EMD2FNN: A strategy combining empirical mode decomposition and factorization machine based neural network for stock market trend prediction

Feng Zhou, Hao-min Zhou, Zhihua Yang, Lihua Yang

PII: S0957-4174(18)30490-1
DOI: 10.1016/j.eswa.2018.07.065
Reference: ESWA 12119

To appear in: *Expert Systems With Applications*

Received date: 1 April 2018
Revised date: 2 July 2018
Accepted date: 29 July 2018

Please cite this article as: Feng Zhou, Hao-min Zhou, Zhihua Yang, Lihua Yang, EMD2FNN: A strategy combining empirical mode decomposition and factorization machine based neural network for stock market trend prediction, *Expert Systems With Applications* (2018), doi: 10.1016/j.eswa.2018.07.065

Highlights

- We propose an improved neural network model to predict the stock prices
- The empirical mode decomposition and factorization machine are used in our approach
- The empirical mode decomposition helps overcome the non-stationarity of stock price
- Factorization Machine helps grasp the nonlinear interactions among the inputs
- The real data sets are used to demonstrate the accuracy of the new approach

EMD2FNN: A strategy combining empirical mode decomposition and factorization machine based neural network for stock market trend prediction☆

Feng Zhou[a], Hao-min Zhou[b], Zhihua Yang[a], Lihua Yang[c,*]

[a]School of Information, Guangdong University of Finance and Economics, Guangzhou 510320, China.
[b]School of Mathematics, Georgia Institute of Technology, Atlanta, GA, 30332, USA.
[c]School of Mathematics, Sun Yat-sen University, Guangzhou 510275, China.

Abstract

Stock market forecasting is a vital component of financial systems. However, the stock prices are highly noisy and non-stationary due to the fact that stock markets are affected by a variety of factors. Predicting stock market trend is usually subject to big challenges. The goal of this paper is to introduce a new hybrid, end-to-end approach containing two stages, the Empirical Mode Decomposition and Factorization Machine based Neural Network (EMD2FNN), to predict the stock market trend. To illustrate the method, we apply EMD2FNN to predict the daily closing prices from the Shanghai Stock Exchange Composite (SSEC) index, the National Association of Securities Dealers Automated Quotations (NASDAQ) index and the Standard & Poor's 500 Composite Stock Price Index (S&P 500), which respectively exhibit oscillatory, upward and downward patterns. The results are compared with predictions obtained by other methods, including the neural network (NN) model, the factorization machine based neural network (FNN) model, the empirical mode decomposition based neural network (EMD2NN) model and the wavelet de-noising-based back propagation (WDBP) neural network model. Under the same conditions, the experiments indicate that the proposed methods perform better than the other ones according to the metrics of Mean Absolute Error (MAE), Root Mean Square Error (RMSE) and Mean Absolute Percentage Error (MAPE). Furthermore, we compute the profitability with a simple long-short trading strategy to examine the trading performance of our models in the metrics of Average Annual Return (AAR), Maximum Drawdown (MD), Sharpe Ratio (SR) and AAR/MD. The performances in two different scenarios, when taking or not taking the transaction cost into consideration, are found economically significant.

Keywords: Empirical Mode Decomposition, Factorization Machine, Neural Network, Stock Market Prediction, Profitability

1. Introduction

Stock market forecasting is always a remarkable topic and has attracted continuous attention in finance. Unfortunately, stock prices exhibit dynamic, non-linear, non-parametric and chaotic properties in nature (Oh & Kim, 2002; Wang, 2003). Many standard statistical and econometric models for forecasting must face significant challenges, such as disobeying the statistical assumptions in dealing with non-stationary time series, or having unsatisfactory forecasting performance due to the requirement of observations to be distributed normally.

There are numerous models and strategies proposed for the stock market predictions. Most of them can be classified into two categories: the ones based on statistical techniques and those using machine learning techniques. In the category of statistical approaches, there are autoregressive integrated moving average (ARIMA), generalized autoregressive conditional heteroskedasticity (GARCH) volatility (Franses & Ghijsels, 1999), and the smooth transition autoregressive model (STAR) (Sarantis, 2001), just to name a few. These approaches are primarily based on the assumptions of stationarity in time series and linearity among normally distributed variables. However, the stationarity, linearity and normality assumptions are not satisfied in real stock markets. On the other side, machine learning models without these restrictive assumptions have been proposed in recent years, and they can outperform the statistical methods (Hansen & Nelson, 2002; Zhang, 2003; Enke & Thawornwong,

☆This research was partially supported by NSFC (Nos. 11771458, 1431015), Guangdong Youth Innovation Talent Project (Nos. 2017KQNCX083), the Major Project of Basic and Applied Research in Guangdong Universities (Nos. 2017WZDXM012). Haomin Zhou has been partially supported by NSF Awards DMS-1419027, DMS-1620345, and ONR Award N000141310408.

*Corresponding author

Email addresses: fengzhou@gdufe.edu.cn (Feng Zhou), hmzhou@math.gatech.edu (Hao-min Zhou), yangzh@gdufe.edu.cn (Zhihua Yang), mcsylh@mail.sysu.edu.cn (Lihua Yang)

2005; Ture & Kurt, 2006). Thus, machine learning approaches, such as support vector machine (SVM) (Kim, 2003; Qian & Gao, 2017), genetic algorithm (Kim & Han, 2000), fuzzy system (Wang, 2002; Shen & Han, 2004), neural network (NN) (Vellido et al., 1999; Chen et al., 2003; Rather et al., 2015) and hybrid methods (Armano et al., 2005; Wang et al., 2011; Patel et al., 2015), have been widely employed in forecasting stock prices.

Although the machine learning based models have achieved remarkable results, there are still limitations. Firstly, they do not have an explicitly mechanism to handle the non-stationarity of stock prices. Secondly, as far as we know, most models do not pay attention to the interactions between features at different scales. For example, for the neural network models or deep network models, which are also widely used in many other applications such as image processing (Krizhevsky et al., 2012; Szegedy et al., 2015; He et al., 2015a), mechanical translation (Bahdanau et al., 2014; Luong et al., 2015), speech recognition (Hinton et al., 2012; Amodei et al., 2015) and so on, the non-linearities mainly handled by the activation functions, and there is few technique addressing the non-linear interactions among the inputs.

In this paper, we propose a 2-stage, end-to-end forecasting model combining Empirical Mode Decomposition (EMD) and Factorization Machine (FM) based Neural Network technique for stock market trend prediction. For convenience, we abbreviate it by EMD2FNN. In the first stage, we utilize EMD (Huang et al., 1998a), which is very efficient in handling the non-stationary data, to decompose the original financial time series into several components called intrinsic mode functions (IMFs). Each extracted IMF contains oscillatory patterns with scales in a narrow range, and it can be viewed as a quasi-stationary component. In the second stage, a FM-based Neural Network (FNN) (Rendle, 2010) technique is constructed using the values of IMFs as inputs to predict the future stock prices trend. Owning to the combination with the FM technology, it makes FNN capable of grasping the factorized interactions among inputs and efficient in computation due to its linear complexity.

There exist many EMD based techniques in dealing with nonlinear and non-stationary time series, and some have been used for data predictions (Huang et al., 1998b, 1999; Yang et al., 2004; Nunes et al., 2005; Yang et al., 2005b, 2006a,b; Bi et al., 2007; Zhang et al., 2008; Jaber et al., 2014). A common theme in these EMD prediction strategies is that each IMF is used as an independent time series, and every IMF has its own machine learning model for predicting. Different from the existing methods, the EMD2FNN uses one forecasting model, i.e., FNN, and all IMFs are fed as the inputs of this FNN. This design makes the prediction mechanism not only simple, easy to execute in practice, but also enable to capture the interactions among different scales.

To evaluate the performance, we apply the EMD2FNN to predict the closing prices from the empirical data sets of Shanghai Stock Exchange Composite (SSEC) index closing prices from January 4th 2012 to December 30th 2016, the National Association of Securities Dealers Automated Quotations (NASDAQ) index from January 4th 2012 to December 30th 2016, and the Standard & Poor's 500 Composite (S&P 500) index covering from January 3rd 2007 to December 30th 2011, which respectively cover oscillatory, upward and downward patterns. We compare the proposed model with several other approaches, such as the single NN and FNN models, and a hybrid model combining the empirical mode decomposition and neural network (EMD2NN) using the performance metrics such as the Mean Absolute Error (MAE), Root Mean Square Error (RMSE) and Mean Absolute Percentage Error (MAPE). Moreover, the EMD2FNN is compared with the wavelet de-nosing-based back propagation (WDBP) neural network model (Wang et al., 2011) which aims at using the wavelet transform to avoid the influences of noise. The results show significant improvements achieved by our model. In addition, we compute the profitability with a simple long-short trading strategy, in two different scenarios - taking or not taking the transaction cost into consideration, to examine the trading performance of our model in terms of the Average Annual Return (AAR), Maximum Drawdown (MD), Sharpe Ratio (SR) and AAR/MD. The performances are found economically significant as well.

The contributions of this paper include the following three points. (1) We propose EMD2FNN, a 2-stage, end-to-end model for the prediction of the stock market trend. The EMD2FNN is composed of EMD and FNN, two techniques that can work together for non-stationary data analysis. The EMD2FNN enjoys benefits from both EMD and FNN. (2) We demonstrate the prediction accuracy, in various metrics, of our methods for data sets that exhibit oscillatory, upward or downward patterns. The numerical experiments show significant improvements in prediction accuracy over the existing methods. (3) We illustrate the effectiveness of our method in a simple long-short trading strategy. The results are prominently better than the benchmark (i.e. buy-and-hold) strategy when evaluating the performance by either taking or not taking the transaction cost into consideration.

