國立政治大學資訊管理研究所

碩士學位論文

以序列型學習演算法預測牛市與熊市之轉折點

The Sequentially-Learning-based Algorithm and the Prediction of the Turning Points of Bull and Bear markets

指導教授：蔡 瑞 煌 博士、盧 敬 植 博士

研究生：張 瑄 芸 撰

中 華 民 國 一〇八 年 七月

DOI:10.6814/NCCU201900325

以序列型學習演算法預測牛市與熊市之轉折點

摘要

在機器學習領域中，對於人工神經網絡（ANN）的架構中，輸入值是實數，輸出值是二元的問題是有挑戰性的，尚未有任何神經網路學習演算法可以解決過度擬合（overfitting）問題，同時可以完美地學習所有訓練數據；另外，在概念飄移環境中要如何去處理離群值的偵測也變得重要，現今的很多資料多為動態性且具概念漂移的特性。

為了解決上述挑戰，本研究提出了 DSM（決策支持機制）和 CSI（強記、軟化、整合）學習演算法。決策支持機制運用了移動視窗的概念，不僅可以識別牛市/熊市中的潛在轉折點檢測，還可以幫助決策者檢視所有轉折點候選者。所提出的 CSI 學習算法具有以下特點：

（1）採用單層隱藏層的神經網絡（ASLFN）和 ReLU 激活函數；

（2）採用 LTS 原理加速訓練時間；

（3）完美的學習所有訓練的數據；

（4）實行正則化（regularization），軟化和整合機制，以減輕過度擬合趨勢的模型。

我們進行了檢測牛市/熊市轉折點的實驗，以驗證所提出的演算法具有有效性和效率，以及偵測出轉折點候選人幫助決策者作出最終決定。

關鍵字：概念飄移、離群值偵測、人工神經網路、決策支援機制、移動視窗

DOI:10.6814/NCCU201900325

The sequentially-learning-based algorithm and the prediction of the turning points of bull and bear markets

Abstract

In the field of machine learning, there is a challenge to the Artificial Neural Networks (ANN) application whose input values are real numbers and the output values are binary. Whether any of ANN learning algorithms can solve the overfitting problem, while it can perfectly learn all of training data. Besides, the problem of outlier detection in the concept environment is becoming an issue. The nature data now has the dynamic and unstable property in the concept drifting environment.

To address the aforementioned challenge, this study proposes the DSM (Decision Support Mechanism) and CSI (Cramming, Softening, and Integrating) learning algorithm. DSM apply the moving window mechanism, and it can not only identify the potential turning point detection in the bull/ bear market but also assist the decision maker to double check merely all of turning point candidates. The proposed CSI learning algorithm has the following features: (1) the adoption of adaptive single-hidden layer feed-forward neural network (ASLFN) and ReLU activation function, (2) the usage of least trimmed squares (LTS) principle to speed up the training time, (3) the practice to precisely learn all training data, and (4) the implementations of the regularization term, the softening and integrating mechanism to alleviate the obtained model from the overfitting tendency. We conduct an experiment of detecting the turning points of bull/bear markets to validate the effectiveness and efficiency of the proposed algorithm in the addressing challenge.

Keywords: concept drifting, outlier detection, artificial neural network, cramming mechanism, softening mechanism, integrating mechanism, moving window.

DOI:10.6814/NCCU201900325

Index

DOI:10.6814/NCCU201900325

Table Index

DOI:10.6814/NCCU201900325

Figure Index

DOI:10.6814/NCCU201900325

1. Introduction

Nowadays, artificial intelligence (AI) is booming and leads the technical revolution in decision making of financial and industrial fields. Financial firms worldwide begin to employing artificial neural networks (ANN) to tackle difficult tasks requiring the detection of data patterns which cannot be done by conventional analytic techniques. Unlike other types of AI, ANN mimic to some extent the processing characteristics of the human brain. As a result, ANN can draw conclusions from incomplete data, recognize patterns as they unfold in real time and forecast the future. They can even learn from mistakes [1]. Many observers believe the ANN based system will eventually provide better decision support to even the best traders and investors.

There is a hybrid approach based on the ANN for time series properties, such as the adaptive time delay neural networks (ATNNs) and the time delay neural networks (TDNNs), with the genetic algorithms (GAs) in detecting temporal patterns for stock market prediction tasks [2]. To estimate many aspects of the ATNN and TDNN design, Kim et al. [2] propose a general method based on trial and error along with various heuristics or statistical techniques is proposed. The results show that the accuracy of the integrated approach is higher than that of the standard ATNN, TDNN and the recurrent neural network (RNN). The disadvantages of [2] are the computational complexity to obtain optimal sets of the number of time delays and network architectural factors at the same time, and the infinite number of possible combinations with control variables that can generate a lot of groups for the general result.

Moreover, in the big data era, there is a challenge to the ANN application whose input values are real numbers and the output values are binary. Whether any of ANN learning algorithms can deal with the overfitting tendency, while it can perfectly learn all of training data. To address the aforementioned challenge, the proposed DSM (Decision Support Mechanism) can assist the decision to double check merely all of turning point candidates. This study introduces a sequentially-learning-based algorithm, called CSI (Cramming, Softening, and Integrating) learning algorithm. That is, the proposed CSI learning algorithm should have the following factors: (1) the adoption of adaptive single-hidden layer feedforward neural network (ASLFN) and ReLU activation function, (2) the usage of least trimmed squares (LTS) principle to speed up the training time, (3) the practice to precisely

DOI:10.6814/NCCU201900325

learn all training data, and (4) the implementations of the regularization term, the softening and integrating mechanism to alleviate the overfitting tendency of the obtained model.

Stock market price is one of the most important indicators of a country's economic growth. That's why determining the exact movements of stock market price is considerably regarded. However, complex and uncertain behaviors of stock market make exact determination impossible and hence strong forecasting models are deeply desirable for investors' financial decision making process. To capture the relationship between the technical indicators and the stock market for the period under investigation, hybrid ANN models are used for selecting the most relevant technical indicators [3]. In addition, [3] simultaneously searches the most appropriate number of hidden neurons in hidden layer and in this respect; proposed models mitigate well-known problem of over-fitting/under-fitting of ANN. But there are two limitations of the model, one is the hidden layer is fixed that may influence the performance of the model. The other limitation is that combinations of predetermined training function and transfer functions may affect quality of ANN.

This study conducts an experiment of detecting the turning points of bull/bear markets to validate the effectiveness and efficiency of the proposed DSM and CSI learning algorithm in addressing the challenge. To increase the computation power and effectively reduce the training time, TensorFlow and GPU are adopted to implement the proposed algorithm. TensorFlow and GPU can speed up the training and enhance the learning performance. TensorFlow is an open source software library for high performance numerical computation. Its flexible architecture allows easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and it comes with strong support for machine learning and deep learning and the flexible numerical computation core is used across many other scientific domains. The experiment results are promising.

The paper is structured as follows. Chapter 2 presents the literature review of previous studies and related algorithms. Chapter 3 describes the learning algorithm in detail. Finally, the experiment design is provided in Chapter 4. The experiment results are shown in Chapter 5. Finally, the conclusion and future work is provided in Chapter 6.

DOI:10.6814/NCCU201900325

2. Literature Review

2.1 The prediction of bull and bear markets

Numerous market analysts propose the strategies to forecast the trending in the stock market price, namely the bull and bear markets. According to several studies in the prediction of the stock market, the popular research tools include statistical models, supervised learning, unsupervised learning, the reinforcement learning, and so on.

Macchiarulo [4] compares machine learning (ML) and technical analysis to explore which one best predicts the stock market and in turn generates the highest return. He concludes that forming a trading strategy with ML can yield higher returns that using most technical indicators. ML performs better in up markets because it uses momentum to its advantage and calculate the optimal weights of the portfolio in the market well predicted of the future direction. However, his sample period is only ten years long and the down market has only 48 observations for training the system. The under-represented bear market makes the predictability of the stock return less convincing in the market downturn.

Chong et al. [5] offer a systematic analysis of the use of deep learning networks for stock market analysis and prediction. Its ability to extract features from a large set of raw data without relying on prior knowledge of predictors makes deep learning potentially attractive for stock market prediction at high frequencies. However, if there exist factors with strong evidence of predictability, exploiting those factors may likely give better performance than simply dumping a huge volume of raw data into the system.

Text-mining provides another interesting thread of methodology on market forecast. Nassirtoussi et al. [6] conduct a thorough literature review on different ML algorithms used in text-mining for market prediction. Researchers use these tools to detect investor sentiment from social media and predict market returns with the information. Their study divides the prediction process into three aspects: pre-processing, ML and the evaluation mechanism, and discusses the development and prospects of future study in each area.

Hanna [7] proposes a top-down approach to identify the bull and bear market states and highlights potential deficiencies in existing ex-post rule-based methodologies and proposes adjustments to address such issues. While early work was inspired by the treatment of business cycles, a principle-based approach is adopted specifically for the treatment of

DOI:10.6814/NCCU201900325

bull and bear market phases. His approach focuses on identifying low-frequency, long-term trends where wide agreement between market observers would be expected.

Pagan and Sossounov [8] use an algorithm to sort a given time series of equity prices into periods that can be designated as bull and bear markets. They define the idea of local peaks and troughs in asset prices and then observe that the proposed definitions mean that the characteristics of such markets come from the stochastic process driving capital gains.

