國立政治大學資訊管理學系
碩士學位論文

適應性學習模型應用於銅價預測

An adaptive learning-based model for copper price forecasting

指導教授：林怡伶 博士、蔡瑞煌 博士

研究生：楊仁瀚 撰

中 華 民 國 一一一 年 一 月

DOI:10.6814/NCCU202200079

Acknowledgement

To my family,

To my advisors, Dr. Rua-Huan Tsaih & Dr. Yi-Ling Lin,

giving me guidance and helping me to complete my research,

To the thesis committee, Dr. Chih-Ping Wei,

To the community that is exploring new learning algorithms.

DOI:10.6814/NCCU202200079

摘要

銅在工業生產過程中扮演著不可或缺的工業原料之一，其價格變動的掌握對於相關的工業計劃與參與者來說至關重要。由於銅價的波動型態經常隨著時間推移而有所變化，往往會造成開發出的預測模型無法有效因應。為了因應銅價的變動特性，在本篇研究中除了檢驗出銅價具有結構性變化的特性並提出適應性學習型預測模型 (ALFM) 在動態變化的環境中學習。因爲結構性轉變在文獻中被證實與概念飄移在本質上存在著相近概念，所以本研究所提出之預測模型中除了加入移動窗口機制來因應銅價背後所存在的概念飄移與結構性轉變，並於自適應單隱藏層前饋神經網路 (ASLFN) 中設計序列型學習 (SS) 機制，以因應類神經網絡在學習具有複雜擬合函數資料時常面臨到梯度消失與擬合過度之問題。

由於 SS 機制是本研究中首次提出，因此其有效性有必要被加以驗證，我們使用長江有色金屬網的銅現貨價進行實驗。實驗結果除了驗證 ALFM 中 SS 機制是有效的之外，即 SS 機制當中的模組安排皆為必要，同時 SS 機制也被證實可以有效解決自適應單隱藏層前饋神經網路所遭遇梯度消失與擬合過度之問題。在所提出的預測模型中移動窗口機制與 SS 機制皆有助於提高預測能力，這使得所提出的 ALFM 比文獻中的其他工具有更好的預測結果，而且訓練時間是可以被接受的。最後，在與文獻中所使用的工具（如：SARIMA、SLFN、SVR、RNN、LSTM 以及 GRU）相比後，可以發現 ALFM 具有更好的預測結果。

關鍵字：自適應單隱藏層前饋神經網路、概念飄移、銅價預測、移動窗口、結構性變化

DOI:10.6814/NCCU202200079

Abstract

An accurate forecasting model for the price volatility of copper plays a vital role in decision-making for industrial projects and related companies. The challenge to deploy models is the change of the data over time, which commonly leads to significant mispredictions. In this paper, the structural change in copper prices has been examined. The adaptive learning-based forecasting model (ALFM) is proposed to learn the patterns under a dynamic changing environment, which combines the moving window mechanism and sequentially structuring (SS) mechanism. The moving window mechanism is used to address the concept drift and structural change behind the copper price. The sequentially structuring (SS) mechanism is designed for the adaptive single hidden layer feed-forward neural network (ASLFN) in response to solving the vanishing gradient and overfitting problems.

The SS mechanism is first proposed in this study and thus should be validated. We use the copper spot prices of Yangtze River (YR) nonferrous metals as application data. The experiment results provide evidence for examining the arrangement of SS mechanism does work in the training process. The proposed ideas of these modules within the SS mechanism can cope with the vanishing gradient or alleviate the overfitting tendency. Furthermore, both the moving window mechanism and SS mechanism in the proposed forecasting model help to improve the prediction ability, which makes the ALFM have better prediction results than other tools in the literature, and the training time is acceptable. The baseline models are seasonal ARIMA model (SARIMA), single-hidden layer feedforward neural network (SLFN), support vector regression (SVR), recurrent neural network (RNN), long short-term memory (LSTM), and gated recurrent unit (GRU).

Keywords: adaptive single-hidden layer feed-forward neural network, concept drift, copper price forecasting, moving window, structural change

DOI:10.6814/NCCU202200079

Table of Contents

Acknowledgement I

摘要 II

Abstract III

Chapter 1 Introduction 1

Chapter 2 Related works 4

2.1 Research on the copper price forecast 4

2.1.1 Traditional statistical methods 4

2.1.2 Learning-based approaches 5

2.1.3 Hybrid models 6

2.1.4 Predictor variables 9

2.2 Structural change and concept drift 11

2.2.1 Structural change of copper prices 11

2.2.2 Concept drift in structural change data 13

2.2.3 Concept drift handling 14

Chapter 3 The proposed adaptive learning-based forecasting model (ALFM) 17

3.1 The moving window mechanism 18

3.2 The sequentially structuring (SS) mechanism 19

Chapter 4 Experimental design 27

4.1 Data description 27

4.2 Structural change test of weekly copper prices of Yangtze River Market 29

4.3 Data preprocessing – the normalization arrangement 31

4.4 Validation and evaluation 31

Chapter 5 Experiment results 34

5.1 SS mechanism validation 34

5.1.1 The learning route 34

DOI:10.6814/NCCU202200079

5.1.2 Number of adopted hidden nodes 39
5.1.3 Training time 41
5.1.4 Validation for reorganizing module 43
5.1.5 Validation for LTS principle 45
5.1.6 Validation for regularizing module 46
5.2 ALFM evaluation 48
Chapter 6 Discussion and Conclusions 53
6.1 Summary 53
6.2 Theoretical and practical contributions 54
6.3 Limitations and future works 55
References 57
Appendix A - The learning process 72
Appendix B - The number of adopted hidden nodes 76
Appendix C - The training time 77
Appendix D - The number of hidden nodes deleted by reorganizing module 78

DOI:10.6814/NCCU202200079

Tables

Table 1. Recent literature and modeling approaches for copper price forecasting 7

Table 2. The potential predictor variables for copper price forecasting.......................................10

Table 3. List of notations...21

Table 4. The description of x-labels and y-label 28

Table 5. The empirical result of sup-F(*k*) and the double maximum tests 30

Table 6. The empirical result of the sequential test...30

Table 7. The four different forecasting learning models 32

Table 8. The learning process of MW-PO-100 version 35

Table 9. The learning process of MW-LTS-0 version 35

Table 10. The learning process of MW-LTS-100 version 35

Table 11. The learning process of MW-LTS-500 version 36

Table 12. The number of adopted hidden nodes 41

Table 13. The hidden node required for each phase..41

Table 14. The average training time...43

Table 15. The average number of irrelevant hidden nodes deleted by reorganizing module 44

Table 16. Forecasting performance of MW-PO-100, and MW-LTS-100.....................................45

Table 17. Forecasting performance of the version with LTS principle..47

Table 18. Out-of-sample performance comparison of the SLFN-based models...........................50

Table 19. Out-of-sample performance comparison of the models 51

DOI:10.6814/NCCU202200079

Figures

Figure 1. Types of concept drift 13
Figure 2. The system flowchart of the proposed ALFM 17
Figure 3. The moving window mechanism 18
Figure 4. The flowchart of the proposed SS mechanism 20
Figure 5. The ASLFN for the SS mechanism 20
Figure 6. The reorganizing module 23
Figure 7. The regularizing module 24
Figure 8. The matching module 24
Figure 9. The estimated three breaks in the copper spot price 31
Figure 10. The *familiar route* executions 37
Figure 11. The *thinking route* executions 37
Figure 12. The *cramming route* executions 38
Figure 13. Evolution of the number of hidden nodes used in the 8th window 39
Figure 14. The accumulated training time 42
Figure 15. The percentage of windows used reorganizing module 43
Figure 16. The training loss and testing loss of MW-PO-100, and MW-LTS-100 46
Figure 17. Out-of-sample performance of MW-PO-100, and MW-LTS-100 46
Figure 18. The training loss and testing loss of the version with LTS principle 48
Figure 19. Out-of-sample performance of MW-LTS-0, MW-LTS-100, and MW-LTS-500 48
Figure 20. Out-of-sample performance comparison of the SLFN-based models 51
Figure 21. Out-of-sample performance comparison of MW-LTS-500 with other models 52

DOI:10.6814/NCCU202200079

Chapter 1 Introduction

With the rapid development of society, non-ferrous minerals have become important raw materials for economic development (Gargano & Timmermann, 2014; Li & Li, 2015; Lasheras et al., 2015; Liu et al., 2020). Copper is one of the main non-ferrous metals playing a vital role in various industries (Dehghani & Bogdanovic, 2018). Due to its versatility and conductivity, more than 90% of industry projects need to rely on copper (Astudillo et al., 2020). The price of raw material accounts for 60% of the total profit of the mining companies (Sharma et al., 2015). The company's internal material preparation or procurement strategy is closely related to the cost of raw materials (He et al., 2016). Generally, the price volatility of the non-ferrous metals not only has a significant impact on the performance of the related companies (Lasheras et al., 2015; Astudillo et al., 2020), but affects various aspects in today's economies (Lasheras et al., 2015; Wang et al., 2019; Astudillo et al., 2020).

Over the past decades, the copper price prediction has attracted the interest of many researchers (Carrasco et al., 2019; Liu et al., 2020). Both academia and practitioners use various tools to grasp the changes in metal price to develop more effective strategies to respond to future changes. The copper price data is a kind of time series data stream with concept drift (Tsymbal, 2004) which is regarded as a similar concept to structural change in the econometrics and statistics literature (Koitsiwe & Adachi, 2017, 2018). The concept drift refers to the concept evolution that emerges when the novel classes appear in the data stream (Tsymbal, 2004). The structural change means the fundamental changes in the ways a market or economy functions (Matsuyama, 2008). Since the main drivers of copper price changes and their impact are significantly different over time, it causes copper prices to have different dynamic patterns in distinct periods (Koitsiwe & Adachi, 2017, 2018). Nowadays, the econometrics and statistics literature contain a large amount of important work related to the issue of structural changes in the copper market.

Due to the structural and conceptual change on the copper price, the developed model without handled correctly will encounter the significant mispredictions (Baier et al., 2020). Wang

DOI:10.6814/NCCU202200079

et al. (2003) argued that the concept drift problem owing to data expiration problem means the model built by the training dataset isn’t consistent with current concepts. The emerging concept evolution leads to the change of fitting function form embedded in the data stream. Many methods are used to split data to facilitate dynamic training of the model, like methods of instance weight (Klinkenberg, 2004), monitoring two different time window distribution (Gama & Kosina, 2014), detection concept change point (Kosina & Gama, 2015), and sliding window (Fornaciari & Grillenzoni, 2017). With the growth of learning-based methods, the sequence-based moving window (Babcock et al., 2001) is an efficient strategy to adapt the model to the changes in time series data (Kashani et al., 2012; Gama et al., 2014). As a result, this research proposed the forecasting model that is able to deal with gradual and abrupt changes in the market structure through the moving window mechanism.

From the research on the copper price forecast, artificial neural networks (ANN) is usually used to model the data with an embedded non-linear fitting function form (Shokry & Espuña, 2018). The single-hidden layer feedforward neural network (SLFN) is one of most used ANN (Wang et al., 2021). There are still some challenges to train NN-based models. One is the problem of vanishing gradient in backward propagation. Vanishing gradient means the gradient descent method used converges to the local optimum, the saddle point, or the plateau (Hu et al., 2021). The derivative or slope will get smaller in these cases, so the training process of neural network are nearly stop. In other words, the new weight values are very similar to the old weight values. The other is the problem of overfitting. Once the complexity of the model exceeds that needed for fitting the training data distribution, the deployed model will do much worse on the test set than on the training set (Guo et al., 2018).

Although many methodologies have been developed to forecast the copper price, these works mainly focus on optimizing and comparing existing algorithms. Often these modeling approaches proposed in the literature cannot simultaneously deal with the concept drift/structural change and non-linear fitting function form embedded in the copper price data stream (Tsymbal,

DOI:10.6814/NCCU202200079

2004; Suárez-Cetrulo et al., 2019). Thus, this study aims at developing the adaptive learning-based forecasting model (ALFM) for learning the patterns embedded in the data stream obtained from a structural change environment to form a forecasting model that is adaptive within the learning process. The proposed ALFM involved the sequentially structuring (SS) mechanism which extends the reasoning neural network (RN) of Tsaih (1998) and resistant learning (RL) of Tsaih & Cheng (2009) to cope with the vanishing gradient and alleviate the overfitting tendency. The ALFM results in an network structure of ASLFN determined by initially installing few hidden nodes and autonomously recruiting and deleting hidden nodes during the learning process.

In addition to providing the evidence that the structure of proposed SS mechanism is functional and the overall ALFM does lead to good performances in copper price forecasting, the research also provides statistical guarantees on the reliability of detected change and meaningful descriptions. To the best of our knowledge, this research represents the first attempt to simultaneously deal with the concept drift/structural change and nonlinearities in copper price forecasting. The proposed ALFM has been proven through a series of validation and evaluation, which also can assist market participants in making decisions in response to the volatility of the copper price.

The remainder of this paper is organized as follows: Chapter 2 makes the relevant literature review including the description of these modeling approaches and input factors selection for copper forecasting issues, and the explanation of structural change and concept drift. The proposed adaptive learning-based forecasting model (ALFM) is presented in Chapter 3. Chapter 4 conducts the experimental design for the AI research issues in fundamentals and applications. Chapter 5 shows the experiment results of validation and evaluation. Finally, the conclusion remarks of this study are summarized in the last chapter.

DOI:10.6814/NCCU202200079

Chapter 2 Related works

2.1 Research on the copper price forecast

The methods for forecasting copper prices are presented in this section and grouped into three types: traditional statistical methods, learning-based approaches, and hybrid models (see Table 1). The predictor variables for copper price are summarized at the end of this section.

2.1.1 Traditional statistical methods

Several statistical techniques are used to conduct the time series forecasting such as the auto-regressive (AR), moving average (MA), auto-regressive moving average (ARMA), auto-regressive integrated moving average (ARIMA), and seasonal ARIMA model (SARIMA). These models are commonly used due to the ease of implementation and explainable for the output result (Angus et al., 2012). The ARIMA model assumed that past values of the series plus previous error terms contain information for the purposes of forecasting. It has demonstrated its superiority in terms of accuracy and precision in predicting the next lag of the time series (Karakoyun & Cibikdiken, 2018). Most these statistical models solely rely on past values and assume that the data is stable (Stevenson, 2007; Schaffer et al., 2021) without considering varied structures and nonlinear patterns of the underlying data (Çinar, 1995; Wang et al., 2019).

To respond to the change issue, some researchers attempted to explain nonlinearities in the conditional mean and some tried to interpret nonlinearities in the conditional variance. The classic non-linear models in conditional mean like threshold autoregression (TAR) model (Tong, 1977), and the exponential autoregressive (ExpAR) model (Ozaki, 1980). A numerous extended versions of the TAR model have been derived to specific situations such as the self-exciting threshold AR (SETAR) model (Tong & Lim, 2009), the smooth transition autoregression (STAR) model (Chan & Tong, 1986). On the other hand, the most characteristic examples of nonlinear models in conditional variance are the autoregressive conditional heteroskedasticity (ARCH) (Engle, 1982) and the generalized autoregressive conditional heteroskedasticity models (GARCH) (Bollerslev,

DOI:10.6814/NCCU202200079

1986). There are some time series exhibiting asymmetries which could be better explained by models that have both a nonlinear conditional mean and a changing conditional variance (Gharleghi et al., 2014; Li & Li, 2015; García & Kristjanpoller, 2019). However, these models are still limited by many variations in the model, which makes it unable to respond to the high volatility data (Abbasimehr et al., 2020; Hu et al., 2020).

2.1.2 Learning-based approaches

With the recent advancement in hardware computation and software enhancements, the artificial intelligence (AI) are introduced for dealing with non-linearity and complexity changes in the data (Wang et al., 2019). Often AI is further subdivided into machine learning (ML) and deep learning (DL), in which the mainly difference is the steps for feature processing. Feature processing is the process of applying domain knowledge to feature extractors to reduce the complexity of the data and generate patterns that make learning algorithms work better. The process is time-consuming and requires specialized knowledge. In ML, most of the characteristics of an application must be determined by experts. The performance of most ML algorithms depends upon the accuracy of the features extracted. Trying to obtain high-level features directly from data is a major difference with DL (Deng & Yu, 2014). Thus, DL reduces the effort of designing a feature extractor for each problem.

In terms of ML techniques, the methods of support vector machine (SVM) and support vector regression (SVR) were widely used in many disciplines due to its remarkable generalization performance (Cao & Tay, 2003; Astudillo et al., 2020). For instance, some financial series forecasting was applied and get the promising result like stock market movement (Kim, 2003; Bao et al., 2004; Huang et al., 2005; Hao & Yu, 2006), and copper price volatility (Parveen et al., 2017; Carrasco et al., 2018; Astudillo et al., 2020). The main feature of theses method is that they solve the problems of high dimensionality and overfitting to a certain degree (Astudillo et al., 2020). Although it has achieved remarkable results in solving the problem of small samples, which are not

DOI:10.6814/NCCU202200079

efficient enough to handle large-scale time series data or do not perform well when the data set has more noise (Nalepa & Kawulok, 2019).

Advanced algorithms derived from the artificial neural networks (ANNs) and recurrent neural networks (RNNs) have gained lots of attentions recently with their applications in many non-linear forecasting problems (Parisi et al., 2008; Khashei & Bijari, 2011). Moreover, some variants of the basic RNN like long short-term memory (LSTM), bidirectional LSTM (BLSTM), gated recurrent unit (GRU) can also stably show better forecasting ability. The major advantage of these neural network (NN)-based models is that they can easily model complex or multi complexes task (Nikzad et al., 2012; Lazli & Boukadoum, 2013), and capture the nonlinear characteristics of time series (Hochreiter & Schmidhuber, 1997; Yu et al., 2005). From the several empirical results, it can be found that the NN-based models have a higher performance compared with basic machine learning techniques and traditional time series models in financial forecasting (Lasheras et al., 2015; Karakoyun & Cibikdiken, 2018; Liao et al., 2020).

2.1.3 Hybrid models

There are many studies on the integration of linear and nonlinear models such as the hybridization of ANNs and the ARIMA model (Zhang, 2003; Khashei & Bijari, 2011; Lasheras et al., 2015). ANNs have been also integrated with the GARCH model (Kristjanpoller & Minutolo, 2015, 2016; Herna´ndez et al., 2017). Several studies combined multiple ANNs and the extended models such as LSTM and GRU to form a deeper network structure (Niu et al., 2020; Hu et al., 2020).

Although hybrid models often give better results than a single model, they still cannot solve the fundamental problem of deep neural network architectures (Sharma et al., 2020). The performance of these models are sensitive to the settings of these hyperparameters (Domhan et al., 2015; Diez-Sierra & del Jesus, 2020). Since there is no specific rule for determining the structure of artificial neural networks, the neural network usually suffers with many limitations such as

DOI:10.6814/NCCU202200079

Table 1. Recent literature and modeling approaches for copper price forecasting

Author(year)	Method	X-Label(s)	Finding
Li & Li (2015)	GARCH, EGARCH,GJR-GARCH, NAGARCH and FIGARCH	The daily price of the three-month copper futures contract of LME	The model averaging methods significantly reduce the forecasting uncertainty. The OLS time-varying weighted model averaging method obtain the best result.
Lasheras et al. (2015)	ARIMA, MLP[a], Elman	The copper prices	Neural network models outperform the ARIMA model.
Liu et al. (2017)	Decision tree	The prices of gold, silver, copper, crude oil, natural gas, lean hogs, and coffee, and the Dow Jones indices	The proposed decision tree model is accurate and reliable in both short (days) and long terms (years).
Carrasco et al. (2018)	SVM	The copper's price at the closure of the London Metal Exchange	The construction model allows the prediction with a margin of error in a period of around of 2 weeks. These data can further be updated with sliding-windows to maintain the accuracy of the prediction.
Dehghani (2018)	GEP Benchmark: time series functions, and multivariate regression methods	Silver price, nickel price, aluminum price, OPEC crude oil price, WTI crude oil price, BRENT crude oil price, and CLP/USD	The correlation score of these selected input variables was at least 80% by utilizing Pearson cross-correlation coefficients. The GEP could forecast metal prices (i.e., copper price) with higher reliability than the classic estimation methods.
Astudillo et al. (2020)	SVR	The closing price of daily copper price	The SVR model can be done by three actual values as predictors regardless of the number of days of the prediction.
Khoshalan et al. (2021)	GEP, ANN, ANFIS[b], and ANFIS-ACO[c]	The price of coal, aluminum, crude oil, gold, iron ore, natural gas, nickel, and silver	Artificial intelligence-based networks are efficient and effective in forecasting copper prices.

DOI:10.6814/NCCU202200079

Zhang et al. (2021)	MLP, NN, KNN[d], SVM, GBT[e], and RF[f]	The price of oil, gold, silver, iron ore, natural gas, USD/CLP, USD/CNY, USD/PEN, USD/AUD, and historical copper prices	MLP neural network can forecast monthly copper price with the highest accuracy. The four exchange rates are highly correlated with the monthly copper prices.
Dehghani & Bogdanovic (2018)	BA-BMMR[g]	The copper price historical datasets	The bat algorithm can effectively improve the classical time series function in copper price forecasting.
García & Kristjanpoller (2019)	A set of time series models, AI models, the hybrid model of time series and AI models	Copper price, Inventories, China's IPI, and Major Currency Index	Adaptive capacity helped to identify the best explanatory variables, window sizes, and model configuration parameters, as well as improve volatility forecast accuracy.
Alameer et al. (2019)	GA-ANFIS[h] Benchmark: ANFIS, SVM, GARCH, ARIMA	The exchange rates of the CLP, PEN, and RMB; the inflation rates of US and China; and prices of gold, silver, iron, and oil and past copper prices	The selected predictor variables power to forecast the monthly volatility of copper prices. The hybrid GA–ANFIS model outperforms other models.
Hu et al. (2020)	ARCH-LSTM-ANN, and GARCH-BLSTM-ANN Benchmark: ANN, LSTM-ANN, BLSTM-ANN, GARCH-ANN,	Copper price; Aluminum price; Zinc price; Gold price; CNY/US; CNY/EUR; CNY/YEN; DJIA index; FTSE 100 index; CSI300 index; crude oil price; Shanghai Interbank Offered Rate	1. The GARCH forecasts can serve as informative features to boost the volatility prediction. 2. Incorporating RNNs (LSTM and BLSTM) into the hybrid GARCH-ANN network can further improve the volatility prediction performance.
Zhang et al. (2021)	PSO-ELM[i], and GA-ELM[j] Benchmark: ELM, and ANN	The price of crude oil, iron ore, gold, silver, natural gas prices, USD/CLP, USD/CNY, USD/PEN, and USD/AUD	The ELM neural network was significant improved by the GA and PSO algorithms in forecasting copper price with high accuracy and reliability.

[a]The multilayer perceptron (MLP); [b]The adaptive neuro-fuzzy inference system (ANFIS); [c]The hybrid model of ANFIS and ant colony optimization algorithm (ACO); [d]The k-nearest neighbors neural network (KNN); [e]The gradient boosting tree algorithm (GBT); [f]The random forest algorithm (RF); [g]The hybrid model of bat algorithm (BA) and Brownian motion with mean reversion (BMMR); [h]The hybrid model of genetic algorithm (GA) and ANFIS; [i]The hybrid model of particle swarm optimization (PSO) and the extreme learning machine (ELM); [j]The hybrid model of GA and the extreme learning machine (ELM)

DOI:10.6814/NCCU202200079

vanishing gradient, overfitting, and computational time (Ooyen & Nienhuis, 1992; Schalkoff, 2007). In response to the definition of an appropriate network structure is not yet fully defined (Tsai et al., 2019), there are several such learning algorithms that can determine an appropriate network architecture automatically like the tiling algorithm (Mezard & Nadal, 1989), the cascade-correlation (CC) algorithm (Fahlman & Lebiere, 1990), the upstart algorithm (Frean, 1990), the softening learning procedure (Tsaih, 1993), the W&S algorithm (Watanabe & Shimizu, 1993), the CTN algorithm (Chen et al., 1994), the reasoning neural networks (Tsaih, 1998), and CSI learning algorithm (Tsai et al., 2019). This study also proposes the sequentially structuring (SS) mechanism to determine an appropriate network architecture of the ASLFN with the real number input/output.