The rest of this paper is organized into following sections. We review the needed ingredients including EMD, FM and neural network in Section 2. Our EMD2FNN model is presented in Section 3, in which a couple of other hybrid approaches are discussed too. We show the simulation results in Section 4, and followed it by a brief conclusion in Section 5.

2. EMD, Neural Networks and FM

2.1 Empirical Mode Decomposition

As an alternative to the traditional methods, such as Fourier or wavelet transforms, EMD has been proven, by numerous studies, to be effective in analyzing non-stationary time series in recent years. It has received considerable attention in terms of interpretations (Flandrin et al., 2004; Chen et al., 2006; Sekine, 2007; Lin et al., 2009; Yang et al., 2014; Zhou et al., 2016) and applications in many disciplines such as ocean science (Huang et al., 1999), biomedicine (Huang et al., 1998b; Yang et al., 2006a), speech signal processing (Yang et al., 2004), image processing (Bi et al., 2007), pattern recognition (Nunes et al., 2005; Yang et al., 2005b, 2006b) and financial forecasting (Zhang et al., 2008; Jaber et al., 2014).

Many different algorithms have been proposed to carry out EMD. Examples include the strategies based on moving average (Smith, 2005), partial differential equation (PDE) (Delechelle et al., 2005; Diop et al., 2010), operators (Peng & Hwang, 2008; Hong et al., 2009; Peng & Hwang, 2010; Oberlin et al., 2012; Hu et al., 2013), filtering (Lin et al., 2009), and optimizations (Pustelnik et al., 2010; Daubechies et al., 2011; Hou & Shi, 2011; Wu, 2013; Ding & Selesnick, 2013; Pustelnik et al., 2014; Zhou et al., 2016). In this paper, we adopt the original EMD method given in (Huang et al., 1998b) for two reasons. Firstly, it is simple, intuitive and efficient in computation. Secondly, it has been demonstrated experimentally in a broad range of applications. Here is the algorithm.

Algorithm 1 Empirical Mode Decomposition

Require: Given a signal $x(t)$
1). Set $r(t) := x(t)$ and $k = 0$
while r(t) is not monotonous **do**
  2). Set $m(t) = r(t)$
  while m(t) is nontrivial **do**
    3). Interpolate between minima (resp. maxima), ending up with some 'envelope' $e_{min}(t)$ (resp. $e_{max}(t)$)
    4). Compute the average $m(t) = (e_{min}(t) + e_{max}(t))/2$
    5). Extract the detail $c(t) = r(t) - m(t)$, and denote $c(t)$ as $r(t)$
  end while
  6). Set $k = k + 1$
  7). Set $\mathrm{imf}_k(t) = c(t)$
  8). Set $r(t) = x(t) - \sum_{i=1}^{k} \mathrm{imf}_i(t)$
end while
Output: $x(t) = \sum_{i=1}^{k} \mathrm{imf}_i(t)$

In essence, the EMD method decomposes a complicated signal into a finite, and often small, number of intrinsic mode functions (IMFs), arranged from high frequencies to low frequencies, based on local characteristic scale, which is defined as the distance between two successive local extrema in the signal. An IMF is a function whose upper and lower envelopes are symmetric. Moreover, the number of zero-crossings and the number of extremes are equal or differ at most by one (Huang et al., 1998a). Each computed IMF contains oscillatory scales in a narrow range, and it is usually viewed as a quasi-stationary component. For example, an IMF extracted from an economic time series with a scale of three months can be viewed as the seasonal component.

2.2 Neural Networks

In general, neural networks possess attributes of learning, generating, parallel processing and error endurance, which make them powerful in solving complex problems. Over the past decades, neural networks have been widely used in many areas, including financial forecasting (Lu, 2010; Omidi et al., 2011; Wang et al., 2011; Chang et al., 2012). Hence, we take the neural network as the benchmark for comparing the models' forecasting accuracy.

The commonly referred neural network contains very diverse structures. For instance, selecting different number of network layers or different activation functions can generate different models with various approximation abilities. In this paper, we employ a neural network with 4-layer as depicted in Figure 1. The four layers are one input layer, two hidden layers and one output layer. We want to note that the pooling and dropout layers, which are often added in the convolutional neural network (CNN) when applied to image processing, are not included in the NN structure.

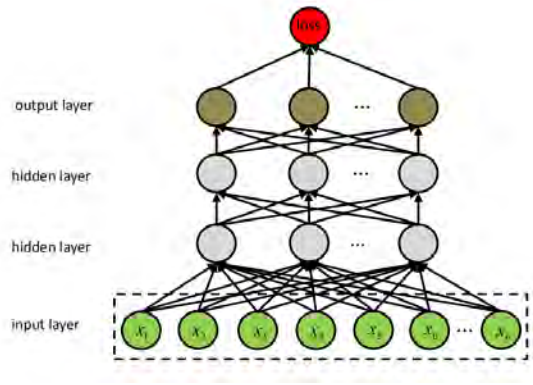

Figure 1: The architecture of neural network (NN).

In the neural network of Figure 1, the neurons of the input layer denote the input features, each neuron of the hidden and output layers is computed through combining the connection weights multiplied by the input values and nonlinear activation function, where the activation function aims to take the nonlinearity into consideration in the model. There are plenty of studies worked on the selection of activation functions, such as the sigmoid (Han & Moraga, 1995), tanh, Relu (Lecun et al., 2015), PRelu (He et al., 2015b), ELU (Clevert et al., 2015) and so on. The loss layer calculates the error between the training and estimated values to adjust the connection weights. In addition, a regularization term is often added to the loss layer to avoid over-fitting.

Assume that there are n input features, m_1 output neurons in the 1st hidden layer, m_2 output neurons in the 2nd hidden layer and o output neurons in the output layer, the forward (forecasting) process of the neural network can be described by the following four steps:

1). The 1st hidden layer: the outputs of the first activation layer are calculated by the following scheme:

$$y_j^I = f(\sum_{i=1}^{n} w_{ij}^I x_i),\ (j = 1, 2, \ldots, m_1), \tag{1}$$

where x_i denotes the value of the i-th feature, w_{ij}^I is the to-be-determined weight related to the i-th input feature and the j-th output neuron, y_j^I is the value of the j-th node and f is the activation function.

2). The 2nd hidden layer: the neurons in the second activation layer are computed by:

$$y_j^H = f(\sum_{i=1}^{m_1} w_{ij}^H y_i^I),\ (j = 1, 2, \ldots, m_2), \tag{2}$$

where w_{ij}^H is the undetermined weight related to the i-th input neuron and the j-th output neuron.

3). The output layer: the outputs of the output layer are given as follows:

$$y_j^O = f(\sum_{i=1}^{m_2} w_{ij}^O y_i^H),\ (j = 1, 2, \ldots, o), \tag{3}$$

where w_{ij}^O is the undetermined connection weight, y_j^O is the value of the j-th node of the output layer.

4). The loss layer: for different tasks, one can choose different loss functions ℓ to calculate the error between the actual and the estimated values from the output layer, such as squared loss, log-loss, softmax (Arjo, 2009). In order to avoid over-fitting, we add the L_2 regularized term in this layer. Hence, the loss function is computed as follows:

$$L(x, y) = \sum_{i=1}^{o} \ell(y_i^O, y_i) + \frac{\alpha}{2}(\sum_{i=1}^{n}\sum_{j=1}^{m_1}(w_{ij}^I)^2 + \sum_{i=1}^{m_1}\sum_{j=1}^{m_2}(w_{ij}^H)^2 + \sum_{i=1}^{m_2}\sum_{j=1}^{o}(w_{ij}^O)^2), \tag{4}$$

where $\alpha > 0$ is a regularized parameter, y_i is the actual values.

All undetermined weights in the NN model are iteratively updated according to the stochastic gradient descent (SGD) method within the back propagation process for the gradients. This process can be regarded as a special case of our improved FNN model described in Section 3.1. Therefore, we do not describe the weight updating process of the NN model here.

2.3 Factorization Machine

FM is originally introduced for collaborative recommendations (Rendle, 2010). Like the support vector machines, FMs form a general class of predictors that are able to estimate reliable parameters under the very high sparsity assumption. FM has a key advantage in learning feature interactions, due to its learning in the latent space. And this is one of the main reasons we select FM in our model. Given a real valued feature vector $\mathbf{x} \in \mathbb{R}^n$, FM estimates the target by modeling all interactions between each pair of features via factorized interaction parameters:

$$\hat{y}(\mathbf{x}) = w_0 + \sum_{i=1}^{n} w_i \cdot x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i \cdot x_j, \tag{5}$$

where the model parameters for calibration are: $w_0 \in \mathbb{R}$ is the global bias, $\mathbf{w} \in \mathbb{R}^n$ denotes linear interaction to the target. The inner product term $\langle \mathbf{v}_i, \mathbf{v}_j \rangle$ captures the factorized interaction, where each $\mathbf{v}_i \in \mathbb{R}^k$ is the latent vector for feature x_i, and k is a user-specified parameter for the dimension of the latent vector. The larger the value of k, the more sensitive the training model.

FM is very flexible, in contrast to the matrix factorization that models the relation of two entities only (He et al., 2017). FM can work with any real valued feature vectors for supervised learning. It enhances the linear or logistic regression using the second order factorized interactions among features. By specifying input features, the study in (Rendle, 2010) shows that FM can mimic many specific factorization models such as the standard matrix factorization, parallel factor analysis, and SVD++ (Koren, 2008). On the other hand, since the factorized interactive term can be rewritten as

$$\sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i \cdot x_j = \frac{1}{2} \sum_{f=1}^{k} ((\sum_{i=1}^{n} v_{if} \cdot x_i)^2 - \sum_{i=1}^{n} v_{if}^2 \cdot x_i^2), \tag{6}$$

this enables that the equation (5) can be computed in a linear time $O(k \cdot n)$. Owing to such a property, FM has been recognized as one of the most effective methods for sparse data prediction. It has brought a plethora of successful applications in industry, and has yielded great promise in a variety of prediction tasks, such as regression, classification and ranking (Oentaryo et al., 2014; Chen et al., 2016; Juan et al., 2016; Bayer et al., 2017).

