Integrating Local Learning to Improve Deep-Reinforcement-Learning-based Pairs Trading Strategies

Wei-Che Chang
Inst. of Computer Science and Engineering
National Yang Ming
Chiao Tung University
Hsinchu City, Taiwan
wcchang.cs09@nycu.edu.tw

Tian-Shyr Dai
Dept. of Information Management and Finance
National Yang Ming
Chiao Tung University
Hsinchu City, Taiwan
cameldai@mail.nctu.edu.tw

Ying-Ping Chen
Dept. of Computer Science
National Yang Ming
Chiao Tung University
Hsinchu City, Taiwan
ypchen@cs.nycu.edu.tw

Chin-Yi Hsieh
Inst. of Computer Science and Engineering
National Yang Ming
Chiao Tung University
Hsinchu City, Taiwan
eason311551164.cs11@nycu.edu.tw

Yu-Wei Chang
Dept. of Computer Science
National Yang Ming
Chiao Tung University
Hsinchu City, Taiwan
alen610227.cs09@nycu.edu.tw

Yu-Han Huang
Dept. of Computer Science
National Yang Ming
Chiao Tung University
Hsinchu City, Taiwan
john01217.cs09@nycu.edu.tw

***Abstract*—Instead of trying to predict unpredictable market trends, influenced by various complex factors that are challenged to be fully captured in a machine learning model, financial experts often adopt pairs trading. This strategy involves simultaneously trading two stocks to eliminate market trends. The portfolio value of a carefully selected stock pair oscillates around a mean level, with investors longing the underpriced and shorting the overpriced portfolios to profit or stop loss when the portfolio's value reverts or diverges significantly.**

The timing of trading actions highly depends on the characteristics of constituent stocks and significantly influences trading performance. Past literature either trains a single machine learning model with all stock pairs' trading data or multiple models, each for a specific stock pair. While the former approach avoids overfitting due to sufficient data, it struggles to capture the unique characteristics of different stock pairs. Conversely, the latter approach focuses on specific pairs but faces limited training data.

To address this dilemma, our paper leverages local learning to recommend the best trading actions. We group trading data by similarity and train each model with data from a specific group. The trained model then predicts optimal actions for stock pairs in that group. Using Gaussian mixture models for data grouping in local learning outperforms other methods in scenarios with limited data for most stock pairs.

***Keywords*—Deep Reinforcement Learning, Local Learning, Pairs Trading, Unsupervised Learning**

I. Introduction

While global learning trains a single machine learning model with the entire dataset to capture general rules or patterns, it often fails to capture specific local patterns unique to subsets of the data. For instance, growth stocks and blue-chip stocks are distinct stock types. The former typically exhibits higher growth rates with significant price volatility, while the latter represents well-established companies with stable earnings but limited rapid growth. It is therefore intuitive to apply local learning to train separate models for these stock types, assuming sufficient data is available. Considering the diversity of stock types necessitates different models to learn distinct patterns, we identify a challenge: excessively categorizing transaction data can lead to overfitting due to the limited training data available for each category. To demonstrate this issue and our approach, we focus on pairs trading (PTS), a popular investment strategy. Past literature has typically trained a machine learning model with trading data for each stock pair individually (referred to as "individual learning") or used all stock pairs (global learning) to predict trading actions. Our paper modifies this learning setup, employing local learning to enhance trading performance by addressing the challenges of insufficient training data and effectively capturing the distinct patterns of various stock types.

To effectively utilize local learning under the constraints of limited training data resulting from partitioning data into groups for training different models, it is crucial to address dataset shift problems. These are often intensified by economic changes and black swan events such as COVID-19. Consequently, financial experts increasingly adopt stable, market-neutral strategies like the pairs trading strategy (PTS). Recently, PTS has gained significant attention in machine learning literature, as seen in works like [1]–[7], among others. PTS mitigates market trend risks by longing one stock and shorting another, thus maintaining a stationary portfolio value (also known as the spread process) with a consistent mean level over time. Investors long (buy) the portfolio when it's underpriced and short (sell) when overpriced, closing the

979-8-3503-5409-6/24/$31.00 ©2024 IEEE
DOI 10.1109/CAI59869.2024.00139

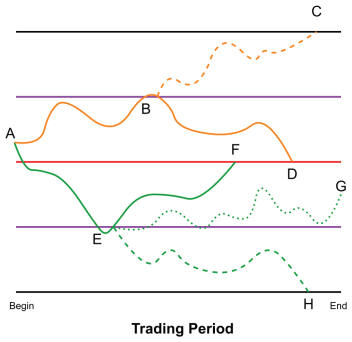

Fig. 1. Trading Period Scenarios

position once the spread process reverts to the mean, thereby profiting from the spread process irrespective of market trend changes.