2.1.4 Predictor variables

To develop an accurate forecasting model of copper prices, previous studies have categorized the potential predictor variables into five groups (see Table 2). In the energy category, many studies have confirmed that crude oil prices are quite feasible as copper price forecasting (Behmiri et al., 2015; Buncic & Moretto, 2015; Liu et al., 2017; Alameer et al., 2019). Since oil is the most commonly used energy source in the mining process, it is often considered as the main factor affecting the price of metals. Furthermore, Liu et al. (2017) observed that when crude oil prices are rising, copper prices also have a significant upward effect.

The word economy also significantly influences the status of copper price. When the rising in market price suggests strong economic health and leads to promote industrial production, the required industrial raw materials is also increasing. Meanwhile, the oil price surge also increases the prices of metal via the effects of inflation (Cologni & Manera, 2008). Orlowski (2017) observed the strong effect of shocks in inflation rates on commodity prices like crude oil and copper. The inflation rates of the US and China are the most commonly used predictor variables since these countries are regarded as a proxy of world inflation (Alameer et al., 2019).

Due to the indivisible relationship between economic conditions and copper prices, good economic conditions will drive consumers' life needs. Coffee prices are considered to be the most

DOI:10.6814/NCCU202200079

significant variable affecting copper prices (Liu et al., 2017; Dehghani, 2018). Other cash crop prices such as cocoa, corn, and wheat cannot get a certain correlation with the copper price (Dehghani, 2018). Although a strong correlation between copper and coffee prices, it is hard to interpret the underlying reason (Alameer et al., 2019).

Table 2. The potential predictor variables for copper price forecasting

Groups	Predictor variables	Representative studies
Energy sources	Crude oil prices	Jiang and Adeli (2005), Behmiri et al. (2015), Liu et al. (2017), Alameer et al. (2019)
Macroeconomics	Inflation rates of US and China	Cologni and Manera (2008), Orlowski (2017), Alameer et al. (2019)
Cash crops	Coffee prices	Liu et al. (2017), Dehghani (2018)
Exchange rates	USD/CLP, USD/PEN, USD/CNY, USD/EURO	Chen (2010), Wets & Rios (2015), Ciner (2017), Zheng et al. (2017), Alameer et al. (2019), Wang and Wang (2019)
Related metal prices	The prices of gold, silver, nickel, aluminum, zinc, iron	Morales and Andreosso-O'Callaghan (2011), Liu et al. (2017), Dehghani (2018), Alameer et al. (2019)

The exchange rates of the major production and consumption countries have been found with high predictive power in predicting copper prices (Chen et al., 2010; Ciner, 2017). For example, Chile has copper mine production accounts for one-third of the world, so the trend of copper prices is often related to the Chilean exchange rate. Wets & Rios (2015) also use the exchange rate between the USD and CLP as an input variable for copper prices prediction, the experiment results show that the Chilean exchange rate has a good prediction effect. The six base industrial metals also have examined the causality relationship with the CNY/USD currency exchange rate (Wang & Wang, 2019). Alameer et al. (2019) pinpointed the main exporter country CLP and PEN whose exchange rate has a positive relationship with copper price. Moreover, the coefficient between copper price and China's exchange rate is slightly negative. This finding is consistent with the result of Zheng et al. (2017) who claimed that nonferrous metals have a positive

DOI:10.6814/NCCU202200079

relationship with exporting countries. Also, the inverse relationship exists between nonferrous metals and major importer countries.

In terms of the functionality and accessibility of copper, since copper is an alternative to precious metals such as gold and silver, the relationship between copper and other precious metals has also been confirmed as an important factor to predict the copper price (Morales & Andreosso-O'Callaghan, 2011; Buncic & Moretto, 2015; Liu et al., 2017). From the empirical results, Alameer et al. (2019) also confirmed that the copper prices has a significant correlation with both gold and silver prices, and the first pinpointed copper has a higher correlation with iron. Additionally, copper has a common movement with nickel, aluminum, zinc, and iron (Rossen, 2015). Dehghani (2018) also found that aluminum, nickel, and zinc have a sufficient degree of correlation with copper prices.

2.2 Structural change and concept drift

2.2.1 Structural change of copper prices

Structural stabilities are essential to the modeling of time series data (Hernes, 1976; Chu et al., 1996). If the estimated model does not keep up with the data generating process, it makes the prediction ineffective (Chu et al., 1996). Structural changes are widely presented in both the statistics and econometrics literature (Perron et al., 2020), which refers to the fundamental changes in the ways a market or economy functions or operates (Matsuyama, 2008). These changes can be caused by factors such as a major change in government policy (Chavas, 2001; Matsuyama, 2008), a war (Mussagy & BigramoAllaro, 2016), natural disasters (Passerini, 2000; Cró & Martins, 2017) or technological revolutions (Chavas, 2001; Krüger, 2008). With time-varying components in financial time series (Talih & Hengartner, 2005), forecasting for these data streams is usually subject to structural changes (Andreou & Ghysels, 2009). The structural change in the acquired data streams is also regarded as concept drift (Alippi et al., 2014), which considers the target concept to be learned from the data stream may change over time (Sarnovsky & Kolarik, 2021).

DOI:10.6814/NCCU202200079

These literatures on structural break detection began with the early works of Quandt (1958) and Chow (1960) who considered tests for structural change for a known single break date. These tests for structural instability require strictly exogenous regressors and a breakpoint specified in advance (Nielsen & Whitby, 2015), thus the test result is subject to estimation error (Hansen, 2001). Quandt (1960) extended the Chow test and proposed taking the largest Chow statistic over all possible break dates. After research offers the solution to the unknown break date (Muthuramu & Maheswari, 2019), more scholars take the upsurge of interest in extending procedures to the various models with an unknown change point (Bai & Perron, 1998). Andrews (1993) and Andrews & Ploberger (1994) provided a comprehensive analysis of the problem of testing for structural change endogenously.

Nevertheless, the literature addressing the issue of multiple structural changes is relatively scarce (Bai & Perron, 1998). Andrews et al. (1996) considered optimal tests in the linear model with known variance. Garcia & Perron (1996) studied the sup Wald test for two changes in a dynamic time series. Yao (1988), Yin (1988), and Yao & Au (1989) investigated the estimation of the number of mean shifts of variables sequence using the Bayesian information criterion. Liu et al. (1997) considered multiple changes in a linear model estimated by least squares and estimate the number of changes using a modified Schwarz' criterion. The considerable development of testing for unknown multiple structural changes can be retroactively traced to Bai and Perron (1998, 2003). The methods allow us for general forms of serial correlation and heteroscedasticity in the errors, lagged dependent variables, trending regressors, as well as different distributions for the mistakes and the regressors across segments. In essence, the methodology is mainly based on the ordinary least squares (OLS) to test for unknown multiple breakpoints, which permit the modeler to endogenously estimate structural breaks without knowing the timing of the breaks in advance. The methodology proposed by Bai and Perron (1998, 2003) is also used in this study.

DOI:10.6814/NCCU202200079

2.2.2 Concept drift in structural change data

The structural change in the econometrics and statistics literature is in essence the same as concept drift in fields of engineering and machine learning (Koitsiwe & Adachi, 2017, 2018). The term concept drift implies the concept changes in the underlying distribution of streaming data over time (Tsymbal, 2004). These changes cause the challenge of models cannot respond to the new data (Tsymbal, 2004). In statistics, a similar problem is the detection of structural changes in time series data (Zeileis et al., 2003; Verbesselt et al., 2010). Such a robust tool to understand the unknown structural breaks was proposed by Bai and Perron (1998, 2003).

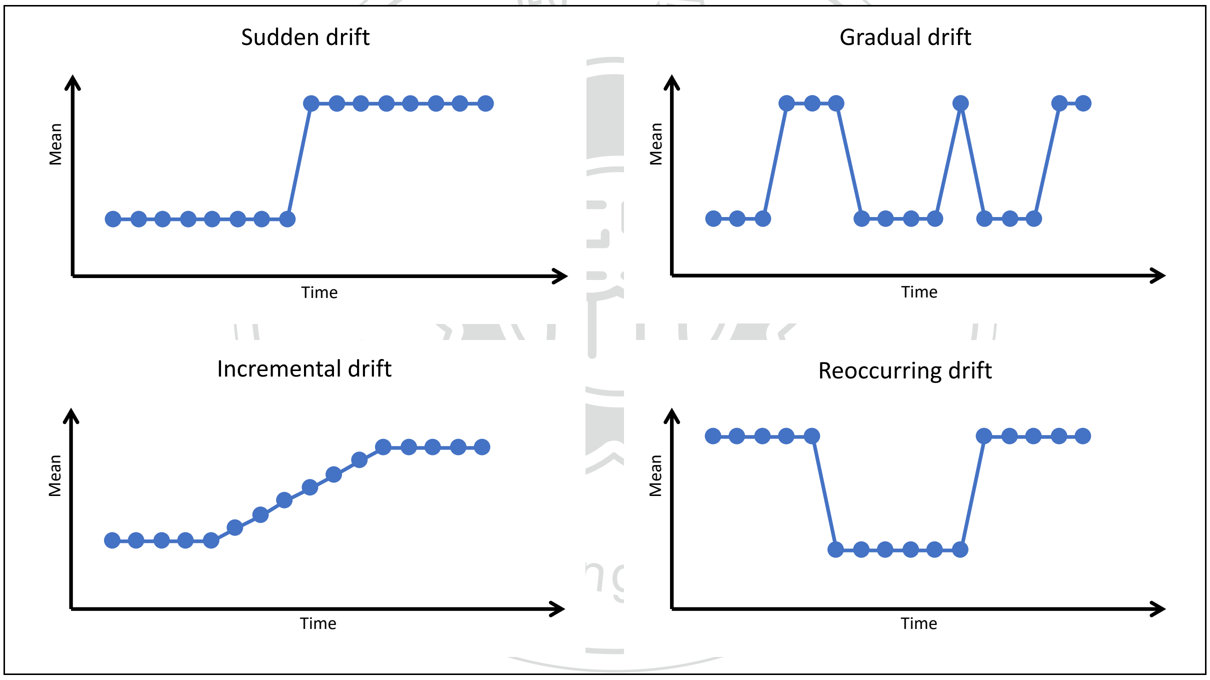


Figure 1. Types of concept drift

Since the concept of change takes different forms, Tsymbal (2004) classified these changes into four structural types: sudden drift, gradual drift, incremental drift, and reoccurring drift (Figure 1). A sudden drift can be seen as the concept of switching from one to another quickly. The occurrence of sudden drift is mainly attributed to the event of force majeure that causes an imbalance between supply and demand, as well as the market which is filled with the effects of a

DOI:10.6814/NCCU202200079

panic atmosphere that exacerbates the rapid change of prices to another type. The characteristics of gradual and incremental drift are slower and more gradual changes. In detail, gradual drift change will go back to the previous pattern for some time, but the drift change incrementally over time is referred to as incremental drift. A gradual drift occurs more often in financial markets whose supply and demand are highly sensitive to the news of the moment. As market operators take different actions from both positive and negative news, causing the price on the market shifts back and forth. The typical example of incremental drift is continuous rising energy and food prices that have fueled higher inflation. A recurring drift is when new concepts that were not seen before, or previously seen concepts may recur after some time. This type of case is often determined by seasonal patterns. The underlying theory behind concept drift is also considered as a structural change in the acquired data stream (Alippi et al., 2014).

2.2.3 Concept drift handling

Since concept drift is an important area that has gained the attention of the last years (Lu et al., 2020), strategies for updating existing learning models according to the drift have become noticeably (Lu et al., 2018). These most usual drift adaptation methods are the following (Farid et al., 2013): instance selection (window-based) approaches, weight-based approaches, and an ensemble of classifiers.

Instance selection approaches focus on selecting the appropriate prior dataset to train a model (Katakis et al., 2010). It adaptively selects instances into a fixed or dynamic sliding window (Farid et al., 2013). These approaches assume older examples are incompatible with new data classification, so it handle concept drift by forgetting some useless old instances (Katakis et al., 2010). Therefore, the last batch of information with the last training instances are employed to train the model. The appropriate window size is the key of these strategies, which is usually determined by the user. In general, the window size can be variable or fixed over time. Bifet & Gavaldà (2007) indicated the length of the window is appropriate for different case or situation. A short window is

DOI:10.6814/NCCU202200079

good in more accurately reflecting the current distribution and ensures fast adaptation in times with concept changes. A large window gives a better performance in stable periods.

A weight-based approach weights instances and deletes the outdated training ones based on their weights (Farid et al., 2013). Although these strategies consider all instances of learning, the degree of informativeness of old instances decreases as time passes. Within this framework, the mechanism should be capable of updating the weighted after each learning process. The ensemble algorithm combines several outputs from different learners to define a final classification (Farid et al., 2013). With a dynamic weighted majority vote phase, the ensemble learners are weighted according to their performance and deletes or creates new learners also based on the global ensemble performance. Outputs from learners are combined to classify instances with a weighted vote mechanism (Katakis et al., 2010).

Although instance selection approaches cannot assign corresponding weights to each instance, they can simply implement the adaptive learning and forgetting process without excessive consumption of resources. In contrast, weight-based approaches can weight instances by their entry time or competence concerning the current concept. However, it may happen that the concept of a particular period has not been sufficiently considered but only described by specific data. Klinkenberg (2004) demonstrated that instance weighting techniques handle concept drift worse than instance selection techniques, which is probably due to overfitting the data. Since the ensemble classifier mainly assembles multiple models to cope with concept drift, it takes a long time to run all the models in addition to training all of them. Another problem is the mechanism that discards or assigns lower weights to models in an ensemble whose global accuracy decreases on the current block of data, even if they are still good learners in the stable part of the data.

The instance selection approaches are also regarded as an efficient strategy to handle time-series with concept drift, (Giannella et al., 2003; Chang & Lee, 2005; Li et al., 2005; Leung & Khan, 2006; Mozafari et al., 2008; Wang et al., 2019). These data streams can be divided into batches comprising the training and testing blocks. Learners are continuously able to periodically

DOI:10.6814/NCCU202200079

learn and improve from these batches. Three types of moving windows are often used (Klinkenberg & Joachims, 2000), namely full-memory window, no-memory window, and moving window with fixed size *n*. The difference among these methods is how to update the instances within a window. In the full-memory window, the old instances cannot be forgotten and kept until the end. In the no-memory window, the most recent of the stream are considered merely. The mechanism behind the moving window with fixed size *n* adopts a first-in-first-out (FIFO) mechanism to discard the irrelevant old instances, keep and add new instances to keep the window with *n* instances. It extracts instances stepwise as equally important contributors to the current concept in a non-stationary environment.

DOI:10.6814/NCCU202200079

Chapter 3 The proposed adaptive learning-based forecasting model (ALFM)

For forecasting copper price under a dynamic changing environment, the study proposes that the ALFM consists of the following parts: the moving window arrangement, and the SS mechanism implemented through the adaptive single hidden layer feed-forward neural network (ASLFN). The ASLFN with a single output node corresponds to the copper price of four weeks later. Eventually, the inferencing model is the final acceptable ASLFN derived via a sequential learning process from the training block in each window. This section focuses on summarizing the proposed ALFM presented in Figure 2. Section 3.1 introduces the moving window mechanism. Then, Section 3.2 systematically explains the process of the SS mechanism.

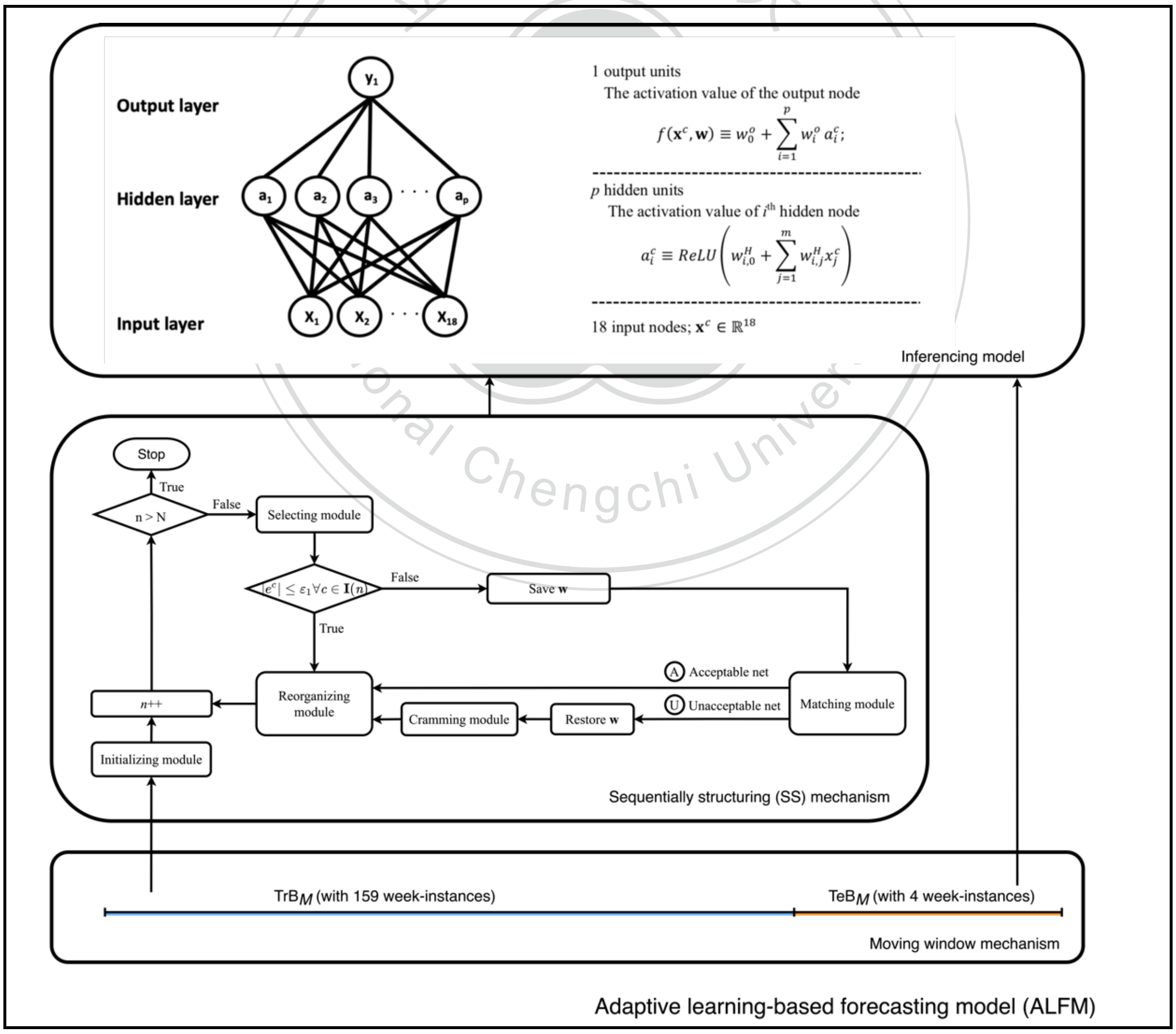
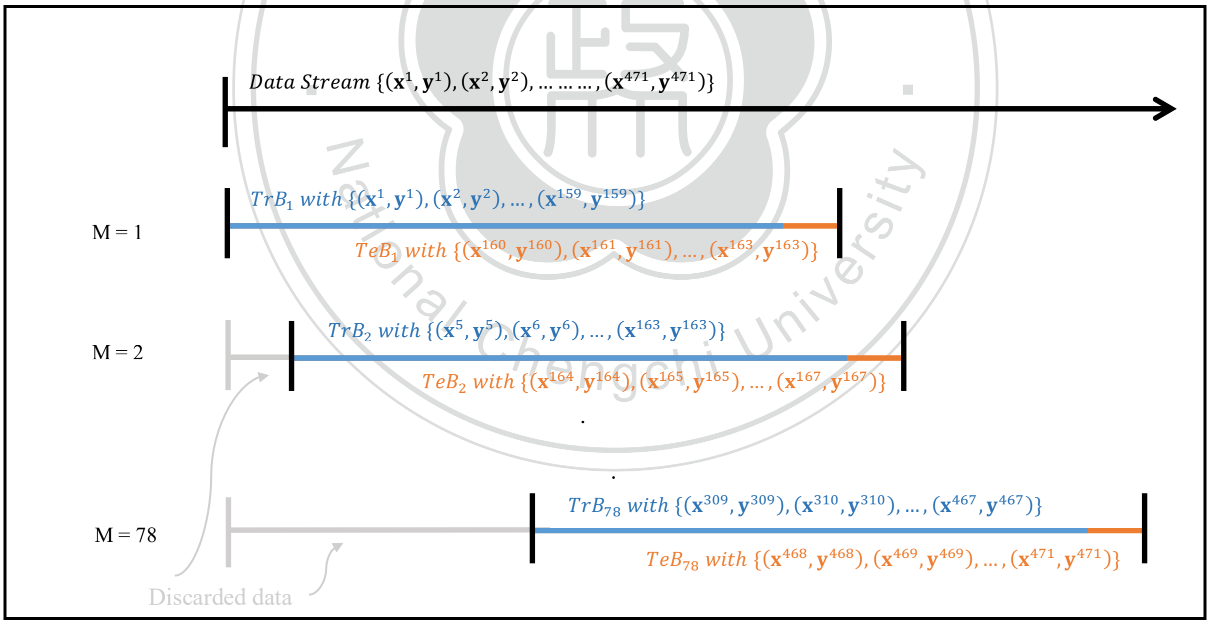

Output layer
Hidden layer
Input layer
y_1
a_1 a_2 a_3 a_p
X_1 X_2 X_{18}
1 output units
The activation value of the output node
$f(\mathbf{x}^c, \mathbf{w}) \equiv w_0^o + \sum_{i=1}^{p} w_i^o a_i^c;$
p hidden units
The activation value of i^{th} hidden node
$a_i^c \equiv ReLU\left(w_{i,0}^H + \sum_{j=1}^{m} w_{i,j}^H x_j^c\right)$
18 input nodes; $\mathbf{x}^c \in \mathbb{R}^{18}$
Inferencing model
Stop
True
n > N
False
Selecting module
$|e^c| \le \varepsilon_1 \forall c \in \mathbf{I}(n)$
False
Save w
True
Reorganizing module
A Acceptable net
U Unacceptable net
Matching module
n++
Cramming module
Restore w
Initializing module
Sequentially structuring (SS) mechanism
TrB$_M$ (with 159 week-instances)
TeB$_M$ (with 4 week-instances)
Moving window mechanism
Adaptive learning-based forecasting model (ALFM)

Figure 2. The system flowchart of the proposed ALFM

DOI:10.6814/NCCU202200079

3.1 The moving window mechanism

To take into the consideration that the concept drift may occur constantly or occasionally and to continually update the learner as new data arrive, the moving window mechanism is adopted not only to prevent data expiration (Wang et al., 2003) but maintain an effective predictive model (Du et al., 2011). The moving window concept of Figure 3 processes the data stream $\{(x^1, y^1), (x^2, y^2), \ldots\}$ into a serial of windows with the index M. In each window, the training block (TrB_M) consists of N = 159 week-instances (approximately 3 years) and the testing block (TeB_M) consists of B = 4 week-instances (approximately a month). For instance, TrB_1 consists of the 1st to 159th instances and TeB_1 consists of the 160th to 163th instances. When the window slides to the second one, TrB_2 has the 5th to 163th instances and TeB_2 has the 164th to 167th instances. There are 78 windows in total.