3. Our Proposed Approaches

3.1 the FNN Model

Inspired by FM, we modify the NN model by incorporating FM ideas and name it as FNN for simplicity. The structure of FNN is shown in Figure 2. The main differences, compared against NN, are located in the hidden layers, where the gray nodes are still achieved by the nonlinear activation function's operation after the linear connections as those appeared in the NN model, but the blue nodes represent the results calculated through the activation function of the factorized interactions. Comparing to the NN model, we summarize the properties of the FNN as follows:

1) From the expression (5), the term $\sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i \cdot x_j$ makes FNN capture the role of factorized interactions between features, which is an advantage that most of the existing generalized linear models do not have.

2) According to equation (6), FNN has the same level (linear) of computation complexity as that of NN model.

The idea combining FM together with NN models has been recently reported for advertisement recommendation system (He & Chua, 2017), in which FM is used to modify the 1st hidden layer only. In this paper we advocate the idea by incorporating FM in other hidden layers as well. We use single-FNN to denote the model that FM is used only at the 1st hidden layer, while multi-FNN is for the case where FM is used in other layers.

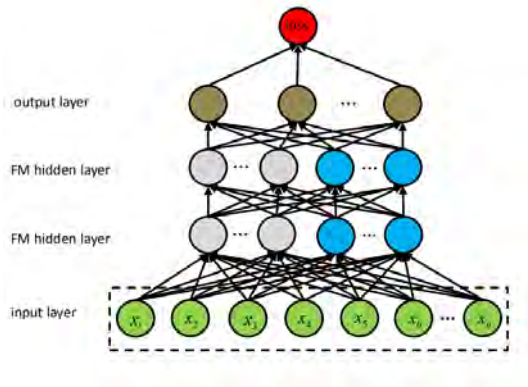

Figure 2: The architecture of FM based neural network (FNN).

Suppose that there are n input features, m_1 output neurons in the 1st hidden layer, m_2 output neurons in the 2nd hidden layer and o output neurons in the output layer. Next, we describe the forward (forecasting) process of FNN.

1). The 1st hidden layer: the neurons in the first hidden layer are partitioned into two parts:

$$y_j^I = f(\sum_{i=1}^{n} w_{ij}^I x_i),\ (j = 1, 2, \ldots, m_1 - k_1), \tag{7}$$

$$y_{m_1-k_1+j}^I = f(\frac{1}{2}((\sum_{i=1}^{n} v_{ij}^I\ x_i)^2 - \sum_{i=1}^{n}(v_{ij}^I)^2\ x_i^2)),\ (j = 1, 2, \ldots, k_1), \tag{8}$$

where x_i denotes the value of i-th feature, y_j^I is the value of the j-th node, w_{ij}^I is the undetermined linear weight related to the i-th input feature and the j-th output neuron, $v_i^I \in \mathbb{R}^{k_1}$ is the undetermined latent weight corresponding to the feature x_i, k_1 is the user-specified dimension of v_i^I, and f is the activation function.

2). The 2nd hidden layer: the outputs of the second hidden layer include the following two cases:

– single-FNN:

$$y_j^H = f(\sum_{i=1}^{m_1} w_{ij}^H y_i^I),\ (j = 1, 2, \ldots, m_2), \tag{9}$$

– multi-FNN:

$$y_j^H = f(\sum_{i=1}^{m_1} w_{ij}^H y_i^I),\ (j = 1, 2, \ldots, m_2 - k_2), \tag{10}$$

$$y_{m_2-k_2+j}^H = f(\frac{1}{2}((\sum_{i=1}^{m_1} v_{ij}^H\ y_i^I)^2 - \sum_{i=1}^{m_1}(v_{ij}^H)^2\ (y_i^I)^2)),\ (j = 1, 2, \ldots, k_2), \tag{11}$$

where w_{ij}^H is the undetermined linear weight related to the i-th input neuron and the j-th output neuron, $v_i^H \in \mathbb{R}^{k_2}$ is the undetermined latent weight corresponding to y_i^I, k_2 is the user-specified dimension of v_i^H.

3). The output layer: the output layer in FNN is the same as that in the NN model, i.e.,

$$y_j^O = f(\sum_{i=1}^{m_2} w_{ij}^O y_i^H),\ (j = 1, 2, \ldots, o), \tag{12}$$

where w_{ij}^O is the undetermined connection weight, y_j^O is the value of the j-th node of the output layer.

4). The loss layer: FNN can be applied to a variety of forecasting tasks by taking different loss functions ℓ, including classification, regression and ranking. For regression, a commonly used loss function is the squared loss defined by $\ell(y_i^O, y_i) = \frac{1}{2}(y_i^O - y_i)^2$. For classification task, the loss function is usually taken as the log-loss $\ell(y_i^O, y_i) = -y_i \log y_i^O - (1 - y_i) \log(1 - y_i^O)$. For ranking task, it optimizes contrastive max-margin loss or pairwise personalized ranking loss. Similar to the loss layer in the NN model, the loss layer in FNN is computed as follows:

$$\begin{aligned} L(x, y) = &\sum_{i=1}^{o} \ell(y_i^O, y_i) + \frac{\alpha}{2}\Big(\sum_{i=1}^{n}\sum_{j=1}^{m_1-k_1}(w_{ij}^I)^2 + \sum_{i=1}^{m_1}\sum_{j=1}^{m_2-k_2}(w_{ij}^H)^2 \\ &+ \sum_{i=1}^{m_2}\sum_{j=1}^{o}(w_{ij}^O)^2 + \sum_{i=1}^{n}\sum_{j=1}^{k_1}(v_{ij}^I)^2 + \sum_{i=1}^{m_1}\sum_{j=1}^{k_2}(v_{ij}^H)^2\Big), \end{aligned} \tag{13}$$

where $\alpha > 0$ denotes the regularized parameter, y_i is the actual values.

To optimize the FNN model, the back propagation process is used to iteratively calculating the gradient in each layer, and then the stochastic gradient descent (SGD) is adopted to update the weights until convergence. The backward (training) process of FNN would be given in Section 3.2.

3.2 the EMD2FNN Model

In this section, we present the EMD2FNN approach. It would be used for two tasks in this paper, determining the changes (directions) and predicting the future prices (trends) in stock prices. The former is a classification problem, and the later a regression task.

There are two stages in EMD2FNN. At the first stage, the original data sequence is decomposed into several components (IMFs) via the EMD algorithms given in **Algorithm 1**. The resulting IMFs are used as the inputs for the FNN model for the prediction. The detailed steps are presented in **Algorithm 2** accompanied by a flowchart shown in Figure 3.

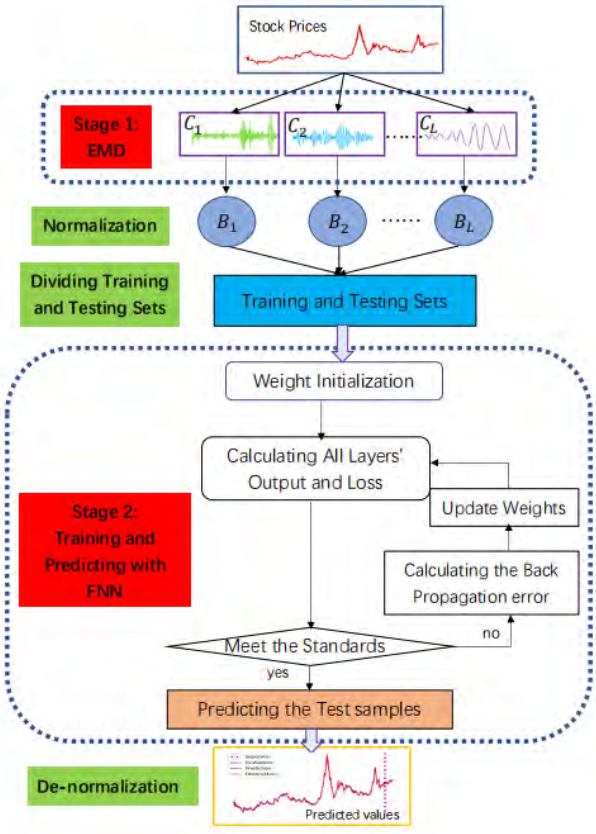

Figure 3: The flowchart of our proposed EMD2FNN.

Algorithm 2 EMD2FNN

Require: Given a stock prices series $\{x_i\}_{i=1}^{\tilde{n}}$

Stage one - the EMD process:

1). Following the **Algorithm 1**, decompose the original stock prices $\{x_i\}_{i=1}^{\tilde{n}}$ into several IMFs $\{C_j\}_{j=1}^{L}$, where $C_j \in \mathbb{R}^{\tilde{n}}$, and denote the number of IMFs as L.

Stage two - the FNN process:

2). Each IMF C_j is normalized into $B_j \in [0,1]$ defined by $B_j = \frac{C_j - \min(\{C_j\}_{j=1}^{L})}{\max(\{C_j\}_{j=1}^{L}) - \min(\{C_j\}_{j=1}^{L})}$.

