m,[1,T_m]}}, \ldots$: Relative price sequences of training datasets
Input: $\boldsymbol{x}_{[1,T]}$: Relative price sequences of testing dataset
Input: M: Number of candidate policies
Input: K, Q: Number samples in support set, Number samples in query set
Input: H: Number of batch training tasks
Output: $S_{[K+w,T]}$: Cumulative wealth sequences in the testing portion

//Data Preparation and Data Preprocessing

1: Based on training datasets, use a clustering method to identify M historically high-performing policies with low similarity as candidate strategies.
2: Initialize $\mathcal{C} = \{\}$
3: **for** each m **do**
4:     Obtain the return sequences of M candidate policies $\boldsymbol{r}^j_{m,[1,T_m]}, j = 1, \ldots, M$ on training portion $\boldsymbol{x}_{m,[1,T_m]}$, and construct task set $\mathcal{C}_m = \{\mathcal{T}\}$
5:     Update $\mathcal{C} = \mathcal{C} \bigcup \mathcal{C}_m$
6: **end for**

//Meta Training
7: Randomly initialize θ
8: **for** each epoch **do**
9:     Randomly select H tasks $\mathcal{B} = \{\mathcal{T}_{m_k,l_k}\}^H_{k=1}$ from $\mathcal{C}$
10:     **for** $\mathcal{T}_{m_k,l_k}$ **do**
11:         Evaluate $\nabla_\theta \mathcal{L}_k(f_\theta(\mathcal{T}^{sup}_{m_k,l_k}))$ with respect to K examples
12:         Compute adapted parameters with gradient descent: $\theta'_k = \theta - \alpha \nabla_\theta \mathcal{L}_k(f_\theta(\mathcal{T}^{sup}_{m_k,l_k}))$
13:     **end for**
14:     Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum^H_{k=1} \mathcal{L}_{\mathcal{T}_k}(f_{\theta'_k}(\mathcal{T}^{que}_{m_k,l_k}))$
15: **end for**
16: Update $\theta^* \leftarrow \theta$

//Meta Testing
17: Initialization:$S_{K+w} = 1$
18: **for** $l = K + w + 1, \ldots, T$ **do**
19:     Set the initial parameter $\theta_0 = \theta^*$
20:     Calculate the returns of M candidate policies in period $(t-1)$ to obtain the support set for the current meta-test task $\mathcal{T}^{sup}_l = \{[\boldsymbol{r}_{l+t-w}, \ldots, \boldsymbol{r}_{l+t-1}]; \hat{\boldsymbol{\omega}}_{l+t}\}^{-1}_{i=-K}$
21:     Perform gradient descent on $\mathcal{T}^{sup}_l$ to quickly obtain $\theta^{new} = \theta_0 - \alpha \nabla_\theta \mathcal{L}_t(f_{\theta_0}(\mathcal{T}^{sup}_t))$

22:     Obtain the mixture weights in period t: $\boldsymbol{\omega}_t = f_{\theta^{new}_t}(\boldsymbol{r}_{t-w}, \ldots, \boldsymbol{r}_{t-1})$
23:     Acquire the portfolios of M candidate policies in period l $(\boldsymbol{b}^1_l, \ldots, \boldsymbol{b}^M_l)$, and set the investment portfolio as $\boldsymbol{b}_l = \sum^M_{j=1} \omega^j_l \boldsymbol{b}^j_l$
24:     The market reveals actual price relative vectors $\boldsymbol{x}_l$
25:     Update the cumulative wealth as $S_l = S_{l-1} \cdot \boldsymbol{b}^\top_l \boldsymbol{x}_l(1 - \delta\|\boldsymbol{b}_l - \widetilde{\boldsymbol{b}}_{l-1}\|_1)$
26: **end for**
***

4. Experiments

4.1. Experimental Setting

Datasets. Experiments are conducted on six real-world datasets of daily stock returns, namely NYSE(O), NYSE(N), SP500, DJIA, TSE, and MSCI. The detailed information for each dataset is shown in Table 3. The datasets were downloaded from the OLPS GitHub repository.