$Data\ Stream\ \{(\mathbf{x}^1,\mathbf{y}^1),(\mathbf{x}^2,\mathbf{y}^2),\ldots\ldots,(\mathbf{x}^{471},\mathbf{y}^{471})\}$
M = 1
$TrB_1\ with\ \{(\mathbf{x}^1,\mathbf{y}^1),(\mathbf{x}^2,\mathbf{y}^2),\ldots,(\mathbf{x}^{159},\mathbf{y}^{159})\}$
$TeB_1\ with\ \{(\mathbf{x}^{160},\mathbf{y}^{160}),(\mathbf{x}^{161},\mathbf{y}^{161}),\ldots,(\mathbf{x}^{163},\mathbf{y}^{163})\}$
M = 2
$TrB_2\ with\ \{(\mathbf{x}^5,\mathbf{y}^5),(\mathbf{x}^6,\mathbf{y}^6),\ldots,(\mathbf{x}^{163},\mathbf{y}^{163})\}$
$TeB_2\ with\ \{(\mathbf{x}^{164},\mathbf{y}^{164}),(\mathbf{x}^{165},\mathbf{y}^{165}),\ldots,(\mathbf{x}^{167},\mathbf{y}^{167})\}$
M = 78
$TrB_{78}\ with\ \{(\mathbf{x}^{309},\mathbf{y}^{309}),(\mathbf{x}^{310},\mathbf{y}^{310}),\ldots,(\mathbf{x}^{467},\mathbf{y}^{467})\}$
$TeB_{78}\ with\ \{(\mathbf{x}^{468},\mathbf{y}^{468}),(\mathbf{x}^{469},\mathbf{y}^{469}),\ldots,(\mathbf{x}^{471},\mathbf{y}^{471})\}$
Discarded data

Figure 3. The moving window mechanism

DOI:10.6814/NCCU202200079

3.2 The sequentially structuring (SS) mechanism

For forecasting copper prices under a dynamic changing environment, the study proposes the sequentially structuring (SS) mechanism presented in Figure 4 and implemented via the ASLFN shown in Figure 5 with the ReLU activation function (Hahnloser et al., 2000) and one output node whose value corresponds to the copper price of four weeks later. The SS mechanism ensures that all observation references can be learnt perfectly by ASLFN, i.e., the forecast error is satisfied by learning goal for all training data. During the training process, all training case in each window is learnt in sequence. There are three alternative routes within the sequentially structuring learning process: 1) *familiar route*: the last acceptable ASLFN is still an acceptable ASLFN, 2) *thinking route*: an acceptable ASLFN is obtained after using matching module, 3) *cramming route*: an acceptable ASLFN is obtained after using cramming module. The learning goal is to find **w** render $| e^c | \leq \varepsilon_1 \; \forall \; c \in \mathbf{I}(n)$, where the ε_1 value is set to be 0.24. Table 3 is the list of notations used in this study. As shown in Figure 4, the proposed SS mechanism contains the modules of initializing module, selecting module, matching module, cramming module, and reorganizing module to learn all instances in TrB_M in sequence. The details are as follows.

At first, the acceptable architecture is set up by the initializing module. It is easy to find an acceptable ASLFN with only 1 hidden unit and the associated **w** rendering the learning goal as true. The initial module directly picks up the first m+1 data $\{(\mathbf{x}^c, y^c): c \in \mathbf{I}(m + 1)\}$ that are linearly independent as the initial m+1 training data and then applying the linear regression mechanism to the data set $\{(\mathbf{x}^c, y^c - \min_{u\in\mathbf{I}(m+1)} y^u): c \in \mathbf{I}(m + 1)\}$, the initializing module of the 1st window results in a set of m+1 weights $\{w_j: j = 0, 1, \ldots, m\}$ to set up an initial ASLFN with merely one hidden node whose $w_{1,j}^H$ equals $w_j \; \forall \; j = 1, 2, \ldots, m$, $w_{1,0}^H$ equals w_0, w_1^o equals 1, and w_0^o equals $\min_{u\in\mathbf{I}(m+1)} y^u$. $\mathbf{I}(m+1)$ is the set of indices of these data. At the end of the initialization module, set $n = m + 1$.

DOI:10.6814/NCCU202200079

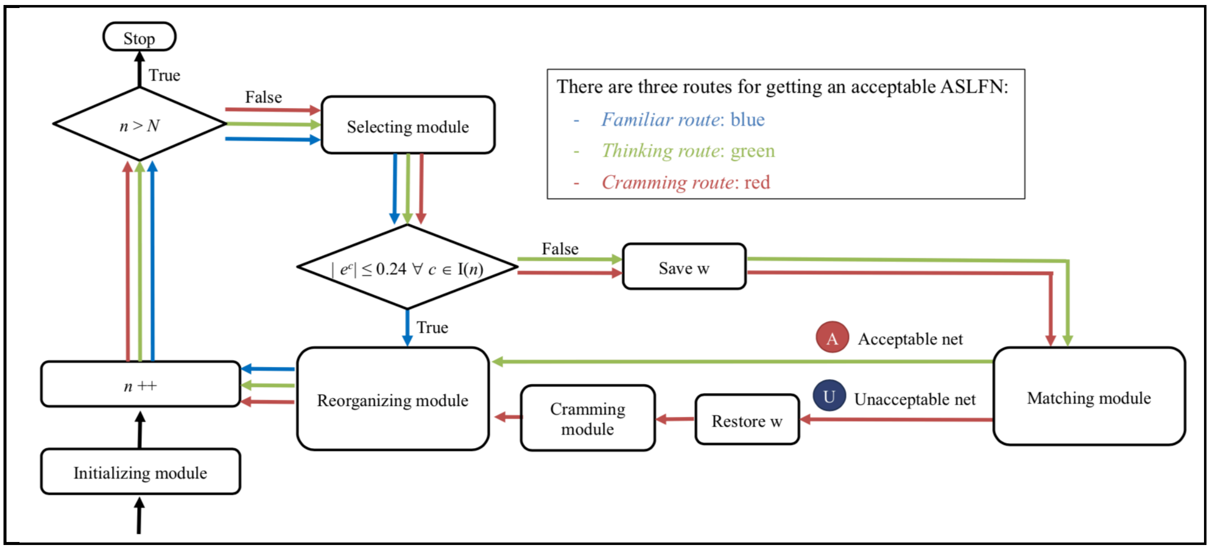
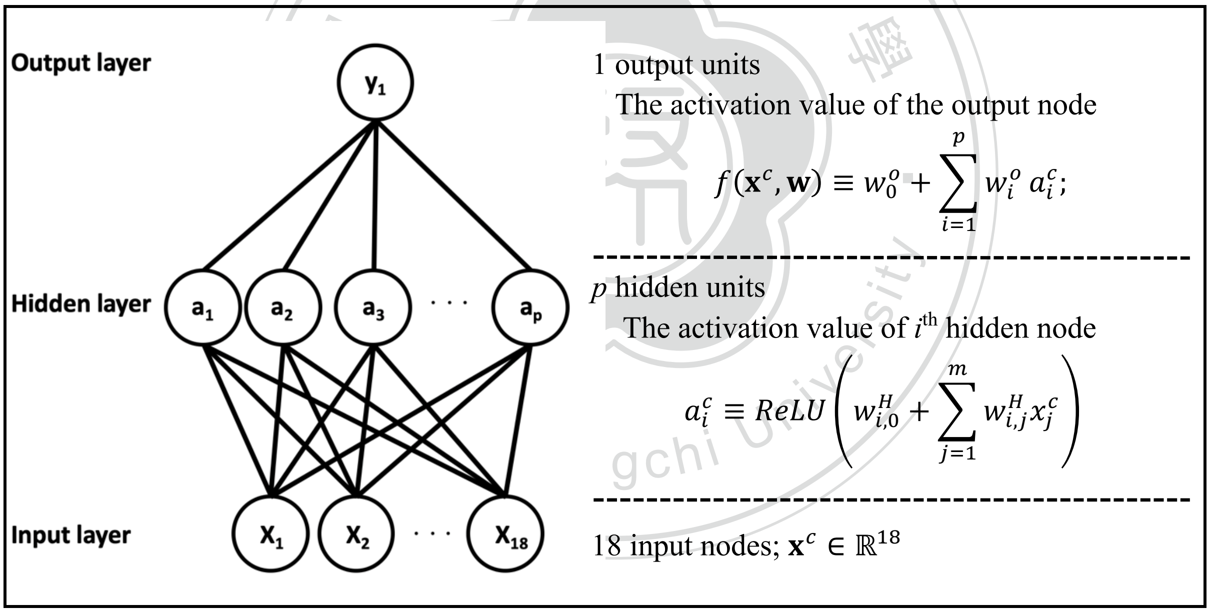


Figure 4. The flowchart of the proposed SS mechanism



1 output units

The activation value of the output node

$$f(\mathbf{x}^c, \mathbf{w}) \equiv w_0^o + \sum_{i=1}^{p} w_i^o\, a_i^c;$$

p hidden units

The activation value of ith hidden node

$$a_i^c \equiv ReLU\left(w_{i,0}^H + \sum_{j=1}^{m} w_{i,j}^H x_j^c\right)$$

18 input nodes; $\mathbf{x}^c \in \mathbb{R}^{18}$

Figure 5. The ASLFN for the SS mechanism

DOI:10.6814/NCCU202200079

Table 3. List of notations

$ReLU(x) \equiv \max(0, x)$;

D: the number of data, $D = 471$;

$\mathbf{x}^c \equiv (x_1^c,\ x_2^c, \dots,\ x_m^c)^{\mathrm{T}} \in \mathrm{R}^m$: the cth input; m=18;

p: the number of adopted hidden nodes; p is adaptable within the training stage;

$w_{i,0}^H$: the bias value of ith hidden node;

$w_{i,j}^H$: the weight between jth input node and ith hidden node, $j = 1, 2, \dots, m$;

$\mathbf{w}_i^H \equiv (w_{i,1}^H,\ w_{i,2}^H, \dots, w_{i,m}^H)^{\mathrm{T}}$, $i = 1, 2, \dots, p$;

$\mathbf{W}^H \equiv (\mathbf{w}_1^H, \mathbf{w}_2^H, \dots, \mathbf{w}_p^H)^{\mathrm{T}}$;

$\mathbf{w}_0^H \equiv (w_{1,0}^H,\ w_{2,0}^H, \dots, w_{p,0}^H)^{\mathrm{T}}$;

w_0^o: the bias value of output node;

w_i^o: the weight between the ith hidden node and the output node;

$\mathbf{w}^o \equiv (w_1^o,\ w_2^o, \dots, w_p^o)^{\mathrm{T}}$;

$\mathbf{w} \equiv \{\mathbf{W}^H, \mathbf{w}_0^H, \mathbf{w}^o, w_0^o\}$;

a_i^c: the activation value of ith hidden node corresponding to $\mathbf{x}^c$;

$\mathbf{a}^c \equiv (a_1^c, a_2^c, \dots, a_p^c)^{\mathrm{T}}$;

$f(\mathbf{x}^c, \mathbf{w})$: the output value of ASLFN corresponding to $\mathbf{x}^c$;

$y^c \in \mathrm{R}$: the target output value corresponding to $\mathbf{x}^c$;

e^c: the difference between $f(\mathbf{x}^c, \mathbf{w})$ and y^c;

$e^c \equiv f(\mathbf{x}^c, \mathbf{w}) - y^c$

Characters in bold represent column vectors, matrices or sets; the superscript H refers to quantities related to the hidden layer; the superscript o refers to the quantities related to the output layer; $(.)^{\mathrm{T}}$ denotes the transpose of (.).

DOI:10.6814/NCCU202200079

The n++ module indicates that the SS mechanism implements a sequential learning process. That is, the instances are picked one by one and, at the n^{th} stage, the SS mechanism learns merely n instances and $\mathbf{I}(n)$ is the set of indices of these data. The stop criterion of the SS mechanism is $n > N$ where $N = 159$ which is the number of entire instances in the training block within a window. At the n^{th} stage, an ASLFN is acceptable when its associated $\mathbf{w}$ makes the learning goal ($|e^c| \le \varepsilon_1$ $\forall$ $c \in \mathbf{I}(n)$) true. Because of the high correlation between the training data sets of consecutive windows, when $M \ge 2$, the ASLFN resulted from the former window can be used as the initial ASLFN of the latter window. For instance, the ASLFN resulted from the first window is used as the initial ASLFN of the second window.

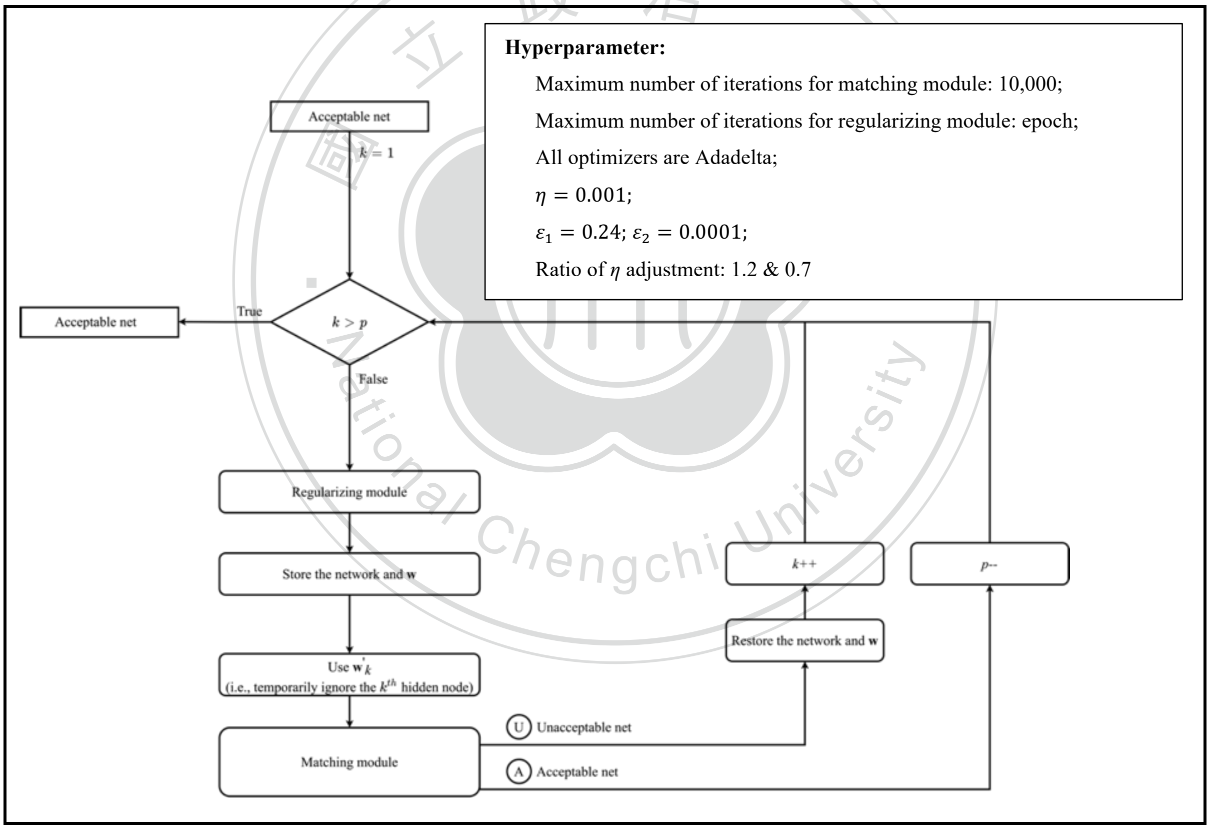
The selecting module which is implemented by the least trimmed squares (LTS) principle (Tsaih & Cheng, 2009) leads to the scenario of learning the easy first and grouping similar instances together. That is, at the n^{th} stage, the LTS principle firstly sorts all N instances of the current window based upon the squared residual $(e^c)^2$ values such that $(\delta^{[(M-1)B+1]})^2 \le (\delta^{[M-1)B+2]})^2 \le \ldots \le (\delta^{[M-1)B+N]})^2$. After that, select the first n instances with smallest squared residuals as the training samples. Let $\mathbf{I}(n)$ be the set of indices of these picked data.

If the learning goal is satisfied after the LTS principle, this means the current ASLFN is familiar to the knowledge we obtained so far and could interpret it. The subsequent training route will take the *familiar route*, which will further lighten the ASFLN structure to prevent the overfitting tendency by the reorganizing module in the case of fulfilling the learning goal. In ANN, overfitting can be attributed to big weights or too many hidden nodes. The process of reorganizing module as illustrated as Figure 6. At the n^{th} stage, the k^{th} hidden node is irrelevant if it is deleted and the learning goal can still be accomplished by merely applying the matching mechanism to $\min_{\mathbf{w}'_i} E_n(\mathbf{w}'_i)$, where $\mathbf{w}'_i \equiv \mathbf{w} - \{w^o_i, w^H_{i,0}, \mathbf{w}^H_i\}$. If an acceptable ASLFN is obtained through matching module, the k^{th} hidden unit is pruned and then goes back to the regularizing module to conduct the tailoring again. The regularizing module applies Adadelta (Zeiler, 2012) to $E_n(\mathbf{w})$ of Eq. (1) to adjust $\mathbf{w}$ until the maximum number of iterations are reached. The process should make

DOI:10.6814/NCCU202200079

sure to keep $|e^c| \leq \varepsilon_1\ \forall\ c \in \mathbf{I}(n)$ during the training process, otherwise the **w** will be restore and immediately end the regularizing step. The flowchart is illustrated in Figure 7. Since $E_n(\mathbf{w})$ has a regularization term, $\min_{\mathbf{w}} E_n(\mathbf{w})$ leads to a preference over small **w**. If an unacceptable ASLFN is obtained, the kth hidden unit is not irrelevant. Then restore the kth hidden unit, and save **w** continuously to check other hidden nodes sequentially. When all hidden nodes are examined, then we assume the current ASLFN has no irrelevant hidden node and go to n++ module.

$$E_n(\mathbf{w}) \equiv \frac{1}{n}\sum_{c=1}^{n}(e^c)^2 + \lambda\|\mathbf{w}\|^2 \tag{1}$$


Acceptable net
$k = 1$
$k > p$
True
Acceptable net
False
Regularizing module
Store the network and $\mathbf{w}$
Use $\mathbf{w}'_k$ (i.e., temporarily ignore the k^{th} hidden node)
Matching module
U Unacceptable net
A Acceptable net
Restore the network and $\mathbf{w}$
k++
p--

Hyperparameter:

Maximum number of iterations for matching module: 10,000;

Maximum number of iterations for regularizing module: epoch;

All optimizers are Adadelta;

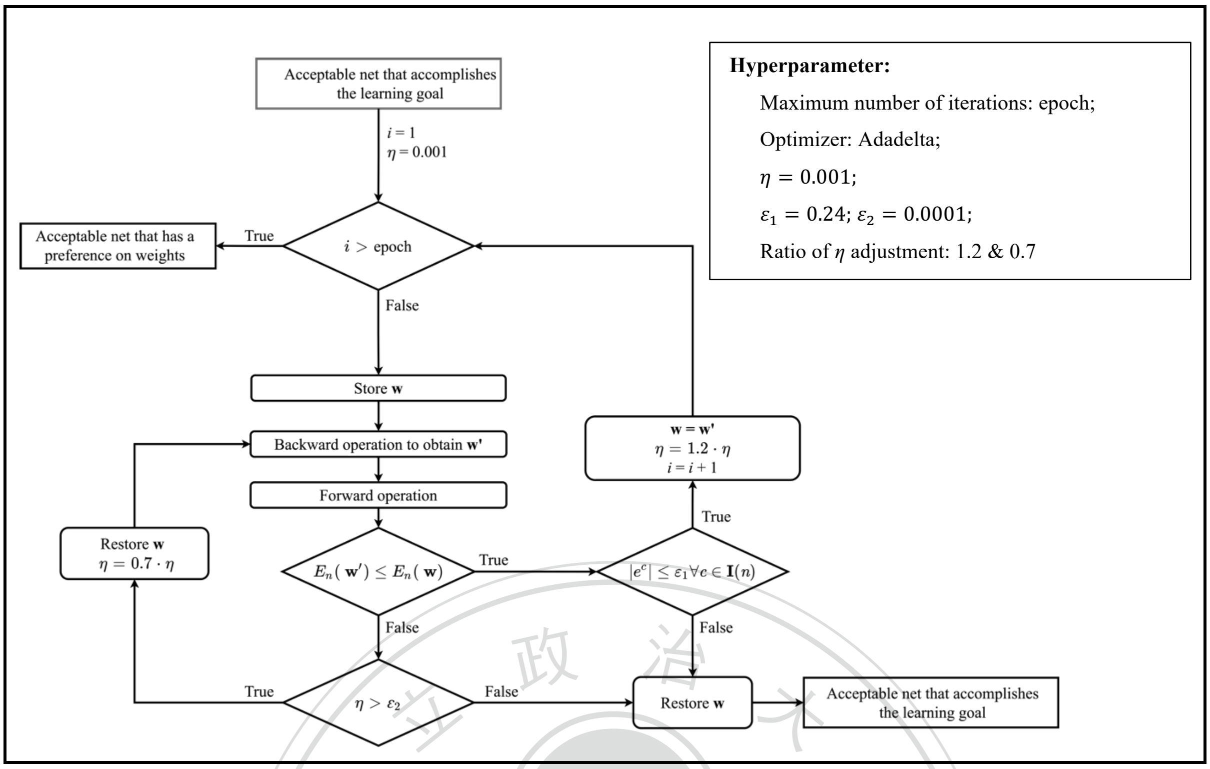
$\eta = 0.001$;

$\varepsilon_1 = 0.24$; $\varepsilon_2 = 0.0001$;

Ratio of η adjustment: 1.2 & 0.7

Figure 6. The reorganizing module

DOI:10.6814/NCCU202200079


Acceptable net that accomplishes the learning goal
$i = 1$
$\eta = 0.001$
$i > \text{epoch}$
True
Acceptable net that has a preference on weights
False
Store **w**
Backward operation to obtain **w'**
Forward operation
$E_n(\mathbf{w}') \leq E_n(\mathbf{w})$
True
$|e^c| \leq \varepsilon_1 \forall c \in \mathbf{I}(n)$
True
w = **w'**
$\eta = 1.2 \cdot \eta$
$i = i + 1$
False
Restore **w**
$\eta = 0.7 \cdot \eta$
False
$\eta > \varepsilon_2$
True
False
Restore **w**
Acceptable net that accomplishes the learning goal

Hyperparameter:

Maximum number of iterations: epoch;

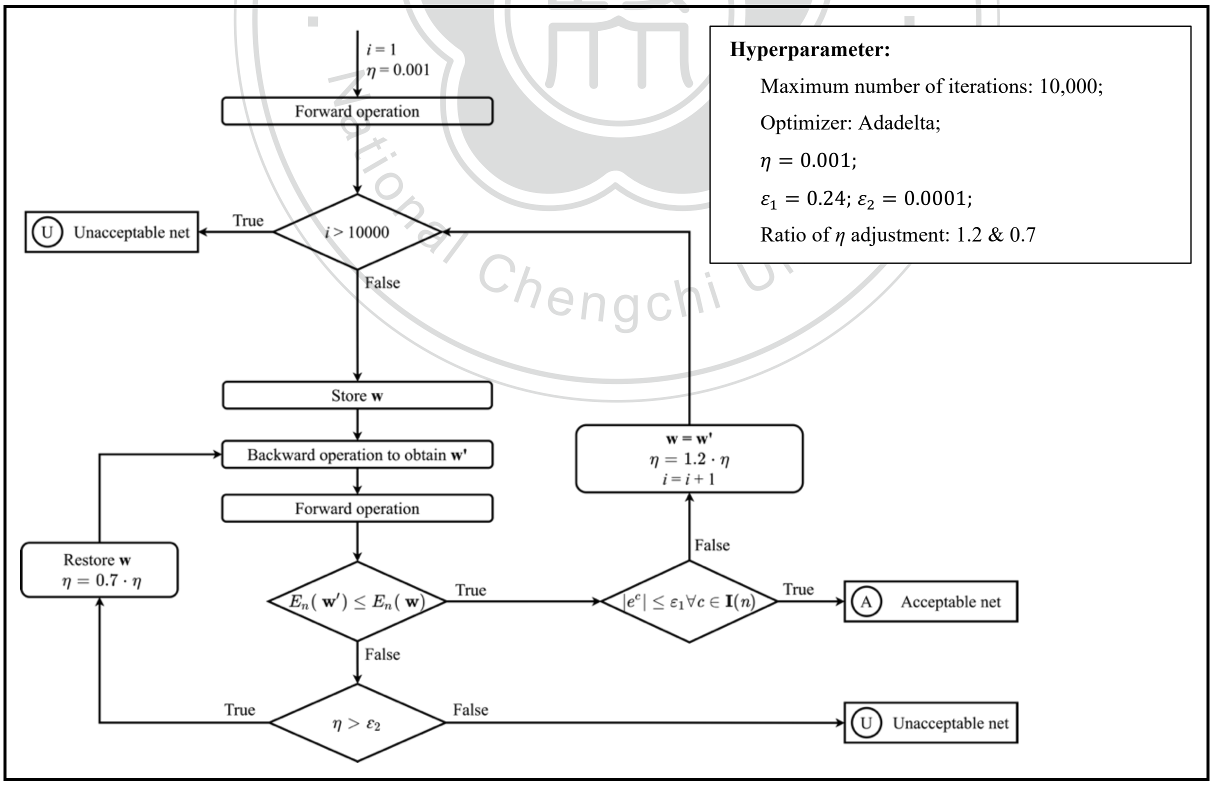
Optimizer: Adadelta;

$\eta = 0.001$;

$\varepsilon_1 = 0.24$; $\varepsilon_2 = 0.0001$;

Ratio of η adjustment: 1.2 & 0.7

Figure 7. The regularizing module



Hyperparameter:

Maximum number of iterations: 10,000;

Optimizer: Adadelta;

$\eta = 0.001$;

$\varepsilon_1 = 0.24$; $\varepsilon_2 = 0.0001$;

Ratio of η adjustment: 1.2 & 0.7

Figure 8. The matching module

DOI:10.6814/NCCU202200079

The matching module applies the generalized delta rule (GDR) to the loss function $E_n(\mathbf{w})$ defined in Eq. (2) to try to make the learning goal true. The workflow of matching module is shown in Figure 8. With the Pytorch framework, the GDR can be implemented by various optimizers such as Adadelta (Zeiler, 2012) in the study. To boost the efficiency of the model, the maximum number of training rounds was set to 10,000. With such a matching mechanism, it may lead to an acceptable ASLFN or an unacceptable ASLFN. The unacceptable ASLFN is attributed to an insufficient number of hidden nodes or the convergence to the local minimum or saddle point of $\mathrm{E}_n(\mathbf{w})$ (Tsaih, 1993).

$$E_n(\mathbf{w}) \equiv \frac{1}{n}\sum_{c=1}^{n}(e^c)^2 \tag{2}$$

When the learning goal is not satisfied after the selecting module, there is one and only one instance κ that is not at the right place and the instance κ is the $[n]^{\text{th}}$ instance. The matching module will first try to be applied to obtain an acceptable ASLFN. The storage module saves $\mathbf{w}$ before running the matching module, in order that the restore module can be used to return to a scene where only one instance not at the right place when the matching module still fails to adjust current ASLFN to acceptable structure. The subsequent route will be determined by the results of the matching module. If the result of the matching module is acceptable ASLFN, then the training style will be the *thinking route* in the n^{th} stage. Otherwise the *cramming route* will be taken.