3). Take the set constituted of M consecutive values from each IMF (i.e., $\{B_{j,k}, B_{j,k+1}, \ldots, B_{j,k+M-1}\}_{j=1}^{L}$) as the input features, and the set constituted of next o consecutive values from original stock prices (i.e., $\{x_{k+M}, x_{k+M+1}, \ldots, x_{k+M+o-1}\}$) as the labels for regression task, where $k = 1, 2, \ldots, \tilde{n} - M - o + 1$. For classification task, the labels are taken as the direction of the stock price change (i.e., 1 if $x_{k+M} \geq x_{k+M-1}$; -1 otherwise), where $k = 1, 2, \ldots, \tilde{n} - M$.

4). Divide the samples into training and testing data sets.

5). The FNN model is learnt on the training data.

6). According to the forward process of FNN described in Section 3.1, using the testing data to measure how well the network is able to extrapolate the unknown samples.

7). For regression task, the predicted values are scaled back to the original range.

The step 5) (i.e., the FNN model) appeared at the stage two in **Algorithm 2** is the only part involving machine learning. The backward (training or learning) process of FNN is detailed in the following points.

1). The gradients of the inputs and weights in the output layer are computed as:

$$\begin{aligned} \frac{\partial L}{\partial y_j^O} =& \frac{\partial \ell}{\partial y_j^O},\ \frac{\partial L}{\partial w_{ij}^O} = \frac{\partial L}{\partial y_j^O} \cdot \left.\frac{\partial f}{\partial z}\right|_{z=\sum_{h=0}^{m_2} w_{hj}^O y_h^H} \cdot y_i^H + \alpha \cdot w_{ij}^O; \\ w_{ij}^O \leftarrow& w_{ij}^O - \eta \cdot \frac{\partial L}{\partial w_{ij}^O}. \end{aligned} \tag{14}$$

where $i = 1, 2, \ldots, m_2,\ j = 1, 2, \ldots, o$ and η denotes the learning rate.

2). The gradienst of the inputs and weights in the 2nd hidden layer are calculated as the following two cases:

$$\frac{\partial L}{\partial y_j^H} = \sum_{i=1}^{o} \frac{\partial L}{\partial y_i^O} \cdot \left.\frac{\partial f}{\partial z}\right|_{z=\sum_{h=0}^{m_2} w_{hj}^O y_h^H} \cdot w_{ji}^O, \tag{15}$$

– single-EMD2FNN:

$$\frac{\partial L}{\partial w_{ij}^H} = \frac{\partial L}{\partial y_j^H} \cdot \left.\frac{\partial f}{\partial z}\right|_{z=\sum_{h=0}^{m_1} w_{hj}^H y_h^I} \cdot y_i^I + \alpha \cdot w_{ij}^H,\ w_{ij}^H \leftarrow w_{ij}^H - \eta \cdot \frac{\partial L}{\partial w_{ij}^H}, \tag{16}$$

– multi-EMD2FNN:

$$\begin{aligned} \frac{\partial L}{\partial w_{ij_1}^H} =& \frac{\partial L}{\partial y_{j_1}^H} \cdot \left.\frac{\partial f}{\partial z}\right|_{z=\sum_{i=0}^{m_1} w_{hj_1}^H y_h^I} \cdot y_i^I + \alpha \cdot w_{ij_1}^H,\ \bar{z}_{j_2} = \frac{1}{2}((\sum_{h=1}^{m_1} v_{hj_2}^H\ y_h^I)^2 - \sum_{h=1}^{m_1} (v_{hj_2}^H)^2\ (y_h^I)^2), \\ \frac{\partial L}{\partial v_{ij_2}^H} =& \frac{\partial L}{\partial y_{(m_2-k_2+j_2)}^H} \cdot \left.\frac{\partial f}{\partial z}\right|_{z=\bar{z}_{j_2}} \cdot [(\sum_{h=1}^{m_1} v_{hj_2}^H y_h^I) y_i^I - v_{ij_2}^H (y_i^I)^2] + \alpha \cdot v_{ij_2}^H, \\ w_{ij_1}^H \leftarrow& w_{ij_1}^H - \eta \cdot \frac{\partial L}{\partial w_{ij_1}^H},\ v_{ij_2}^H \leftarrow v_{ij_2}^H - \eta \cdot \frac{\partial L}{\partial v_{ij_2}^H}, \end{aligned} \tag{17}$$

where $i = 1, 2, \ldots, m_1,\ j_1 = 1, 2, \ldots, m_2 - k_2,\ j_2 = 1, 2, \ldots, k_2$ and η is the learning rate.

3). The gradients of the inputs and weights in the 1st hidden layer are computed by:

– single-EMD2FNN:

$$\frac{\partial L}{\partial y_j^I} = \sum_{i=1}^{m_2} \frac{\partial L}{\partial y_i^H} \cdot \left.\frac{\partial f}{\partial z}\right|_{z=\sum_{h=0}^{m_1} w_{hj}^H y_h^I} \cdot w_{ij}^H, \tag{18}$$

– multi-EMD2FNN:

$$\begin{aligned}
\hat{z}_{j_1} =& \sum_{h=0}^{m_1} w^H_{hj_1} y^I_h,\ \tilde{z}_{j_2} = \frac{1}{2}((\sum_{h=1}^{m_1} v^H_{hj_2} \cdot y^I_h)^2 - \sum_{h=1}^{m_1} (v^H_{hj_2})^2 \cdot (y^I_h)^2), \\
\frac{\partial L}{\partial y^I_j} =& \sum_{i=1}^{m_2-k_2} \frac{\partial L}{\partial y^H_i} \left.\frac{\partial f}{\partial z}\right|_{z=\hat{z}_i} w^H_{ij} + \sum_{i=1}^{k_2} \frac{\partial L}{\partial y^H_{m_2-k_2+i}} \left.\frac{\partial f}{\partial z}\right|_{z=\tilde{z}_i} [(\sum_{h=1}^{m_1} v^H_{hj} y^I_h) v^H_{ij} - (v^H_{ij})^2 y^I_i],
\end{aligned} \tag{19}$$

$$\begin{aligned}
\frac{\partial L}{\partial w^I_{ij_3}} =& \frac{\partial L}{\partial y^I_{j_3}} \cdot \left.\frac{\partial f}{\partial z}\right|_{z=\sum_{h=0}^{n} w^I_{hj_3} x_h} \cdot x_i + \alpha \cdot w^I_{ij_3},\ \bar{z}_{j_4} = \frac{1}{2}((\sum_{h=1}^{n} v^I_{hj_4} \cdot x_h)^2 - \sum_{h=1}^{n} (v^I_{hj_4})^2 \cdot x_h)^2), \\
\frac{\partial L}{\partial v^I_{ij_4}} =& \frac{\partial L}{\partial y^I_{(m_1-k_1+j_4)}} \cdot \left.\frac{\partial f}{\partial z}\right|_{z=\bar{z}_{j_4}} \cdot [(\sum_{h=1}^{n} v^I_{hj_4} x_h) x_i - v^I_{ij_4} (x_i)^2] + \alpha \cdot v^I_{ij_4}, \\
w^I_{ij_3} \leftarrow& w^I_{ij_3} - \eta \cdot \frac{\partial L}{\partial w^I_{ij_3}},\ v^I_{ij_4} \leftarrow v^I_{ij_4} - \eta \cdot \frac{\partial L}{\partial v^I_{ij_4}},
\end{aligned} \tag{20}$$

where $i = 1, 2, \ldots, n$, $j = 1, 2, \ldots, m_1$, $j_1 = 1, 2, \ldots, m_2 - k_2$, $j_2 = 1, 2, \ldots, k_2$, $j_3 = 1, 2, \ldots, m_1 - k_1$, $j_4 = 1, 2, \ldots, k_1$, η denotes the learning rate, and the single-EMD2FNN shares the equation (20) with the case of multi-EMD2FNN.

It is worth noting that the EMD2FNN need only one prediction model, in which all normalized IMFs are fed as the inputs. This gives several advantages for EMD2FNN:

- It makes the prediction mechanism simple and easy to control in practice.
- Since each IMF is not completely independent to the others, EMD2FNN is able to excavate the interactions among IMFs.

As mentioned in section 3.1, both single-FNN and multi-FNN structures can be used in the second stage for prediction. We call them single-EMD2FNN and multi-EMD2FNN respectively.

We want to remark that after replacing the FNN model by NN, one can construct another hybrid approach called EMD2NN that will be compared with our model in the next section. In fact, the idea of using EMD together with neural network has been recently reported in a few applications, e.g., wind speed series prediction (Liu et al., 2012), tourism demand prediction (Chen et al., 2012), and water temperature prediction (Liu et al., 2016). A common structure in those studies is that every IMF has its own machine learning model for predicting, and the neural network models are completely independent of each other, including the training and predicting stages. In contrast, our EMD2FNN uses FM to exploit the interactions between IMFs and that helps to improve the prediction accuracy significantly.

4. Simulation Results and Evaluation

4.1 Experimental Data

The data[1] for our experiments are the Shanghai Stock Exchange Composite (SSEC) index, the National Association of Securities Dealers Automated Quotations (NASDAQ) index and the Standard & Poor's 500 Composite Stock Price Index (S&P 500). For the SSEC and NASDAQ data sets, their total number of values are 1214 collected from trading days ranged from January 4th 2012 to December 30th 2016. For the S&P 500, the size of series is 1260. It is taken from January 3rd 2007 to December 30th 2011. Figure 4 shows the original time series, from which it covers oscillatory (top panel), upward (middle panel) and downward (bottom panel) patterns. We partition them into training sets (80% of the total trading days) and testing sets (20% of the total trading days). Table 1 describes their statistics, where N_{all} and N_{out} denote the sizes of the whole data and the testing samples respectively.

[1] All the data in this paper are downloaded from www.finance.yahoo.com.