Table 3: Datasets Information

Dataset Name	Region	Time Interval	Number of Assets	Frequency
NYSE(O)	America	5651	36	Daily
NYSE(N)	America	6431	23	Daily
SP500	America	1276	25	Daily
DJIA	America	507	30	Daily
TSE	Canada	1259	88	Daily
MSCI	China	1043	24	Daily

Baseline. We compare our method with 14 traditional policies listed in Table 2.

Experimental Setup. The transaction cost rate for buying and selling assets is set to 0.0005, as referenced from the configuration in [11]. The number of candidate policies M is set as 4. The historical window size w is 5. The sample size K of support set of each task is 10, while the sample size Q of the query set is 1. The epoch of each task is 20. The base-parameter learning rate α is 0.001, and the meta-learning rate β is 0.0005.

4.2. Comparison of LMPS-SMO to Baselines

Single-market test. For each dataset, We utilize the first three quarters of the data for meta-training purposes, while the remaining one quarter of the data is employed for meta-testing. Table 4 and Figure 6 summarize the comparison results of LMPS-SMO and the baseline method.

From Table 4, it can be observed that the LMPS-SMO method outperforms the candidate policies in terms of cumulative wealth and Sharpe ratio across all datasets. As shown in Figure 6, the LMPS-SMO method generally tends to learn one or two candidate policies while also being able to adjust policy allocations in a timely manner, resulting in its overall performance surpassing that of the individual learned candidate policies. Although

Table 4: Single-market performance of different strategies in 6 datasets

Datasets	NYSE(O)		NYSE(N)		SP500	
Strategy	Cumulative Wealth	Sharpe Ratio	Cumulative Wealth	Sharpe Ratio	Cumulative Wealth	Sharpe Ratio
LMPS-SMO	**67.44**	19.05	**7.39**	**3.55**	1.68	17.41
BAH	2.24	6.30	1.25	1.08	0.78^c	-10.44^c
CRP	2.68	8.93	1.43^c	1.46^c	0.82	-8.55
UP	2.68	8.93	1.44	1.50	0.82	-8.59
BestSoFar	0.94	-0.20	1.77	2.61	0.59	-11.41
EG	2.68^c	8.90^c	1.44	1.51	0.82	-8.62
ONS	3.84^c	9.09^c	0.59	-0.74	1.02	0.78
Anticor	11.00^c	9.46^c	5.32	3.16	1.48^c	12.14^c
PAMR	62.33	**19.51**	1.59^c	0.72^c	1.15	3.79
CWMR	42.45	17.46	1.49	0.62	1.14	3.59
OLMAR	41.34	14.60	1.61	0.58	**1.86**	**20.26**
RMR	56.53^c	16.38^c	1.46^c	0.47^c	1.67	15.97
WMAMR	28.31	13.28	0.24	-1.55	1.13^c	3.21^c
BNN	1.01	0.02	5.95^c	2.73^c	0.72^c	-8.21^c
CORN	53.00	18.24	1.83	1.43	1.13	3.59
Datasets	DJIA		TSE		MSCI	
Strategy	Cumulative Wealth	Sharpe Ratio	Cumulative Wealth	Sharpe Ratio	Cumulative Wealth	Sharpe Ratio
LMPS-SMO	**2.00**	**208.71**	3.23	34.42	**2.38**	**90.77**
BAH	1.02	3.86	0.89	-7.79	1.37	31.26
CRP	1.05	8.88	0.86	-9.23	1.36	29.88
UP	1.05	8.66	0.86^c	-9.23^c	1.36	30.02
BestSoFar	0.86^c	-34.19^c	0.56	-10.35	1.27	25.74
EG	1.05	8.62	0.86	-9.31	1.36^c	29.99^c
ONS	1.30	49.49	0.81	-4.68	1.25	15.03
Anticor	1.50^c	80.3^c	1.86	12.29	1.42	20.64
PAMR	1.80^c	171.64^c	1.93^c	17.95^c	1.69^c	35.04^c
CWMR	1.81	172.48	1.94	18.15	1.72	36.26
OLMAR	1.45	73.95	1.00	0.05	1.59	28.40
RMR	1.44	71.74	2.06^c	12.81^c	1.80	39.18
WMAMR	1.15	21.33	**4.93**	**55.29**	1.75^c	37.45^c
BNN	1.08^c	11.05^c	0.80	-3.8	1.39	20.92
CORN	0.85	-20.21	0.76^c	-4.76^c	2.17^c	87.63^c

$\cdot^c$: Candidate policies selected for each dataset.