Chen [9] investigates whether macroeconomic variables can predict the bear markets. He uses both parametric and non-parametric approaches to identify the market trends, then ex-amines macroeconomic variables to see whether they are useful predictors. Chen [10] pro-poses the theory whether the empirical linkages between stock returns and trading volume differ over the fluctuations of stock markets, for example, whether the return–volume relation is asymmetric in bull and bear stock markets. It is found that the stock return is capable of predicting trading volume in both bear and bull markets. Therefore, we can adopt these potential factors into our experiment design to forecast the market trend.

In this paper, the information of both macroeconomic variables and historical performances of the stock market itself are fed into an ANN to predict the future market trends. We present a brand-new learning algorithm to perfectly learn thorough training cases and implement the practical experiment in the prediction of turning points of bull and bear markets.

2.2 Concept drifting

In the real world, concepts are often not stable but change with time. Typical examples of this are weather prediction rules and customers' preferences. The underlying data distribution may change as well. Often these changes make the model built on old data inconsistent with the new data, and regular updating of the model is necessary. The definition of concept drifting proposed by Tsymbal [11] is that it complicates the task of learning a model from data and requires special approaches, different from commonly used techniques, which treat arriving instances as equally important contributors to the final concept.

In order to build model in concept drifting environment, Elwell & Polikar [12] introduce

DOI:10.6814/NCCU201900325

the ensemble of classifiers-based approach for incremental learning of concept drift, characterized by nonstationary environments (NSEs), where the underlying data distributions change over time. Krawczyk & Woźniak [13] develop efficient classifiers to cope with the concept drifting environment. They come up with several different strategies for incremental learning and forgetting mechanism. In addition, they evaluate forgetting mechanism on the basis of several real data streams. Obtained results confirmed the usability of proposed classifier to the problem of data stream classification with the presence of concept drift.

2.3 The single-hidden layer feed-forward neural networks with single output node

The single-hidden layer feed-forward neural networks (SLFN) with single output node is defined as follows:

$$a_i^c \equiv ReLU(w_{i0}^H + \sum_{j=1}^{m} w_{ij}^H x_j^c) \tag{1}$$

$$f(\mathbf{x}^c, \mathbf{w}) \equiv w_0^o + \sum_{i=1}^{p} w_i^o a_i^c \tag{2}$$

where a_i^c is the activation value of i^{th} hidden node; $ReLU(x_i) \equiv \begin{cases} x_i, \text{ if } x_i \geq 0 \\ 0, \text{ if } x_i < 0 \end{cases}$; m is the amount of input nodes; $\mathbf{x}^c \equiv (x_1^c, x_2^c, \ldots, x_m^c)^{\mathrm{T}}$ is the c^{th} input vector; w_{i0}^H is the threshold value of the i^{th} hidden node; w_{ij}^H is the weight between the i^{th} hidden node and the j^{th} input node; m is the amount of input nodes; p is the amount of hidden nodes; $f(\mathbf{x}^c, \mathbf{w})$ is the activation value of the output node; w_0^o is the threshold value of the output node; w_i^o is the weight between the output node and the i^{th} hidden node; $\mathbf{w}_i^H \equiv (w_{i0}^H, w_{i1}^H, w_{i2}^H, \ldots, w_{im}^H)^{\mathrm{T}}$; $\mathbf{w}^H \equiv (\mathbf{w}_1^{H^T}, \mathbf{w}_2^{H^T}, \ldots, \mathbf{w}_{\mathrm{p}}^{H^T})^{\mathrm{T}}$; $\mathbf{w}^o \equiv (\mathrm{w}_0^o, \mathrm{w}_1^o, \mathrm{w}_2^o, \ldots, \mathrm{w}_{\mathrm{p}}^o)^{\mathrm{T}}$; and $\mathbf{w}^{\mathbf{T}} \equiv (\mathbf{w}^{o\mathrm{T}}, \mathbf{w}^{H\mathrm{T}})$. The superscript H refers to quantities related to the hidden layer throughout the paper; the superscript o refers to quantities related to the output layer throughout the paper. In this study, a character with bold format represents a vector, a matrix, or a set, and the superscript T means transpose.

DOI:10.6814/NCCU201900325

Through this SLFN, the input vector x^c is transformed into the values of hidden nodes $\mathbf{a}^c \equiv (a_1^c, a_2^c, \ldots, a_p^c)$ first, then the value of output node $f(\mathbf{x}^c, \mathbf{w})$ is generated by $\mathbf{a}^c$ [14][15][16][17][18].

2.4 The back-propagation learning algorithm associated with SLFN with single output node

Let N denote the total amount of reference observations for the training and y^c the desired output value of the output node corresponding to the c^{th} case. The following loss function is adopted:

$$E_n(\mathbf{w}) \equiv \frac{\sum_{c=1}^{n}(e^c)^2}{n} + \frac{0.001}{\mathrm{p+1+(m+1)p}}\left(\sum_{i=0}^{\mathrm{p}}(w_i^{\mathrm{o}})^2 + \sum_{i=1}^{\mathrm{p}}\sum_{j=0}^{\mathrm{m}}\left(w_{ij}^{\mathrm{H}}\right)^2\right) \quad (3)$$

where $e^c \equiv f(\mathbf{x}^c, \mathbf{w}) - y^c$ and there is a regularization term.

In general, the generalized delta rule is used in the weight-tuning mechanism for the SLFN. For instance, Figure 1 shows the weight-tuning mechanism implementing the momentum version of the generalized delta rule with an automatic adjustment of learning rate η and γ and tiny ε_2 and ε_3 values. [14] But it leads to either an acceptable SLFN or an unacceptable SLFN due to the defective SLFN or the convergence to local minimum solution. In Figure 1, the A and B represent the SLFN is acceptable and unacceptable, respectively. Within the weight-tuning procedure, we can repeatedly inspect if the new SLFN is better than the old SLFN. The design of tiny ε_2 and ε_3 values can help identify if the current SLFN is defective or the convergence leads to a local minimum of a good SLFN or a global minimum of a defective SLFN.

DOI:10.6814/NCCU201900325

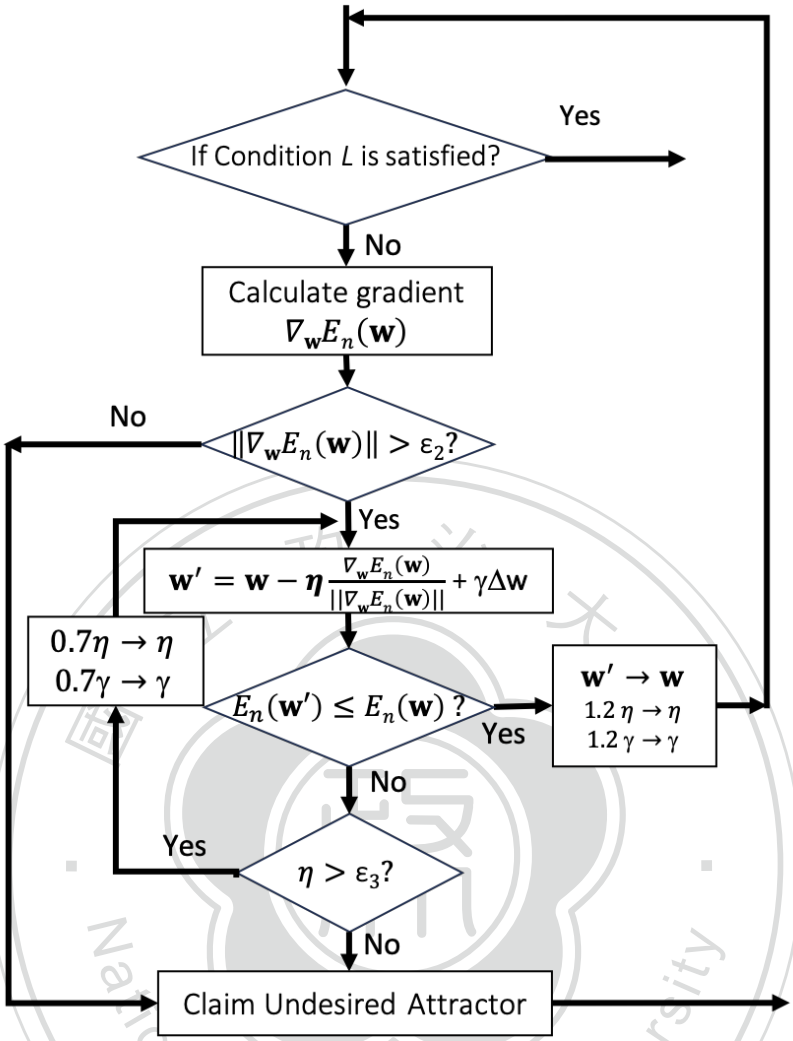

Figure 1: The momentum version of the generalized delta rule. [14]

2.5 The adaptive single-hidden layer feed-forward neural networks with single output node

When *p*, the amount of adopted hidden nodes, is variable during the training process, the SLFN becomes an adaptive single-hidden layer feed-forward neural networks (ASLFN).