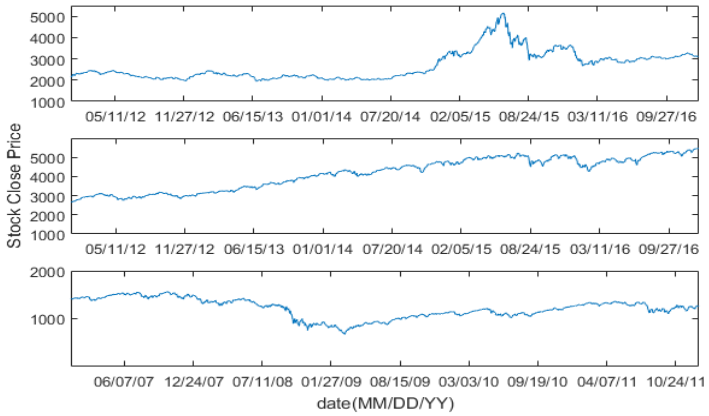

Figure 4: The original data. Top: SSEC. Middle: NASDAQ. Bottom: S&P500.

Table 1: The data and their descriptive statistical analysis.

Name	N_{all}	N_{out}	**Mean**	**Std.**	**Data range**
SSEC	1214	242	2665.19	667.26	2012.01.04-2016.12.30
NASDAQ	1214	242	4161.88	825.13	2012.01.04-2016.12.30
S&P500	1260	252	2677.27	672.07	2007.01.03-2011.12.30

4.2 the Setup in Models

To evaluate the performance of the proposed single-EMD2FNN and multi-EMD2FNN models, we first compare them with other models in forecasting the stock prices to determine the prediction accuracy. And then, they are used in predicting the trend of the price change to prove their reliability in trading with a simple long-short trading strategy.

Suppose that the stock price series is decomposed into L IMFs via the EMD, we take the set constituted of M consecutive values from each IMF as the input samples, the activation function f from the hidden layers as tanh. The activation function f, the size of output neurons in the output layer and the loss function in the loss layer are set as the identity function, M and squared loss respectively for the case of stock price prediction, i.e., $f(x) = x$, $o = M$ and $\ell(y_i^O, y_i) = \frac{1}{2}(y_i^O - y_i)^2$, where y_i is the true stock price and y_i^O denotes the predicted stock price. In the case of trend prediction, they are taken as the sigmoid function, 1 and the log-loss respectively, i.e., $f(x) = 1/(1 + \exp(-x))$, $o = 1$ and $\ell(y_i^O, y_i) = -y_i \log y_i^O - (1 - y_i)\log(1 - y_i^O)$, where y_i is the true direction of the price change, i.e., $y_i = 1$ if the $(i + 1)$-th price is larger than the i-th price, and $y_i = -1$ otherwise, y_i^O denotes the predicted value.

After the training and testing data sets divided, the FNN model will be learnt on the training samples with the input size $n = M * L$. In this paper, all undetermined weights are assigned with normal distributed random values initially, and then updated according to the back-propagation process on the training samples, which is concretely depicted in Section 3.1.2. For the hyper-parameters m_1, m_2, k_1, k_2 mentioned in Section 3.1.1 and 3.1.2, to reduce the FNN model's complexity in our experiments, we fix $m_2 = \lfloor \frac{m_1}{2} \rfloor$, $k_1 = m_1 - 1$, $k_2 = m_2 - 1$. The rest values of hyper-parameter m_1, the learning rate η in back-propagation process and the regularized parameter α are obtained by the grid search algorithm, which aims at finding the optimal parameters from various parameter combinations via minimizing the squared loss on training set.

For the parameters in the NN model, including the parameters m_1, m_2, o, and the activation functions f from the hidden layers and the output layer, we take the similar setup as FNN, i.e., $m_2 = \lfloor \frac{m_1}{2} \rfloor$, the activation functions from the hidden layers are set to be tanh, the activation function f and the size of output neurons in the output layer and the loss function in the loss layer are set as $f(x) = x$, $o = M$ and $\ell(y_i^O, y_i) = \frac{1}{2}(y_i^O - y_i)^2$ respectively for the case of stock price prediction; as $f(x) = 1/(1+\exp(-x))$, $o = 1$ and $\ell(y_i^O, y_i) = -y_i \log y_i^O - (1-y_i)\log(1-y_i^O)$ respectively for the case of predicting the direction of price change.

4.3 Evaluation Criteria

Three accuracy measures are chosen to evaluate the predicted values $\hat{y} \in \mathbb{R}^{o \times N}$ relative to the actual closing price $y \in \mathbb{R}^{o \times N}$, where N is the sample size. They are mean absolute error (MAE), root mean square error (RMSE) and mean absolute percentage error (MAPE). Their definitions are given in Table 2. In our experiments, we use all three measures to judge the prediction performance. Smaller values of these indexes indicate more accurate forecast. When the results are not consistent among the indexes, we choose the relatively more stable one, MAPE as suggested by Makridakis (Makridakis, 1993), to be the main reference substance.

Table 2: Evaluation indexes.

Measure	**Expression**
MAE	$\frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{o}\lvert\hat{y}_{in}-y_{in}\rvert$
RMSE	$\sqrt{\frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{o}(\hat{y}_{in}-y_{in})^2}$
MAPE	$\frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{o}\lvert\frac{\hat{y}_{in}-y_{in}}{y_{in}}\rvert$

In addition, Table 3 lists all statistical metrics used for examining the trading performance in this study. The Sharpe Ratio (SR) measures the risk-adjusted return. AAR/MD is slightly modified from the Calmar ratio and is calculated as the Average Annual Return (AAR) derived by the Maximum Drawdown (MD) for the whole duration considered. For the MD, defined as the largest accumulated percentage loss due to a sequence of drops over a investment horizon, a lower output means a better performance. For the Average Annual Return, Sharpe Ratio and AAR/MD, a higher output is better.

Table 3: Description of the measures of trading performance. r_i is the return in year i; r_m denotes the realized return from the trade; r_b is the return of risk-free; σ is the standard deviation of the difference $E[r_m - r_b]$; R_t is the return at date t.

Statistical measure	**Describe**
Trade Counts	Total number of entered trades
Average Annual Return	AAR=$(\prod_{i=1}^{N}(1+r_i))^{\frac{1}{N}}-1$
Sharpe Ratio	SR=$\frac{E[r_m-r_b]}{\sigma}$
Maximum Drawdown	MD=$\max_{\tau\in(0,T)}(\max_{\tau\in(t,\tau)}(R_t-R_\tau))$
Average Annual Return/Maximum Drawdown	AAR/MD

4.4 Empirical Mode Decomposition

From Figure 4, it can be seen that the data includes long term tendencies, periodic vibrations and noise. These patterns can be extracted from the observed data by EMD. The IMFs of SSEC are depicted in Figure 5 as the frequencies changing from high to low, and plots in Figure 6 show the IMFs of NASDAQ and S&P 500 by the EMD technique.

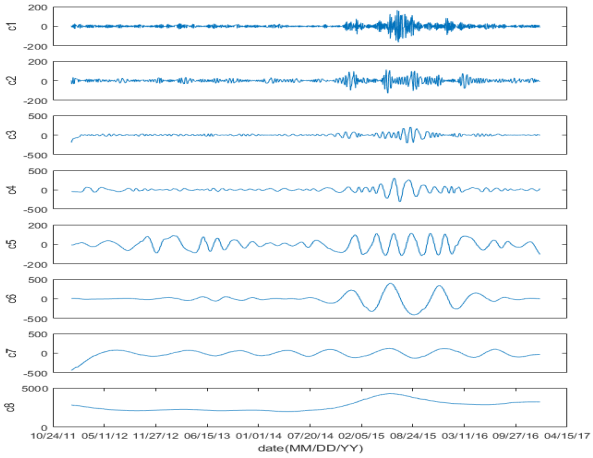

Figure 5: The IMFs of SSEC data.

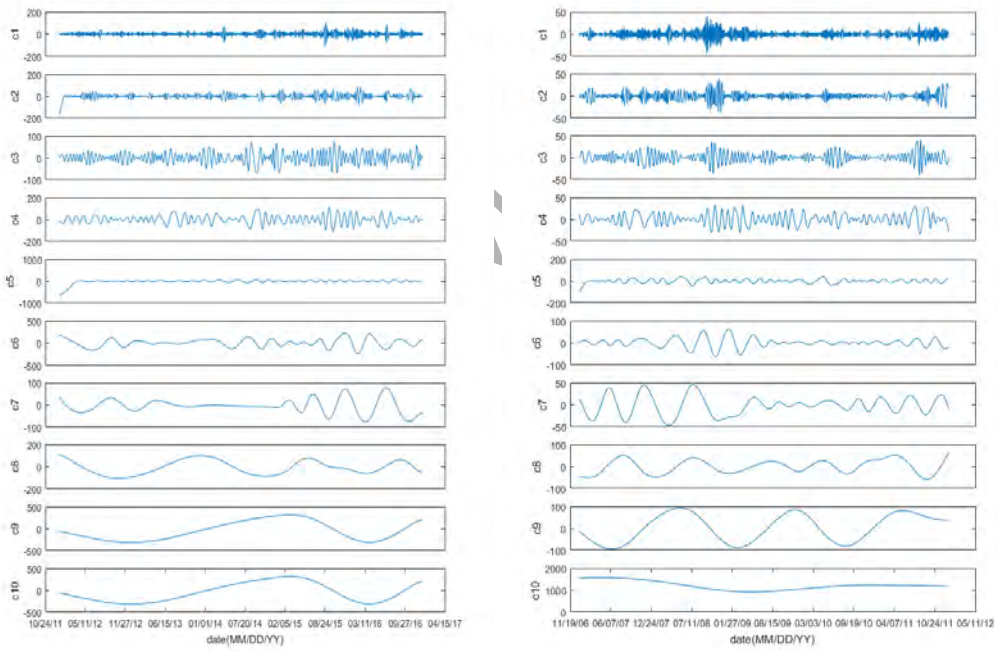

Figure 6: Left: the IMFs of NASDAQ data; Right: the IMFs of S&P500 data.