Figure 6: Single-market cumulative return curves of different strategies in 6 datasets.

some traditional policies perform well across these datasets, the LMPS-SMO method consistently ranks high in terms of return and risk control metrics. By learning from the FTL and pattern-matching strategies, the LMPS-SMO strategy achieves higher returns, while learning from the benchmark and FTW strategies steers it towards stability, thereby effectively reducing risk.

Cross-Market test: We use four datasets (NYSE(O), NYSE(N), DJIA and SP500) to construct training tasks and select candidate policies. We then evaluate LMPS-CMO method using the TSE and MSCI datasets as the

test set for online portfolio management. The results are presented in Table 5 and Figure 7.

Table 5: Cross-market performance of different policies in TSE and MSCI datasets

Datasets	TSE		MSCI	
Strategy	Culmulative Wealth	Sharpe Ratio	Culmulative Wealth	Sharpe Ratio
LMPS-CMO	**149.57**	**17.54**	6.73	13.30
BAH	1.55	5.10	0.93	-0.55
CRP	1.51	4.76	0.95	-0.40
UP	1.51	4.78	0.95	-0.39
BestSoFar	0.42	-2.04	0.43	-6.07
EG	1.51^c	4.76^c	0.95^c	-0.40^c
ONS	1.34	1.28	0.85	-1.00
Anticor	10.40^c	6.81^c	2.93^c	6.51^c
PAMR	95.07^c	15.01^c	5.91^c	11.86^c
CWMR	71.49	13.76	6.07	12.04
OLMAR	21.36	6.50	7.76	13.10
RMR	90.24	11.38	**8.09**	**13.51**
WMAMR	43.32	9.55	3.92	8.03
BNN	1.01^c	0.02^c	1.44^c	2.21^c
CORN	2.92	3.03	4.22	10.74

$\cdot^c$: Candidate policies selected for each dataset.

The results demonstrate that within a framework of cross-learning from multiple datasets, the Meta-LMPS-online algorithm maintains strong performance. Whether on the TSE or MSCI datasets, the LMPS-CMO strategy showcases higher cumulative wealth and Sharpe ratios compared to the four candidate policies it learned. Moreover, it manages to contain volatility and maximum drawdown within a narrower range. This underscores the algorithm's robust learning capacity and exceptional adaptability to various environments.

4.3. Ablation studies

We carry out a series of ablation studies to examine the affect of meta-training and meta-testing on the performance of the Meta-LMPS-online algorithm. For this purpose, we designed the following three controlled experiences to compare against the completed Meta-LMPS-online algorithm.

- E2E-online: Without using the mixture policies learning framework, the network takes the return sequences of assets as input and the allo-
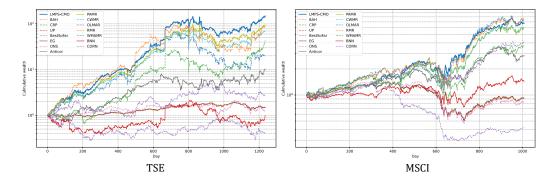
Figure 7: Cross-market cumulative return curves of different strategies in TSE and MSCI datasets

cation ratios for each asset as output. Apart from this, other settings remain the same as the LMPS-online algorithm.

- LMPS-MIRP: Without using meta-training to generate initial parameters for meta-testing, it randomly generates initial parameters and updates these parameters during the meta-testing phase.

- LMPS-FixedInit: During testing phase, it consistently uses the initial parameters generated in the meta-training phase to determine policy allocations, without any updating in the meta-testing period.

Single-market test: The final results are displayed in Table 6 and Figure 8. We include the BAH strategy for comparison to observe the overall market state.