While the selection of feasible stock pairs and appropriate investment weights for constructing PTS can be derived using various statistical approaches, as reviewed in [8], the timing for executing trading actions, which significantly influences investment performance, is widely studied in related machine learning literature. Unlike the works of [3], [6], which determined actions by directly predicting spread movements, this paper follows the approach of most PTS works. It establishes that open position and stop loss actions are triggered when the spread process meets the respective opening or stop loss thresholds. As illustrated in Fig. 1, which is modified from Fig. 2 of [5], we open positions when the spread process diverges enough to meet the open threshold (denoted by purple lines) and close them when it converges back to the mean level (denoted by the red line). Additionally, positions may be closed for a stop loss when the spread process diverges further, reaching the stop loss threshold (denoted by black lines). Notably, increasing the price difference between the open threshold and the mean level can enhance potential profit per trade, but at the expense of reduced opportunities to open positions. Moving the stop-loss threshold farther from the mean price level decreases the likelihood of prematurely cutting profitable trades, but it increases the potential losses from non-profitable ones. Therefore, selecting appropriate thresholds according to the properties of the constituent stocks in the portfolio, to maximize PTS profits, presents an interesting yet challenging problem.

To perform PTS trading with suitable open and stop-loss thresholds, this paper, following the settings in [4], [5], divides each trading day's hours into the formation period (the first 150 minutes after market opening) and the trading period (the remaining time). We assess the eligibility of stock pairs for PTS by applying the Johansen cointegration test to stock price processes during the formation period, then trading eligible pairs in the trading period. It's evident that not all stock pairs are consistently feasible for PTS trading each day. For instance, pairs like Coca Cola and Pepsi, in the soft drink category, often show high correlation and are frequently suitable for PTS. However, many stock combinations are only occasionally eligible. For example, in the Taiwan stock market data set, over 90% of stock pairs are feasible for trading only about once per year. This sporadic eligibility makes predicting feasible thresholds for these pairs challenging due to the limited training data available for PTS trading.

Previous PTS literature adopts either individual or global learning approaches to predict PTS thresholds. To capture specific stock price patterns of constituent stocks in a pair, [1], [2], [3], and [7] trained eligible trading data of each stock pair with one unique machine learning model. From now on, we refer to this method as "individual learning." Although this approach allows the model to fully learn the features specific to that stock pair, it also fails to train models for stock pairs with limited eligible trading data. On the other hand, [4]–[6] adopted the global learning concept (see [10]) by training a single machine learning model with all eligible trading data of all stock pairs during the training period. Although the resulting model captures the general properties for PTS, it fails to capture the specific properties of individual pairs.

To capture the advantages of the aforementioned individual and global learning while alleviating their drawbacks, this article utilizes local learning [11] to capture specific properties of various constituent stocks without the constraints of limited training data. Specifically, all stock pairs are divided into clusters based on their spread patterns using unsupervised learning. Then, each machine learning model is trained with the trading data of stock pairs belonging to the same cluster. Consequently, each model can learn the specific patterns of its cluster, generating open and stop-loss thresholds that maximize profits according to the cluster's properties. Note that our approach avoids losing trading opportunities by grouping seldom traded stock pairs into clusters with similar characteristics. We set the number of clusters to ensure that each contains enough training data, thus avoiding overfitting. Since the spread processes in this paper are constructed based on the Johansen cointegration test [12], which can be treated as complex linear combinations of Gaussian white noises, we cluster these processes using the Gaussian Mixture Model (GMM). Empirical results indicate that this distribution clustering approach (i.e., GMM) performs better than other common unsupervised learning clustering methods.

This paper is organized as follows: In Section II, we review previous machine-learning-based PTS studies and the concepts of global and local learning. Section III details our proposed model. Section IV compares the PTS performance by training with different data clustering and illustrates the superiority of local learning with GMM clustering. Section V concludes this paper.