If an unacceptable ASLFN is obtained through matching module, the restore module restores the saved $\mathbf{w}$ to get back the scenario that there is one and only one instance not at the right place. An acceptable ASLFN will be got by the *cramming route*, in which the cramming module is implemented to add three extra hidden nodes to the existing ASLFN. To begin with, the cramming module creates an m-vector $\boldsymbol{\gamma}$ of length one such that can meet the condition of $\boldsymbol{\gamma}^{\mathrm{T}}(\mathbf{x}^c - \mathbf{x}^\kappa) \neq 0 \ \forall\ c \in \mathbf{I}(n)\text{-}\{\kappa\}$, and then picks up a small number ζ complies the condition of $(\zeta + \boldsymbol{\gamma}^{\mathrm{T}}(\mathbf{x}^c - \mathbf{x}^\kappa))*(\zeta - \boldsymbol{\gamma}^{\mathrm{T}}(\mathbf{x}^c - \mathbf{x}^\kappa)) < 0 \ \forall\ c \in \mathbf{I}(n)\text{-}\{\kappa\}$. Furthermore, the cramming module lets $p + 3 \rightarrow p$, adds

DOI:10.6814/NCCU202200079

the new (p-2)th, (p-1)th, and pth hidden nodes to the existing ASLFN, and then assigns their associated weights as Eq. (3) - (5) to make the condition $|e^c| \leq \varepsilon_1 \; \forall \; c \in \mathbf{I}(n)$ true:

$$\mathbf{w}_{p-2}^{H} = \boldsymbol{\gamma},\ w_{p-2,0}^{H} = \zeta\text{-}\boldsymbol{\gamma}^{\mathrm{T}}\mathbf{x}^{\boldsymbol{\kappa}},\ w_{p-2}^{o} = \frac{y^{\kappa} - w_0^o - \sum_{i=1}^{p-3} w_i^o a_i^{\kappa}}{\zeta} \tag{3}$$

$$\mathbf{w}_{p-1}^{H} = \boldsymbol{\gamma},\ w_{p-1,0}^{H} = \text{-}\boldsymbol{\gamma}^{\mathrm{T}}\mathbf{x}^{\boldsymbol{\kappa}},\ w_{p-1}^{o} = \frac{-2(y^{\kappa} - w_0^o - \sum_{i=1}^{p-3} w_i^o a_i^{\kappa})}{\zeta} \tag{4}$$

$$\mathbf{w}_{p}^{H} = \boldsymbol{\gamma},\ w_{p,0}^{H} = \text{-}\zeta\text{-}\boldsymbol{\gamma}^{\mathrm{T}}\mathbf{x}^{\boldsymbol{\kappa}},\ w_{p}^{o} = \frac{y^{\kappa} - w_0^o - \sum_{i=1}^{p-3} w_i^o a_i^{\kappa}}{\zeta} \tag{5}$$

DOI:10.6814/NCCU202200079

Chapter 4 Experimental design

In the era of big data and artificial intelligence (AI), the new learning algorithms becomes more high dimensionality and nonlinearity. This makes mathematical proof unfeasible and must be validated and evaluated through a series of empirically experiment. With experiment results, this study examines the following AI fundamental research issues: 1) whether the corresponding learning process of SS mechanism does take the proposed three alternative routes; 2) whether the matching module and the cramming module help cope with the encountered vanishing gradient; 3) whether the reorganizing module help alleviate the overfitting tendency. This study further examines the following AI application research issue: whether the ALFM has a better forecasting accuracy than other tools in the literature and the total amount of training time is acceptable.

4.1 Data description

A total of 471 weekly copper prices of Yangtze River (YR) nonferrous metals from October 31, 2011 to December 21, 2020 are used. According to the literature review in Section 2.1.4, we identified 18 predictor variables. We did not consider cash crop as input variables, since there is no sufficient support for predicting copper prices. Compared to short-term forecasting, medium-term forecasting is the most practiced in many literatures but more complex and critical in many applications, such as raw material procurement planning and national budgeting (Astudillo et al., 2020). Thus, the four-week-ahead forecasting is adopted in this study. Table 4 shows the detailed description of x-labels and y-label used in this study, where there are 18 inputs and 1 output.

DOI:10.6814/NCCU202200079

Table 4. The description of x-labels and y-label

Input variable	Description
x_1^t	The weekly crude oil price of New York Mercantile Exchange at time epoch *t*.
x_2^t	The weekly copper spot price of YR nonferrous metals at time epoch *t*.
x_3^t	The weekly copper spot price of YR nonferrous metals at time epoch *t*-1.
x_4^t	The weekly copper spot price of YR nonferrous metals at time epoch *t*-2.
x_5^t	The weekly copper spot price of YR nonferrous metals at time epoch *t*-3.
x_6^t	The weekly copper spot price of London Metal Exchange at time epoch *t*.
x_7^t	The weekly gold spot price of FX Broker at time epoch *t*.
x_8^t	The weekly silver spot price of FX Broker at time epoch *t*.
x_9^t	The weekly nickel spot price of London Metal Exchange at time epoch *t*.
x_{10}^t	The weekly aluminum spot price of London Metal Exchange at time epoch *t*.
x_{11}^t	The weekly zinc spot price of London Metal Exchange at time epoch *t*.
x_{12}^t	The weekly iron spot price of London Metal Exchange at time epoch *t*.
x_{13}^t	Inflation rates of US at time epoch *t*.
x_{14}^t	Inflation rates of China at time epoch *t*.
x_{15}^t	The weekly USD/CLP dollar exchange rate at time epoch *t*.
x_{16}^t	The weekly USD/PEN dollar exchange rate at time epoch *t*.
x_{17}^t	The weekly USD/CNY dollar exchange rate at time epoch *t*.
x_{18}^t	The weekly USD/EURO dollar exchange rate at time epoch *t*.
Output variable	**Description**
y_1^t	The weekly copper spot price of YR nonferrous metals at time epoch *t*+4.

DOI:10.6814/NCCU202200079

4.2 Structural change test of weekly copper prices of Yangtze River Market

Bai and Perron (1998, 2003) proposed the methodology for the unknown multiple structural breaks test. The method can further determine these break dates for changes in the sequence that changes the inherent in the system concept. The maximum permitted number of breaks is at $M = 5$, and a trimming $\varepsilon = 0.15$ is used to determine the minimal number of observations in each segment. To impose the minimum structure on the data, we allow for different distributions of both the regressors and the errors in the different subsamples. The research first utilizes the sup-F(k) test, the double maximum tests like UDmax, and WDmax to confirm the structural changes. The result of the sup-F(k) tests are all significant for $k = 1, 2, ..., 5$. The double maximum tests UDmax and WDmax to test the null hypothesis of no structural break versus an unknown number of changes given the upper bound of five breaks indicate that the series presents at least one break in this structure. The results of these tests are given in Table 5.

Subsequently, the numbers of structural breaks as well as break dates can be determined using the sequential test $\sup F_T(l+1|l)$ or using other information criteria such as BIC and LWZ. From the result of the sequential test (Table 6), the null hypothesis cannot be rejected when $l = 3$, which suggests there are three breaks in the time series. In terms of information criteria, BIC selects 4 breaks and the LWZ selects 3 breaks. Since the heterogeneity of different structural stages is not considered in the procedure of BIC and LWZ testing, the results of the sequential test are preferred in this study. These estimated break dates also are determined at June 17, 2013, July 06, 2015, and November 16, 2016, by sequential test.

Testing for structural change has always been an important issue before conducting the financial time series analysis and forecasting. Since a myriad of political and economic factors can cause the relationships among economic variables to change over time, Bai and Perron (1998, 2003) proposed the estimation of multiple structural shifts in a linear model estimated based on the ordinary least squares. Figure 9 shows the copper price together with the breaks identified for all the periods of observations (October 31, 2011, to December 21, 2020). Through investigating the

DOI:10.6814/NCCU202200079

historical events and market situation around the breakpoints, we could find some relevant evidence economically. Since the impact of financial events on the commodity market is continuous, these breaks are illustrated by the events that occurred in the corresponding month.

Table 5. The empirical result of sup-F(*k*) and the double maximum tests

	F-statistic	Critical value
$Sup\ F_T(1)$	507.4440*	8.58
$Sup\ F_T(2)$	586.2324*	7.22
$Sup\ F_T(3)$	900.5784*	5.96
$Sup\ F_T(4)$	675.0136*	4.99
$Sup\ F_T(5)$	545.7206*	3.91
$UD\ max$	900.5784*	8.88
$WD\ max$	1296.470*	9.91

*denotes that the tests are significant at a 5% level

Table 6. The empirical result of the sequential test

	F-statistic	Critical value
$Sup\ F_T(1\|0)$	507.4440*	8.58
$Sup\ F_T(2\|1)$	185.1933*	10.13
$Sup\ F_T(3\|2)$	455.4091*	11.14
$Sup\ F_T(4\|3)$	10.49810	11.83

*denotes that the tests are significant at a 5% level

DOI:10.6814/NCCU202200079

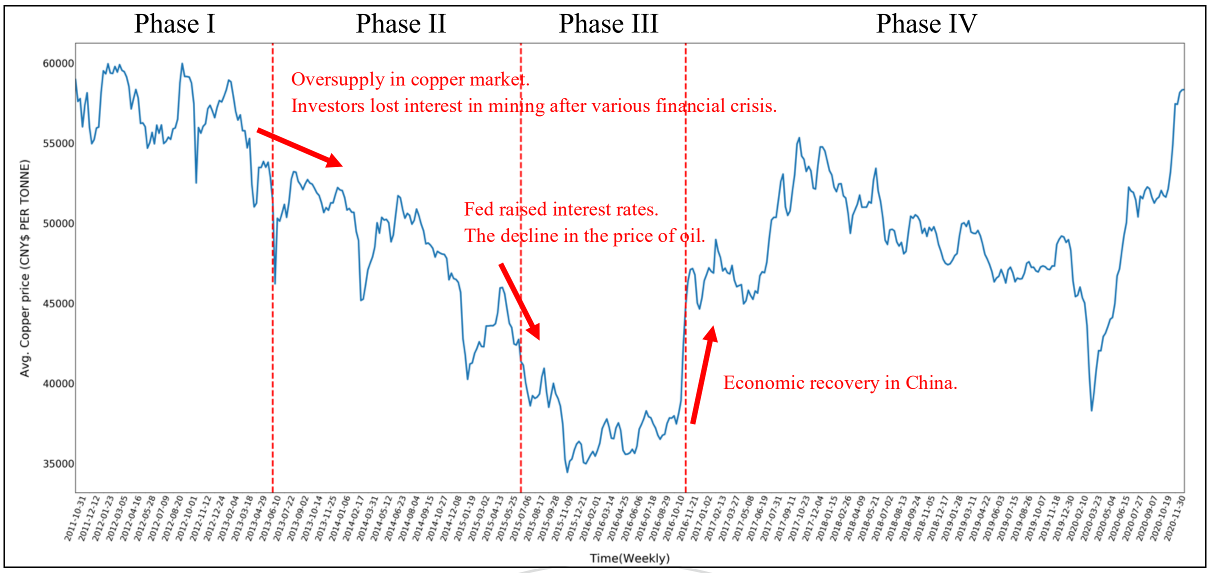


Figure 9. The estimated three breaks in the copper spot price

4.3 Data preprocessing – the normalization arrangement

All (training and test) data streams are normalized in the range [0, 1] by the min-max normalization method whose equation is presented as Eq. (6). To justify the efficiency of the proposed ALFM, the result generated from the inferencing phase is compared with the true value after inverse.

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}} \in [0,1] \tag{6}$$

4.4 Validation and evaluation

The sequentially structuring (SS) mechanism implemented through an ASLFN is proposed to cope with the complex non-linearity. This is the new proposed mechanism in this study and thus should be validated. To check whether the arrangement of SS mechanism does work in the training process, copper prices are used in the experiment. The module arrangement of SS mechanism needs to cope with the vanishing gradient of the matching module and alleviate the overfitting tendency of the SS mechanism. We build four different versions of the SS mechanism which are the

DOI:10.6814/NCCU202200079

difference in selecting modules and regularizing configuration shown as Table 7. For each version, all real-world data streams are split by moving window mechanism based on the rule of 159 training instances and 4 test instances in each window as detailed in Section 3.1. After that, there are 78 windows are used for training and testing. In this research, we implemented the selecting module by the LTS principle which ensures that the simple cases are trained first and adaptively to learn another similar case during the training process. We will check the LTS principle does display a good principle for learning to compare with pre-order (PO) principle which merely follows the original sequence to train all training data.

Additionally, the regularization module in the SS mechanism used to alleviate the overfitting tendency by adding the regularization term to the loss function presented as Eq. (1). The module will adjust **w** to $\min_{\mathbf{w}} E_n(\mathbf{w})$ at maximum number of iterations unless the adjusted ASLFN does not satisfy the learning goal ($|e^c| \le \varepsilon_1 \ \forall \ c \in \mathbf{I}(n)$). We set the maximum epoch for the regularizing module at 0, 100, and 500 to understand the effort of regularizing module when the number of training times the regularizing module is activated increases.

Table 7. The four different forecasting learning models

Version	The selecting module	Epoch for regularizing module
MW-PO-100	PO	100
MW-LTS-0	LTS	0
MW-LTS-100	LTS	100
MW-LTS-500	LTS	500

Except for the configuration of the selecting module and the regularizing module, for each version, the acceptable ASLFN will be obtained from one of the three alternative routes during the learning process. The occurrence percentages of these three routes are recorded to understand the learning styles and the effort of both the matching module and the cramming module for the

DOI:10.6814/NCCU202200079

proposed four versions. Furthermore, we can use how many irrelevant hidden nodes have been deleted during the training process to validate the reorganizing module.

There is some other information can be used to evaluate the efficiency of these four versions such as training time, the amount of adopted hidden nodes. These measurements can be seen as whether the version requires more time or resources to learn a new input/output relationship. These results correlate with the learning style results described above. When training requires a matching module or a cramming module to obtain an acceptable ASLFN, more training time and resources are incurred.

Finally, to check whether the proposed ALFM can have better forecast results in copper price forecasting. In the testing phase, the performance measurement such as mean absolute error (MAE), mean absolute percentage error (MAPE), and root mean square error (RMSE) is contained to check their performance of test data set. Finally, we want to confirm whether the best configuration among these versions has better accuracy of forecasting than other methods in the current literature. The five of the most efficient computational intelligence techniques (i.e., SLFN, SVM, RNN, LSTM, and GRU) and one of the classic volatility prediction models (i.e., SARIMA) are compared. These proposed four version will be developed with Pytorch (pytorch 1.5.1 + cuda 101) and GPU (GeForce GTX 1070 Ti) to accelerate the learning process on Ubuntu 20.04.1.

DOI:10.6814/NCCU202200079

Chapter 5 Experiment results

In this section, the results of validation and evaluation are discussed. We will validate that the module arrangement of the SS mechanism is necessary by comparing the proposed versions which are different in the selecting module and the regularizing module. In Section 5.1, these experiment results include the learning route, the number of adopted hidden nodes, training time, as well as the module validation for reorganizing module, LTS principle, and regularizing module. Then, in Section 5.2, the best configuration among these versions is evaluated by comparing with SLFN-based models and commonly used models in the relevant literature.

5.1 SS mechanism validation

5.1.1 The learning route

Regarding the new coming training sample, there are three ways for getting an acceptable SLFN. The frequency of these alternative routes being taken is used to not only understand the learning style of these four versions but also confirm whether the matching module and cramming module help these four versions to obtain acceptable ASLFN during the training process.

In most cases for each version, an acceptable ASLFN is obtained via the *familiar route* but it is still necessary to learn some data by the *thinking route* and the *cramming route*. For the MW-PO-100 version, there are approximately 88.14% of the training samples through *familiar route*, 3.60% of the training samples can be learned by *thinking route*, and the rest 8.26% of the training samples via *cramming route* (Table 8, see Appendix Table A1 for details). The circumstances of other versions with LTS principle are: the MW-LTS-0 version takes *familiar route* to learn about 84.41% of the training samples, 11.33% of the training samples could be learned by *thinking route*, and the remaining 4.26% of the training samples were learned by *cramming route* (Table 9, see Appendix Table A2 for details); the MW-LTS-100 version takes *familiar route* to learn about 94.56% of the training samples, 1.31% of the training samples could be learned by *thinking route*, and the remaining 4.13% of the training samples were learned by *cramming route* (Table 10, see

DOI:10.6814/NCCU202200079

Appendix Table A3 for details); the MW-LTS-500 version takes *familiar route* to learn about 97.46% of the training samples, 0.79% of the training samples could be learned by *thinking route*, and the remaining 1.75% of the training samples were learned by *cramming route* (Table 11, see Appendix Table A4 for details).

Table 8. The learning process of MW-PO-100 version

	Familiar route	*Thinking route*	*Cramming route*
Mean	88.14%	3.60%	8.26%
S.D.	12.51%	2.43%	12.13%
Min	30.82%	0.00%	0.00%
Max	100.00%	10.69%	64.78%

Table 9. The learning process of MW-LTS-0 version

	Familiar route	*Thinking route*	*Cramming route*
Mean	84.41%	11.33%	4.26%
S.D.	10.44%	9.33%	7.92%
Min	47.80%	0.63%	0.00%
Max	98.11%	35.85%	51.57%

Table 10. The learning process of MW-LTS-100 version

	Familiar route	*Thinking route*	*Cramming route*
Mean	94.56%	1.31%	4.13%
S.D.	9.54%	1.65%	9.47%
Min	47.80%	0.00%	0.00%
Max	100.00%	8.81%	51.57%

DOI:10.6814/NCCU202200079

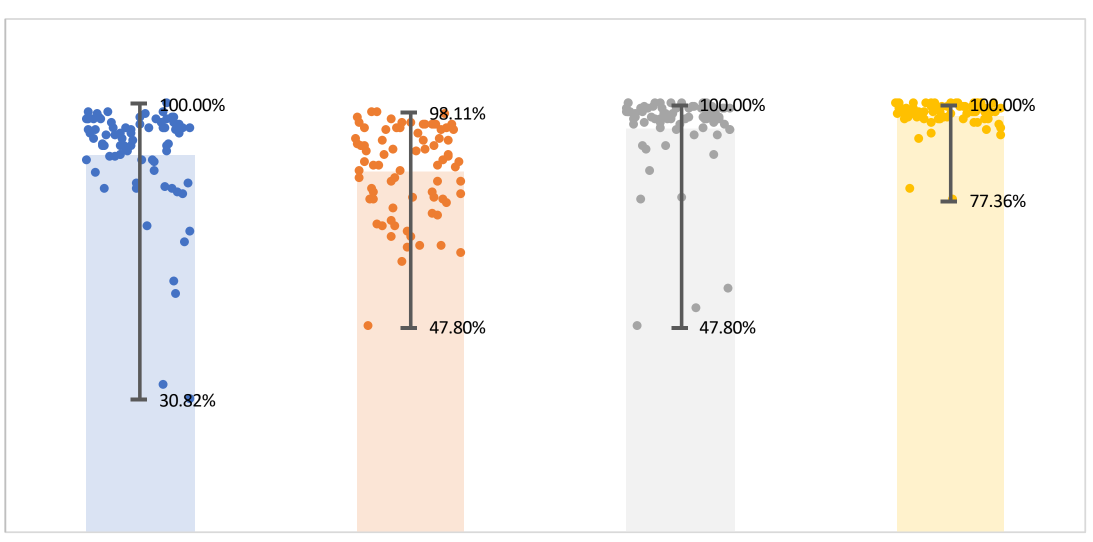
Table 11. The learning process of MW-LTS-500 version

	Familiar route	*Thinking route*	*Cramming route*
Mean	97.46%	0.79%	1.75%
S.D.	3.61%	1.10%	3.29%
Min	77.36%	0.00%	0.00%
Max	100.00%	5.03%	21.38%

As for the comparison between MW-LTS-100 and MW-PO-100, it is found that the version with LTS principle can increase from 88.14% to 94.56% of acceptable ASLFN driven from the *familiar route* (Figure 10). Accordingly, the MW-LTS-100 takes *thinking route* and *cramming route* less than the MW-PO-100. The occurrence of *thinking route* decreased from 3.60% to 1.31% (Figure 11), and the *cramming route* executions decreased from 8.26% to 4.13% (Figure 12).

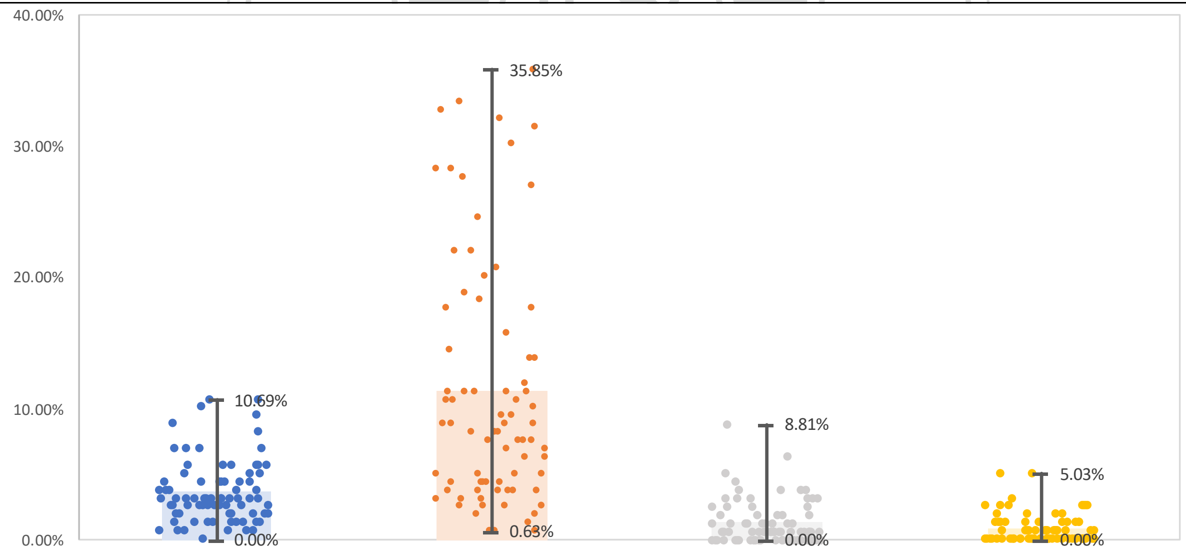
From the comparison among MW-LTS-0, MW-LTS-100, and MW-LTS-500, we also found that the regularizing module is activated more often, the percentage of acceptable ASLFN by the *familiar route* increases. Among these four versions, MW-LTS-500 is the most frequent via the *familiar route* to obtain an acceptable ASLFN structure. The *familiar route* occurrence percentage of MW-LTS-500, MW-LTS-100, and MW-LTS-0 are 97.46%, 94.56%, and 84.41%, respectively (Figure 10). Correspondingly, the regularizing module is activated more often, the percentage of the *thinking route* and the *cramming route* decreases. The *thinking route* occurrence percentage of MW-LTS-0, MW-LTS-100, and MW-LTS-500 are 11.33%, 1.31%, and 0.79%, respectively (Figure 11); the *cramming route* occurrence percentage of MW-LTS-0, MW-LTS-100, and MW-LTS-500 are 4.26%, 4.13%, and 1.75%, respectively (Figure 12). These results demonstrate how the adoption of the LTS principle and regularizing module impacts the learning style, as well as verify the necessity of both the matching module and cramming module in SS mechanism. These modules are designed to address the vanishing gradient situation in the adaptive learning process.