The results show that the LMPS-SMO strategy outperforms others in cumulative returns across 6 datasets. The E2E-online strategy follows market trends but generally performs worse than the mixture policies learning algorithm, demonstrating the benefits of the mixture policies approach. LMPS-FixedInit tends to perform well for specific datasets but lacks adaptability across different market conditions, highlighting the importance of meta-testing. LMPS-MIRP also performs worse than LMPS-SMO, emphasizing the need for meta-training.

Cross-market test: Due to the lack of cross-market transferability in the E2E-online method, we only compare LMPS-CMO with LMPS-MIRP, LMPS-FixInit, and BAH in the ablation experiments for cross-market test. The results are displayed in Table 7 and Figure 9.

22

Table 6: Ablation study of the LMPS-SMO method in 6 datasets

Datasets	Strategy	Culmulative Wealth	Annualized Return	Annualized Volatility	Maximum Drawdown	Sharpe Ratio
NYSE(O)	**LMPS-SMO**	**67.44**	**111.76%**	5.93%	27.43%	**19.05**
	E2E-online	2.99	21.74%	4.64%	44.00%	4.68
	LMPS-MIRP	16.70	65.80%	5.73%	35.05%	11.48
	LMPS-FixedInit	54.70	105.18%	6.50%	43.60%	16.17
	BAH	2.26	15.80%	**2.47%**	**20.37%**	6.39
NYSE(N)	**LMPS-SMO**	**7.39**	**37.04%**	10.43%	83.48%	**3.55**
	E2E-online	1.87	10.36%	6.67%	68.19%	1.55
	LMPS-MIRP	3.77	23.25%	12.00%	78.67%	1.94
	LMPS-FixedInit	3.82	23.51%	7.99%	75.33%	2.94
	BAH	1.24	3.45%	**3.21%**	**53.59%**	1.07
SP500	**LMPS-SMO**	**1.68**	**52.21%**	3.12%	22.48%	**17.41**
	E2E-online	0.80	-16.72%	**1.79%**	30.48%	-9.34
	LMPS-MIRP	1.40	32.61%	3.04%	**22.46%**	10.71
	LMPS-FixedInit	1.11	9.15%	3.25%	31.16%	2.82
	BAH	0.78	-19.24%	1.84%	32.46%	-10.48
DJIA	**LMPS-SMO**	**2.00**	**400.82%**	1.90%	17.83%	**208.71**
	E2E-online	1.20	53.73%	1.61%	17.79%	33.47
	LMPS-MIRP	1.47	148.09%	1.37%	14.36%	108.18
	LMPS-FixedInit	1.29	82.32%	1.30%	**10.91%**	63.21
	BAH	0.99	-2.87%	**1.16%**	20.31%	-2.47
TSE	**LMPS-SMO**	**3.23**	**158.84%**	4.91%	32.22%	**34.42**
	E2E-online	0.98	-1.87%	1.29%	**29.48%**	-1.46
	LMPS-MIRP	1.93	74.91%	5.17%	51.13%	14.50
	LMPS-FixedInit	2.04	83.35%	5.55%	39.52%	15.01
	BAH	0.88	-10.46%	**1.25%**	30.02%	-8.39
MSCI	**LMPS-SMO**	**2.38**	**146.76%**	1.62%	**12.40%**	**90.77**
	E2E-online	1.47	49.38%	2.29%	21.67%	21.58
	LMPS-MIRP	1.77	81.26%	1.84%	18.69%	44.20
	LMPS-FixedInit	1.70	73.80%	2.11%	20.00%	34.98
	BAH	1.37	38.58%	**1.22%**	12.69%	31.71

Figure 8: Ablation study of the LMPS-SMO method in 6 datasets
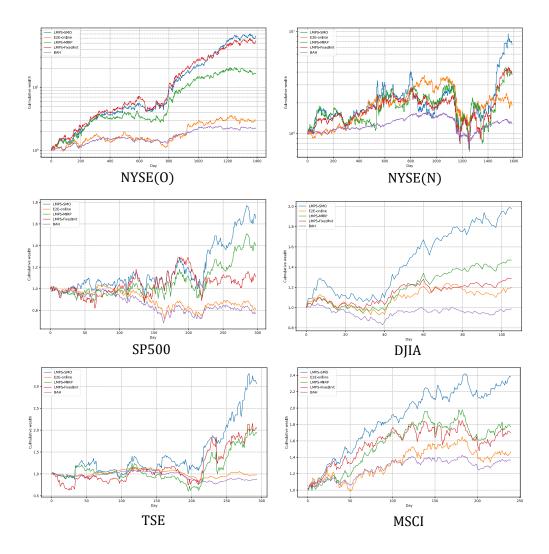
The results indicate that LMPS-CMO outperforms the other three strategies in terms of cumulative returns and Sharpe ratio, effectively mitigating risks. This highlights the robust learning capability of LMPS-CMO. In practical applications, compared to traditional end-to-end models, the proposed model efficiently utilizes data from various markets and historical data.