II. Literature Review

Two similar concepts of local learning were presented by [11] and [13]. The former approach predicted for each testing data point by training a machine learning model with predetermined amounts of nearby training data. However, this method

is inefficient as it requires searching for nearby training data to train the model for each prediction. The latter approach, known as the "mixture of experts", divided all training data into different clusters and trains each machine learning model with data from the corresponding cluster. For instance, [13] used the expectation-maximization algorithm to divide training data into clusters for training local neural network models. This concept is commonly used to train support vector machines (SVM)s in recent studies. [14] proposed a DTSVM that uses a decision tree to divide the training data and trained each local SVM with a group of data. [15] introduced a CSVM that divides training data with the K-means method and adds global regularization to prevent overfitting in each local SVM. Similarly, KSVM ([16]) and BCSVM ([17]) also used the K-means method to divide training data for local SVMs.

[1], [2], and [4] used reinforcement learning (RL) to select open and stop-loss thresholds from a heuristically determined set, a practice that limits profits as shown in [5]. They demonstrated significant profit improvement by proposing a representative threshold mechanism utilizing the training dataset. Our paper integrates RL, the representative-threshold concept, and local learning to enhance PTS performance.

Individual learning was adopted by [1], [2], [3], and [7] to train each RL model with specific stock pair trading data. [1] transformed the threshold selection problem into a multi-arm bandit problem using an RL with one state. [2] utilized spread processes of the same stock pair as states to train a deep Q network (DQN) with six heuristic actions, reflecting a combination of open and stop-loss thresholds, as proposed by [18]. [3] designed features as the state of a double deep Q-Network (DDQN) to capture the mean-reverting property of the spread process. [7] proposed a hybrid deep reinforcement learning (DRL) method consisting of two independent DRL networks: one for predicting spread movements and another for selecting stop-loss thresholds to minimize significant losses. While individual learning is adept at capturing each stock pair's unique characteristics, it is often limited by insufficient training data. For instance, some stock pairs may lack sufficient eligible trading days within the training period, or eligibility may only become apparent during testing. This issue can impair RL models from recommending accurate thresholds, leading to loss trading opportunities. Unlike prior studies which manually selected up to 38 highly correlated stock pairs for PTS, our local-learning-based approach enables our DRL to suggest thresholds for a wider array of infrequently traded pairs. Our experiments show that trading up to 4211 stock pairs in the Taiwan stock markets significantly increases overall profits. Omitting these pairs would markedly degrade performance.

In contrast, [4], [5], and [6] adopted global learning to train a single machine learning model with all trading data. [5] proposed a representation labeling mechanism, replacing the manually selected thresholds used in past literature. [4] improved [2]'s framework by integrating a deep learning model to predict the probability of a spread process losing its stationary properties. [6] utilized dueling DQN to predict spread process movements and introduced reward shaping to expedite learning. Although global learning uses all PTS-eligible stock pairs, preventing the loss of infrequent trading opportunities, experiments show it struggles to capture specific characteristics of certain stock pairs, leading to subpar trading performance. Conversely, our proposed local learning approach groups stock pairs with similar statistical properties, enabling machine learning models to better grasp the characteristics of spread processes within these groups.

III. Proposed Method

The structure of our DDQN, which is designed for predicting PTS open and stop-loss thresholds using representation labeling and local learning, is depicted in Fig. 2. We categorize all trading days (denoted as $D_1, \ldots$) into training and testing periods. For each trading day D_n in the training period, we divide the trading hours into two parts: the formation period (first 150 minutes) and the trading period (remaining hours). Following [4] and [5], we use stock price data from the formation period to select eligible stock pairs for PTS, which are then traded during the trading period. Specifically, we apply the Johansen cointegration test to the price processes of all stock pairs in the formation periods to determine eligibility for D_n. Finally, we compile eligible stock pairs from all trading days $D_1, \ldots, D_N$ within the training period. This compilation helps us identify a set of representative open and stop-loss thresholds, as suggested by [5], to form the action set A for our DDQN.

While some price processes are highly correlated, making their stock pairs frequently eligible for PTS, the eligibility of pairs composed of less correlated stocks can be rare. To properly execute PTS by capturing various spread patterns contributed by different constituent stocks, we consider three learning methods. The global learning adopted by [4]–[6], individual learning by [1]–[3], [7], and our proposed local learning illustrated in the central block of Fig. 2. The details are further in Fig. 3.