DOI:10.6814/NCCU202200079



	MW-PO-100	MW-LTS-0	MW-LTS-100	MW-LTS-500
Mean	88.14%	84.41%	94.56%	97.46%
S.D.	12.51%	10.44%	9.54%	3.61%
Min	30.82%	47.80%	47.80%	77.36%
Max	100.00%	98.11%	100.00%	100.00%

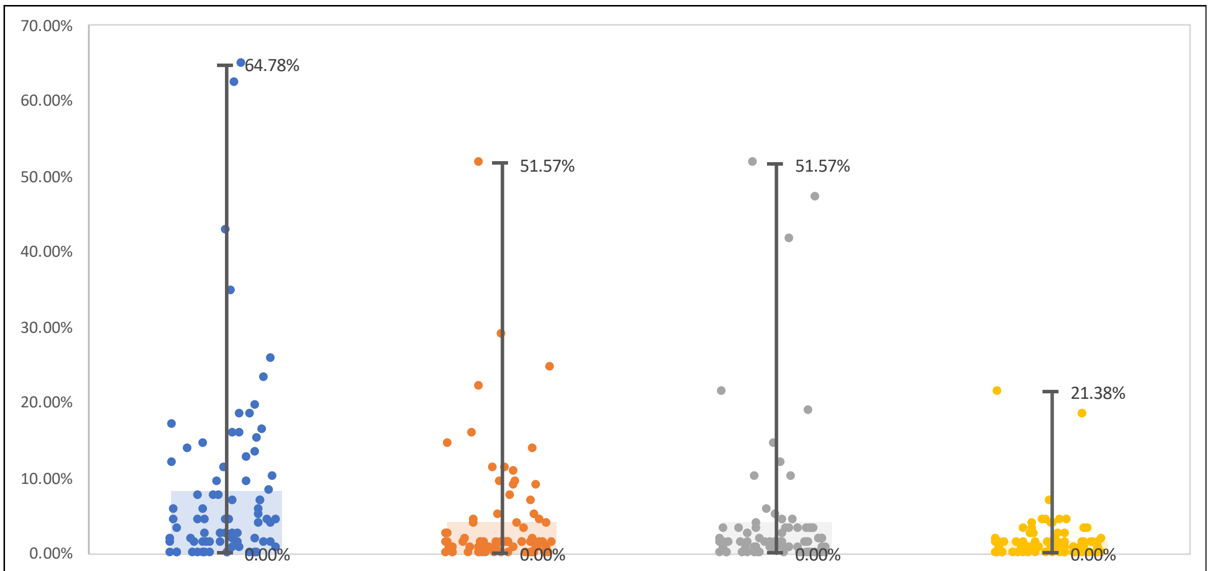
Figure 10. The *familiar route* executions



	MW-PO-100	MW-LTS-0	MW-LTS-100	MW-LTS-500
Mean	3.60%	11.33%	1.31%	0.79%
S.D.	2.43%	9.33%	1.65%	1.10%
Min	0.00%	0.63%	0.00%	0.00%
Max	10.69%	35.85%	8.81%	5.03%

Figure 11. The *thinking route* executions

DOI:10.6814/NCCU202200079



	MW-PO-100	MW-LTS-0	MW-LTS-100	MW-LTS-500
Mean	8.26%	4.26%	4.13%	1.75%
S.D.	12.13%	7.92%	9.47%	3.29%
Min	0.00%	0.00%	0.00%	0.00%
Max	64.78%	51.57%	51.57%	21.38%

Figure 12. The *cramming route* executions

Although these four versions mostly use the *familiar route* to obtain acceptable ASLFN (Figure 10), all versions still rely on the *thinking route* (Figure 11) and the *cramming route* (Figure 12) to learn some data. For example, the MW-LTS-500 is the version which less trigger matching module and cramming module. It still learns 0.79% of the data in a window through the matching module and 1.75% of the data in a window through the cramming module. Among these four versions, MW-LTS-0 is the version most commonly uses the *thinking route* to obtain an acceptable ASLFN, with an average of 11.33% of the data should be learnt via matching modules within a window. In terms of the *cramming route*, the MW-PO-100 is the most frequent trigger cramming module among these four versions, about 8.26% of the data in a window on average. Therefore we can verify that both matching module and cramming module in SS mechanism are necessary to solve the vanishing gradient situation.

DOI:10.6814/NCCU202200079

5.1.2 Number of adopted hidden nodes

The number of hidden nodes helps to verify that the fitting function is different among versions since the number of hidden nodes used implies the complexity of the fitting function behind the network (LeCun et al., 2015). We select the 8th window (one of the 78 windows) to illustrate the hidden node evolution within the window to understand the adaptive learning status of the four versions.

We can find the following observations from Figure 13. First, the adoption of LTS principle can effectively reduce the hidden nodes compared to PO principle. For example, the MW-PO-100 version triggers cramming actions frequently from the 89th stage of the learning process resulting in 54 hidden nodes at the end. In contrast, the MW-LTS-100 version only add 6 new hidden nodes (two triggers for the cramming module and each trigger add three extra hidden nodes) during the last two stages and eventually uses 7 hidden nodes.

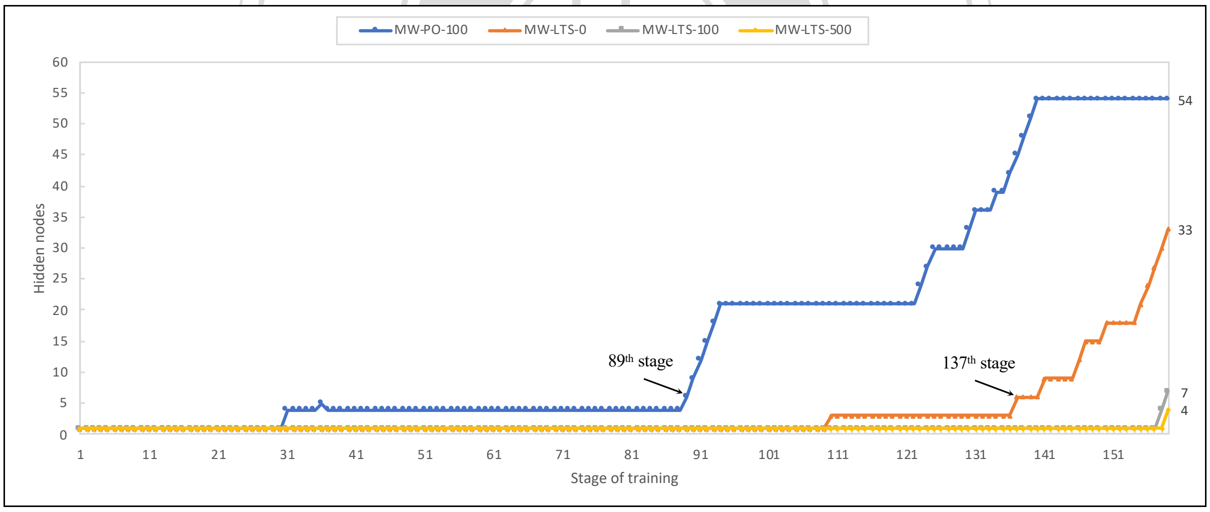


Figure 13. Evolution of the number of hidden nodes used in the 8th window

When the LTS principle is used, the more number of iterations to apply the regularization module the more it reduces the number of hidden nodes employed. It can be found from the comparison among the MW-LTS-0, MW-LTS-100, and MW-LTS-500. In the version of MW-LTS-0 that does not use the regularizing module, the hidden nodes are frequently added from the 137th stage. The MW-LTS-100 and MW-LTS-500 versions, which apply the regularization module,

DOI:10.6814/NCCU202200079

add additional hidden nodes in the last few stages. The number of hidden nodes used in MW-LTS-0, MW-LTS-100 and MW-LTS-500 are 33, 7, and 4, respectively. In summary, MW-LTS-100 and MW-LTS-500 are the versions that require the least number of hidden nodes among these four versions. We can conclude that using both the LTS principle and the regularizing module within SS mechanism help to obtain a lighter SLFN architecture since the required hidden node less means fewer parameters are required in the network.

From Table 12 (see Appendix Table B1 for details), it can also be found that SS mechanism with LTS principles and regularization modules contribute to lighter neural network architecture. The average number of the adopted hidden nodes for MW-LTS-100 and MW-LTS-500 is namely 15.99 and 7.85, which is smaller than MW-PO-100 and MW-LTS-0 with the average number of the adopted hidden nodes of 36.97 and 17.76. Because the *cramming route* is used to learn the new coming training sample by adding an additional hidden node, the number of adopted hidden nodes is correspond to it. In terms of the execution of the *cramming route*, the adoption of both the LTS principle and the regularization module can reduce the learning of new incoming training samples through the *cramming route* (Figure 12). Therefore, these versions (i.e., MW-LTS-100, and MW-LTS-500) use the least number of hidden nodes.

Among these versions, the S.D. of MW-LTS-500 and MW-LTS-100 are the first two minimum values of 14.94 and 31.67, respectively. That is smaller than the value of 53.77 and 34.97 got from versions of MW-PO-100 and MW-LTS-0. Similar results can be found in the range of the adopted hidden nodes. In 78 windows, both versions of MW-PO-100 and MW-LTS-0 require hidden nodes between 1 and 286, and 1 and 245, respectively. This is more than the variation of hidden nodes required by MW-LTS-100 and MW-LTS-500. In summary, it can be found that using both the LTS principle and regularizing module can obtain less average and dispersion of the adopted hidden nodes.

DOI:10.6814/NCCU202200079

Table 12. The number of adopted hidden nodes

	MW-PO-100	MW-LTS-0	MW-LTS-100	MW-LTS-500
Mean	36.97	17.76	15.99	7.85
S.D.	53.77	34.97	31.67	14.94
Min	1	1	1	1
Max	286	245	197	102

According to the three structural breakpoints detected by the methodology proposed by Bai and Perron (1998, 2003), the four phases of copper price change are distinguished by these breakpoints (Figure 9). The number of hidden nodes required for each of the four phase is different regardless of the version (Table 13). These four versions have similarities in the number of hidden nodes required such as phase I and phase III are more than phase II and phase IV. Since each phase has different patterns and requires different hidden nodes, it is necessary to propose such an SS mechanism that can automatically adjust the ASLFN structure (i.e., add and delete hidden nodes of the hidden layer) based on the current phase.

Table 13. The hidden node required for each phase

	MW-PO-100	MW-LTS-0	MW-LTS-100	MW-LTS-500
Phase I (1st-21st window)	40.00	19.45	23.70	9.80
Phase II (22nd-48th window)	38.48	11.81	8.44	5.33
Phase III (49th-66th window)	47.67	34.28	20.89	11.67
Phase IV (67th-78th window)	15.50	4.92	14.00	5.08

5.1.3 Training time

The total training time required for the MW-LTS-500 was 28,153.37 seconds, which is the lowest cumulative training time of the four versions. (Figure 14). The following are MW-LTS-100 and MW-LTS-0, which took a total of 41,705.25, and 43,764.77 seconds, respectively. The major

DOI:10.6814/NCCU202200079

difference among MW-LTS-500, MW-LTS-100, and MW-LTS-0 is the maximum number of times (i.e., 500, 100, and 0) the regularizing module can be executed. We can get the conclusion that the larger the maximum number of times the regularizing module is executed, the less we need to trigger the cramming module (MW-LTS-500 only triggers cramming module to learn 1.75% of training data in Figure 12). When the cramming module does not need to be triggered, it means that there is no need to add additional hidden nodes for training. In addition, the reduction in the number of hidden nodes used also reduces the time for the reorganization module to traverse all the adopted hidden nodes. We regard that using regularizing module is helpful to reduce the training time.

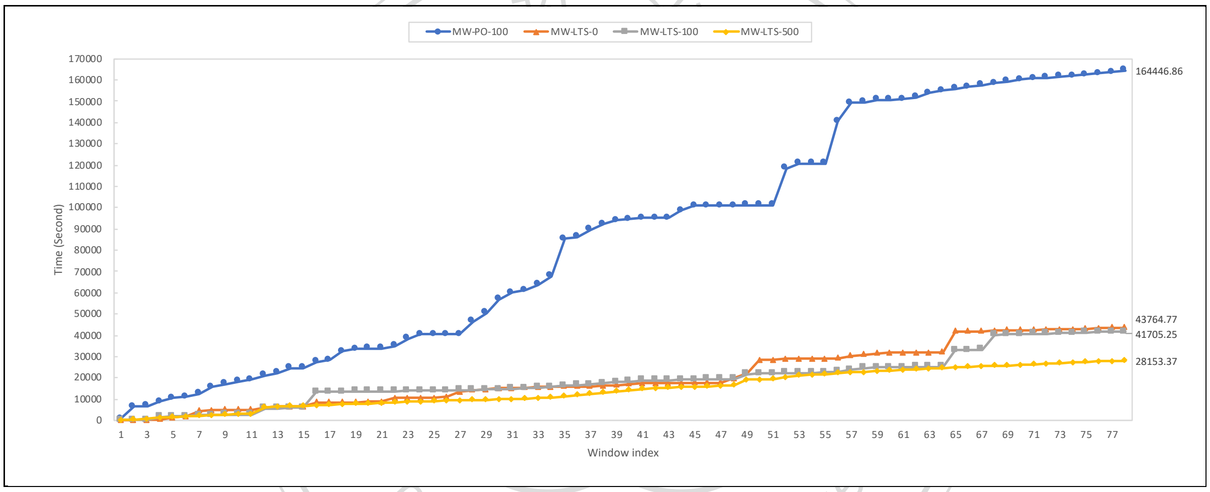


Figure 14. The accumulated training time

It is worth noting that if the LTS principle is used instead of the PO principle, the time required decreases from 164,446.86 to 41,705.25 seconds when the comparison of the MW-PO-100 with MW-LTS-100. The difference in training time between the MW-LTS-100 and MW-PO-100 versions in terms of their learning routes is applicable to the aforementioned conclusion.

The same results can also be found from the average training results presented as Table 14 (see Appendix Table C1 for details). The time required to train a window on average for MW-LTS-500, MW-LTS-100 and MW-LTS-0 is 360.94, 534.68, and 561.09 seconds, respectively. When the number of activation of the regularizing module is set to more, the training time can be reduced.

DOI:10.6814/NCCU202200079

Comparing MW-PO-100 with MW-LTS-100, we can observe that the MW-LTS-100 takes an average of 534.68 seconds per window less than the MW-PO-100, which takes 2108.29 seconds. In summary, the learning style of the version that uses both the LTS principle and the regularizing module can effectively save the required training time.

Table 14. The average training time

(Units: second)

	MW-PO-100	MW-LTS-0	MW-LTS-100	MW-LTS-500
Mean	2108.29	561.09	534.68	360.94
S.D.	3650.71	1388.54	1457.91	443.63
Min	35.31	10.37	45.29	161.73
Max	19790.97	9912.40	7685.57	3150.11

5.1.4 Validation for reorganizing module

To verify the functionality of reorganizing module which is designed to alleviate the overfitting tendency, the number of potential irrelevant hidden nodes that have been deleted is used to measure the effort of the reorganizing module as shown in Table 15 (see Appendix Table D1 for details). These four versions enabled the reorganization module to delete more than one potentially irrelevant hidden node in over 80% of the 78 windows (Figure 15).

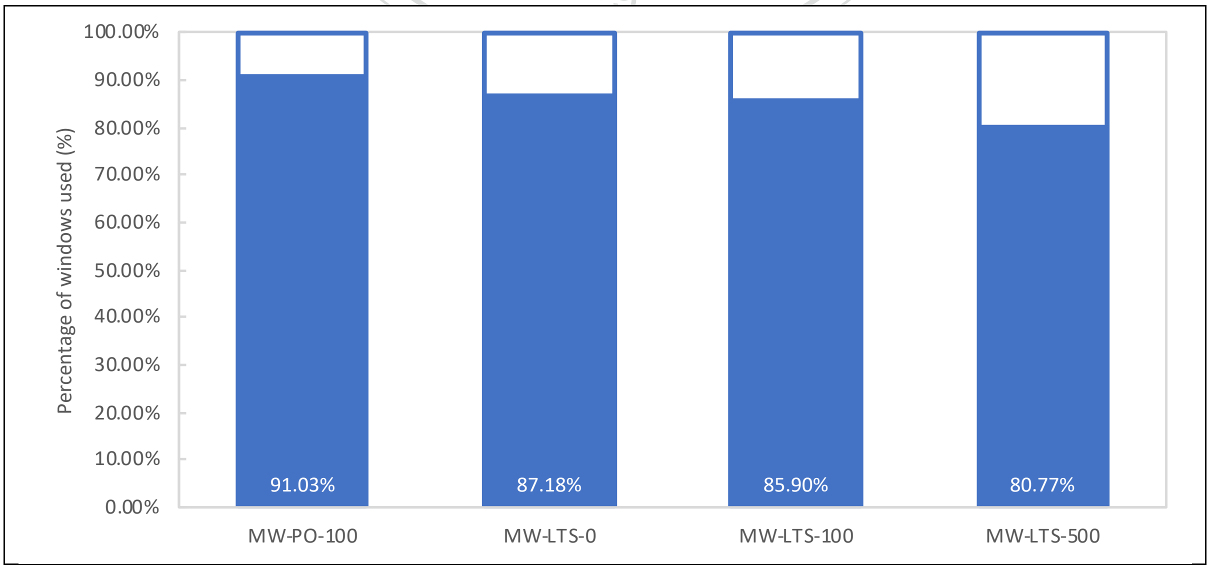


Figure 15. The percentage of windows used reorganizing module

DOI:10.6814/NCCU202200079

Table 15. The average number of irrelevant hidden nodes deleted by reorganizing module

	MW-PO-100	MW-LTS-0	MW-LTS-100	MW-LTS-500
Mean	39.01	18.94	16.31	8.17
S.D.	54.72	35.44	31.59	15.07
Min	0	0	0	0
Max	300	250	196	103

The average deletion amount of 39.01 with PO principle is much higher than the average deletion amount of 16.31 with LTS principle, which can be found from the comparison of MW-PO-100 and MW-LTS-100. In addition, it can be found that as the maximum number of epochs for the regularizing module increases, the average number of hidden node deleted also decreases gradually. For example, MW-LTS-0 deletes an average of 18.94 hidden nodes in a window, which is much more than the values of 16.31 and 8.17 obtained by MW-LTS-100 and MW-LTS-500 respectively.

In Table 12, we found that the versions that use both LTS principle and regularizing module (i.e., MW-LTS-100 and MW-LTS-500) employ fewer hidden nodes than the other versions. Table 15 also shows that both versions have the lowest number of potentially irrelevant hidden nodes deleted by the reorganizing module. We can argue that using both LTS principle and regularizing module is less likely to add unnecessary hidden nodes in the learning process. This help to improve the learning efficiency. Overall, the reorganizing module is used in all four versions to delete irrelevant hidden nodes. We can conclude that the reorganization module is necessary to exist in the SS mechanism.

DOI:10.6814/NCCU202200079

5.1.5 Validation for LTS principle

For the purpose of verifying the effect of the LTS principle, the MW-LTS-100 version is compared with the MW-PO-100 version. As illustrated as Figure 10, a higher frequency of MW-LTS-100 compared to MW-PO-100 mostly use the *familiar route* to obtain an acceptable ASLFN. On the other hand, the MW-PO-100 is the version that more often takes *cramming route* to obtain an acceptable ASLFN. Therefore, the number of hidden nodes required for MW-PO-100 is relatively higher than MW-LTS-100 during the training process. We can confirm that the adoption of the LTS principle enhances learning efficiency.

The version with LTS principle not only allows for more efficient learning but also helps to improve forecasting performance (Table 16). At the training stage, the lower MAE, MAPE, and RMSE can be gained from MW-LTS-100. Similar results can be found at the testing stage, the MAE of MW-LTS-100 is 2427.44, which is smaller than MW-PO-100 with a MAE of 3305.56. The MAPE of MW-LTS-100 is 5.50, being smaller than that the MAPE of MW-PO-100 is 7.40. The RMSE of MW-LTS-100 is 2562.26, which is smaller than MW-PO-100 with a RMSE of 3435.47. Consequently, we can confirm that the version using the LTS principle has more efficient for forecasting than the version using the PO principle. We also evaluated the performance of the model based on the loss during training and testing process. The results show that the version with the LTS principle has lower training loss and testing loss, and the difference is closer. This can be seen as adopting the LTS principle also helps to avoid overfitting (Figure 16). The forecasting performances of the MW-PO-100, and MW-LTS-100 are presented in Figure 17.