Tsaih and Cheng [16] propose the resistant learning outlier detection that SLFN can adapt the weight dynamically during training. Besides, they also come up with both robustness analysis and deletion diagnostics. The ideas of robustness analysis is proposed by Rousseeuw and Van Driessen [19] features for deriving an (initial) subset of m+1 reference observations to fit the linear regression model, ordering the residuals of all N observations

DOI:10.6814/NCCU201900325

at each stage and then augmenting the reference subset gradually based upon the smallest trimmed sum of squared residuals principle. In deletion diagnostics section, this idea is employed with the diagnostic quantity as the number of pruned hidden nodes when one observation is excluded from the reference pool. That means the ASLFN will exclude the potential outlier at early stage prevent the SLFN from learning it.

2.6 Least Trimmed Squares estimator

In the literature [16] of the LTS estimator, if $\hat{\mathbf{w}}$ denotes any estimate of w, then least squares estimator (LSE) is defined to the $\hat{w}$ that minimizes $\sum_{c=1}^{N} (e^c)^2$, in which

$$e^c = y^c - f(\mathbf{x}^c, \mathbf{w}) \tag{4}$$

The generalized delta rule proposed by Rumelhart et al. [20] is a kind of nonlinear LSE. If only the smallest q of those ordered squared residuals are included in the summation, then the LTS estimator is defined as the estimate $\hat{w}$ that minimizes $\sum_{c=1}^{q} (e^{[c]})^2$, where e^c is defined in equation (4) and $(e^{[c]})^2$ denotes the ordered squared residuals; that is, $(e^{[1]})^2 \leq (e^{[2]})^2 \leq \ldots \leq (e^{[N]})^2$. Zaman et al. [21] suggest that $\lfloor 0.75N \rfloor$ is a reasonable value for q in most empirical studies, in which $\lfloor x \rfloor$ is the largest integer not larger than the value x.

Atkinson and Cheng [22] adapt the forward search algorithm to obtain the LTS estimator. The forward search algorithm consists of randomly choosing an (initial) subset of observations to fit the linear regression model, ordering the residuals of all N observations, and then augmenting the subset gradually by including extra observations based upon the smallest sum of q squared residuals principle.

2.7 Moving Windows

Gama et al. [23] propose the approach of the moving windows to data management is to maintain a predictive model consistent with a set of recent examples. The data structure store as the type of first-in-first-out (FIFO). At each time step, the learning algorithm builds a new model using the examples from the training window. The model is updated following two processes: a learning process (update the model based on the new data) and a forgetting

DOI:10.6814/NCCU201900325

process (discard data that is moving out of the window). The key challenge is to select an appropriate window size. A short window reflects the current distribution more accurately; thus, it can ensure fast adaptation in times with concept changes, but during stable periods, a too short window worsens the performance of the system. A large window gives a better performance in stable periods, but it reacts to concept changes more slowly. In general, the training window size can be fixed or variable over time. In our experiment design, we store in memory a fixed number of the training data. Whenever a new example arrives, it is saved to memory and the oldest one is discarded. This simple adaptive learning method is often used as a baseline in evaluation of new algorithms.

Figure 2 presents the conceptual moving window mechanism [24]. Due to the FIFO technique, it will dispose of the oldest data as time goes by. This mechanism will reflect in discarding the out-of-date data and retain the up-to-date data.

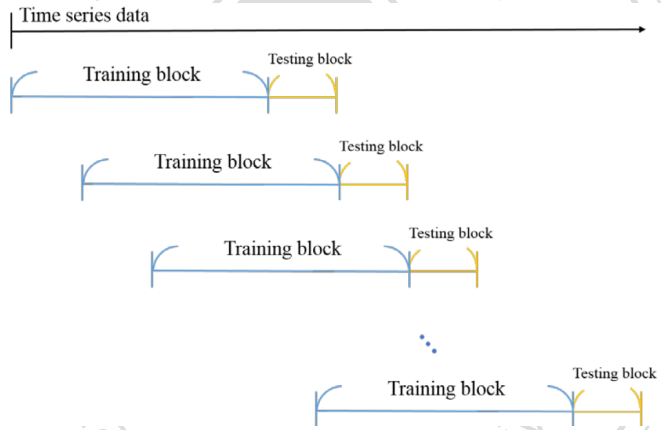

Figure 2: Moving window concept. [24]

2.8 TensorFlow & GPU

TensorFlow, a flexible data flow-based programming model, as well as single machine and distributed implementations of this programming model. The system is borne from real-world experience in conducting research and deploying more than one hundred machine learning projects throughout a wide range of Google products and services. [25] TensorFlow supports both large-scale training and inference: it efficiently uses hundreds of powerful (GPU-enabled) servers for fast training, and it runs trained models for inference in production on various platforms. At the same time, it is flexible enough to support ex-

DOI:10.6814/NCCU201900325

perimentation and research into new machine learning models and system-level optimizations [26]. Figure 3 presents the example computation graph of TensorFlow. In a TensorFlow graph, each node has zero or more inputs and zero or more outputs, and represents the instantiation of an *operation*. Values that flow along normal edges in the graph (from outputs to inputs) are *tensors*, arbitrary dimensionality arrays where the underlying element type is specified or inferred at graph-construction time.

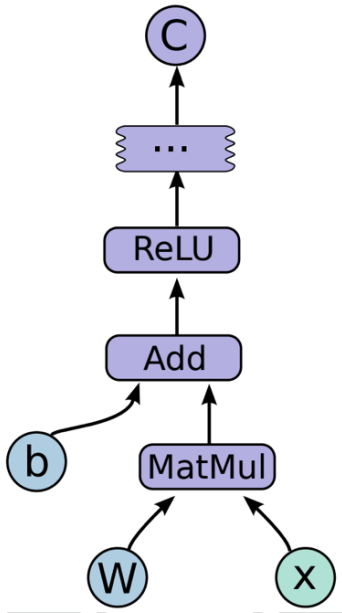

Figure 3: Corresponding computation graph. [26]

TensorFlow takes computations described using a dataflow-like model and maps them onto a wide variety of different hardware platforms, ranging from running inference on mobile device platforms such as Android and iOS to modest-sized training and inference systems using single machines containing one or many GPU cards to large-scale training systems running on hundreds of specialized machines with thousands of GPUs [26].

DOI:10.6814/NCCU201900325

3. The proposed DSM and CSI learning algorithm

3.1 The proposed DSM

In order to provide decision support for turning point issue in bull and bear markets, we derive the proposed CSI learning algorithm. Furthermore, we choose the incremental learning technique (that is, the moving window) in our DSM to help us handling the data expiration problem. With time passing, the older time series data won't be learned in the learning procedure. In contrast, the incoming time series data will be taken into consideration. The proposed DSM is shown in Table 1. There are one training block and one testing block in each window. M is the index of current window. N is the sample size of training block. B is the sample size of testing block.

Table 1. The proposed DSM.

Step 1.1: Use the CSI learning algorithm stated in Table 2 to learn the training block $\{(x^{(M-1)*B+1}, y^{(M-1)*B+1}), (x^{(M-1)*B+2}, y^{(M-1)*B+2}), \ldots, (x^{(M-1)*B+N}, y^{(M-1)*B+N})\}$ to obtain an acceptable SLFN, the majority of training block, and the standard deviation of the majority (σ).
Step 1.2: Output the turning point candidates in the training block.
Step 2.1: Use the obtained SLFN and σ to detect whether there are turning point candidates within the testing block $\{(x^{(M-1)*B+N+1}, y^{(M-1)*B+N+1}), (x^{(M-1)*B+N+2}, y^{(M-1)*B+N+2}), \ldots, (x^{M*B+N}, y^{M*B+N})\}$.
Step 2.2: Output the turning point candidates in the testing block.
Step 3: For more data, $M \leftarrow M+1$ and go to Step 1.1; otherwise, STOP.

In Step 1.1, and set the first window's M to 1 and use N training data of the current window to obtain an acceptable SLFN via the CSI learning algorithm. This step will make the SLFN to learn a non-linear fitting function f. Besides, we also get the majority of data in the training block (i.e., the learnt N*(1-k) data) and their associated σ, the standard deviation of residuals regarding the learnt N*(1-k) data.

In Step 1.2, the last (N*k) data, recognized as the *potential turning points*, will be examined their deviance information via the σ. The "deviance information" is the distance

DOI:10.6814/NCCU201900325

between the fitting function f and the desired output. Any instance that has larger residual (e.g., large than 3σ) will be recognized as the turning point candidate. We will output the potential outliers as *turning point candidates* to the decision maker.

In Step 2.1, we use the obtained SLFN and σ from Step 1.1 to determine whether any instance in testing block is the turning point candidate or not via its deviance information and σ. In Step 2.2, any instance that has larger residual (e.g., large than 3σ) will be recognized as the turning point candidate and be outputted to the decision maker.

In Step 3, the stopping criteria. If there is no more incoming data, this DSM will slide to next window and go back to Step 1. Otherwise, the DSM will be finished. In this research, the time series data is sorted sequentially in order.

3.2 The proposed CSI learning algorithm

This study has revised the softening learning procedure [14] to get the learning concept of CSI algorithm: when we encounter a new case (a new input/output relationship), we first check whether it is familiar to us. If it is familiar to us, there is no spontaneous learning effort involved. Later all learnt cases within our knowledge system are integrated. If it is unfamiliar, we might cram this unfamiliar case first. The cramming results in a strict rule regarding to this unfamiliar case. Then we will soften the strictness of the new case and do our best to integrate all learnt cases within our knowledge system.