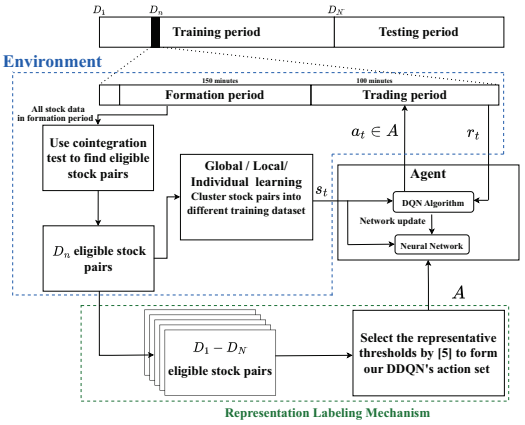

Fig. 2. Flow chart of DDQN in PTS

The global learning approach trains a DDQN using all PTS-eligible trading data from the training period, effectively capturing general PTS properties. However, it overlooks the unique characteristics of different stock pairs. In contrast, individual learning segments PTS-eligible transaction data by stock pairs, training a specialized DDQN for each pair. While this method excels in capturing pair-specific properties, it falls short for less correlated stocks due to inadequate eligible data. Consequently, this leads to missing opportunities for profit from these infrequently traded pairs.

To leverage the strengths of both global and individual learning while minimizing their weaknesses, we adopt the local learning concept. This involves first dividing the stock pairs into several clusters based on their statistical properties. Then, for each cluster, we compile a corresponding training dataset from all PTS-eligible trading data of the pairs in that cluster. The DDQN, trained with this dataset, predicts trading actions for pairs within the same cluster. Notably, some stock pairs may only become eligible for PTS during the testing period and are not grouped into any cluster during training. In such instances, we assign these pairs to clusters which are based on the classification determined in the training stage. We then input each spread process s_t during the formation period into the trained DDQN of the assigned cluster, and recommend an action a_t from the action set A.

To cluster stock pairs, we consider unsupervised learning methods, categorized into hierarchical-based, density-based, centroid-based, and distribution-based approaches [25]. Given the unknown nature of PTS-eligible pairs and spread processes in the testing dataset during training, hierarchical and density-based methods are less feasible. Consequently, we opt for the most representative centroid-based and distribution-based methods, namely k-means [26] and Gaussian Mixture Model (GMM) [27], in our experiments. The spread process, resembling a linear combination of normal random white noise[1], makes GMM a suitable choice for clustering.

We use the dataset determined in Fig. 3 to train a DDQN (proposed in [18], [19]) with specific inputs, actions, and reward functions defined as follows.

- Inputs: For each trading day, the first 150-minute price processes of constituent stocks and the spread process s_t are input to the DDQN.
- Action: Each action a_t is a combination of an opening and a stop-loss threshold selected from the action set A constructed by the representation labeling mechanism proposed by [5]. The action recommended by the DDQN is used to trade the stock pair during the trading period.
- Reward functions: We adopted the reward function from [2]. Denote the PTS profit and loss by r_t. We open the position when the spread meets the open threshold, as illustrated by B and E in Fig. 1. The profit or loss of the trade can be categorized into the following scenarios:
  - Normal close: The spread sucessfully reverts back to the mean level (like D and F in Fig. 1) and we close the position to make profit with the reward function:

$$r_t = 1000 \times r_t \tag{1}$$

  - Stop-loss close: We close the position to stop the loss when the spread reaches the stop-loss threshold (like C and H) with the reward function:

$$r_t = -1000 \times |r_t| \tag{2}$$

  - Forced exit: To avoid the risk for keeping the PTS portfolio overnight, the portfolio is forced to close at the end of the trading day (like G) if neither normal nor stop-loss close events are triggered. The reward function is defined as

$$r_t = -500 \times |r_t| \tag{3}$$

[1] [5] modeled this as a vector error correction model in Equation (1)