Table 16. Forecasting performance of MW-PO-100, and MW-LTS-100

Models	Training stage			Testing stage		
	MAE	MAPE	RMSE	MAE	MAPE	RMSE
MW-PO-100	1379.93	3.02	1824.07	3305.56	7.40	3435.47
MW-LTS-100	**1153.27**	**2.52**	**1510.34**	**2427.44**	**5.50**	**2562.26**

Values in bold represent the best model

DOI:10.6814/NCCU202200079

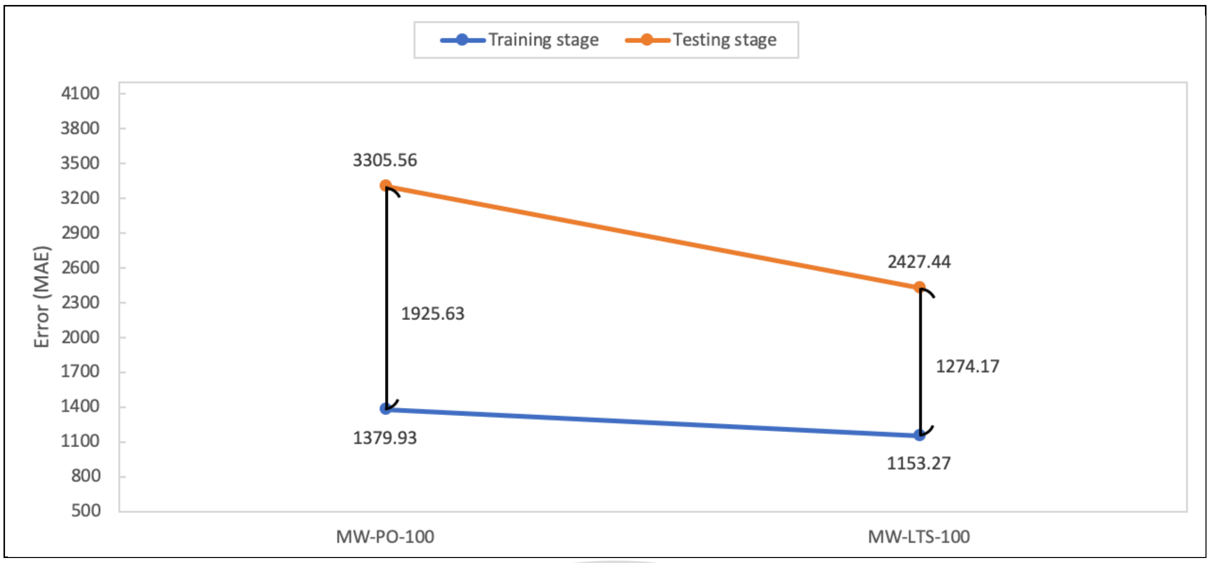


Figure 16. The training loss and testing loss of MW-PO-100, and MW-LTS-100

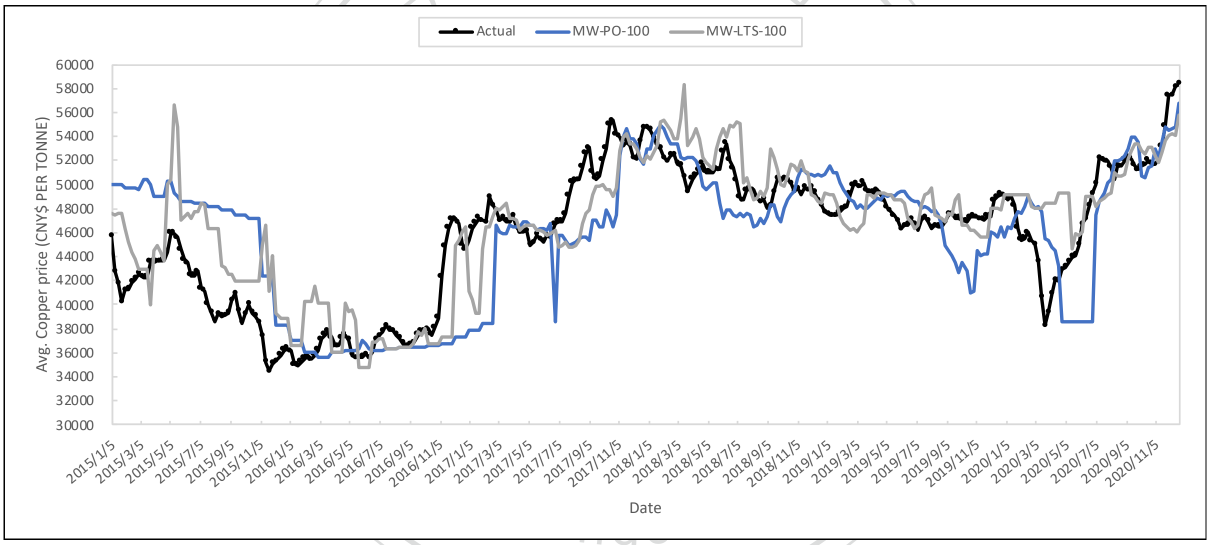


Figure 17. Out-of-sample performance of MW-PO-100, and MW-LTS-100

5.1.6 Validation for regularizing module

The effect of regularizing module has been validated from a comparison of the MW-LTS-0, MW-LTS-100, and MW-LTS-500 versions. Several results show that the MW-LTS-0 (the version that does not contain the regularizing module) takes long time as well as lots of hidden nodes for training. This can be regarded as regularizing module is helpful to improve the training efficiency. Also, the MAE, MAPE, and RMSE are used to measure the forecasting error during the

DOI:10.6814/NCCU202200079

training and testing process. In the training phase, the lowest MAE, MAPE and RMSE can be obtained from MW-LTS-500, while the highest MAE, MAPE and RMSE can be obtained from MW-LTS-0 (Table 17). The same results were obtained at the testing stage. The version that includes both LTS principle and the regularization module is not only useful for effective learning but also has superior forecasting performance.

After confirming the effect of regularizing module on both effectiveness and efficiency, we can further look at the effect of the maximum number of executions of regularizing module on effectiveness and efficiency. The MW-LTS-500 can be found with less hidden node and training time than the MW-LTS-100. For example, the MW-LTS-500 version takes 360.94 seconds and employs 7.85 hidden nodes on average for a window during the training process, which is more efficient than the MW-LTS-100 version that takes 534.68 seconds, and employs 15.99 hidden nodes. In addition, setting a larger number of activations also enhances the forecasting power of the network at the testing stage. The lowest of MAE, MAPE, and RMSE can be obtained from the MW-LTS-500 version. When the maximum number of iterations of the regularization module is set higher, the difference between the training loss and the test loss is closer (Figure 18). Therefore, the more regularizing times used, the less time it will take to train and the more it will help to avoid overfitting. The actual and forecasted values during testing via MW-LTS-0, MW-LTS-100, and MW-LTS-500 as shown in Figure 19.

Table 17. Forecasting performance of the version with LTS principle

Models	Training stage			Testing stage		
	MAE	MAPE	RMSE	MAE	MAPE	RMSE
MW-LTS-0	1820.24	4.02	2314.32	3271.44	7.43	3450.17
MW-LTS-100	1153.27	2.52	1510.34	2427.44	5.50	2562.26
MW-LTS-500	**1112.11**	**2.43**	**1453.04**	**1923.63**	**4.19**	**2071.42**

Values in bold represent the best model

DOI:10.6814/NCCU202200079

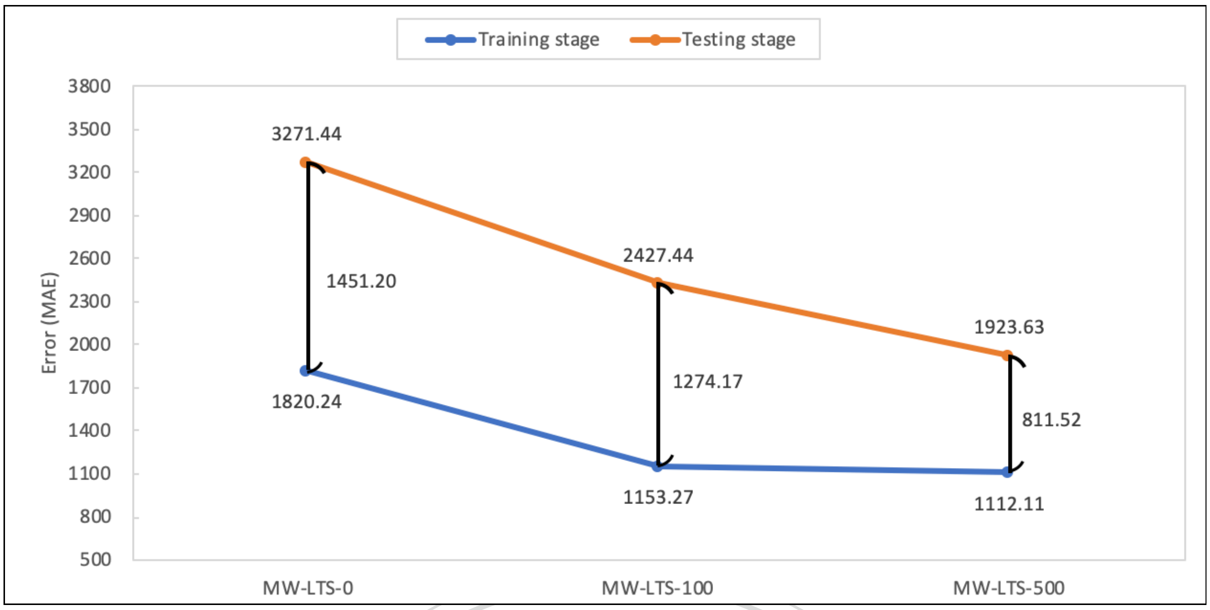


Figure 18. The training loss and testing loss of the version with LTS principle

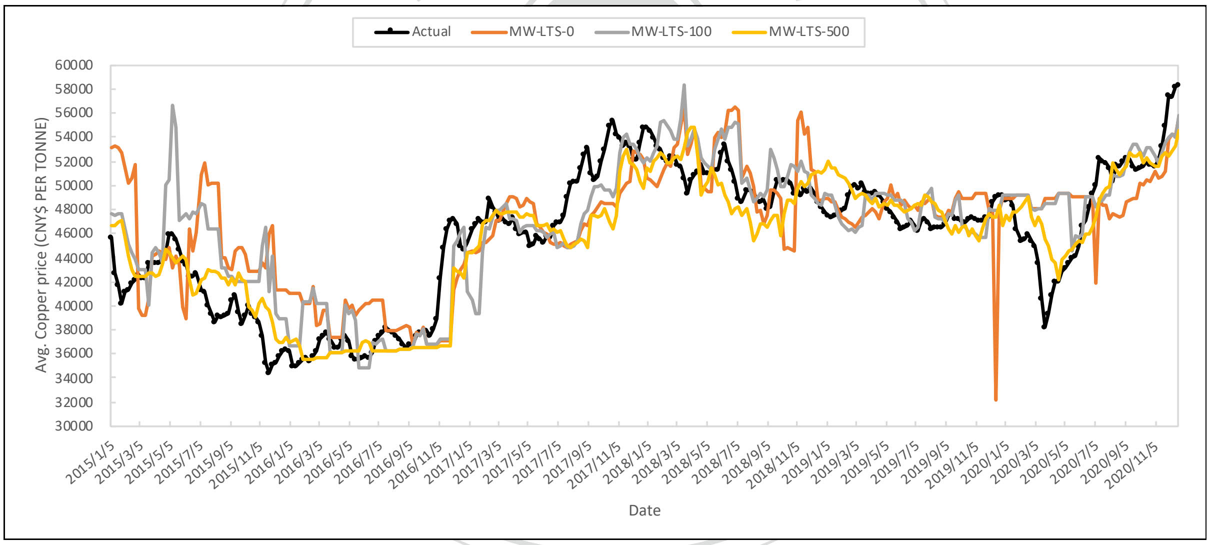


Figure 19. Out-of-sample performance of MW-LTS-0, MW-LTS-100, and MW-LTS-500

5.2 ALFM evaluation

In this section, MW-LTS-500 which is the best forecasting capacity among proposed versions is used to measure ALFM forecast performance. Since MW-LTS-500 is a SLFN-based forecasting model that includes both the moving window mechanism and the SS mechanism. In order to confirm that these mechanisms are helpful for the prediction performance, the MW-LTS-

DOI:10.6814/NCCU202200079

500 is first compared to LTS-500, MW-SLFN, and SLFN. These models are based on SLFN, but LTS-500 and MW-SLFN only contain SS mechanism and moving window mechanism, respectively. The number of adopted hidden nodes in the hidden layer of the SLFN-based forecasting model is set to 8 units which is the average number of hidden nodes used by the MW-LTS-500 to train a window on average. The hyperparameter settings of the corresponding mechanisms are the same as those of MW-LTS-500. In other words, if the forecasting model contains a moving window mechanism, the length of the training block and testing block are set to values of 159 and 4, respectively; Or if the SS mechanism is included, the selection module follows the LTS principle, the maximum number of training iterations of the regularization module is set to 500, and the hyperparameters of the other modules are set as shown in Figure 6 - Figure 8.

MW-LTS-500 is further compared with six popular methods in the literature (i.e., SARIMA, SLFN, SVM, RNN, LSTM, and GRU). All MAE, MAPE, and RMSE aforementioned were used for measuring and evaluating the performance and accuracy of the proposed model in comparison with other forecasting models. These models without moving window module use the convenient ratios to divide the training and test data into the ratio of 80:20, in which the first 376 observations (from October 31, 2011, to February 25, 2019) were used for training the model, whereas the final 95 observations (from March 4, 2019, to December 21, 2020) were employed to validate the performance of the proposed model. We also use the same period of data to train and test the MW-LTS-500.

The hyperparameters of each model are given as follows: for the SARIMA model, the notation for these elements is specified as SARIMA(trend autoregression order, trend difference order, trend moving average order)(seasonal autoregressive order, seasonal difference order, seasonal moving average order, and the number of time steps for a single seasonal period) whose value is set as SARIMA(1,1,0)(0,1,2,53). These values are determined by the *auto_arima* function in python, which performs a random search over a hyperparameter space; for the SVR model, the kernel type is set to radial basis, the kernel coefficient is set to 0.1; for the NN-based models (i.e.,

DOI:10.6814/NCCU202200079

SLFN, RNN, LSTM, and GRU), the structure is set as a single hidden layer and contains only 8 neurons, and the activation function of the hidden node is set as ReLU with Adam optimizer (Kingma & Ba, 2014) to adjust the parameters. During the training period, each training was conducted with a batch size of 14, and the training was repeated 300 times. Since the RNN-based models (i.e., RNN, LSTM, and GRU) learn in a sequence-to-sequence way, the input variable used is not only the copper price, but the rest of the variables all contain a lag time of 4 units.

Table 18 summarizes the performance measure of the SLFN-based models used in this study. Comparison with LTS-500 and SLFN, the lower loss (i.e., MAE, MAPE, and RMSE) can be obtained by LTS-500. This can be seen as an SS mechanism that is helpful in forecasting performance. Comparing MW-SLFN and SLFN, the model with moving window mechanism also contributes to reduce forecasting error. The MW-LTS-500 is best model among these network architectures from the result of forecasting performance. We can confirm the incorporation of a moving window mechanism with the SS mechanism is effective for four-weeks ahead copper price forecasts. The forecasting result of SLFN-based model for test data are plotted in Figure 20.

Table 18. Out-of-sample performance comparison of the SLFN-based models

Model	MAE	MAPE	RMSE
ALFM (MW-LTS-500)	**1576.39**	**3.33**	**2080.75**
LTS-500	2361.18	4.87	3367.10
MW-SLFN	2436.76	4.99	3270.72
SLFN	3649.32	7.77	4570.24

Values in bold represent the best model

DOI:10.6814/NCCU202200079

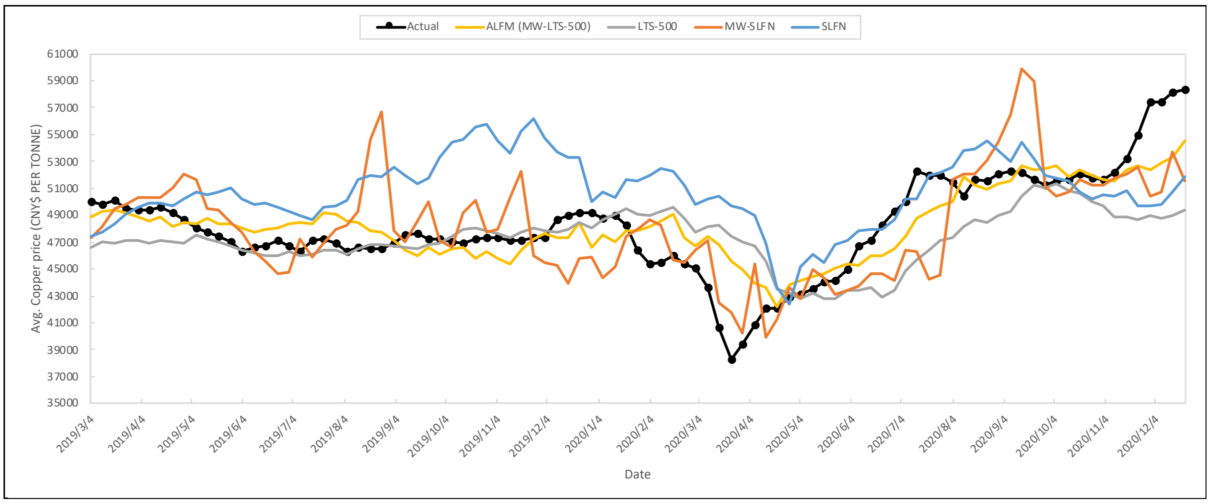


Figure 20. Out-of-sample performance comparison of the SLFN-based models

The proposed MW-LTS-500 also presents several noteworthy contributions to improve the performance of the copper price forecasting for four-weeks ahead. Table 19 indicates that among these forecasting model used for comparison, the lowest MAE, MAPE, and RMSE have been gained from the proposed forecasting model (i.e., MW-LTS-500). This suggests that the ASLFN with a moving window mechanism and an SS mechanism can increase the forecasting accuracy on the copper price. The forecasting result of each model for test data are plotted in Figure 21.

Table 19. Out-of-sample performance comparison of the models

Model	MAE	MAPE	RMSE
ALFM (MW-LTS-500)	**1576.39**	**3.33**	**2080.75**
SARIMA	2873.98	6.02	3764.86
SLFN	3649.32	7.77	4570.24
SVR	3359.37	6.85	4318.43
RNN	2646.85	5.50	3686.23
LSTM	2584.56	5.46	3319.30
GRU	3354.81	6.87	4228.00

Values in bold represent the best model

DOI:10.6814/NCCU202200079

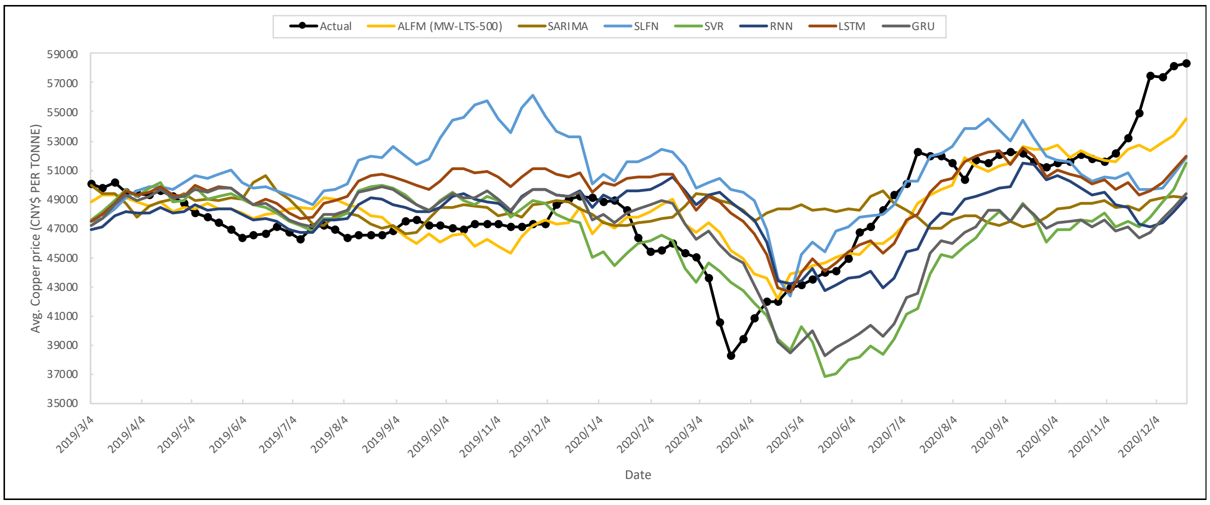


Figure 21. Out-of-sample performance comparison of MW-LTS-500 with other models

DOI:10.6814/NCCU202200079

Chapter 6 Discussion and Conclusions

6.1 Summary

Forecasting copper prices with high accuracy and reliability are critical for investors and businesses. Investigation results showed that copper prices have complicated movements from 2011 to 2020, and accurate forecasting fluctuations in copper prices is a challenge. This study proposes an adaptive learning-based forecasting model (ALFM) to cope with the concept drift and the complex fitting function form behind the copper price. ALFM consists of two parts: the moving window mechanism, and the sequentially structuring (SS) mechanism. The moving window mechanism first changes the data into the form of a window, which is used to cope with structural and conceptual change. The SS mechanism is implemented in ASLFN to avoid the vanishing gradient and overfitting issues that are often faced by modeling neural networks. Since the SS mechanism was first proposed and developed in this study, it needs to be further validated.

Through the empirical result, we have several major findings that can contribute to the literature. We confirm that all modules of the SS mechanism are necessary. Each version is learned during the training process through the following three paths： *familiar route*, *thinking route*, and *cramming route*. When the *thinking route* and *cramming route* are performed, it means that the matching module and the cramming module have been activated to learn new material, respectively. The necessity of the reorganizing module in the SS mechanism has also been verified since the reorganizing module is activated in all versions to delete irrelevant neurons during the training process. Also, both matching module and cramming module help to solve vanishing gradient from the evidence of both modules are required for all versions to ensure learning all training data set.

Comparing the MW-PO-100 and MW-LTS-100 versions, LTS principle helps to improve the learning efficiency and prediction of the neural network. In the LTS version, both MW-LTS-100 and MW-LTS-500 require less training time and forecasting error than MW-LTS-0, which is considered necessary for the regularizing module and helps the neural network to learn in an

DOI:10.6814/NCCU202200079

efficient way and reduce the overfitting phenomenon. Comparing MW-LTS-100 and MW-LTS-500, we can further find that the effectiveness and efficiency of learning are more obvious when the number of activations increases.

The MW-LTS-500 is the best architecture of the adaptive learning-based forecasting model. To verify the proposed forecasting model does lead to good performances on copper price volatility, we compare the proposed MW-LTS-500 with other state-of-art models (i.e., SARIMA, SLFN, SVM, RNN, LSTM, and GRU). The results demonstrate the superiority of the MW-LTS-500 model overall comparison methods. The numerical results imply that the MW-LTS-500 model provides the lowest MAE, MAPE, and RMSE of 1576.39, 3.33, and 2080.75, respectively.

To the best of our knowledge, the research not only examines the structural and conceptual change on copper price but also proposes the tool to solve it. We have conducted a variety of experiments to validate the proposed ALFM which has a notable forecasting power in copper price forecasting. Such an adaptive network structure can systematically adjust the number of adopted hidden nodes during the learning process. This allows us to solve the challenges of modeling NN-based models such as vanishing gradient, and overfitting tendency.

6.2 Theoretical and practical contributions

To fill the theoretical gap, the ALFM proposed in the literature simultaneously deals with the challenges (i.e., concept drift/structural change, and non-linear fitting function form) embedded in the copper price data stream. First, this study validates the issue of structural change in copper prices by the methodology proposed by Bai and Perron (1998, 2003). Based on the previous literature, we find that the concept is similar to the term “concept drift” in fields of engineering and machine learning. Hence, the method of concept drift handling (i.e., moving window mechanism) is used to address the issue of underlying structural change behind the copper price.

Additionally, NN-based models are often used to cope with the non-linear fitting function form in the data stream. But there are some fundamental challenges associated with the NN-based model like vanishing gradient, and overfitting. This study proposed the SS mechanism for the

DOI:10.6814/NCCU202200079

ASLFN in response to these issues. Eventually, an adaptive learning-based forecasting model (ALFM) for learning the patterns embedded in the data stream obtained from a structural change environment to form a forecasting model that is adaptive within the learning process. Through the empirical results, the ALFM outperforms state-of-the-art models within an acceptable training time.

The experimental results obtained with real data confirm the proposed ALFM works quite well in capturing the overall medium-term trend. In practice, medium-term forecasting brings great benefits in energy management and production planning in order to ensure continuous production and raw material sufficiency. An accurate and robust model can further guide related copper manufacturing companies, policymakers, and investors to catch the volatility of copper prices in a dynamic environment. As a result, copper producers can plan their procurement decision and production operations more efficiently and responsively. Policymakers can regulate the market more effectively based on accurate price forecasting. Investors can better design profitable medium-term investment strategies.

6.3 Limitations and future works

Despite the remarkable contributions of this study, there is some direction for future research regarding the modeling and learning of the proposed forecasting model, ASLFN. First, the length of the training period *N* and the testing period *B* in the study are based on try-and-error for all the periods of observations (October 31, 2011, to December 21, 2020). Due to the high volatility and structural and conceptual changes in the copper price, it frequently moves in a different way. If the moving window mechanism is chosen to solve for different intervals of copper prices, the setting of the window size must be revisited or the moving-window scheme with variable can also be attempted in practice. Second, all the modules in the SS mechanism have been confirmed necessary and can effectively solve the vanishing gradient and overfitting phenomena. However, there are some modules (i.e., cramming module, and reorganizing module) whose processes can be further optimized to improve efficiency. The cramming module is used to solve

DOI:10.6814/NCCU202200079

the vanishing gradient by recruiting extra hidden nodes and the reorganizing module is used to prune irrelevant hidden nodes to prevent poor network generalization capabilities.

In terms of the application, there are some future works. Wahab & Adewuyi (2021) confirmed the characteristics of structural breaks and non-linearity in the other metal prices (i.e., gold, silver, platinum, and palladium). Meanwhile, these phenomena also exist in the time series of macro-economic variables (Nasir & Vo, 2020), and the financial market (Mahata et al., 2020). It would be interesting to confirm whether there would be any major changes in the results when the proposed technique can be utilized as a forecasting tool to deal with other forecasting problems with structural and conceptual change.