Based upon the aforementioned learning concept, the proposed sequentially-learning-based algorithm contains the cramming, softening and integrating mechanisms as shown in Table 2, the details are as follows.

In the training phase, there are $\{(\mathbf{x}^1, y^1), (\mathbf{x}^2, y^2), \ldots, (\mathbf{x}^N, y^N)\}$, in which $y^c \in \{-1, 1\}$ is the desired output associated with $\mathbf{x}^c \in R^m$; y^c is the desired output value of the output node associated with $\mathbf{x}^c$; and N is the total amount of all reference observations for training. Based upon the aforementioned CSI learning procedure, the training case is learnt in sequence.

At the n^{th} stage of handling n reference observations, through minimizing the loss function $E_n(\mathbf{w}) \equiv \frac{\sum_{c\in \mathbf{I}(n)}(f(\mathbf{x^c},w)-\mathbf{y^c})^2}{n}+\frac{0.001}{p+1+p(m+1)}(\sum_{i=0}^{p}(w_i^o)^2 + \sum_{i=1}^{p}\sum_{j=0}^{m}(w_{ij}^H)^2)$, the learning goal is to seek $\mathbf{w}$ where $f(\mathbf{x^c}, \mathbf{w}) \geq \nu \ \forall\, c \in \mathbf{I}_1(n)$ and $f(\mathbf{x^c}, \mathbf{w}) \leq -\nu \ \forall\, c \in$

DOI:10.6814/NCCU201900325

$\mathbf{I}_2(\mathrm{n})$, with $1 > \nu > 0$ and the alternative goal of learning is to seek $\mathbf{w}$ that satisfies the condition L regarding $\{f(\mathbf{x}^c, \mathbf{w}),\ \forall\, c \in \mathbf{I}(n)\}$.

Table 2. The proposed CSI learning algorithms.

Step 1: Use two reference observations $\{(\mathbf{x}^1, y^1), (\mathbf{x}^2, y^2)\}$ with $y^1 * y^2 = -1$ to set up an acceptable SLFN estimate with one hidden node. Set $n = 3$.
Step 2: If $n > N*(1\text{-}k)$, STOP.
Step 3: Pick up the first n reference observations $\{(\mathbf{x}^c, y^c)\}$ which are sorted by all N reference observations' squared residuals in ascending order. Let $\mathbf{I}(n)$ be the set of indices of these picked observations.
Step 4: If the condition L regarding $\{f(\mathbf{x}^c, \mathbf{w})$, all $c \in \mathbf{I}(n)\}$ is satisfied, go to Step 7; otherwise, there is one and only one $\kappa \in \mathbf{I}(n)$ that is not at the right place and $\kappa = [n]$.
Step 5: Save $\mathbf{w}$.
Step 6: Apply the weight-tuning mechanism to $\min_{\mathbf{w}} E_n(\mathbf{w})$ to adjust $\mathbf{w}$ until one of the following two cases occurs : (1) If the condition L regarding $\{f(\mathbf{x}^c, \mathbf{w})$, all $c \in \mathbf{I}(n)\}$ is satisfied, go to Step 7. (2) If the condition L is not satisfied, restore $\mathbf{w}$ and then apply the cramming mechanism to add four extra hidden nodes to the existing SLFN to obtain an acceptable SLFN estimate.
Step 7: Apply the softening and integrating mechanism to prune the irrelevant hidden node, $n+1 \rightarrow n$; go to Step 2.

Step 1 uses two reference observations $\{(\mathbf{x}^1, y^1), (\mathbf{x}^2, y^2)\}$ with $y^1 * y^2 = -1$ to set up an SLFN estimate with one hidden node. It is easy to find a set of $\mathbf{w}$ associated with merely one hidden node that can render the condition L regarding $\{f(\mathbf{x}^c, \mathbf{w}), c \in \mathbf{I}(n)\}$ satisfied.

Step 2 defines the stopping criterion of the proposed learning algorithm. The symbol k can be referred to the percentage of potential outlier. Clearly, the majority consists of at least (1-k) % data. For example, if there is approximately at least 97% non-outliers and at most 3% outliers, the SLFN will take 97% data into the learning practice while building the SLFN.

DOI:10.6814/NCCU201900325

In Step 3, in order to reduce learning time, we choose our reference observation of the n^{th} stage by the LTS principle. Pick up the first n reference observations $\{(\mathbf{x}^c, y^c)\}$ which are sorted by all N reference observations' squared residuals in ascending order. Let $(e^c)^2 = \left(y^c - f(\mathbf{x}^c, \mathbf{w})\right)^2$, and $(e^{[1]})^2 \le (e^{[2]})^2 \le \ldots \le (e^{[n]})^2$. At the n^{th} stage, the cases with smallest n of the ordered squared residuals are included in the summation. Let $\mathbf{I}(n)$ be the set of indices of these observations and the following loss function is adopted:

$$E_n(\mathbf{w}) \equiv \frac{\sum_{c\in\mathbf{I}(n)}(f(x^c,w)-y^c)^2}{n} + \frac{0.001}{p+1+p(m+1)}\left(\sum_{i=0}^{p}(w_i^o)^2 + \sum_{i=1}^{p}\sum_{j=0}^{m}(w_{ij}^H)^2\right) \quad (5)$$

where $\frac{0.001}{p+1+p(m+1)}\left(\sum_{i=0}^{p}(w_i^o)^2 + \sum_{i=1}^{p}\sum_{j=0}^{m}(w_{ij}^H)^2\right)$ is the regularization term.

Step 4 states that, at the beginning of n^{th} stage, the SLFN is acceptable if its $\mathbf{w}$ and $\mathbf{r}$ can render the condition L regarding $\{f(\mathbf{x}^c, \mathbf{w}), \text{all } c \in \mathbf{I}(n)\}$ satisfied. As shown in Figure 4, we represent the cases of the class 1 with cross and the ones of class 2 with circle.

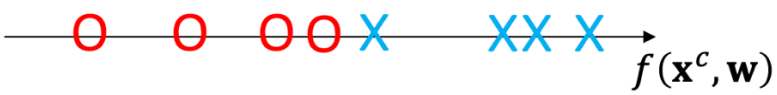

Figure 4: The illustration of the condition L.

The condition L regarding $\{f(\mathbf{x}^c, \mathbf{w}), \text{all } c \in \mathbf{I}(n)\}$ is satisfied implied there is no overlap between cross and circle and thus $\alpha > \beta$. If the condition L regarding $\{f(\mathbf{x}^c, \mathbf{w}), \text{all } c \in \mathbf{I}(n)\}$ is not satisfied, there is one and only one case κ that is not at the right place. Note that the κ^{th} case is the $[n]^{th}$ case since Step 3 has implemented the LTS principle. Step 5 stores $\mathbf{w}$ such that when we restore $\mathbf{w}$ at Step 6.2.1, we get back the scenario of there is one and only one [n] case that is not at the right place.

In Step 6, the weight-tuning mechanism will implement the adapted version of RMSProp Optimizer with an automatic adjustment of learning rate η shown in Figure 5, trying to accomplish the condition L and accelerate the training time. With such a weigh-tuning mechanism, it may lead to an acceptable SLFN or an unacceptable SLFN.

DOI:10.6814/NCCU201900325

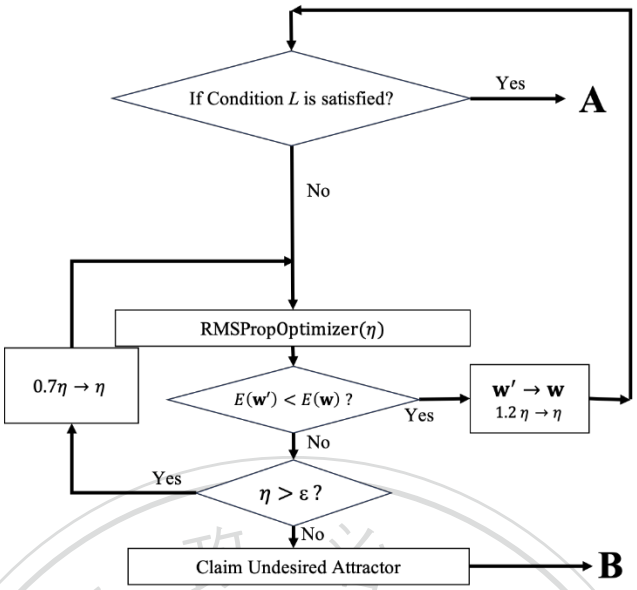

Figure 5: The adapted version of RMSProp Optimizer of the weight-tuning mechanism.

The unacceptable SLFN may be due to the defect of the SLFN or the convergence to local minimum solution. If the weight-tuning mechanism leads to an unacceptable SLFN, the cramming mechanism in Step 6.2 is implemented to add extra hidden nodes to obtain an acceptable SLFN. Table 3 shows the proposed cramming mechanism. The cramming mechanism creates a SLFN that, given n reference observations, will render the condition L regarding $\{f(\mathbf{x}^c, \mathbf{w})$, all $c \in \mathbf{I}(n)\}$ satisfied when given any reference observation as input. Step 6.2.a lets ζ be a given small number and tries to find two parallel hyperplanes whose normal vector is the m-vector $\boldsymbol{\gamma}$ of length one. $\boldsymbol{\gamma}$ is not vertical to the $(\mathbf{x}^c - \mathbf{x}^{[n]})$ $\forall$ $c \in \mathbf{I}(n) - \{[n]\}$. Moreover, the multiple of $(\zeta + \boldsymbol{\gamma}^{\mathrm{T}}(\mathbf{x}^c - \mathbf{x}^{[n]}))$ and $(\zeta - \boldsymbol{\gamma}^{\mathrm{T}}(\mathbf{x}^c - \mathbf{x}^{[n]}))$ will smaller than 0. Step 6.2.b assigns the new weights and bias of these four new added hidden nodes to satisfy the condition L regarding the output node.