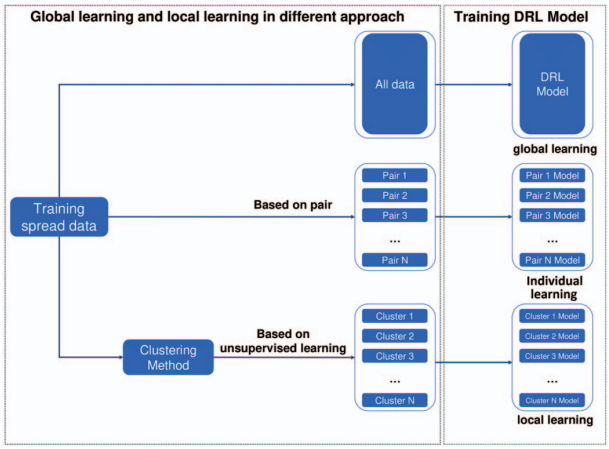

Fig. 3. Global, Individual, and Local Learning for PTS

To avoid the overfitting problem, we use the validation set to check the win rate of PTS in each (training) epoch and select the model with the highest win rate for executing PTS during the testing period. For the subsequent experiments, we used the package provided by [30] and the hyper-parameter settings are shown in Table I.

TABLE I
DETAILED HYPER-PARAMETER SETTINGS USED IN PACKAGE BY [30].

class name	hyper-parameter
DQNAgent	gamma=0.95, batch_size=64, nb_steps_warmup=1000, train_interval=1000 target_model_update=0.001, delta_clip=1
SequentialMemory	limit=100000, window_length=1
LinearAnnealedPolicy	inner_policy=EpsGreedyQPolicy(), attr='eps', value_max=1.0, value_min=0, value_test=0
Adam	lr=0.001, decay=0

IV. Empirical Results

We assess PTS performance using stocks from the Taiwan Top 50 ETF (0050) and the Taiwan Mid-Cap 100 ETF (0051), comparing global, individual, and local learning methods as

TABLE II
COMPARISON OF CLUSTERING METHODS FROM TOP 29 TO TOP 425.

Training Method	**individual learning**	**global learning**	**local learning**	
Clustering Method	x	x	k-**means**	**GMM**
Top 29 (count≥110)				
Profit (thousands)	**157.03**	145.29	154.42	151.29
Total open number	**1153**	1082	1130	1105
Win rate (%)	**92.11**	91.4	91.95	91.95
Normal close rate (%)	**89.68**	88.91	89.38	89.5
Sharpe ratio (daily-based)	**12.6667**	11.9756	12.663	12.6585
Sharpe ratio (trade-based)	**0.8184**	0.7655	0.8068	0.8028
Profit per open (thousands)	0.1362	0.1343	0.1367	**0.1369**
MDD	**-2.04**	-2.19	**-2.04**	-2.19
Top 63 (count≥100)				
Profit (thousands)	**351.38**	348.23	332.46	336.04
Total open number	2449	**2492**	2355	2333
Win rate (%)	**90.24**	89.57	89.43	90.18
Normal close rate (%)	88.81	88.12	88.2	**89.07**
Sharpe ratio (daily-based)	15.0411	14.3802	**15.159**	14.2769
Sharpe ratio (trade-based)	**0.7759**	0.6973	0.7708	0.7691
Profit per open (thousands)	0.1435	0.1397	0.1412	**0.144**
MDD	**-1.95**	-3.41	**-1.95**	-2.2
Top 127 (count≥90)				
Profit (thousands)	653.09	683.51	665.01	**692.24**
Total open number	4740	4736	4854	**4959**
Win rate (%)	88.59	**89.53**	88.79	**89.53**
Normal close rate (%)	86.52	**88.37**	87.62	88.34
Sharpe ratio (daily-based)	12.7439	**13.0446**	11.3814	12.4109
Sharpe ratio (trade-based)	0.6167	0.6202	0.6178	**0.6256**
Profit per open (thousands)	0.1378	**0.1443**	0.137	0.1396
MDD	-10.36	**-6.59**	-17.51	-9.98
Top 237 (count≥80)				
Profit (thousands)	1212.68	1278.89	1295.94	**1356.47**
Total open number	8369	8404	8728	**8891**
Win rate (%)	86.84	88.16	87.92	**88.47**
Normal close rate (%)	84.47	87.14	86.88	**87.45**
Sharpe ratio (daily-based)	11.3445	11.3686	11.012	**12.1073**
Sharpe ratio (trade-based)	0.5427	0.5802	0.5522	**0.5843**
Profit per open (thousands)	0.1449	0.1522	0.1485	**0.1526**
MDD	-17.29	-25.98	**-16.25**	-17.43
Top 425 (count≥70)				
Profit (thousands)	2213.36	2463.95	2476.94	**2546.43**
Total open number	13486	13736	**14423**	14339
Win rate (%)	85.66	86.9	86.98	**87.52**
Normal close rate (%)	83.61	86.19	86.14	**87.17**
Sharpe ratio (daily-based)	9.3011	9.6444	9.5473	**10.1049**
Sharpe ratio (trade-based)	0.4891	0.5274	0.5235	**0.5495**
Profit per open (thousands)	0.1641	**0.1794**	0.1717	0.1776
MDD	-21.67	**-7.73**	-47.63	-19.26

TABLE III
COMPARISON OF CLUSTERING METHODS FROM TOP 732 TO TOP 4211.