Table 7: Ablation study of the LMPS-CMO method in TSE and MSCI datasets

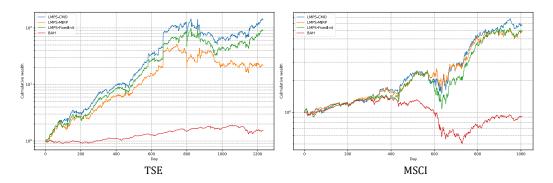| Dataset	Strategy	Culmulative Wealth	Annualized Return	Annualized Volatility	Maximum Drawdown	Sharpe Ratio
TSE	**LMPS-CMO**	**149.57**	**171.96**%	10.03%	65.21%	**17.54**
	LMPS-MIRP	21.80	86.80%	9.49%	68.99%	9.15
	LMPS-FixedInit	90.20	149.13%	10.10%	67.20%	14.76
	BAH	1.54	9.18%	**1.81**%	**30.02**%	5.08
TSE	**LMPS-CMO**	**6.73**	**60.89**%	4.56%	39.36%	**13.30**
	LMPS-MIRP	5.86	55.11%	4.57%	**36.40**%	12.05
	LMPS-FixedInit	5.96	55.76%	4.70%	57.14%	11.88
	BAH	0.91	-2.22%	**3.15**%	64.75%	-0.71

Figure 9: Ablation study of the LMPS-CMO method in TSE and MSCI datasets

5. Conclusion

In this paper, we introduce the Meta-LMPS-online algorithm for online portfolio selection problem, applying the meta-learning approach to mixture policies. The core of meta-learning lies in extracting patterns and rules from diverse market data or historical trends, empowering decision-makers to quickly adapt to new investment landscapes. Meanwhile, the mixture policies learning framework enhances stability and versatility by consolidating weighted combinations of multiple candidate polices, thereby refining overall performance. This paper introduces LMPS-SMO strategy for individual markets and LMPS-CMO strategy for multiple markets, effectively addressing the limitations of end-to-end networks in learning from diverse markets. Empirical tests validate the effectiveness of meta-learning approach and the mixture policies learning framework, highlighting the substantial aid provided by this algorithm in addressing online portfolio selection problems.

6. Acknowledgments

References

[1] Amit Agarwal, Elad Hazan, Satyen Kale, and Robert E. Schapire. Algorithms for portfolio management based on the newton method. In *Proceedings of the 23rd International Conference on Machine Learning*, 2006.

[2] Karhan Akcoglu, Petros Drineas, and Ming-Yang Kao. Fast universalization of investment strategies. *SIAM Journal on Computing*, 2004.

[3] Chaher Alzaman. Deep learning in stock portfolio selection and predictions. *Expert Systems with Applications*, 2024.

[4] Allan Borodin, Ran El-Yaniv, and Vincent Gogan. Can we learn to beat the best stock. In *Advances in Neural Information Processing Systems*, 2003.

[5] Zeynep Cipiloglu Yildiz and Selim Baha Yildiz. A portfolio construction framework using lstm-based stock markets forecasting. *International Journal of Finance & Economics*, 2022.

[6] Mark HA Davis and Andrew R Norman. Portfolio selection with transaction costs. *Mathematics of Operations Research*, 1990.

[7] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, 2017.

[8] Li Gao and Weiguo Zhang. Weighted moving average passive aggressive algorithm for online portfolio selection. In *2013 5th International Conference on Intelligent Human-Machine Systems and Cybernetics*, 2013.