DOI:10.6814/NCCU202200079

References

Abbasimehr, H., Shabani, M., & Yousefi, M. (2020). An optimized model using LSTM network for demand forecasting. *Computers and Industrial Engineering*, *143*, 106435.

Alameer, Z., Elaziz, M. A., Ewees, A. A., Ye, H., & Jianhua, Z. (2019). Forecasting copper prices using hybrid adaptive neuro-fuzzy inference system and genetic algorithms. *Natural Resources Research*, *28*(4), 1385–1401.

Alippi, C., Liu, D., Zhao, D., & Bu, L. (2014). Detecting and reacting to changes in sensing units: The active classifier case. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *44*(3), 353–362.

Andreou, E., & Ghysels, E. (2009). Structural breaks in financial time series. *Handbook of Financial Time Series*, 839–870.

Andrews, D. W. (1993). Tests for parameter instability and structural change with unknown change point. *Econometrica: Journal of the Econometric Society*, *61*, 821–856.

Andrews, D. W. K., Lee, I., & Ploberger, W. (1996). Optimal changepoint tests for normal linear regression. *Journal of Econometrics*, *70*(1), 9–38.

Andrews, D. W., & Ploberger, W. (1994). Optimal tests when a nuisance parameter is present only under the alternative. *Econometrica: Journal of the Econometric Society*, 1383–1414.

Angus, A., Casado, M. R., & Fitzsimons, D. (2012). Exploring the usefulness of a simple linear regression model for understanding price movements of selected recycled materials in the UK. *Resources, Conservation and Recycling*, *60*, 10–19.

Astudillo, G., Carrasco, R., Fernández-Campusano, C., & Chacón, M. (2020). Copper price prediction using support vector regression technique. *Applied Sciences*, *10*(19), 1–10.

Babcock, B., Datar, M., & Motwani, R. (2001). Sampling from a moving window over streaming data. *In 2002 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2002)*.

DOI:10.6814/NCCU202200079

Bai, J., & Perron, P. (1998). Estimating and testing linear models with multiple structural changes. *Econometrica*, *66*(1), 47.

Bai, J., & Perron, P. (2003). Computation and analysis of multiple structural change models. *Journal of Applied Econometrics*, *18*(1), 1–22.

Baier, L., Hofmann, M., Kühl, N., Mohr, M., & Satzger, G. (2020). Handling concept drifts in regression problems-the error intersection approach. *ArXiv Preprint ArXiv:2004.00438*.

Bao, Y., Lu, Y., & Zhang, J. (2004). Forecasting stock price by SVMs regression. *International Conference on Artificial Intelligence: Methodology, Systems, and Applications*, 295–303.

Behmiri, B., N., Manera, M., Behmiri, N. B., & Manera, M. (2015). The role of outliers and oil price shocks on volatility of metal prices. *Resources Policy*, *46*, 139–150.

Bifet, A., & Gavaldà, R. (2007). Learning from time-changing data with adaptive windowing. *Proceedings of the 2007 SIAM International Conference on Data Mining*, 443–448.

Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, *31*(3), 307–327.

Buncic, D., & Moretto, C. (2015). Forecasting copper prices with dynamic averaging and selection models. *The North American Journal of Economics and Finance*, *33*, 1–38.

Cao, L. J., & Tay, F. E. H. (2003). Support vector machine with adaptive parameters in financial time series forecasting. *IEEE Transactions on Neural Networks*, *14*(6), 1506–1518.

Carrasco, R., Fernández-Campusano, C., Soto, I., Lagos, C., Krommenacker, N., Banguera, L., & Durán, C. (2019). Copper price variation forecasts using genetic algorithms. *International Conference on Applied Technologies*, 284–296.

Carrasco, R. R., Astudillo, G., Soto, I., Chacon, M. M. M., & Fuentealba, D. (2018). Forecast of copper price series using vector support machines. *2018 7th International Conference on Industrial Technology and Management (ICITM)*, 380–384.

DOI:10.6814/NCCU202200079

Chan, K. S., & Tong, H. (1986). On estimating thresholds in autoregressive models. *Journal of Time Series Analysis*, *7*(3), 179–190.

Chang, J. H., & Lee, W. S. (2005). estWin: Online data stream mining of recent frequent itemsets by sliding window method. *Journal of Information Science*, *31*(2), 76–90.

Chavas, J. P. (2001). Structural change in agricultural production: Economics, technology and policy. *Handbook of Agricultural Economics*, *1*, 263–285.

Chen, Y. C., Rogoff, K. S., & Rossi, B. (2010). Can exchange rates forecast commodity prices? *The Quarterly Journal of Economics*, *125*(3), 1145–1194.

Chen, Y. Q., Thomas, D. W., & Nixon, M. S. (1994). Generating-shrinking algorithm for learning arbitrary classification. *Neural Networks*, *7*(9), 1477–1489.

Chow, G. C. (1960). Tests of equality between sets of coefficients in two linear regressions. *Econometrica: Journal of the Econometric Society*, *28*, 591–605.

Chu, C. S. J., Stinchcombe, M., & White, H. (1996). Monitoring structural change. *Econometrica: Journal of the Econometric Society*, *64*(5), 1045–1065.

Çinar, A. (1995). Nonlinear time series models for multivariable dynamic processes. *Chemometrics and Intelligent Laboratory Systems*, *30*(1), 147–158.

Ciner, C. (2017). Predicting white metal prices by a commodity sensitive exchange rate. *International Review of Financial Analysis*, *52*, 309–315.

Cologni, A., & Manera, M. (2008). Oil prices, inflation and interest rates in a structural cointegrated VAR model for the G-7 countries. *Energy Economics*, *30*(3), 856–888.

Cró, S., & Martins, A. M. (2017). Structural breaks in international tourism demand: Are they caused by crises or disasters? *Tourism Management*, *63*, 3–9.

Dehghani, H. (2018). Forecasting copper price using gene expression programming. *Journal of Mining and Environment*, *9*(2), 349–360.

DOI:10.6814/NCCU202200079

Dehghani, H., & Bogdanovic, D. (2018). Copper price estimation using bat algorithm. *Resources Policy*, *55*, 55–61.

Deng, L., & Yu, D. (2014). Deep learning: methods and applications. *Foundations and Trends in Signal Processing*, *7*(3–4), 197–387.

Diez-Sierra, J., & del Jesus, M. (2020). Long-term rainfall prediction using atmospheric synoptic patterns in semi-arid climates with statistical and machine learning methods. *Journal of Hydrology*, *586*, 124789.

Domhan, T., Springenberg, J. T., & Hutter, F. (2015). Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Du, W., Chen, Z. P., Zhong, L. J., Wang, S. X., Yu, R. Q., Nordon, A., Littlejohn, D., & Holden, M. (2011). Maintaining the predictive abilities of multivariate calibration models by spectral space transformation. *Analytica Chimica Acta*, *690*(1), 64–70.

Engle, R. F. (1982). Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. *Econometrica: Journal of the Econometric Society*, 987–1007.

Fahlman, S. E., & Lebiere, C. (1990). The cascade-correlation learning architecture. *Advances in Neural Information Processing Systems*, 524–532.

Farid, D. M., Zhang, L., Hossain, A., Rahman, C. M., Strachan, R., Sexton, G., & Dahal, K. (2013). An adaptive ensemble classifier for mining concept drifting data streams. *Expert Systems with Applications*, *40*(15), 5895–5906.

Fornaciari, M., & Grillenzoni, C. (2017). Evaluation of online trading systems: Markov-switching vs time-varying parameter models. *Decision Support Systems*, *93*, 51–61.

Frean, M. (1990). The upstart algorithm: A method for constructing and training feedforward neural networks. *Neural Computation*, *2*(2), 198–209.

DOI:10.6814/NCCU202200079

Gama, J., & Kosina, P. (2014). Recurrent concepts in data streams classification. *Knowledge and Information Systems*, *40*(3), 489–507.

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Computing Surveys (CSUR)*, *46*(4), 1–37.

García, D., & Kristjanpoller, W. (2019). An adaptive forecasting approach for copper price volatility through hybrid and non-hybrid models. *Applied Soft Computing Journal*, *74*, 466–478.

Garcia, R., & Perron, P. (1996). An analysis of the real interest rate under regime shifts. *The Review of Economics and Statistics*, 111–125.

Gargano, A., & Timmermann, A. (2014). Forecasting commodity price indexes using macroeconomic and financial predictors. *International Journal of Forecasting*, *30*(3), 825–843.

Gharleghi, B., Md Nor, A. H. S., & Sarmidi, T. (2014). Application of the threshold model for modelling and forecasting of exchange rate in selected ASEAN countries. *Sains Malaysiana*, *43*(10), 1609–1622.

Giannella, C., Han, J., Pei, J., Yan, X., & Yu, P. S. (2003). Mining frequent patterns in data streams at multiple time granularities. *Next Generation Data Mining*, *212*, 191–212.

Guo, H., Li, S., Li, B., Ma, Y., & Ren, X. (2018). A new learning automata-based pruning method to train deep neural networks. *IEEE Internet of Things Journal*, *5*(5), 3263–3269.

Hahnloser, R. H., Sarpeshkar, R., Mahowald, M. A., Douglas, R. J., & Seung, H. S. (2000). Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, *405*(6789), 947–951.

Hansen, B. E. (2001). The new econometrics of structural change: Dating breaks in U.S. labor productivity. *Journal of Economic Perspectives*, *15*(4), 117–128.

DOI:10.6814/NCCU202200079

Hao, W., & Yu, S. (2006). Support vector regression for financial time series forecasting. *International Conference on Programming Languages for Manufacturing*, 825–830.

He, B., Huang, H., & Yuan, K. (2016). Managing supply disruption through procurement strategy and price competition. *International Journal of Production Research*, *54*(7), 1980–1999.

Herna´ndez, E., Kristjanpoller, W., Hernández, E., & Herna´ndez, E. (2017). Volatility of main metals forecasted by a hybrid ANN–GARCH model with regressors. *Expert Systems with Applications*, *84*, 290–300.

Hernes, G. (1976). Structural change in social processes. *American Journal of Sociology*, *82*(3), 513–547.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Comput*, *9*(8), 1735–1780.

Hu, Y., Ni, J., & Wen, L. (2020). A hybrid deep learning approach by integrating LSTM-ANN networks with GARCH model for copper price volatility prediction. *Physica A: Statistical Mechanics and Its Applications*, *557*, 124907.

Hu, Z., Zhang, J., & Ge, Y. (2021). Handling vanishing gradient problem using artificial derivative. *IEEE Access*, *9*, 22371–22377.

Huang, W., Nakamori, Y., & Wang, S. Y. (2005). Forecasting stock market movement direction with support vector machine. *Computers & Operations Research*, *32*(10), 2513–2522.

Jiang, X., Adeli, H., Shi, L., Chu, L. K., Chen, Y. H., Jiang, X., & Adeli, H. (2005). Dynamic wavelet neural network model for traffic flow forecasting. *Journal of Transportation Engineering*, *131*(10), 771–779.

DOI:10.6814/NCCU202200079

Karakoyun, E. S., & Cibikdiken, A. O. (2018). Comparison of arima time series model and lstm deep learning algorithm for bitcoin price forecasting. *The 13th Multidisciplinary Academic Conference in Prague*, *2018*, 171–180.

Kashani, M. N., Aminian, J., Shahhosseini, S., & Farrokhi, M. (2012). Dynamic crude oil fouling prediction in industrial preheaters using optimized ANN based moving window technique. *Chemical Engineering Research and Design*, *90*(7), 938–949.

Katakis, I., Tsoumakas, G., & Vlahavas, I. (2010). Tracking recurring contexts using ensemble classifiers: An application to email filtering. *Knowledge and Information Systems*, *22*(3), 371–391.

Khashei, M., & Bijari, M. (2011). A novel hybridization of artificial neural networks and ARIMA models for time series forecasting. *Applied Soft Computing*, *11*(2), 2664–2675.

Khoshalan, H. A., Shakeri, J., Najmoddini, I., & Asadizadeh, M. (2021). Forecasting copper price by application of robust artificial intelligence techniques. *Resources Policy*, *73*, 102239.

Kim, K. J. (2003). Financial time series forecasting using support vector machines. *Neurocomputing*, *55*(1–2), 307–319.

Kingma, D. P., & Ba, J. L. (2014). Adam: A method for stochastic optimization. *ArXiv Preprint ArXiv:1412.6980*.

Klinkenberg, R. (2004). Learning drifting concepts: Example selection vs. example weighting. *Intelligent Data Analysis*, *8*(3), 281–300.

Klinkenberg, R., & Joachims, T. (2000). Detecting concept drift with support vector machines. *ICML*, 487–494.

Koitsiwe, K., & Adachi, T. (2017). The impact of structure change on copper prices. *Geo-Resources Environment and Engineering (GREE)*, *2*, 68–71.

DOI:10.6814/NCCU202200079

Koitsiwe, K., & Adachi, T. (2018). The role of financial speculation in copper prices. *Applied Economics and Finance*, *5*(4), 87.

Kosina, P., & Gama, J. (2015). Very fast decision rules for classification in data streams. *Data Mining and Knowledge Discovery*, *29*(1), 168–202.

Kristjanpoller, W., & Minutolo, M. C. (2015). Gold price volatility: A forecasting approach using the artificial neural network–GARCH model. *Expert Systems with Applications*, *42*(20), 7245–7251.

Kristjanpoller, W., & Minutolo, M. C. (2016). Forecasting volatility of oil price using an artificial neural network–GARCH model. *Expert Systems with Applications*, *65*, 233– 241.

Krüger, J. J. (2008). Productivity and structural change: A review of the literature. *Journal of Economic Surveys*, *22*(2), 330–363.

Lasheras, F. S., de Cos Juez, F. J., Sánchez, A. S., Krzemień, A., & Fernández, P. R. (2015). Forecasting the COMEX copper spot price by means of neural networks and ARIMA models. *Resources Policy*, *45*(September 2015), 37–43.

Lazli, L., & Boukadoum, M. (2013). Hidden neural network for complex pattern recognition: A comparison study with multi-neural network based approach. *International Journal of Life Science and Medical Research*, *3*(6), 234–245.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*, 436–444.

Leung, C. K.-S., & Khan, Q. I. (2006). DSTree: A tree structure for the mining of frequent sets from data streams. *Sixth International Conference on Data Mining (ICDM'06)*, 928–932.

Li, G., & Li, Y. (2015). Forecasting copper futures volatility under model uncertainty. *Resources Policy*, *46*, 167–176.

Li, J., Maier, D., Tufte, K., Papadimos, V., & Tucker, P. A. (2005). No pane, no gain: Efficient evaluation of sliding-window aggregates over data streams. *Acm Sigmod Record*, *34*(1), 39–44.

DOI:10.6814/NCCU202200079

Liao, R., Boonyakunakorn, P., Harnpornchai, N., & Sriboonchitta, S. (2020). Forecasting the exchange rate for USD to RMB using RNN and SVM. *Journal of Physics: Conference Series*, *1616*(1), 12050.

Liu, C., Hu, Z., Li, Y., & Liu, S. (2017). Forecasting copper prices by decision tree learning. *Resources Policy*, *52*, 427–434.

Liu, J., Wu, S., & Zidek, J. V. (1997). On segmented multivariate regressions. *Statistica Sinica*, 497–525.

Liu, Y., Yang, C., Huang, K., & Gui, W. (2020). Non-ferrous metals price forecasting based on variational mode decomposition and LSTM network. *Knowledge-Based Systems*, *188*, 105006.

Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., & Zhang, G. (2018). Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering*, *31*(12), 2346–2363.

Lu, J., Liu, A., Song, Y., & Zhang, G. (2020). Data-driven decision support under concept drift in streamed big data. *Complex & Intelligent Systems*, *6*(1), 157–163.

Mahata, A., Bal, D. P., & Nurujjaman, M. (2020). Identification of short-term and long-term time scales in stock markets and effect of structural break. *Physica A: Statistical Mechanics and Its Applications*, *545*, 123612.

Matsuyama, K. (2008). Structural change. *The New Palgrave Dictionary of Economics*, *2*.

Mezard, M., & Nadal, J. P. (1989). Learning in feedforward layered networks: The tiling algorithm. *Journal of Physics A: Mathematical and General*, *22*(12), 2191.

Morales, L., & Andreosso-O'Callaghan, B. (2011). Comparative analysis on the effects of the Asian and global financial crises on precious metal markets. *Research in International Business and Finance*, *25*(2), 203–227.

DOI:10.6814/NCCU202200079

Mozafari, B., Thakkar, H., & Zaniolo, C. (2008). Verifying and mining frequent patterns from large windows over data streams. *2008 IEEE 24th International Conference on Data Engineering*, 179–188.

Mussagy, I. H., & BigramoAllaro, H. (2016). Structural change in Mozambique: Economic performance before and after the civil war. *Journal of Economic and Sustainable Development*, *7*, 119–125.

Muthuramu, P., & Maheswari, T. U. (2019). Tests for structural breaks in time series analysis: A review of recent development. *Shanlax International Journal of Economics*, *7*(4), 66–79.

Nalepa, J., & Kawulok, M. (2019). Selecting training sets for support vector machines: A review. *Artificial Intelligence Review*, *52*(2), 857–900.

Nasir, M. A., & Vo, X. V. (2020). A quarter century of inflation targeting & structural change in exchange rate pass-through: Evidence from the first three movers. *Structural Change and Economic Dynamics*, *54*, 42–61.

Nielsen, B., & Whitby, A. (2015). A joint chow test for structural instability. *Econometrics*, *3*(1), 156–186.

Nikzad, M., Movagharnejad, K., & Talebnia, F. (2012). Comparative study between neural network model and mathematical models for prediction of glucose concentration during enzymatic hydrolysis. *International Journal of Computer Applications, 56*(1).

Niu, T., Wang, J., Lu, H., Yang, W., & Du, P. (2020). Developing a deep learning framework with two-stage feature selection for multivariate financial time series forecasting. *Expert Systems with Applications*, *148*, 113237.

Ooyen, A. Van, & Nienhuis, B. (1992). Improving the convergence of the back-propagation algorithm. *Neural Netw*, *5*(3), 465–471.

DOI:10.6814/NCCU202200079

Orlowski, L. T. (2017). Volatility of commodity futures prices and market-implied inflation expectations. *Journal of International Financial Markets, Institutions and Money*, *51*, 133–141.

Ozaki, T. (1980). Non-linear time series models for non-linear random vibrations. *Journal of Applied Probability*, *17*(1), 84–93.

Parisi, A., Parisi, F., & D´ıaz, D. (2008). Forecasting gold price changes: Rolling and recursive neural network models. *Journal of Multinational Financial Management*, *18*(5), 477– 487.

Parveen, N., Zaidi, S., & Danish, M. (2017). Support vector regression prediction and analysis of the copper (II) biosorption efficiency. *Indian Chemical Engineer*, *59*(4), 295–311.

Passerini, E. (2000). Disasters as agents of social change in recovery and reconstruction. *Natural Hazards Review*, *1*(2), 67–72.

Perron, P., Yamamoto, Y., & Zhou, J. (2020). Testing jointly for structural changes in the error variance and coefficients of a linear regression model. *Quantitative Economics*, *11*(3), 1019–1057.

Quandt, R. E. (1958). The estimation of the parameters of a linear regression system obeying two separate regimes. *Journal of the American Statistical Association*, *53*(284), 873–880.

Quandt, R. E. (1960). Tests of the hypothesis that a linear regression system obeys two separate regimes. *Journal of the American Statistical Association*, *55*(290), 324–330.

Rossen, A. (2015). What are metal prices like? Co-movement, price cycles and long-run trends. *Resources Policy*, *45*, 255–276.

Sarnovsky, M., & Kolarik, M. (2021). Classification of the drifting data streams using heterogeneous diversified dynamic class-weighted ensemble. *PeerJ Computer Science*, *7*, 459.

DOI:10.6814/NCCU202200079

Schaffer, A. L., Dobbins, T. A., & Pearson, S. A. (2021). Interrupted time series analysis using autoregressive integrated moving average (ARIMA) models: A guide for evaluating large-scale health interventions. *BMC Medical Research Methodology*, *21*(1), 1–12.

Schalkoff, R. J. (2007). Pattern recognition. *Wiley Encyclopedia of Computer Science and Engineering*.

Sharma, R., Pachori, R. B., & Sircar, P. (2020). Seizures classification based on higher order statistics and deep neural network. *Biomedical Signal Processing and Control*, *59*, 101921.

Sharma, R., Saxena, A., & Vagrecha, K. (2015). Supply chain optimization of zinc industry: Opportunities, strategies and challenges. *Global Journal of Enterprise Information System*, *7*(3), 62–70.

Shokry, A., & Espuña, A. (2018). The ordinary kriging in multivariate dynamic modelling and multistep-ahead prediction. *Computer Aided Chemical Engineering*, *43*, 265–270.

Stevenson, S. (2007). A comparison of the forecasting ability of ARIMA models. *Journal of Property Investment & Finance*, *25*(3), 223–240.

Suárez-Cetrulo, A. L., Cervantes, A., & Quintana, D. (2019). Incremental market behavior classification in presence of recurring concepts. *Entropy*, *21*(1), 25.

Talih, M., & Hengartner, N. (2005). Structural learning with time-varying components: Tracking the cross-section of financial time series. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *67*(3), 321–341.

Tong, H. (1977). Some comments on the Canadian lynx data. *Journal of the Royal Statistical Society: Series A (General)*, *140*(4), 432–436.

Tong, H., & Lim, K. S. (2009). Threshold autoregression, limit cycles and cyclical data. *Exploration Of A Nonlinear World: An Appreciation of Howell Tong's Contributions to Statistics*, 9–56.

DOI:10.6814/NCCU202200079

Tsai, Y. H., Jheng, Y. J., & Tsaih, R. H. (2019). The cramming, softening and integrating learning algorithm with parametric ReLU activation function for binary input/output problems. *2019 International Joint Conference on Neural Networks (IJCNN)*, 1–7.

Tsaih, R. H., & Cheng, T. C. (2009). A resistant learning procedure for coping with outliers. *Annals of Mathematics and Artificial Intelligence*, *57*(2), 161–180.

Tsaih, R. R. (1993). The softening learning procedure. *Mathematical and Computer Modelling*, *18*(8), 61–64.

Tsaih, R. R. (1998). An explanation of reasoning neural networks. *Mathematical and Computer Modelling*, *28*(2), 37–44.

Tsymbal, A. (2004). The problem of concept drift: Definitions and related work. *Computer Science Department, Trinity College Dublin*, *106*(2), 58.

Verbesselt, J., Hyndman, R., Newnham, G., & Culvenor, D. (2010). Detecting trend and seasonal changes in satellite image time series. *Remote Sensing of Environment*, *114*(1), 106–115.

Wahab, B. A., & Adewuyi, A. O. (2021). Analysis of major properties of metal prices using new methods: Structural breaks, non-linearity, stationarity and bubbles. *Resources Policy*, *74*, 102284.

Wang, C., Zhang, X., Wang, M., Lim, M. K., & Ghadimi, P. (2019). Predictive analytics of the copper spot price by utilizing complex network and artificial neural network techniques. *Resources Policy*, *63*, 101414.

Wang, H., Fan, W., Yu, P. S., & Han, J. (2003). Mining concept-drifting data streams using ensemble classifiers. *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 226–235.

Wang, J., Lu, S., Wang, S.-H., & Zhang, Y.-D. (2021). A review on extreme learning machine. *Multimedia Tools and Applications 2021*, 1–50.

DOI:10.6814/NCCU202200079

Wang, T., & Wang, C. (2019). The spillover effects of China’s industrial growth on price changes of base metal. *Resources Policy*, *61*, 375–384.

Watanabe, E., & Shimizu, H. (1993). Algorithm for pruning hidden units in multilayered neural network for binary pattern classification problem. *Proceedings of 1993 International Conference on Neural Networks*, 327–330.

Wets, R. J. B., & Rios, I. (2015). Modeling and estimating commodity prices: Copper prices. *Mathematics and Financial Economics*, *9*(4), 247–270.

Yao, Y. C. (1988). Estimating the number of change-points via Schwarz’criterion. *Statistics & Probability Letters*, *6*(3), 181–189.

Yao, Y. C., & Au, S. T. (1989). Least-squares estimation of a step function. *The Indian Journal of Statistics*, 370–381.