4.5 Results by the EMD2FNN Model

After the FNN model set up and the IMFs obtained from the EMD technique, we discuss how to select the window size M. Table 4-6 describe the results obtained from the multi-EMD2FNN model on SSEC, NASDAQ and S&P 500 of selecting M as $3, 4, 5$ respectively. Figure 7 depicts the results graphically with bar charts. From them, the results of $M = 3$ outperform the others, so we take $M = 3$ in this paper.

Table 4: The performance of different window sizes M on SSEC.

M	training data			testing data		
	MAE	**RMSE**	**MAPE**	**MAE**	**RMSE**	**MAPE**
3	35.0131	54.8787	0.0128	37.1778	61.5138	0.0123
4	52.2786	73.7618	0.0193	43.5513	66.7576	0.0143
5	51.5614	71.8362	0.0195	54.8499	89.9065	0.0152

Table 5: The performance of different window sizes M on NASDAQ.

M	training data			testing data		
	MAE	**RMSE**	**MAPE**	**MAE**	**RMSE**	**MAPE**
3	36.4873	49.4673	0.0092	52.4773	70.4576	0.0108
4	41.0351	55.9020	0.0104	62.4763	82.4402	0.0128
5	50.7281	62.8714	0.0135	65.0425	83.1258	0.0150

Table 6: The performance of different window sizes M on S&P500.

M	training data			testing data		
	MAE	**RMSE**	**MAPE**	**MAE**	**RMSE**	**MAPE**
3	13.0210	17.5318	0.0117	13.0396	17.6591	0.0105
4	15.1725	20.3924	0.0137	15.1386	20.3978	0.0122
5	15.5838	20.7445	0.0139	16.1700	22.1277	0.0130

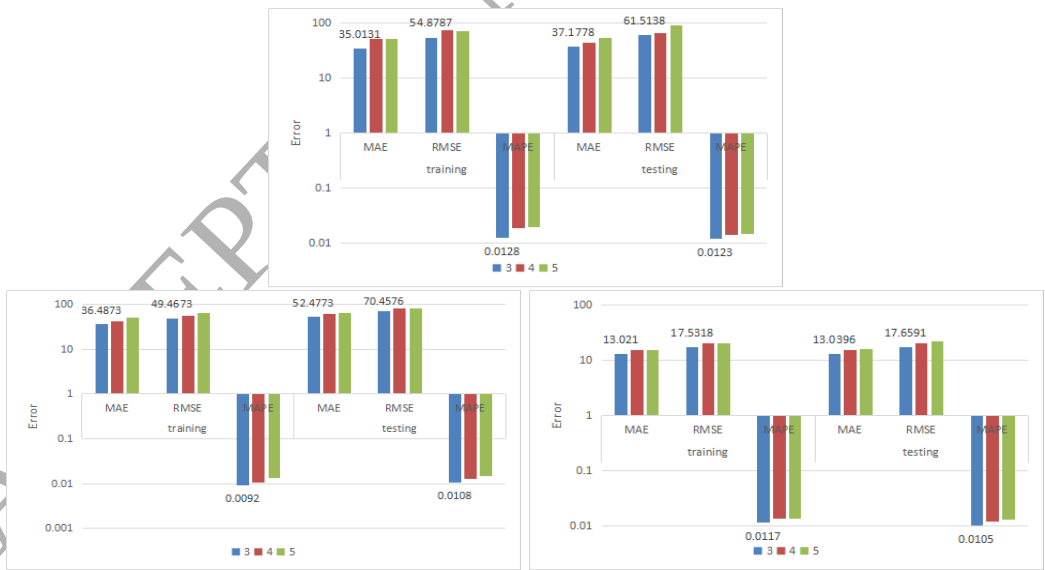

Figure 7: The graphic illustrations of the performance of selecting different M. Top: SSEC; Bottom left: NASDAQ; Bottom right: S&P 500.

Figure 8-10 depict the results of the single-EMD2FNN and multi-EMD2FNN, where the blue curves denote the observations, the cyan ones characterize the results of the single-EMD2FNN, and the red ones are the results of the multi-EMD2FNN model. The errors are also shown in the figures. It can be seen from the figures that both the fittings and predictions look reasonably accurate, and the results obtained by the models are similar.

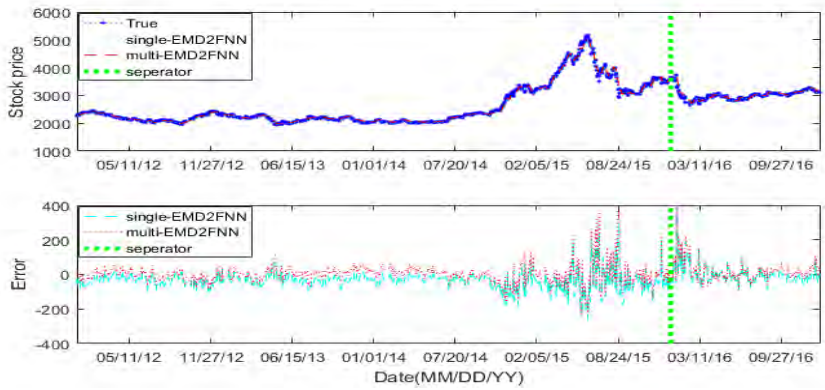

Figure 8: The performances of EMD2FNN models on SSEC. The top panel depicts the fitting and prediction capabilities of different models, and the bottom panel gives the errors of fitting and prediction from different models, where the blue curve denotes the observation, the cyan and red ones represent the performances from the single-EMD2FNN and multi-EMD2FNN models respectively, the green dotted line is the boundary between the training and testing sets.

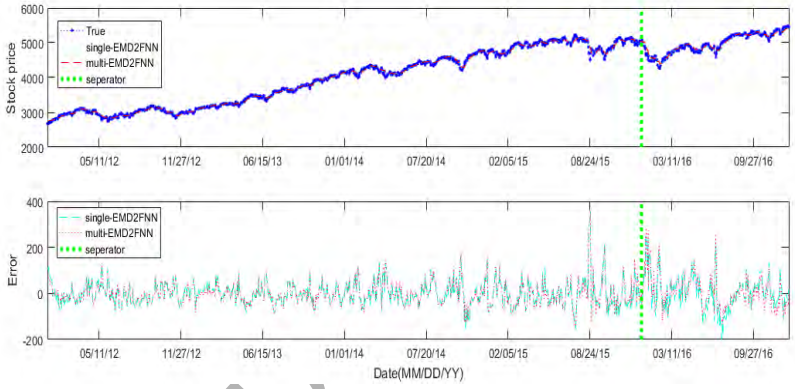

Figure 9: The performances of EMD2FNN models on NASDAQ. The top panel depicts the fitting and prediction capabilities of different models, and the bottom panel gives the errors of fitting and prediction from different models, where the blue curve denotes the observation, the cyan and red ones represent the performances from the single-EMD2FNN and multi-EMD2FNN models respectively, the green dotted line is the boundary between the training and testing sets.

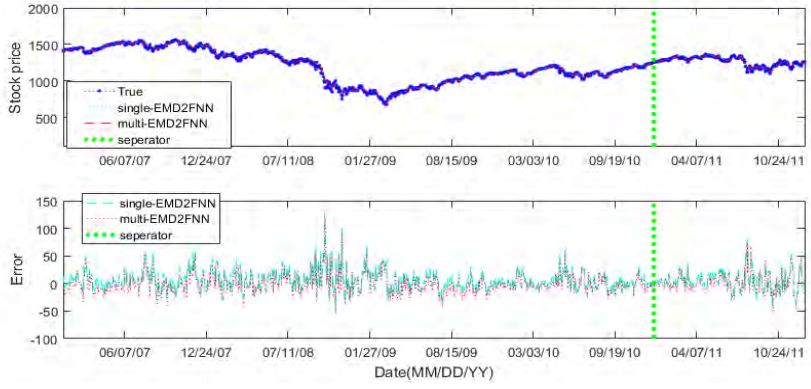

Figure 10: The performances of EMD2FNN models on S&P 500. The top panel depicts the fitting and prediction capabilities of different models, and the bottom panel gives the errors of fitting and prediction from different models, where the blue curve denotes the observation, the cyan and red ones represent the performances from the single-EMD2FNN and multi-EMD2FNN models respectively, the green dotted line is the boundary between the training and testing sets.

4.6 the Discussion of Predicted Accuracy

In order to evaluate the predicted accuracy of the proposed EMD2FNN (single-EMD2FNN and multi-EMD2FNN) models, we compare them with the NN, FNN (single-FNN and multi-FNN), and the EMD2NN models.

For the NN, single-FNN, multi-FNN, EMD2NN and EMD2FNN-based models, their predicted values on SSEC, NASDAQ and S&P 500 are depicted in Figure 11, 12 and 13 respectively with the corresponding colors and curve types, where each top panel shows the predicted values on the whole testing data set, each bottom panel depicts the results zoomed in to see detailed local values. In order to clearly observe the performance of the proposed models, we mark the ranges that the proposed models prominently surpass the other ones, and vice versa. We use colored rectangles to box the regions where the EMD2FNN is better than the others (red), the others are better than EMD2FNN (blue), and they can't be clearly tell which one is better (green). According to the results, we have that:

1). From the top panels of Figures 11-13, it indicates that all neural network based models have good prediction ability, especially for the trend of stock prices.

2). In the bottom panel of Figure 11, the SSEC values ranged from March 1st, 2016 to May 20th, 2016, which contain both upward and downward patterns, there are more red rectangles covering the majority of the interval. This indicates that the EMD2FNN models predicted values closer to the true stock prices than those obtained by the other models.