DOI:10.6814/NCCU201900325

Table 3. The proposed cramming mechanism in Step 6.2.

Step 6.2.1: Restore **w**

Step 6.2.a: Let ζ be a given small number. Find an m-vector $\boldsymbol{\gamma}$ of length one such that $\boldsymbol{\gamma}^{\mathrm{T}}(\mathbf{x}^{c}-\mathbf{x}^{[n]}) \neq 0$ all $c \in \mathbf{I}(n)-\{[n]\}$ and $(\zeta+\boldsymbol{\gamma}^{\mathrm{T}}(\mathbf{x}^{c}-\mathbf{x}^{[n]}))*(\zeta-\boldsymbol{\gamma}^{\mathrm{T}}(\mathbf{x}^{c}-\mathbf{x}^{[n]})) < 0$ all $c \in \mathbf{I}(n)-\{[n]\}$.

Step 6.2.b:

(i) For the case of $y^{[n]} = 1.0$, let p+4 → p and add four new hidden nodes p-3th, p-2th, p-1th and pth to the existing SLFN with $w_{p-3,0}^{H} = \zeta-\boldsymbol{\gamma}^{\mathrm{T}}\mathbf{x}^{[n]}$, $\mathbf{w}_{p-3}^{H} = \boldsymbol{\gamma}$, $w_{p-2,0}^{H} = -\boldsymbol{\gamma}^{\mathrm{T}}\mathbf{x}^{[n]}$, $\mathbf{w}_{p-2}^{H} = \boldsymbol{\gamma}$, $w_{p-1,0}^{H} = -\boldsymbol{\gamma}^{\mathrm{T}}\mathbf{x}^{[n]}$, $\mathbf{w}_{p-1}^{H} = \boldsymbol{\gamma}$, $w_{p0}^{H} = -\zeta-\boldsymbol{\gamma}^{\mathrm{T}}\mathbf{x}^{[n]}$, $\mathbf{w}_{p}^{H} = \boldsymbol{\gamma}$, and

$$w_{p-3}^{o} = -w_{p-2}^{o} = -w_{p-1}^{o} = w_{p}^{o} = 1.01\left(\max_{u\in \boldsymbol{I}_2(n)} \sum_{i=1}^{p-2} w_i^o a_i^u - \sum_{i=1}^{p-2} w_i^o a_i^{[n]}\right)/\zeta$$

(ii) For the case of $y^{[n]} = -1.0$, let p+4 → p and add four new hidden nodes p-3th, p-2th, p-1th and pth to the existing SLFN with $w_{p-3,0}^{H} = \zeta-\boldsymbol{\gamma}^{\mathrm{T}}\mathbf{x}^{[n]}$, $\mathbf{w}_{p-3}^{H} = \boldsymbol{\gamma}$, $w_{p-2,0}^{H} = -\boldsymbol{\gamma}^{\mathrm{T}}\mathbf{x}^{[n]}$, $\mathbf{w}_{p-2}^{H} = \boldsymbol{\gamma}$, $w_{p-1,0}^{H} = -\boldsymbol{\gamma}^{\mathrm{T}}\mathbf{x}^{[n]}$, $\mathbf{w}_{p-1}^{H} = \boldsymbol{\gamma}$, $w_{p0}^{H} = -\zeta-\boldsymbol{\gamma}^{\mathrm{T}}\mathbf{x}^{[n]}$, $\mathbf{w}_{p}^{H} = \boldsymbol{\gamma}$, and

$$w_{p-3}^{o} = -w_{p-2}^{o} = -w_{p-1}^{o} = w_{p}^{o} = 1.01\left(\min_{u\in \boldsymbol{I}_1(n)} \sum_{i=1}^{p-2} w_i^o a_i^u - \sum_{i=1}^{p-2} w_i^o a_i^{[n]}\right)/\zeta$$

Because of cramming mechanism, all of N reference observations can be learnt perfectly at the end of training. But the SLFN may turn out to be overfitting due to recruiting too many hidden nodes. To prevent the overfitting dilemma, the softening and integrating mechanism of the Step 7 is implemented to prune any identified irrelevant hidden node. At the n^{th} stage, the k^{th} hidden node is irrelevant if it is deleted and the condition L can still be accomplished by merely applying the weight-tuning mechanism to $\min_{\mathbf{w}_k'} E_n(\mathbf{w}_k')$, where $\mathbf{w}_k' \equiv \mathbf{w}-\{\mathbf{w}_k^o, \mathbf{w}_k^H\}$.

Table 4 shows the proposed softening and integrating mechanisms implemented in Step 7. The details are as follow.

Step 7.1 applies the weight-tuning mechanism one hundred times to $\min_{\mathbf{w}} E_n(\mathbf{w})$ to adjust **w**, while keeping the learning goal satisfied. Since there is a regularization term

DOI:10.6814/NCCU201900325

defined in $E_n(\mathbf{w})$, applying the weight-tuning mechanism one hundred times to $\min_{\mathbf{w}} E_n(\mathbf{w})$ can "soften" the large values of $\mathbf{w}$, and thus may solve the tendency of overfitting.

In Step 7.2, we calculate the g'_k value regarding each hidden node. That is, let $\boldsymbol{w}'_k \equiv \mathbf{w}\text{-}\{w_k^o, \boldsymbol{w}_k^H\}$ all k. Then, we calculate $f(\boldsymbol{x}^c, \boldsymbol{w}'_k) \equiv f(\mathbf{x}^c, \mathbf{w}) - w_k^o a_k^c$ all $c \in \mathbf{I}(n)$ all k, and $\alpha'_k \equiv \min_{c \in \boldsymbol{I}_1(n)} f(\boldsymbol{x}^c, \boldsymbol{w}'_k)$, $\beta'_k \equiv \max_{c \in \boldsymbol{I}_2(n)} f(\boldsymbol{x}^c, \boldsymbol{w}'_k)$, and $g'_k \equiv \alpha'_k\text{-}\beta'_k$ all k.

The fact $\max_{1 \le k \le p} g'_k = 0$ means that some hidden node is irrelevant and can be pruned directly. Thus, Step 7.3 picks up the i^{th} hidden node, where i is the first index of $arg\max_{k \in \Omega} g'_k$, and prune it directly. Then go to step 7.1 to keep inspecting if the SLFN has any other irrelevant hidden node.

Let Θ be the tolerance quantity regarding the irrelevance. With the Θ guidance, the fact $-\Theta > \max_{1 \le k \le p} g'_k$ means that we assume the current SLFN has no irrelevant hidden node and go to Step 2. Meanwhile, the fact $0 > \max_{1 \le k \le p} g'_k \ge -\Theta$ means that we assume the current SLFN may have some potentially irrelevant hidden nodes. Thus, Step 7.5 picks up the i^{th} hidden node, where i is the first index of $arg\max_{k \in \Omega} g'_k$, where $\Omega \equiv \{k: 0 > g'_k \ge -\Theta\}$, to temporarily prune the k^{th} hidden node, and then apply the weight-tuning mechanism to $\min_{\mathbf{w}_i} E_n(\mathbf{w}'_i)$ to adjust $\mathbf{w}'_i$ to inspect if the i^{th} hidden node is irrelevant or not. If the i^{th} hidden node is irrelevant, then go to step 7.1 to keep inspecting if the SLFN has any other irrelevant hidden node. If the i^{th} hidden node is not irrelevant, then we assume the current SLFN has no irrelevant hidden node and go to Step 2.

With the proposed CSI learning algorithm, all the reference observations can be perfectly learnt by the cramming, softening, and integrating mechanisms.

DOI:10.6814/NCCU201900325

Table 4. The softening and integrating mechanism in Step 7.

Step 7.1: Apply the weight-tuning mechanism 100 times to $\min_{\mathbf{w}} E_n(\mathbf{w})$ to adjust **w**, while keeping the condition *L* regarding {*f*(**x**^{c}, **w**), all $c \in \mathbf{I}(n)$} satisfied.

Step 7.2: Calculate $g'_k \; \forall\, k$, where $\boldsymbol{w}'_k \equiv \mathbf{w} - \{w_k^o,\ \boldsymbol{w}_k^H\}$, $f(\boldsymbol{x}^c, \boldsymbol{w}'_k) \equiv f(\mathbf{x}^c, \mathbf{w}) - w_k^o a_k^c$, $\alpha'_k \equiv \min_{c \in \boldsymbol{I}_1(n)} f(\boldsymbol{x}^c, \boldsymbol{w}'_k)$, $\beta'_k \equiv \max_{c \in \boldsymbol{I}_2(n)} f(\boldsymbol{x}^c, \boldsymbol{w}'_k)$, and $g'_k \equiv \alpha'_k - \beta'_k$.

Step 7.3: If $-\Theta > \max_{1 \le k \le p} g'_k$, where Θ is a given constant, go to Step 2.