Yin, Y. Q. (1988). Detection of the number, locations and magnitudes of jumps. *Communications in Statistics. Stochastic Models*, *4*(3), 445–455.

Yu, L., Wang, S., & Lai, K. K. (2005). A novel nonlinear ensemble forecasting model incorporating GLAR and ANN for foreign exchange rates. *Computers and Operations Research*, *32*(10), 2523–2541.

Zeileis, A., Kleiber, C., Krämer, W., & Hornik, K. (2003). Testing and dating of structural changes in practice. *Computational Statistics & Data Analysis*, *44*(1–2), 109–123.

Zeiler, M. D. (2012). ADADELTA: An adaptive learning rate method. *ArXiv Preprint ArXiv:1212.5701*.

Zhang, G. P. (2003). Time series forecasting using a hybrid ARIMA and neural network model. *Neurocomputing*, *50*, 159–175.

Zhang, H., Nguyen, H., Bui, X.-N., Pradhan, B., Mai, N.-L., & Vu, D.-A. (2021). Proposing two novel hybrid intelligence models for forecasting copper price based on extreme learning machine and meta-heuristic algorithms. *Resources Policy*, *73*, 102195.

DOI:10.6814/NCCU202200079

Zhang, H., Nguyen, H., Vu, D.-A., Bui, X.-N., & Pradhan, B. (2021). Forecasting monthly copper price: A comparative study of various machine learning-based methods. *Resources Policy*, *73*, 102189.

Zheng, Y., Shao, Y., & Wang, S. (2017). The determinants of Chinese nonferrous metals imports and exports. *Resources Policy*, *53*, 238–246.

DOI:10.6814/NCCU202200079

Appendix A - The learning process

Table A1. The learning process of MW-PO-100 version

Wi[a]	FR[b]	TR[c]	CR[d]	Wi	FR	TR	CR	Wi	FR	TR	CR
1	96.86%	0.63%	2.52%	27	88.68%	10.69%	0.63%	53	88.05%	9.43%	2.52%
2	79.87%	4.40%	15.72%	28	71.07%	5.66%	23.27%	54	96.23%	3.77%	0.00%
3	93.71%	1.89%	4.40%	29	79.87%	3.14%	16.98%	55	94.34%	5.66%	0.00%
4	87.42%	1.26%	11.32%	30	78.62%	3.14%	18.24%	56	33.96%	1.26%	64.78%
5	88.05%	2.52%	9.43%	31	89.94%	0.63%	9.43%	57	58.49%	6.92%	34.59%
6	94.34%	1.26%	4.40%	32	93.08%	5.03%	1.89%	58	90.57%	8.81%	0.63%
7	92.45%	1.89%	5.66%	33	80.50%	5.66%	13.84%	59	94.34%	4.40%	1.26%
8	83.65%	3.14%	13.21%	34	79.25%	4.40%	16.35%	60	91.19%	8.18%	0.63%
9	90.57%	3.77%	5.66%	35	55.35%	1.89%	42.77%	61	96.86%	3.14%	0.00%
10	91.82%	0.63%	7.55%	36	93.08%	4.40%	2.52%	62	96.23%	2.52%	1.26%
11	86.79%	3.14%	10.06%	37	88.68%	4.40%	6.92%	63	69.81%	10.69%	19.50%
12	86.79%	1.26%	11.95%	38	92.45%	1.89%	5.66%	64	96.23%	2.52%	1.26%
13	86.79%	0.63%	12.58%	39	89.94%	2.52%	7.55%	65	97.48%	1.26%	1.26%
14	84.28%	1.26%	14.47%	40	93.71%	3.77%	2.52%	66	96.23%	2.52%	1.26%
15	96.86%	3.14%	0.00%	41	94.97%	2.52%	2.52%	67	98.11%	0.63%	1.26%
16	81.13%	3.14%	15.72%	42	96.86%	3.14%	0.00%	68	96.23%	2.52%	1.26%
17	93.71%	1.26%	5.03%	43	94.34%	5.66%	0.00%	69	95.60%	2.52%	1.89%
18	79.87%	1.89%	18.24%	44	67.30%	6.92%	25.79%	70	95.60%	0.63%	3.77%
19	89.31%	3.14%	7.55%	45	86.16%	6.92%	6.92%	71	94.34%	1.26%	4.40%
20	94.34%	4.40%	1.26%	46	93.08%	5.03%	1.89%	72	91.82%	3.77%	4.40%
21	100.00%	0.00%	0.00%	47	98.11%	1.89%	0.00%	73	89.94%	5.66%	4.40%
22	89.94%	3.14%	6.92%	48	98.11%	1.89%	0.00%	74	89.94%	5.66%	4.40%
23	81.13%	3.77%	15.09%	49	96.86%	3.14%	0.00%	75	94.34%	1.26%	4.40%
24	87.42%	4.40%	8.18%	50	94.97%	5.03%	0.00%	76	95.60%	2.52%	1.89%
25	97.48%	2.52%	0.00%	51	89.94%	10.06%	0.00%	77	93.71%	3.14%	3.14%
26	96.23%	2.52%	1.26%	52	30.82%	6.92%	62.26%	78	93.08%	3.14%	3.77%

[a]Window index; [b]*Familiar route*; [c]*Thinking route*; [d]*Cramming route*

DOI:10.6814/NCCU202200079

Table A2. The learning process of MW-LTS-0 version

Wi[a]	FR[b]	TR[c]	CR[d]	Wi	FR	TR	CR	Wi	FR	TR	CR
1	66.67%	33.33%	0.00%	27	73.58%	1.89%	24.53%	53	72.33%	27.04%	0.63%
2	66.04%	32.70%	1.26%	28	93.08%	2.52%	4.40%	54	82.39%	17.61%	0.00%
3	76.73%	22.01%	1.26%	29	93.71%	3.77%	2.52%	55	82.39%	17.61%	0.00%
4	85.53%	9.43%	5.03%	30	93.71%	3.14%	3.14%	56	74.21%	22.01%	3.77%
5	81.76%	6.92%	11.32%	31	96.23%	2.52%	1.26%	57	77.36%	11.32%	11.32%
6	81.76%	8.81%	9.43%	32	87.42%	11.32%	1.26%	58	79.25%	11.32%	9.43%
7	71.70%	6.29%	22.01%	33	89.94%	8.81%	1.26%	59	77.99%	20.75%	1.26%
8	88.68%	4.40%	6.92%	34	91.19%	8.18%	0.63%	60	62.89%	35.85%	1.26%
9	91.82%	7.55%	0.63%	35	94.34%	4.40%	1.26%	61	86.79%	11.95%	1.26%
10	89.31%	10.69%	0.00%	36	94.34%	5.03%	0.63%	62	90.57%	9.43%	0.00%
11	91.82%	8.18%	0.00%	37	94.34%	5.03%	0.63%	63	78.62%	20.13%	1.26%
12	77.99%	7.55%	14.47%	38	88.05%	6.92%	5.03%	64	94.97%	2.52%	2.52%
13	91.19%	7.55%	1.26%	39	85.53%	10.69%	3.77%	65	47.80%	0.63%	51.57%
14	89.31%	10.06%	0.63%	40	86.16%	5.03%	8.81%	66	98.11%	0.63%	1.26%
15	95.60%	4.40%	0.00%	41	88.05%	10.69%	1.26%	67	98.11%	1.26%	0.63%
16	86.16%	3.14%	10.69%	42	96.86%	3.14%	0.00%	68	94.97%	3.77%	1.26%
17	95.60%	4.40%	0.00%	43	66.67%	32.08%	1.26%	69	97.48%	0.63%	1.89%
18	85.53%	13.84%	0.63%	44	84.91%	14.47%	0.63%	70	71.07%	28.30%	0.63%
19	71.07%	28.30%	0.63%	45	94.97%	3.77%	1.26%	71	91.19%	7.55%	1.26%
20	94.97%	4.40%	0.63%	46	68.55%	31.45%	0.00%	72	89.94%	8.81%	1.26%
21	81.76%	18.24%	0.00%	47	84.28%	15.72%	0.00%	73	84.28%	11.32%	4.40%
22	77.36%	13.84%	8.81%	48	79.25%	5.03%	15.72%	74	79.87%	18.87%	1.26%
23	69.81%	30.19%	0.00%	49	77.36%	8.81%	13.84%	75	94.97%	3.77%	1.26%
24	93.71%	6.29%	0.00%	50	68.55%	2.52%	28.93%	76	94.97%	3.77%	1.26%
25	88.68%	11.32%	0.00%	51	75.47%	24.53%	0.00%	77	95.60%	3.77%	0.63%
26	94.34%	1.89%	3.77%	52	64.78%	27.67%	7.55%	78	89.94%	8.18%	1.89%

[a]Window index; [b]*Familiar route*; [c]*Thinking route*; [d]*Cramming route*

DOI:10.6814/NCCU202200079

Table A3. The learning process of MW-LTS-100 version

Wi[a]	FR[b]	TR[c]	CR[d]	Wi	FR	TR	CR	Wi	FR	TR	CR
1	99.37%	0.00%	0.63%	27	99.37%	0.63%	0.00%	53	95.60%	3.14%	1.26%
2	98.74%	0.00%	1.26%	28	96.86%	3.14%	0.00%	54	98.11%	1.89%	0.00%
3	97.48%	1.26%	1.26%	29	96.86%	0.00%	3.14%	55	97.48%	2.52%	0.00%
4	84.28%	1.26%	14.47%	30	98.74%	1.26%	0.00%	56	94.34%	1.26%	4.40%
5	94.34%	0.00%	5.66%	31	96.23%	0.63%	3.14%	57	88.05%	0.00%	11.95%
6	93.71%	5.03%	1.26%	32	96.23%	0.63%	3.14%	58	89.94%	0.00%	10.06%
7	89.31%	0.63%	10.06%	33	96.23%	0.63%	3.14%	59	96.86%	1.26%	1.89%
8	97.48%	1.26%	1.26%	34	96.23%	0.63%	3.14%	60	97.48%	1.89%	0.63%
9	99.37%	0.00%	0.63%	35	92.45%	2.52%	5.03%	61	95.60%	3.77%	0.63%
10	98.74%	0.63%	0.63%	36	96.23%	0.63%	3.14%	62	98.11%	1.89%	0.00%
11	98.74%	1.26%	0.00%	37	96.23%	0.63%	3.14%	63	98.74%	0.00%	1.26%
12	77.99%	0.63%	21.38%	38	96.23%	0.63%	3.14%	64	97.48%	0.63%	1.89%
13	99.37%	0.00%	0.63%	39	96.23%	1.26%	2.52%	65	47.80%	0.63%	51.57%
14	99.37%	0.00%	0.63%	40	96.86%	0.00%	3.14%	66	98.74%	0.63%	0.63%
15	100.00%	0.00%	0.00%	41	96.86%	0.63%	2.52%	67	98.74%	0.63%	0.63%
16	56.60%	1.89%	41.51%	42	100.00%	0.00%	0.00%	68	52.20%	0.63%	47.17%
17	100.00%	0.00%	0.00%	43	98.74%	0.63%	0.63%	69	97.48%	1.26%	1.26%
18	95.60%	3.77%	0.63%	44	94.97%	3.14%	1.89%	70	96.86%	2.52%	0.63%
19	89.94%	8.81%	1.26%	45	92.45%	3.14%	4.40%	71	99.37%	0.00%	0.63%
20	99.37%	0.00%	0.63%	46	100.00%	0.00%	0.00%	72	96.23%	0.00%	3.77%
21	100.00%	0.00%	0.00%	47	100.00%	0.00%	0.00%	73	96.86%	1.89%	1.26%
22	99.37%	0.63%	0.00%	48	100.00%	0.00%	0.00%	74	98.11%	0.00%	1.89%
23	93.71%	6.29%	0.00%	49	77.36%	3.77%	18.87%	75	98.74%	0.00%	1.26%
24	100.00%	0.00%	0.00%	50	94.97%	4.40%	0.63%	76	98.74%	0.00%	1.26%
25	96.23%	3.77%	0.00%	51	97.48%	2.52%	0.00%	77	98.11%	0.63%	1.26%
26	94.97%	3.14%	1.89%	52	93.71%	3.14%	3.14%	78	98.11%	0.63%	1.26%

[a]Window index; [b]*Familiar route*; [c]*Thinking route*; [d]*Cramming route*

DOI:10.6814/NCCU202200079

Table A4. The learning process of MW-LTS-500 version

Wi[a]	FR[b]	TR[c]	CR[d]	Wi	FR	TR	CR	Wi	FR	TR	CR
1	99.37%	0.00%	0.63%	27	100.00%	0.00%	0.00%	53	93.08%	2.52%	4.40%
2	99.37%	0.00%	0.63%	28	100.00%	0.00%	0.00%	54	98.74%	1.26%	0.00%
3	96.86%	1.26%	1.89%	29	100.00%	0.00%	0.00%	55	100.00%	0.00%	0.00%
4	91.82%	1.26%	6.92%	30	100.00%	0.00%	0.00%	56	95.60%	0.00%	4.40%
5	95.60%	0.00%	4.40%	31	99.37%	0.63%	0.00%	57	98.11%	1.26%	0.63%
6	94.34%	5.03%	0.63%	32	98.74%	1.26%	0.00%	58	98.11%	1.89%	0.00%
7	96.86%	0.63%	2.52%	33	100.00%	0.00%	0.00%	59	98.11%	0.63%	1.26%
8	98.74%	0.63%	0.63%	34	96.23%	0.63%	3.14%	60	96.86%	2.52%	0.63%
9	99.37%	0.00%	0.63%	35	93.71%	2.52%	3.77%	61	96.23%	0.00%	3.77%
10	99.37%	0.63%	0.00%	36	96.23%	0.63%	3.14%	62	97.48%	0.00%	2.52%
11	100.00%	0.00%	0.00%	37	96.23%	0.63%	3.14%	63	98.74%	0.00%	1.26%
12	77.36%	1.26%	21.38%	38	96.23%	0.63%	3.14%	64	98.11%	0.00%	1.89%
13	99.37%	0.00%	0.63%	39	95.60%	1.26%	3.14%	65	96.86%	2.52%	0.63%
14	99.37%	0.00%	0.63%	40	97.48%	0.00%	2.52%	66	98.11%	0.63%	1.26%
15	100.00%	0.00%	0.00%	41	97.48%	0.00%	2.52%	67	98.11%	1.26%	0.63%
16	94.97%	5.03%	0.00%	42	100.00%	0.00%	0.00%	68	97.48%	1.26%	1.26%
17	99.37%	0.00%	0.63%	43	99.37%	0.00%	0.63%	69	98.74%	0.00%	1.26%
18	97.48%	1.26%	1.26%	44	98.11%	1.26%	0.63%	70	96.86%	2.52%	0.63%
19	99.37%	0.00%	0.63%	45	96.23%	2.52%	1.26%	71	98.11%	0.00%	1.89%
20	99.37%	0.00%	0.63%	46	100.00%	0.00%	0.00%	72	96.23%	0.00%	3.77%
21	100.00%	0.00%	0.00%	47	100.00%	0.00%	0.00%	73	98.11%	0.63%	1.26%
22	100.00%	0.00%	0.00%	48	99.37%	0.63%	0.00%	74	98.74%	0.00%	1.26%
23	96.86%	1.89%	1.26%	49	79.87%	1.89%	18.24%	75	98.11%	0.63%	1.26%
24	100.00%	0.00%	0.00%	50	97.48%	2.52%	0.00%	76	98.11%	0.63%	1.26%
25	100.00%	0.00%	0.00%	51	98.11%	1.89%	0.00%	77	98.74%	0.00%	1.26%
26	98.11%	0.63%	1.26%	52	92.45%	3.14%	4.40%	78	98.74%	0.00%	1.26%

[a]Window index; [b]*Familiar route*; [c]*Thinking route*; [d]*Cramming route*

DOI:10.6814/NCCU202200079

Appendix B - The number of adopted hidden nodes

Table B1. The number of adopted hidden nodes

Wi[a]	V1[b]	V2[c]	V3[d]	V4[e]	Wi	V1	V2	V3	V4	Wi	V1	V2	V3	V4
1	12	1	2	2	27	4	56	1	1	53	10	3	7	21
2	72	7	2	2	28	99	15	1	1	54	1	1	1	1
3	21	7	4	4	29	81	6	15	1	55	1	1	1	1
4	54	9	63	30	30	87	9	1	1	56	286	18	21	21
5	45	48	28	21	31	45	1	15	1	57	151	51	57	4
6	22	46	4	2	32	9	3	15	1	58	4	42	48	1
7	27	102	39	6	33	57	6	15	1	59	6	6	9	6
8	54	33	7	4	34	77	1	15	15	60	4	6	4	4
9	27	2	2	2	35	201	3	15	9	61	1	6	3	18
10	36	1	2	1	36	9	1	15	15	62	6	1	1	12
11	48	1	1	1	37	32	1	15	15	63	94	6	6	6
12	51	63	102	102	38	23	18	15	15	64	6	12	9	9
13	60	6	4	2	39	27	18	12	15	65	6	245	109	4
14	69	3	2	2	40	9	39	15	12	66	6	1	1	1
15	1	1	1	1	41	9	3	12	12	67	6	1	1	1
16	76	48	197	1	42	1	1	1	1	68	6	1	115	3
17	14	1	1	2	43	1	7	4	4	69	9	9	6	6
18	78	4	4	4	44	96	3	9	3	70	18	3	3	3
19	27	3	6	4	45	30	6	21	7	71	21	3	3	3
20	6	3	3	3	46	7	1	1	1	72	21	3	12	12
21	1	1	1	1	47	1	1	1	1	73	21	18	3	3
22	27	42	1	1	48	1	60	1	1	74	21	3	3	6
23	69	1	1	2	49	1	60	80	78	75	21	6	6	6
24	30	1	1	1	50	1	120	3	1	76	9	6	6	6
25	1	1	1	1	51	1	1	1	1	77	15	3	6	6
26	6	15	9	6	52	273	37	15	21	78	18	3	4	6

[a]Window index; [b]MW-PO-100; [c]MW-LTS-0; [d]MW-LTS-100; [e]MW-LTS-500

DOI:10.6814/NCCU202200079

Appendix C - The training time

Table C1. The training time (Units: second)

Wi[a]	V1[b]	V2[c]	V3[d]	V4[e]	Wi	V1	V2	V3	V4	Wi	V1	V2	V3	V4
1	672.46	14.71	78.60	214.25	27	53.18	2411.46	55.42	164.16	53	2287.99	99.98	118.34	861.97
2	5909.43	38.19	104.62	213.66	28	5955.66	911.81	56.05	164.70	54	40.69	21.85	47.98	195.54
3	209.72	115.99	140.93	315.79	29	4135.71	275.33	194.52	165.19	55	42.00	22.45	50.05	196.14
4	2054.24	346.61	1152.03	512.59	30	6204.83	552.18	54.70	163.66	56	19790.97	393.72	425.47	642.05
5	1795.54	808.95	358.19	459.95	31	3246.61	53.50	261.43	163.84	57	8745.53	945.57	932.80	257.90
6	508.86	449.73	143.91	256.80	32	1046.85	184.12	290.65	161.73	58	109.53	647.54	718.35	218.37
7	1312.93	2619.47	539.39	336.97	33	2710.87	220.27	319.57	164.08	59	1270.42	588.74	263.40	453.25
8	3360.70	520.48	88.62	225.70	34	4104.14	26.89	306.31	432.27	60	40.65	170.32	78.18	232.48
9	1444.45	41.47	77.96	221.68	35	17492.62	233.66	515.27	691.64	61	40.47	65.66	93.26	405.30
10	1160.07	11.86	65.15	183.92	36	919.65	30.16	389.00	497.47	62	968.40	14.56	48.41	320.51
11	860.64	25.53	51.81	183.63	37	3196.84	29.76	420.48	533.76	63	2112.23	101.42	131.61	313.20
12	1838.00	1357.84	2884.57	3150.11	38	2691.35	312.84	446.97	550.89	64	1193.85	118.01	102.99	253.02
13	1337.79	38.24	80.44	220.42	39	1705.24	323.98	461.49	556.68	65	747.36	9912.40	7685.57	215.18
14	2245.49	81.50	77.42	223.06	40	563.63	575.14	507.90	597.91	66	804.50	62.14	73.39	246.52
15	43.75	12.34	48.14	192.85	41	612.80	164.64	367.21	684.63	67	855.21	36.56	73.02	202.04
16	2894.16	1926.41	7500.54	211.68	42	35.31	10.37	45.92	174.10	68	873.06	52.26	7142.41	247.12
17	717.73	14.38	49.78	223.76	43	37.63	55.73	76.31	213.34	69	894.25	323.60	77.31	193.49
18	3994.10	51.83	90.96	261.05	44	3422.69	108.75	135.66	253.50	70	973.40	56.09	105.24	226.97
19	1060.46	54.70	103.91	228.13	45	2182.99	199.80	264.74	245.56	71	551.95	90.03	172.64	324.39
20	300.97	182.20	81.95	231.63	46	147.95	15.87	45.29	199.13	72	299.88	91.76	189.34	357.82
21	38.89	12.31	51.68	192.31	47	42.88	12.75	46.33	196.39	73	386.00	167.64	128.58	268.94
22	1244.70	1702.86	69.43	193.97	48	39.42	2001.34	46.01	197.44	74	454.78	93.99	149.30	245.42
23	3350.21	63.24	91.06	253.68	49	37.91	2591.34	2328.15	2601.12	75	636.12	280.76	299.28	209.93
24	1993.89	12.01	49.97	184.40	50	39.34	6076.16	321.97	230.95	76	378.94	269.26	84.28	216.81
25	49.80	15.53	49.45	168.62	51	41.14	17.65	55.37	209.84	77	647.40	250.76	85.24	218.09
26	85.41	542.62	102.28	209.01	52	17307.20	365.15	168.92	912.26	78	842.42	66.03	114.40	235.07

[a]Window index; [b]MW-PO-100; [c]MW-LTS-0; [d]MW-LTS-100; [e]MW-LTS-500

DOI:10.6814/NCCU202200079

Appendix D - The number of hidden nodes deleted by reorganizing module

Table D1. The reorganizing module executions

Wi[a]	V1[b]	V2[c]	V3[d]	V4[e]	Wi	V1	V2	V3	V4	Wi	V1	V2	V3	V4
1	1	0	2	2	27	5	28	8	5	53	275	37	14	21
2	15	0	6	3	28	16	59	0	0	54	9	2	6	20
3	72	6	4	7	29	99	18	1	0	55	0	0	0	0
4	21	22	7	7	30	81	9	14	0	56	24	1	1	1
5	54	12	62	30	31	87	14	1	0	57	300	21	21	20
6	44	47	30	22	32	45	4	15	0	58	150	54	57	3
7	22	49	13	8	33	12	3	15	0	59	4	42	48	1
8	36	102	38	5	34	58	8	15	1	60	5	6	8	5
9	54	34	8	5	35	80	4	24	24	61	3	6	4	4
10	27	1	3	1	36	204	5	15	9	62	1	5	2	18
11	36	0	1	0	37	10	3	15	15	63	5	1	1	12
12	54	4	1	1	38	36	4	15	15	64	94	6	6	6
13	51	63	101	103	39	26	18	15	15	65	6	13	8	8
14	60	6	5	3	40	30	21	12	15	66	6	250	111	9
15	68	2	1	1	41	12	42	15	12	67	6	3	3	3
16	0	1	2	0	42	8	2	11	11	68	6	6	0	4
17	86	47	196	2	43	0	0	0	0	69	6	1	115	3
18	23	0	0	4	44	28	7	4	4	70	9	9	6	6
19	87	4	4	3	45	99	3	9	2	71	18	6	3	9
20	27	3	6	4	46	32	5	20	6	72	21	6	9	9
21	5	2	2	2	47	6	0	0	0	73	21	6	15	15
22	7	1	0	0	48	0	1	0	0	74	21	21	9	3
23	30	41	0	5	49	0	63	2	1	75	21	3	3	6
24	78	0	0	1	50	0	63	80	77	76	21	6	6	6
25	29	0	0	0	51	0	119	2	0	77	9	6	6	6
26	1	1	1	1	52	25	0	1	1	78	15	9	8	6

[a]Window index; [b]MW-PO-100; [c]MW-LTS-0; [d]MW-LTS-100; [e]MW-LTS-500

DOI:10.6814/NCCU202200079