Training Method	**individual learning**	**global learning**	**local learning**	
Clustering Method	x	x	k-**means**	**GMM**
Top 732 (count≥60)				
Profit (thousands)	3316.73	3877.7	**3879.0**	3876.75
Total open number	20188	21803	**21815**	21219
Win rate (%)	83.82	85.74	85.63	**86.45**
Normal close rate (%)	82.22	85.68	85.83	**86.99**
Sharpe ratio (daily-based)	7.7419	**7.9828**	7.8604	7.7394
Sharpe ratio (trade-based)	0.4145	0.4641	0.4573	**0.472**
Profit per open (thousands)	0.1643	0.1779	0.1778	**0.1827**
MDD	**-21.49**	-42.72	-33.84	-40.0
Top 1227 (count≥50)				
Profit (thousands)	5125.19	**6252.54**	6122.04	6123.08
Total open number	29287	**32548**	31402	30918
Win rate (%)	82.03	84.29	84.15	**84.64**
Normal close rate (%)	80.81	84.91	84.54	**85.36**
Sharpe ratio (daily-based)	5.0752	**5.5276**	5.289	5.2811
Sharpe ratio (trade-based)	0.3492	0.3965	0.3922	**0.4006**
Profit per open (thousands)	0.175	0.1921	0.195	**0.198**
MDD	**-54.71**	-55.82	-54.73	-60.75
Top 1909 (count≥40)				
Profit (thousands)	7596.62	9633.21	9491.22	**9921.89**
Total open number	39962	41999	42592	**42742**
Win rate (%)	80.28	83.31	83.02	**84.29**
Normal close rate (%)	79.22	84.68	84.41	**85.99**
Sharpe ratio (daily-based)	3.5588	3.9784	3.9427	**4.0362**
Sharpe ratio (trade-based)	0.2947	0.3456	0.3404	**0.3543**
Profit per open (thousands)	0.1901	0.2294	0.2228	**0.2321**
MDD	-87.91	-75.95	**-72.23**	-78.58
Top 2869 (count≥30)				
Profit (thousands)	10924.98	13423.6	14002.16	**14957.67**
Total open number	51320	48417	53436	**57904**
Win rate (%)	78.96	82.64	82.48	**82.97**
Normal close rate (%)	77.74	83.23	83.94	**84.15**
Sharpe ratio (daily-based)	2.5454	2.798	2.7967	**2.9943**
Sharpe ratio (trade-based)	0.2659	**0.3169**	0.3088	0.3136
Profit per open (thousands)	0.2129	**0.2772**	0.262	0.2583
MDD	-233.15	**-83.28**	-282.75	-103.68
Top 4211 (count≥20)				
Profit (thousands)	15214.26	19618.35	19505.69	**20727.17**
Total open number	64062	**67767**	65520	65777
Win rate (%)	78.09	81.63	82.2	**82.99**
Normal close rate (%)	76.84	82.47	83.33	**84.5**
Sharpe ratio (daily-based)	2.0067	2.2313	2.2388	**2.3606**
Sharpe ratio (trade-based)	0.2455	0.282	0.2896	**0.301**
Profit per open (thousands)	0.2375	0.2895	0.2977	**0.3151**
MDD	**-320.66**	-368.01	-382.37	-345.7