Step 7.4: If $\max_{1 \le k \le p} g'_k > 0$, prune the i^{th} hidden node, in which i is the first index of $\arg\max_{1 \le k \le p} g'_k$, p-1 → p, $\boldsymbol{w}'_i$ → **w**, and go to Step 7.1.

Step 7.5: If $0 \ge \max_{1 \le k \le p} g'_k \ge -\Theta$,

(a) store **w**.

(b) temporarily prune the i^{th} hidden node, in which i is the first index of $\arg\max_{k \in \Omega} g'_k$, where $\Omega \equiv \{k: 0 > g'_k \ge -\Theta\}$.

(c) apply the weight-tuning mechanism to $\min_{\boldsymbol{w}'_i} E_n(\boldsymbol{w}'_i)$ to adjust $\boldsymbol{w}'_i$ until one of the following two cases occurs:

(1) If an acceptable result is obtained, then prune the i^{th} hidden node, p-1 → p, $\boldsymbol{w}'_i$ → **w**, and go to Step 7.1.

(2) If an unacceptable result is obtained, then restore the i^{th} hidden node, restore **w**, and go to Step 2.

DOI:10.6814/NCCU201900325

4. Experiment design

4.1 The bull and bear market in US stock market

Although the market trend has been widely touted by media, there has never been a globally agreed upon definition on the bull and bear markets. A bull market sees persistently rising stock prices, strong investor sentiment, lower volatility, higher trading volume, among other performance behavior. A bear market often features otherwise. However, there is no consensus on how much the stock prices are rising or falling in what kind of an extended period of time constitute a bull or bear market. Hanna [7] proposes a top-down approach to identify the bull and bear market states and discusses three different methods. Gonzalez et al. [27] use a formal turning-point identification to pinpoint the break points of different market trends, and they find that the bull and bear markets are associated with significantly different and persistent mean return shifts.

4.2 Variable description

In addition to the consideration of the y-label of the attribute of bull and bear markets, the purpose of this study is to use potential factors as the x-label to predict the trend of the turning point. Traditional technical analysis often compares prices with 3-month or 12-month moving average to see the general trend of the prices. Chen [9][10] further investigates whether macroeconomic variables can predict the bull and bear markets, not only the other way round as people believe.

We use these trustworthy theories to set up our inputs and output data. Among the different market trend identification approaches, we apply Pagan and Sossounov [8] method. They propose the approach which is influenced by Bry and Boschan [28] but adapted to stock prices in accordance with Dow Theory. The output we used in the CSI learning algorithm to demonstrate the prediction of the bull/bear market can be defined as the theory of Pagan and Sossounov [8]. They represent the monthly situation where the phase is between peak and trough. We use the moment of peak and trough as the turning point to construct the indicator of y-label. We assign the bull market (peak) as value 1, and the bear market (trough) as value -1. Chen [9] proposes the macroeconomic variables

DOI:10.6814/NCCU201900325

which is associated with the bull and bear markets. The x-labels we apply in the CSI learning algorithm include term spreads (3M-10Y), term spreads (3M-5Y), inflation rates, industrial production growth, money stocks (M1), money stocks (M2), federal funds rates, unemployment rates, stock return, and trading volume. The description of x-labels and y-label are shown in Table 5.

In sum, there are 12 inputs for the network (including term spreads (3M-10Y), term spreads (3M-5Y), inflation rates, nominal rate of return, industrial production growth, money stocks (M1), money stocks (M2), federal funds rates, unemployment rates, stock return, trading volume, and the extent that the current index is above/below the past 3- and 12-month average); there is only one output of the network (i.e., bull market and bear market).

Table 5. The description of x-labels and y-label.

Input	Attribute	Description
x_1	Term spreads (3M-10Y) [9]	The difference between the 3-month treasury bill rate and the 10-year treasury constant maturity rate.
x_2	Term spreads (3M-5Y) [9]	The difference between the 3-month treasury bill rate and the 5-year treasury constant maturity rate.
x_3	Inflation rates [9]	Related to consumer price.
x_4	Industrial production growth [9]	The annual percentage increase in industrial production (includes manufacturing, mining, and construction).
x_5	Money stocks (M1) [9]	M1 includes funds that are readily accessible for spending.
x_6	Money stocks (M2) [9]	M2 includes a broader set of financial assets held principally by households.
x_7	Federal funds rates [9]	The federal funds rate is the interest rate at which depository institutions trade federal funds (balances held at Federal Reserve Banks) with each other overnight.
x_8	Unemployment rates [9]	Unemployment rate is the number of unemployed people as a percentage of the labour force, where the

DOI:10.6814/NCCU201900325

		latter consists of the unemployed plus those in paid or self-employment.
x_9	Stock return [10]	S&P 500 index.
x_{10}	Trading volume [10]	S&P 500 trading volume.
x_{11}	3-month MA	The current S&P index is above/below the past 3-month average.
x_{12}	12-month MA	The current S&P index is above/below the past 12-month average.

Out-put	Attribute	Description
y	Bull market [7]	The S&P index that extending both directions of current 8-month window, ant determines the highest peak in the window. The bull market starts from the trough and ends at the peak, representing a rising market.
	Bear market [7]	The S&P index that extending both directions of current 8-month window, ant determines the lowest trough in the window. The bear market starts from the peak and ends at the trough, representing a falling market.

In order to compare the performance and the evaluation of the CSI learning algorithm, we categorize the desired output y^c into the bull or bear market. When we have identified the tendency of the bull and bear market, we then determine whether there is a turning point.

4.3 The description of collected data

We focus on the US stock market and use the S&P 500 index to identify the market trend (although Dow Jones Industrial Average is more popular among investors, it only consists 30 stocks and is calculated with less advanced technique), and we collect the monthly data of inputs and output value in monthly frequency.

DOI:10.6814/NCCU201900325

Babcock, Datar and Motwani [29] define a sequence-based window by choosing a specified size uniformly over a "moving window" of the last elements in a data stream. The moving window mechanism has been adopted in many fields. For example, Kashani et al. [30] use a robust ANN with the moving window concept to build a dynamical model for predicting the crude oil fouling behavior. They claim that the implementation of moving window updates the model whenever a new data block comes in and helps to catch the slowly changing dynamic trends.

Because the stock market is considered to be the type of time series data, we set data range from January 1978 to December 2017. The experiment adopts totally 480 observation data, a sample size of 360 for the training block and a sample size of 12 for the testing block in Figure 6. For instance, at the beginning, the training block consists of the 1st to 360th instances and the testing block consists of the 361th to 372th instances. As time passes, the first twelve instances are discarded and the training block slides to the 13th to 372th instances. The testing block simultaneously slides to the 373th to 384th instances. This process continues until there are no further data. Thus, there are 10 windows in total. The advantage of using moving window is to set up the initial acceptable SLFN model in the first window, and use this model to transfer to the second window. Due to the high correlation between the first and the second window, we can assume that the acceptable SLFN will be similar and accelerate the training time. In order to figure the candidate turning point, we apply the proposed DSM to justify and evaluate both the training block and the testing block.

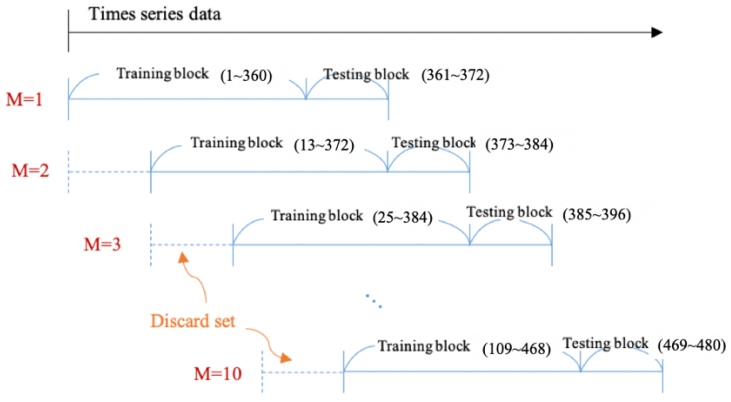

Figure 6: The experiment design of moving window.

DOI:10.6814/NCCU201900325

With a view to evaluating the performance of detecting outliers with this proposed DSM in concept drifting environment, we have to clarify the σ by deviance information.

$$\sigma = \sqrt{\frac{\sum_{c \in N \times (1-k)} (f(x^c) - y^c)^2}{N \times (1-k)}} \tag{6}$$

$$Y_t = f(X_t) + \epsilon_t \,,\ t \geq 0 \tag{7}$$

Here we first calculate the σ by the standard deviation between the desired output of acceptable SLFN and the actual output in the majority of each window. In the research of the potential turning point, we assign the *k* to the 3.3%, that is, for 360 instances in each window, there are approximately 348 major instances, and 12 potential turning points. Secondly, for the training block and the testing block, we calculate the difference between acceptable SLFN and the actual output and set it as ϵ_t in equation (7). If $|\epsilon_t| \geq 3\sigma$, where σ is "the standard deviation obtained from the training block", the instance is treated as the *potential turning point*.

4.4 The proposed learning algorithm and its implementation

The experiment is implemented by the DSM and proposed CSI learning algorithm, which is conducted with TensorFlow and GPU to speed up the training and enhance the learning performance. TensorFlow can result in better performance of the system, in particular with respect to data transfers and memory usage and it comes with strong support for machine learning and deep learning and the flexible numerical computation core is used across many other scientific domains. Besides, the experiment environment is shown as Table 6.