3). The bottom panel of Figure 12 is the NASDAQ data ranged from September 10th, 2016 to December 30th, 2016. In the figure, there are three red rectangle boxes showing that the predicted values from the EMD2FNN models are better than those from the other models. And the overall coverage of the red rectangle boxes is the largest.

4). The data of S&P 500 from January 3rd, 2007 to April 5th, 2007 is shown in the bottom panel of Figure 13. There are three red rectangles, two blue boxes and two green ones. Again, the range covered by the red boxes is the largest one showing the EMD2FNN produce better predication results in general.

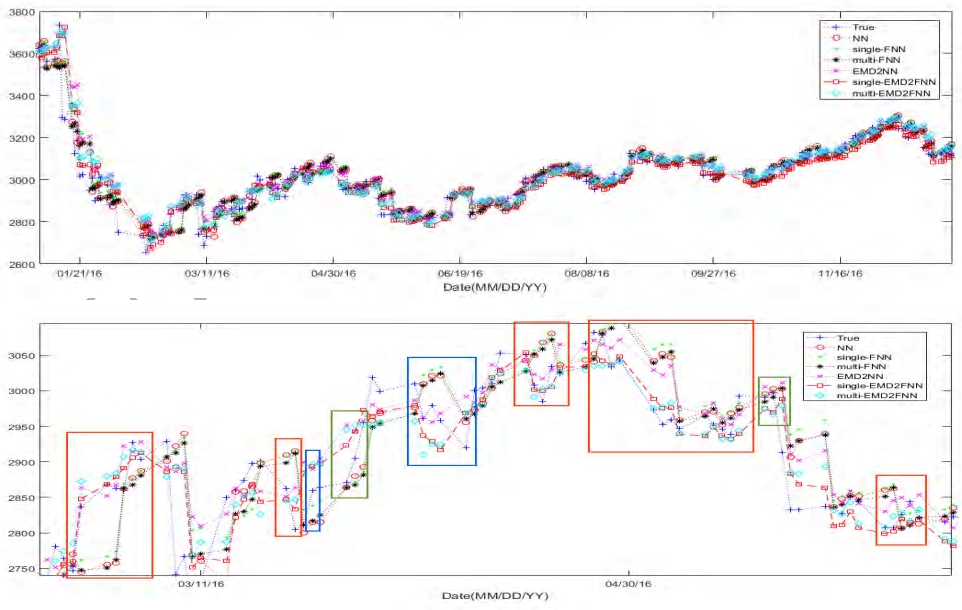

Figure 11: The performance of the models including NN, single-FNN, multi-FNN, EMD2NN, single-EMD2FNN and multi-EMD2FNN on SSEC. Top: all the predicted values covering the whole testing data set; Bottom: the detail with enlarged scale of the top panel, where red rectangle boxes denote the EMD2FNN models surpass the other ones, the green ones denote the other models exceed the EMD2FNN models, and the blue ones denote that it is blurry to judge which model predicts better.

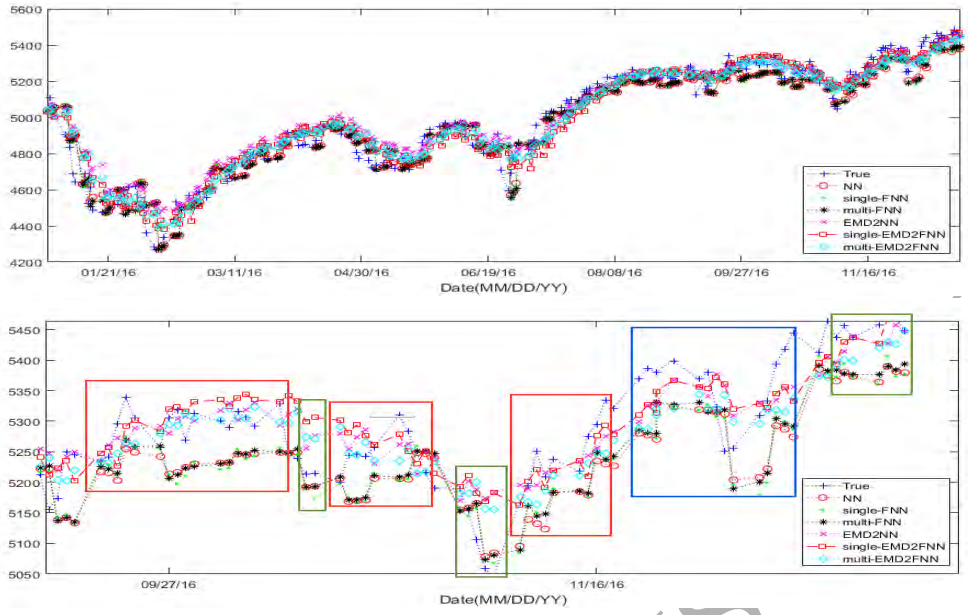

Figure 12: The performance of the models including NN, single-FNN, multi-FNN, EMD2NN, single-EMD2FNN and multi-EMD2FNN on NASDAQ. Top: all the predicted values covering the whole testing data set; Bottom: the detail with enlarged scale of the top panel, where red rectangle boxes denote the EMD2FNN models surpass the other ones, the green ones denote the other models exceed the EMD2FNN models, and the blue ones denote that it is blurry to judge which model predicts better.

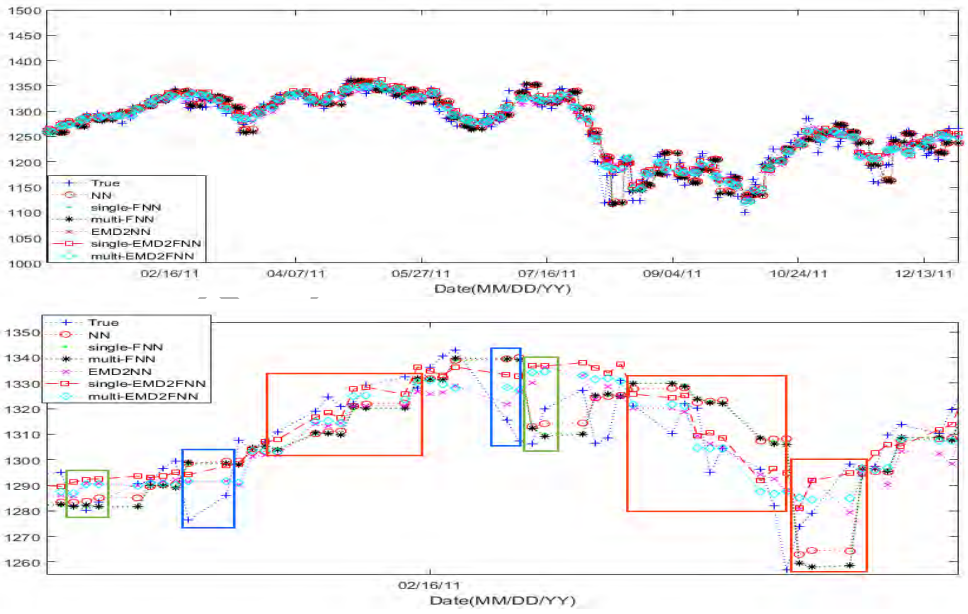

Figure 13: The performance of the models including NN, single-FNN, multi-FNN, EMD2NN, single-EMD2FNN and multi-EMD2FNN on S&P 500. Top: all the predicted values covering the whole testing data set; Bottom: the detail with enlarged scale of the top panel, where red rectangle boxes denote the EMD2FNN models surpass the other ones, the green ones denote the other models exceed the EMD2FNN models, and the blue ones denote that it is blurry to judge which model predicts better.

In order to know more clearly about the effects from FM and EMD techniques, we evaluate the models with the indexes listed in Table 2. Table 7-9 show the gains of the single-FNN, multi-FNN, EMD2NN, single-EMD2FNN and mulit-EMD2FNN models compared with the benchmark (NN) model on SSEC, NASDAQ and S&P 500. The smaller the gain value, the better the model improves. Besides, Figure 14 plots the metrics' values with bar chart. From the results, it can conclude that:

- Compared with the benchmark model, the values indicate both EMD and FM techniques can improve the accuracies. For the EMD2FNN based models, the MAPE indexes are respectively improved as high as 1.9‰ and 1.0‰ for the training and testing data in the SSEC data set. The same conclusion can be obtained for the NASDAQ data set (the accuracies of MAPE are improved as high as 0.4‰ and 1.6‰ for the training and testing data respectively), and S&P 500 data set (the accuracies of MAPE are improved as high as 3.7‰ and 3.6‰ for the training and testing data respectively).
- Compared with the other models, the gain values of the EMD2FNN based models are basically smaller. It indicates the single-EMD2FNN or multi-EMD2FNN are the best among these models, and both EMD and FM techniques are useful in improving the accuracies.
- Consider the differences between multi-EMD2FNN and single-EMD2FNN structures, the gain values of multi-EMD2FNN model are generally smaller than those of single-EMD2FNN model. It indicates the multi-EMD2FNN model is more robust than single-EMD2FNN.

Table 7: The performances of different models on SSEC.

Models	**training data**			**testing data**		
	MAE Gain	**RMSE Gain**	**MAPE Gain**	**MAE Gain**	**RMSE Gain**	**MAPE Gain**
Benchmark: NN	0.0000	0.0000	0.0‰	0.0000	0.0000	0.0‰
single-FNN	-0.1408	-0.5235	-0.0‰	-0.4923	0.4900	-0.1‰
multi-FNN	1.001	0.1077	-0.0‰	0.1374	**-0.1435**	-0.1‰
EMD2NN	-7.0951	-16.1093	**-2.1‰**	0.3309	10.0282	-0.2‰
single-EMD2FNN	-0.9250	-14.5796	-1.9‰	**-2.8577**	1.6113	**-1.0‰**
multi-EMD2FNN	**-7.1983**	**-19.8979**	-1.8‰	**-1.5161**	5.0094	**-0.4‰**

Table 8: The performances of different models on NASDAQ.