defined in Fig. 3. The results are presented in Tables II and III. Our training period spans from January 2015 to October 2016, with November to December 2016 as the validation period, and January 2017 to December 2018 as the testing period. We rank stock pairs by the count of PTS-eligible trading days in the training period and trade the top N most frequent pairs using the DDQN outlined in Fig. 2. For instance, the top 29 pairs each have over 110 eligible trading days, while the top 63 pairs have at least 100. In our local learning experiments, we employ k-means and GMM to cluster stock pairs into three groups. We employ various financial indicators to assess profitability and risk, including total profit, total open numbers, win rate, normal close rate, daily-based and trade-based Sharpe ratios, average profit per open, and max drawdown (MDD). Total profit is the sum of profits and losses across all trading days in the testing period. Total open numbers represent the count of executed PTS trades. The Win rate is the proportion of profitable trades. The normal close rate indicates the frequency of trades where the spread process reverts to the mean level, as shown in Fig. 1. Sharpe ratios are calculated either daily or per trade, measuring risk-adjusted returns. The profit per open is the mean profit in thousand Taiwan dollars per trade. Lastly, MDD measures the maximum cumulative daily loss during the testing period. Together, these indicators provide a holistic view of our trading strategy's effectiveness and risk profile. The best-performing indicators are highlighted in red for easy comparison.

Individual learning, popular in past literature, shows superior results with frequently traded stock pairs (as evident in the Top 29 and Top 63 scenarios). However, as we expand our scope to include less frequent pairs, global and local learning methods start to outshine individual learning. This shift is attributed to the limited training data available for each pair, which often leads to overfitting problems in in-

dividual learning. Moreover, trading a wider range of stock pairs, including less frequent ones, contributes to increased profits, underscoring the potential profitability of these pairs. Notably, studies like [1]–[3], [7] overlooked this opportunity by ignoring less frequent pairs, potentially missing out on both trading opportunities and additional profits. To effectively trade these less frequent pairs without the constraints of insufficient training data, adopting global or local learning strategies presents a more viable solution.

We next compare the global learning approach, also prevalent in past studies like [4], [5], with k-means-based and GMM-based local learning methods proposed in this paper. The comparative results are presented in the last three columns of Tables II and III. Generally, GMM-based local learning surpasses the global learning approach, as it better captures the diverse characteristics of different stock pair clusters. Notably, GMM's superiority over k-means in clustering trading data is evident; it more accurately captures spread process characteristics. Since spread processes, based on the Johansen cointegration test [12], resemble complex linear combinations of Gaussian white noises (as noted in [28] and [29]), GMM's ability to classify stock pairs into statistically similar clusters enhances the DDQN's effectiveness. Furthermore, k-means, essentially a simplified version of GMM without covariance consideration, falls short in comparison, explaining GMM's significant performance edge.

V. Conclusion

Past PTS research predominantly relies on global or individual learning approaches, which either fail to capture the diverse characteristics of stock pairs or miss trading opportunities with infrequent pairs. In contrast, this paper introduces a method that clusters trading data based on the statistical properties of stock pair spreads. We then train individual DQNs with data from each specific cluster, effectively capturing the unique attributes of different stock pairs without overfitting. Our experimental results demonstrate that this approach, particularly the use of GMM for data division, effectively captures the statistical nuances of spread processes, viewed as complex Gaussian white noise combinations. This method surpasses existing learning approaches in PTS performance.

References

[1] Fallahpour, S., Hakimian, H., Taheri, K., & Ramezanifar, E. (2016). Pairs trading strategy optimization using the reinforcement learning method: a cointegration approach. Soft Computing, 20, 5051-5066.

[2] Kim, T., & Kim, H. Y. (2019). Optimizing the pairs-trading strategy using deep reinforcement learning with trading and stop-loss boundaries. Complexity, 2019, 1-20.

[3] Brim, A. (2020, January). Deep reinforcement learning pairs trading with a double deep Q-network. In 2020 10th Annual Computing and Communication Workshop and Conference (CCWC) (pp. 0222-0227). IEEE.

[4] Lu, J. Y., Lai, H. C., Shih, W. Y., Chen, Y. F., Huang, S. H., Chang, H. H., ... & Dai, T. S. (2022). Structural break-aware pairs trading strategy using deep reinforcement learning. The Journal of Supercomputing, 78(3), 3843-3882.

[5] Kuo, W. L., Chang, W. C., Dai, T. S., Chen, Y. P., & Chang, H. H. (2022). Improving Pairs Trading Strategies Using Two-Stage Deep Learning Methods and Analyses of Time (In) variant Inputs for Trading Performance. IEEE Access, 10, 97030-97046.

[6] Wang, C., Sandås, P., & Beling, P. (2021, May). Improving pairs trading strategies via reinforcement learning. In 2021 International Conference on Applied Artificial Intelligence (ICAPAI) (pp. 1-7). IEEE.