Table 6. The experiment environment.

Property	Tool
Operating System	Ubuntu 16.04.5 LTS
GPU	NVIDIA GeForce GTX 1080 Ti
Language	Python 3.6
API	TensorFlow 1.12.0
IDE	Jupyter Notebook

DOI:10.6814/NCCU201900325

5. Experiment results

5.1 The performance evaluation

In the performance evaluation, we set the rule that how to evaluate the DSM's performance in this research. According to the non-turning point or turning point defined when designing the experiment by judging ϵ_t, we can compare the identification result to the proposed DSM. There are four possible outcomes: (the term's fist character is on behalf of the actual type, and the second character is on behalf of the predicted type identified by the proposed DSM).

(1) N─T: The non-turning-point that has been incorrectly specified as turning point candidate.

(2) T─N: The actual turning point that has been incorrectly specified as non-turning point.

(3) T─T: The actual turning point that has been correctly specified as turning point candidate.

(4) N─N: The actual non-turning point that has been correctly specified as non-turning point.

In our experiment, we will record the training time of the training block and the amount of the adapted hidden nodes in each moving window to examine the effectiveness and efficiency of the proposed CSI learning algorithm and the DSM.

5.2 The experiment results

As mentioned in section 4.2, we will take the *x* and *y* variables to predict the trend of the turning point. We set data range from January 1978 to December 2017 of the monthly data. The experiment adopts totally 480 observation data, a sample size of 360 for the training block, a sample size of 12 for the testing block, and there are totally of 10 moving windows. For the majority in each training block, there are 348 instances; and for the potential turning points in the training block, there are 12 instances.

DOI:10.6814/NCCU201900325

Table 7 shows the outcome of the distribution of the actual turning points of the proposed CSI learning algorithm and the DSM. For the most part of actual turning points, it will be categorized to the potential turning points. The overall average of the proportion of actual turning points in the potential turning points is 54.17% and the standard deviation is 7.08%. In comparison, there are fewer actual turning points in the majority of the training block than the potential turning points. The overall average of the proportion of actual turning points in the majority of the training block is 1.29% and the standard deviation is 0.24%.

Table 7. The distribution of the actual turning points in each window.

M	**Training block**				**Testing block**	
	Majority		Potential turning point			
	Non-turning point	Turning point	Non-turning point	Turning point	Non-turning point	Turning point
1	344 (98.85%)	4 (1.15%)	4 (33.33%)	8 (66.67%)	12 (100%)	0 (0%)
2	345 (99.14%)	3 (0.86%)	5 (41.67%)	7 (58.33%)	11 (91.67%)	1 (8.33%)
3	344 (98.85%)	4 (1.15%)	5 (41.67%)	7 (58.33%)	12 (100%)	0 (0%)
4	343 (98.56%)	5 (1.44%)	7 (58.33%)	5 (41.67%)	10 (83.33%)	2 (16.67%)
5	342 (98.27%)	6 (1.73%)	6 (50%)	6 (50%)	12 (100%)	0 (0%)
6	343 (98.56%)	5 (1.44%)	5 (41.67%)	7 (58.33%)	12 (100%)	0 (0%)
7	343 (98.56%)	5 (1.44%)	6 (50%)	6 (50%)	12 (100%)	0 (0%)
8	344 (98.85%)	4 (1.15%)	6 (50%)	6 (50%)	12 (100%)	0 (0%)
9	343 (98.56%)	5 (1.44%)	5 (41.67%)	7 (58.33%)	12 (100%)	0 (0%)
10	344 (98.85%)	4 (1.15%)	6 (50%)	6 (50%)	12 (100%)	0 (0%)
Average	309.2 (98.71%)	4.1 (1.29%)	4.95 (45.83%)	5.95 (54.17%)	10.6 (97.5%)	0.3 (2.5%)
Standard deviation	0.85 (0.24%)	0.85 (0.24%)	0.85 (7.08%)	0.85 (7.08%)	0.67 (7.08%)	0.67 (7.08%)

In Table 8, there are four possible outcomes of the proposed DSM in each window. As far as the turning point detection is concerned, we use Table 9 and Table 10 to explain the relationship between the actual turning points and the turning point candidates. Table

DOI:10.6814/NCCU201900325

9 shows the amount of actual turning points, the prediction of the proposed CSI learning algorithm and DSM in each window. For the prediction in the training block, we can distinguish more than 50% actual turning points. For the prediction in the testing block, there are only three turning points in two windows. In other words, we can precisely identify non-turning point in the testing block and prevent the wrong classification.

Table 10 points up the relationship between the actual turning points and the turning point candidates. The proportion of turning point candidates that are actual turning points in the training block is 73/382. It explains that the proposed CSI learning algorithm and DSM will capture dominant actual turning points for the turning point candidates. Moreover, the proportion of actual turning points identified as turning point candidates in the training block is 66.67%, and depicts the turning point candidates is highly probability belong to the actual turning points. On the other hand, there is not necessarily a turning point of every window in the testing block, for the proportion, we can detect 2 actual turning points in 4 turning point candidates. For the proportion of actual turning points identified as turning point candidates in the testing block is for 3 turning point candidates, we can identify 2 actual turning points.

Table 8. The experiment results in each window.

M	**Training block**								**Testing block**			
	Majority				Potential turning point							
	N-N	N-T	T-T	T-N	N-N	N-T	T-T	T-N	N-N	N-T	T-T	T-N
1	319	25	3	1	2	2	7	1	12	0	0	0
2	323	22	2	1	4	1	7	0	11	0	1	0
3	336	8	2	2	3	2	4	3	12	0	0	0
4	311	32	3	2	3	4	2	3	9	1	1	1
5	309	33	4	2	3	3	3	3	11	1	0	0
6	307	36	3	2	3	2	4	3	12	0	0	0
7	309	34	3	2	4	2	4	2	12	0	0	0
8	315	29	3	1	4	2	4	2	12	0	0	0
9	305	38	3	2	5	2	4	1	12	0	0	0
10	314	30	3	1	4	2	5	1	12	0	0	0

DOI:10.6814/NCCU201900325

Aver-age	314.8	28.7	2.9	1.6	3.5	2.2	4.4	1.9	11.17	0.333	0.333	0.167
Stand-ard devia-tion	9.27	8.73	0.57	0.52	0.85	0.79	1.58	1.10	0.97	0.42	0.42	0.32

Table 9. The experiment results of turning point in each window.

M	**The amount of actual turning points**		**Training block**				**Testing block**	
			Majority		Potential turning point			
	Training block	Testing block	T-T	T-N	T-T	T-N	T-T	T-N
1	12	0	3 (75%)	1 (25%)	7 (87.5%)	1 (12.5%)	NA	NA
2	10	1	2 (66.66%)	1 (33.33%)	7 (100%)	0 (0%)	1 (100%)	0 (0%)
3	11	0	2 (50%)	2 (50%)	4 (57.14%)	3 (42.86%)	NA	NA
4	10	2	3 (60%)	2 (40%)	2 (40%)	3 (60%)	1 (50%)	1 (50%)
5	12	0	4 (67.33%)	2 (33.67%)	3 (50%)	3 (50%)	NA	NA
6	12	0	3 (60%)	2 (40%)	4 (57.14%)	3 (42.86%)	NA	NA
7	11	0	3 (60%)	2 (40%)	4 (67%)	2 (33%)	NA	NA
8	10	0	3 (75%)	1 (25%)	4 (67%)	2 (33%)	NA	NA
9	10	0	3 (60%)	2 (40%)	4 (80%)	1 (20%)	NA	NA
10	10	0	3 (75%)	1 (25%)	5 (83%)	1 (17%)	NA	NA
Average			2.9 (64.83%)	1.6 (35.17%)	4.4 (68.84%)	1.9 (31.16%)	1 (75%)	0.5 (25%)
Standard Deviation			0.57	0.52	1.58	1.1	0	0.71

DOI:10.6814/NCCU201900325

	(8.37%)	(8.37%)	(18.63%)	(18.63%)	(35.36%)	(35.36%)

Table 10. The evaluation of the turning point in each window.

M	Amount of actual turning points		Amount of turning point candidates		Proportion of turning point candidates that are actual turning points		Proportion of actual turning points identified as turning point candidates	
	Training block	Testing block	Training block	Testing block	Training block	Testing block	Training block	Testing block
1	12	0	37	0	10/37 (27.03%)	NA	10/12 (83.33%)	NA
2	10	1	32	1	9/32 (28.13%)	1/1 (100%)	9/10 (90%)	1/1 (100%)
3	11	0	16	0	6/16 (37.5%)	NA	6/11 (54.55%)	NA
4	10	2	41	2	5/41 (12.2%)	1/2 (50%)	5/10 (50.00%)	1/2 (50%)
5	12	0	43	1	7/43 (16.28%)	0/1	7/12 (58.33%)	NA
6	12	0	45	0	7/45 (15.56%)	NA	7/12 (58.33%)	NA
7	11	0	43	0	7/43 (16.28%)	NA	7/11 (63.64%)	NA
8	10	0	38	0	7/38 (18.42%)	NA	7/10 (70%)	NA
9	10	0	47	0	7/47 (14.89%)	NA	7/10 (70%)	NA
10	10	0	40	0	8/40 (20%)	NA	8/10 (80%)	NA
Average	10.8	0.3	38.2	0.4	73/382 (19.11%)	2/4 (50%)	73/108 (67.59%)	2/3 (66.67%)
Standard Deviation	0.92	0.67	8.90	0.70	NA	NA	NA	NA

DOI:10.6814/NCCU201900325

The additional evaluation of this research shows in Table 11. The average of the training time in training block is 1day 13hrs 17mins. Window 1 spends the much time to get the acceptable SLFN via the proposed CSI learning algorithm and DSM. And for another window, the training time is accelerated and the average training time does not exceed to two days. The amount of adopted hidden nodes is gradually increasing because of the moving window. The moving window will discard the older data and update the new information. Overall, the average of the amount of adopted hidden nodes 148.3.