[9] Riccardo Grazzi, Valentin Flunkert, David Salinas, Tim Januschowski, Matthias W. Seeger, and Cédric Archambeau. Meta-forecasting by combining global deep representations with local adaptation. *CoRR*, 2021.

[10] Fengchen Gu, Zhengyong Jiang, and Jionglong Su. Application of features and neural network to enhance the performance of deep reinforcement learning in portfolio management. In *2021 IEEE 6th International Conference on Big Data Analytics (ICBDA)*, 2021.

[11] Sini Guo, Jiawen Gu, Christopher H. Fok, and Wai-Ki Ching. Online portfolio selection with state-dependent price estimators and transaction costs. *European Journal of Operational Research*, 2023.

[12] László Györfi, Gábor Lugosi, and Frederic Udina. Nonparametric kernel-based sequential investment strategies. *Mathematical Finance: An International Journal of Mathematics, Statistics and Financial Economics*, 2006.

[13] Myoung Hoon Ha, Seung-geun Chi, Sangyeop Lee, Yujin Cha, and Moon Byung-Ro. Evolutionary meta reinforcement learning for portfolio optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference*, 2021.

[14] Elad Hazan and C. Seshadhri. Efficient learning algorithms for changing environments. In *Proceedings of the 26th Annual International Conference on Machine Learning*, 2009.

[15] David P. Helmbold, Robert E. Schapire, Yoram Singer, and Manfred K. Warmuth. On-line portfolio selection using multiplicative updates. *Mathematical Finance*, 1998.

[16] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[17] Dingjiang Huang, Junlong Zhou, Bin Li, Steven C. H. Hoi, and Shuigeng Zhou. Robust median reversion strategy for online portfolio selection. *IEEE Transactions on Knowledge and Data Engineering*, 2016.

[18] J. L. Kelly. A new interpretation of information rate. *The Bell System Technical Journal*, 1956.

[19] Sang Il Lee and Seong Joon Yoo. Threshold-based portfolio: The role of the threshold and its applications. *The Journal of Supercomputing*, 2017.

[20] Bin Li and Steven C. H. Hoi. On-line portfolio selection with moving average reversion, 2012.

[21] Bin Li and Steven C. H. Hoi. Online portfolio selection: A survey. *ACM Computing Surveys*, 2014.

[22] Bin Li, Steven C. H. Hoi, and Vivekanand Gopalkrishnan. Corn: Correlation-driven nonparametric learning approach for portfolio selection. *ACM Transactions on Intelligent Systems and Technology*, 2011.

[23] Bin Li, Steven C. H. Hoi, Peilin Zhao, and Vivekanand Gopalkrishnan. Confidence weighted mean reversion strategy for on-line portfolio selection. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 2011.

[24] Bin Li, Peilin Zhao, Steven C. H. Hoi, and Vivekanand Gopalkrishnan. Pamr: Passive aggressive mean reversion strategy for portfolio selection. *Machine Learning*, 2012.

[25] Bin Li, Jialei Wang, Dingjiang Huang, and Steven CH Hoi. Transaction cost optimization for online portfolio selection. *Quantitative Finance*, 2018.

[26] Xuelong Li, Hongli Li, and Yongsheng Dong. Meta learning for task-driven video summarization. *IEEE Transactions on Industrial Electronics*, 67(7):5778–5786, 2019.

[27] Hong Lin, Yong Zhang, and Xingyu Yang. Online portfolio selection of integrating expert strategies based on mean reversion and trading volume. *Expert Systems with Applications*, 238:121472, 2024.

[28] Shaohui Ma and Robert Fildes. *Large-scale time series forecasting with meta-learning.* 2023.

[29] Yilin Ma, Ruizhu Han, and Weizhong Wang. Portfolio optimization with return prediction using deep learning and machine learning. *Expert Systems with Applications*, 2021.

[30] Michael JP Magill and George M Constantinides. Portfolio selection with transactions costs. *Journal of Economic Theory*, 1976.

[31] Xavier Martínez-Berbero, Roberto Cervelló-Royo, and Javier Ribal. Portfolio optimization with prediction-based return using long short-term memory neural networks: Testing on upward and downward european markets. *Computational Economics*, 2024.