[7] Kim, S. H., Park, D. Y., & Lee, K. H. (2022). Hybrid deep reinforcement learning for pairs trading. Applied Sciences, 12(3), 944.

[8] Krauss, C. (2017). Statistical arbitrage pairs trading strategies: Review and outlook. Journal of Economic Surveys, 31(2), 513-545.

[9] Liu, R., Wang, F., Yang, B., & Qin, S. J. (2019). Multiscale kernel based residual convolutional neural network for motor fault diagnosis under nonstationary conditions. IEEE Transactions on Industrial Informatics, 16(6), 3797-3806.

[10] Yen, J., Wang, L., & Gillespie, C. W. (1998). Improving the interpretability of TSK fuzzy models by combining global learning and local learning. IEEE Transactions on fuzzy Systems, 6(4), 530-537.

[11] L. Bottou and V. Vapnik, "Local Learning Algorithms," in Neural Computation, vol. 4, no. 6, pp. 888-900, Nov. 1992, doi: 10.1162/neco.1992.4.6.888.

[12] Johansen, S. (1995). Likelihood-based inference in cointegrated vector autoregressive models. OUP Oxford.

[13] Jacobs, R. A., Jordan, M. I., Nowlan, S. J., & Hinton, G. E. (1991). Adaptive mixtures of local experts. Neural computation, 3(1), 79-87.

[14] Chang, F., Guo, C. Y., Lin, X. R., & Lu, C. J. (2010). Tree decomposition for large-scale SVM problems. The Journal of Machine Learning Research, 11, 2935-2972.

[15] Gu, Q., & Han, J. (2013, April). Clustered support vector machines. In Artificial intelligence and statistics (pp. 307-315). PMLR.

[16] Do, T. N., & Poulet, F. (2017). Parallel learning of local SVM algorithms for classifying large datasets. In Transactions on Large-Scale Data-and Knowledge-Centered Systems XXXI: Special Issue on Data and Security Engineering (pp. 67-93). Springer Berlin Heidelberg.

[17] Hatem, A. F., & Amir, F. A. (2021). Decision boundary clustering for efficient local SVM [J]. Applied Soft Computing Journal, 110(5).

[18] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... & Hassabis, D. (2015). Human-level control through deep reinforcement learning. nature, 518(7540), 529-533.

[19] Van Hasselt, H., Guez, A., & Silver, D. (2016, March). Deep reinforcement learning with double q-learning. In Proceedings of the AAAI conference on artificial intelligence (Vol. 30, No. 1).

[20] Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C., & Liu, C. (2018). A survey on deep transfer learning. In Artificial Neural Networks and Machine Learning–ICANN 2018: 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4-7, 2018, Proceedings, Part III 27 (pp. 270-279). Springer International Publishing.

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

[22] Chen, J., Luo, C., Pan, L., & Jia, Y. (2021). Trading strategy of structured mutual fund based on deep learning network. Expert Systems with Applications, 183, 115390.

[23] Chen, C. T., Chen, A. P., & Huang, S. H. (2018, July). Cloning strategies from trading records using agent-based reinforcement learning algorithm. In 2018 IEEE international conference on agents (ICA) (pp. 34-37). IEEE.

[24] Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine learning, 8, 229-256.

[25] Kameshwaran, K., & Malarvizhi, K. (2014). Survey on clustering techniques in data mining. International Journal of Computer Science and Information Technologies, 5(2), 2272-2276.

[26] MacQueen, J. (1967, June). Some methods for classification and analysis of multivariate observations. In Proceedings of the fifth Berkeley symposium on mathematical statistics and probability (Vol. 1, No. 14, pp. 281-297).

[27] Gupta, M. R., & Chen, Y. (2011). Theory and use of the EM algorithm. Foundations and Trends® in Signal Processing, 4(3), 223-296.

[28] Huck, N., & Afawubo, K. (2015). Pairs trading and selection methods: is cointegration superior?. Applied Economics, 47(6), 599-613.

[29] Rad, H., Low, R. K. Y., & Faff, R. (2016). The profitability of pairs trading strategies: distance, cointegration and copula methods. Quantitative Finance, 16(10), 1541-1558.

[30] Plappert, M. (2016). keras-rl. https://github.com/keras-rl/keras-rl.