Table 11. The evaluation of the training time and the amount of adopted hidden nodes.

M	Training time in the training block	The amount of adopted hidden nodes
1	2 days 3 hrs 48 mins	105
2	1 day 13 hrs 29 mins	95
3	1 day 10 hrs 16 mins	120
4	1 day 11 hrs 28 mins	148
5	1 day 8 hrs 35 mins	131
6	1 day 15 hrs 48 mins	159
7	1 day 11 hrs 56 mins	164
8	1 day 14 hrs 21 mins	186
9	1 day 10 hrs 2 mins	182
10	1 day 9 hrs 47 mins	193
Average	1day 13hrs 17mins	148.3
Standard deviation	5 hrs 35 mins	34.53

DOI:10.6814/NCCU201900325

6. Conclusion and future work

Returning briefly to the study subjective of the detection of the turning point in the bull/ bear markets, this thesis has described the proposed CSI learning algorithm and DSM methods used in this investigation. As for the experiment results, it explains the relationship between the actual turning points and the turning point candidates, we can distinguish more than 50% actual turning points in the training block. Although there are only three turning points in two windows of the testing block, we can precisely identify non-turning points in the testing block in the performance and avoid the wrong classification. That is to say, we present that the proposed CSI learning algorithm and DSM will capture dominant actual turning points for the turning point candidates. Moreover, the average proportion of the actual turning points that has been identified as turning point candidates is 67.6%, and depicts the turning point candidates is highly probability belong to the actual turning points. The proposed CSI learning algorithm and DSM will make effort to provide a decision-making support to the decision maker in the detection of the turning points.

In DSM, there are many hyper-parameters like N, B, ε, k, and 3σ (the coefficient of the σ) needed to be set depending on the data nature and the specific requirements regarding the detection usage. Similarly, Θ, η, and ε are the hyper-parameters in the CSI learning algorithm. In order to conduct different experiment design, we can tune these hyper-parameters in different arrangements.

This thesis is to use potential factors as the x-label to predict the trend of the turning point in the bull and bear markets. For the prediction measure, we can adopt the x variables to predict the following monthly value of both x and y variables. Furthermore, we can build the simulation of x-label to develop the diverse application of the financial trend, for example, the y-label can be assigned as the indicator of the business cycle.

In comparison of the turning point detection in the bull/ bear markets in the domain of the ANN, there is no learning model in econometrics so far. We can progress more conceivable variables to establish the general rule and use other machine learning models to enhance the SLFN model.

DOI:10.6814/NCCU201900325

Reference

[1] R. R. Trippi, and E. Turban, Neural networks in finance and investing: Using artificial intelligence to improve real world performance, McGraw-Hill, Inc., 1992.

[2] H. J. Kim, and K. S. Shin, "A hybrid approach based on neural networks and genetic algorithms for detecting temporal patterns in stock markets," Applied Soft Computing, vol. 7, no. 2, pp. 569-576, 2007.

[3] M. Göçken, M. Özçalıcı, A. Boru, and A. T. Dosdoğru, "Integrating metaheuristics and artificial neural networks for improved stock price prediction," Expert Systems with Applications, vol. 44, pp. 320-331, 2016.

[4] A. Macchiarulo, "Predicting and beating the stock market with machine learning and technical analysis," Journal of Internet Banking and Commerce, vol. 23, no. 1, pp. 1-22, 2018.

[5] E. Chong, C. Han, and F. C. Park, "Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies," Expert Systems with Applications, vol. 83, pp. 187-205, 2017.

[6] A. K. Nassirtoussi, S. Aghabozorgi, T. Y. Wah, and D. C. L. Ngo, "Text mining for market prediction: A systematic review," Expert Systems with Applications, vol. 41, no. 16, pp. 7653-7670, 2014.

[7] A. J. Hanna, "A top-down approach to identifying bull and bear market states.," International Review of Financial Analysis, vol. 55, pp. 93-110, 2018

[8] A. R. Pagan, and K. A. Sossounov, "A simple framework for analysing bull and bear markets," Journal of Applied Econometrics, vol. 18, no. 1, pp. 23-46, 2003.

[9] S. S. Chen, "Predicting the bear stock market: Macroeconomic variables as leading indicators," Journal of Banking & Finance, vol. 33, no. 2, pp. 211-223, 2009.

[10] S. S. Chen, "Revisiting the empirical linkages between stock returns and trading volume," Journal of Banking & Finance, vol. 36, no. 6, pp. 1781-1788, 2012.

[11] A. Tsymbal, "The problem of concept drift: definitions and related work," Computer Science Department, Trinity College Dublin, vol. 106, no. 2, pp. 58, 2004.

[12] R. Elwell, & R. Polikar, "Incremental learning of concept drift in nonstationary environments," IEEE Transactions on Neural Networks, vol. 22, no. 10, pp. 1517-1531, 2011.

DOI:10.6814/NCCU201900325

[13] B. Krawczyk, & M. Woźniak, “Diversity measures for one-class classifier ensembles,” Neurocomputing, vol. 126, pp. 36-44, 2014.

[14] R. R. Tsaih, “The softening learning procedure,” Mathematical and computer modelling, vol. 18, no. 8, pp. 61-64, 1993.

[15] R. R. Tsaih, “An explanation of reasoning neural networks,” Mathematical and Computer Modelling, vol. 28, no. 2, pp. 37-44, 1998.

[16] R. H. Tsaih, and T. C. Cheng, “A resistant learning procedure for coping with outliers,” Annals of Mathematics and Artificial Intelligence, vol. 57, no. 2, pp. 161-180, 2009.

[17] S. Y. Huang, R. H. Tsaih, and F. Yu, “Topological pattern discovery and feature extraction for fraudulent financial reporting,” Expert Systems with Applications, vol. 41, no. 9, pp. 4360-4372, 2014.

[18] R. H. Tsaih, B. S. Kuo, T. H. Lin, and C. C. Hsu, “The use of big data analytics to predict the foreign exchange rate based on public media: A machine-learning experiment,” IT Professional, vol. 20, no. 2, pp. 34-41, 2018.

[19] P. J. Rousseeuw, & K. Van Driessen, “Computing LTS regression for large data sets,” Data mining and knowledge discovery, vol. 12, no. 1, pp. 29-45, 2006.

[20] D. E. Rumelhart, G. E. Hinton, & J. L. McClelland, “A general framework for parallel distributed processing,” Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1, no. 45-76, pp. 26, 1986.

[21] A. Zaman, P. J. Rousseeuw, & M. Orhan, “Econometric applications of high-breakdown robust regression techniques,” Economics Letters, vol. 71, no. 1, pp. 1-8, 2001.

[22] A. C. Atkinson, & T. C. Cheng, “On robust linear regression with incomplete data,” Computational statistics & data analysis, vol. 33, no. 4, pp. 361-380, 2000.

[23] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, “A survey on concept drift adaptation,” ACM computing surveys (CSUR), vol. 46, no. 4, pp. 44, 2014.

[24] C. W. Lin, A Decision Support Mechanism for Outlier Detection in the Concept Drifting Environment, Master's thesis, 2015.

DOI:10.6814/NCCU201900325

[25] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, ... & M. Kudlur, "Tensorflow: A system for large-scale machine learning," In 12th {USENIX} Symposium on Operating Systems Design and Implementation, {OSDI} 16, pp. 265-283, 2016.

[26] M. Abadi, and TensorFlow, A. A. B. P., "Large-scale machine learning on heterogeneous distributed systems," In Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation, OSDI'16, Savannah, GA, USA , pp. 265-283, 2016.

[27] L. Gonzalez, J. G. Powell, J. Shi, & A. Wilson, "Two centuries of bull and bear market cycles," International Review of Economics & Finance, vol. 14 no. 4, pp. 469-486, 2005.

[28] G. Bry, & C. Boschan, "Front matter to" Cyclical Analysis of Time Series: Selected Procedures and Computer Programs," In Cyclical analysis of time series: Selected procedures and computer programs, pp. 13-2. NBER, 1971.

[29] B. Babcock, S. Babu, M. Datar, R. Motwani, & J. Widom, "Models and issues in data stream systems," In Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, pp. 1-16, ACM, 2002.

[30] M. N. Kashani, J. Aminian, S. Shahhosseini,& M. Farrokhi, "Dynamic crude oil fouling prediction in industrial preheaters using optimized ANN based moving window technique," Chemical Engineering Research and Design, vol. 90, no. 7,pp. 938-949, 2012.

DOI:10.6814/NCCU201900325