[32] Hui Niu, Siyuan Li, and Jian Li. Metatrader: An reinforcement learning approach integrating diverse policies for portfolio optimization. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, 2022.

[33] Akhter Mohiuddin Rather. Lstm-based deep learning model for stock prediction and predictive optimization model. *EURO Journal on Decision Processes*, 2021.

[34] Jun Shu, Qi Xie, Lixuan Yi, Qian Zhao, Sanping Zhou, Zongben Xu, and Deyu Meng. Meta-weight-net: Learning an explicit mapping for sample weighting. *Advances in neural information processing systems*, 32, 2019.

[35] Erik Sorensen, Ryan Ozzello, Rachael Rogan, Ethan Baker, Nate Parks, and Wei Hu. Meta-learning of evolutionary strategy for stock trading. *Journal of Data Analysis and Information Processing*, 2020.

[36] M. Thomas, A. Cover, and C. B. Universal portfolios. *Mathematical Finance*, 1991.

[37] Ayse Sinem Uysal, Xiaoyue Li, and John M. Mulvey. End-to-end risk budgeting portfolio optimization with neural networks. *Papers*, 2021.

[38] Ricardo Vilalta and Youssef Drissi. A perspective view and survey of meta-learning. *Artificial Intelligence Review*, 2002.

[39] Gerald Woo, Chenghao Liu, Doyen Sahoo, Akshat Kumar, and Steven Hoi. Deeptime: Deep time-index meta-learning for non-stationary time-series forecasting. *arXiv preprint arXiv:2207.06046*, 2022.

[40] Xingyu Yang, Xiaoteng Zheng, Jiahao Li, and Qingmei Huang. Aggregating closing position experts for online portfolio selection. *Applied Economics Letters*, pages 1–12, 2024.

[41] Chao Zhang, Zihao Zhang, Mihai Cucuringu, and Stefan Zohren. A universal end-to-end approach to portfolio optimization via deep learning, 2021.

Appendix A. Background of Meta-Learning

Meta-learning is a newly flourished research direction in the machine learning field. The goal of meta-learning is to learn prior knowledge from many similar tasks such that the learned knowledge can be fast adapted to new unseen tasks. In contrast to traditional machine learning method involves manually adjusting hyperparameters, meta-learning aims to achieve automatic hyperparameters adjustment. Therefore, meta-learning can be considered as learning beyond the traditional levels of machine learning. The distinction between traditional supervised machine learning and meta-learning is illustrated in Table 8.

Table 8: The difference between machine learning and meta-learning

	Supervised Machine Learning	Meta-Learning
Object of fitting	Data pairs$\{(x_i, y_i)\}_{i=1}^{n}$ Feature & Label	Tasks$\{\mathcal{T}_i^{sup}, \mathcal{T}_i^{que}\}_{i=1}^{m}$ Support set & Query set
Trained parameters	Learning machine parameters for predicting data labels	Meta-learning machine parameters for predicting machine learning hyperparameters
Evaluation metric	Accuracy of predictions on training data labels	Learning performance obtained on the query set by the learning method under the hyperparameters assignments obtained from the support set

MAML(Model-Agnostic Meta-Learning) is one of the classic approaches that applies meta-learning to Few-Shot Learning (FSL). Its primary setup involves initializing hyperparameters as the initial values for model training. The objective of the meta-learning machine in MAML is to learn these initial values from a training task set and then directly apply them to prediction tasks. This allows achieving a good training result for the target task after a few iterations. The complete training process of MAML is shown in the algorithm 4.

Algorithm 4 MAML

Input: $p(\mathcal{T})$: distribution over tasks
Input: α, β: step size hyperparameters
Input: K, Q: the number of samples in the Support Set and Query Set
Input: f_θ: the candidate model
 1: randomly initialize θ
 2: **while** not done **do**
 3:    Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
 4:    **for** $\mathcal{T}_i$ **do**
 5:        Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta(\mathcal{T}_i^{sup}))$ with respect to K examples
 6:        Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta(\mathcal{T}_i^{sup}))$
 7:    **end for**
 8:    Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'}(\mathcal{T}_i^{que}))$
 9: **end while**