Models	**training data**			**testing data**		
	MAE Gain	**RMSE Gain**	**MAPE Gain**	**MAE Gain**	**RMSE Gain**	**MAPE Gain**
Benchmark: NN	0.0000	0.0000	0.0‰	0.0000	0.0000	0.0‰
single-FNN	0.1552	-1.1318	0.1‰	-3.7139	-2.6285	-0.7‰
multi-FNN	-0.1061	-0.1676	-0.1‰	-1.8645	-0.8479	-0.3‰
EMD2NN	1.8459	-0.9002	1.3‰	-3.7176	2.0391	-0.5‰
single-EMD2FNN	**-1.1246**	**-3.0685**	**-0.2‰**	**-5.1145**	**-4.4529**	**-0.9‰**
multi-EMD2FNN	**-1.6514**	**-3.3447**	**-0.4‰**	**-8.6644**	**-6.8494**	**-1.6‰**

Table 9: The performances of different models on S&P500.

Models	**training data**			**testing data**		
	MAE Gain	**RMSE Gain**	**MAPE Gain**	**MAE Gain**	**RMSE Gain**	**MAPE Gain**
Benchmark: NN	0.0000	0.0000	0.0‰	0.0000	0.0000	0.0‰
single-FNN	0.012	-0.1202	-0.1‰	-0.1356	-0.2523	-0.2‰
multi-FNN	-0.0053	-0.1218	-0.1‰	-0.1223	-0.3279	-0.1‰
EMD2NN	-2.2764	-4.0868	-2.7‰	-3.2361	-5.0912	-2.7‰
single-EMD2FNN	**-3.9365**	**-4.9230**	**-3.6‰**	**-3.8347**	**-5.5865**	**-3.1‰**
multi-EMD2FNN	**-4.0252**	**-5.8653**	**-3.7‰**	**-4.3960**	**-6.3499**	**-3.6‰**

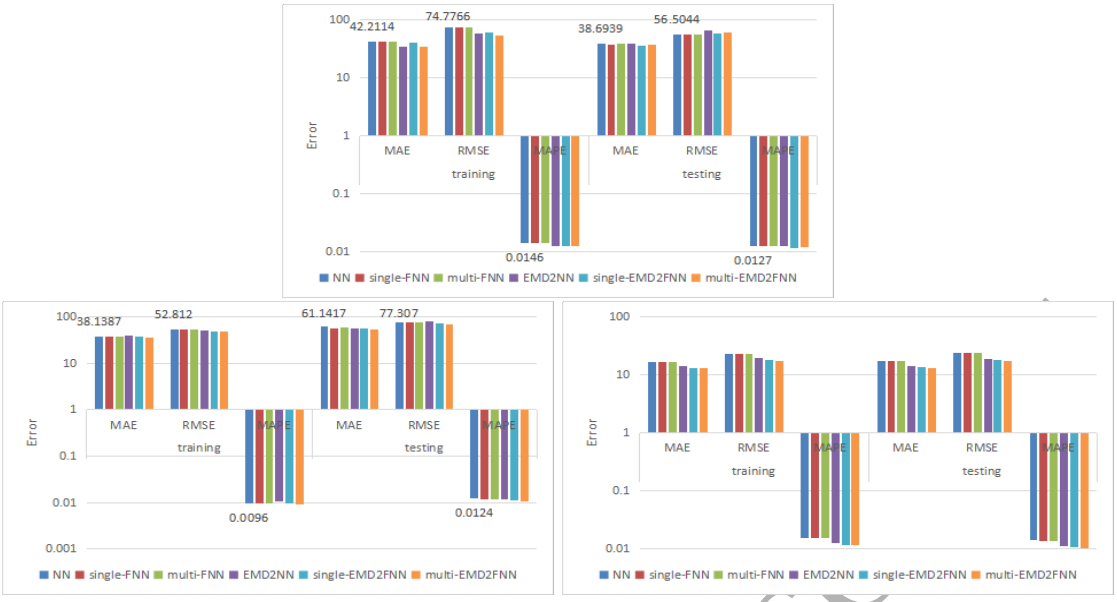

Figure 14: The performance of the forecasting models on three datasets. Top: SSEC; Bottom left: NASDAQ; Bottom right: S&P 500.

Finally, the EMD2FNN models are compared against the WDBP technique that is proposed in (Wang et al., 2011). The data from (Wang et al., 2011) is also the closing prices collected on the Shanghai Stock Exchange. The total number of values is from 204 trading months, from January 1993 to December 2009. Table 10 lists the results obtained from the WDBP (Wang et al., 2011) and the EMD2FNN models. From the table, the MAPE index is reduced from 19.48% to 6.47% and 5.68% corresponding to single-EMD2FNN and multi-EMD2FNN respectively. This shows that the proposed technique has achieved a significant improvement in predicting accuracy, and also proves that the multi-EMD2FNN is superior to the single-EMD2FNN.

Table 10: The performances of multi-EMD2FNN and WDBP on SSEC.

Models	**training data**			**testing data**		
	MAE	**RMSE**	**MAPE**	**MAE**	**RMSE**	**MAPE**
WDBP	–	–	–	675.9543	847.5841	0.1948
single-EMD2FNN	83.7839	105.7786	0.0597	**186.4379**	**235.7480**	**0.0647**
multi-EMD2FNN	99.4046	136.8320	0.0673	**164.1967**	**215.8639**	**0.0568**

4.7 the Discussion on Profitability

Since having a high prediction accuracy does not necessarily imply that the strategy makes profit, we compute the profitability with a simple long-short trading strategy to prove the reliability of the EMD2FNN models in trading, where we follow the philosophy of (Zhou et al., 2011) in viewing strategies as nonlinear transforms of the input time series that help revealing the properties of the price generating process.

For each model, the so-called long-short strategy we use consists of buying (respectively selling) if the price is predicted to increase (respectively decrease) from today to the next day. More specifically, one buys one unit of the index (position +1) if the price is predicted to increase. This strategy is kept until the first prediction for a price decrease is obtained, at which time one sells the index (position −1), i.e., one sells the existing position and further borrows one additional unit of the index and sells it. This short position −1 is reversed to position +1 at the next prediction for an upward index direction, by buying two units of the index, where one unit is used to cancel the short position and the other one is to be the net long position. And so on.

We compare the performance of this simple long-short strategy for both single-EMD2FNN and multi-EMD2FNN prediction models to (i) the buy-and-hold strategy, i.e., enter the market with buying one unit of index and holding the long position until the end; and (ii) a look-ahead 'ex post trading strategy', which would be realized if one had perfect foresight of the sign of the price change from today to tomorrow. Although, the 'ex post trading

strategy' can never be achieved in practice, and is expected to give extraordinary large returns, it provides an interesting upper bound of the performance.

Table 11 reports the performance metrics defined in Table 3 of the simple long-short strategy using our proposed prediction models, together with those of the buy-and-hold (denoted as **Benchmark**) and of the look-ahead 'ex post trading strategy'. As expected, the **Ex post** strategy would achieve a whooping 285% of average annual return (AAR) and zero drawdown since perfect foresight is assumed. In contrast, the **Benchmark** strategy delivers -1% of AAR with maximum drawdowns 19%. Remarkably, the single-EMD2FNN and multi-EMD2FNN models respectively have the AAR close to 13% and 25% with small maximum drawdowns in the range of 5%. Their Sharpe ratios are also impressive. From it, we can also see that the multi-EMD2FNN performs much better than the single-EMD2FNN.

Table 11: The performance of the simple long-short strategy without transaction cost using EMD2FNN-based prediction models, together with those of the buy-and-hold (denoted as **Benchmark**) and of the look-ahead 'ex post trading strategy' (denoted as **Ex post**) for S&P500.

Transaction cost	**Measure**	**Ex post**	**Benchmark**	**single-EMD2FNN**	**multi-EMD2FNN**
No	AAR	2.85	-0.01	0.13	**0.25**
	MD	0.00	0.19	0.05	**0.04**
	SR	10.05	0.07	1.72	**2.60**
	AAR/MD	$+\infty$	-0.05	2.60	**6.25**
	Trade Counts	59	1	17	22

It is often the case that very good results are reported in the absence of transaction cost (as well as when neglecting slippage and other real market frictions), while it wipes out all profits. In such a case, the prediction ability of a model would be a statistical reality but would be economically irrelevant. The market would still provide apparent arbitrage opportunities, which would be however in-exploitable due to the market frictions.

Transaction cost mainly consists of stamp duty, commission and transfer fees. In addition, an order to buy at a given price may not be implemented at the desired price due to lack of counter-parties (lack of liquidity and fast moving prices), leading to adverse slippage. In order to simplify the process of estimating the impact of transaction cost, we use a rather conservative value of 0.3% for each trade, to include all possible cost and frictions. This means that a given transaction consisting of buying an index and then selling it in the future will be exposed to a total of 0.6% of transaction cost.

Table 12 shows that, as expected, transaction cost reduces dramatically the AAR and other performance indicators. However, the multi-EMD2FNN still gives impressive results with AAR 14% and maximum drawdown 4%. Meanwhile, the single-EMD2FNN have the metrics with AAR 11% and maximum drawdown 17%.

Table 12: The performance of the simple long-short strategy with transaction cost using EMD2FNN-based prediction models, together with those of the buy-and-hold (denoted as **Benchmark**) and of the look-ahead 'ex post trading strategy' (denoted as **Ex post**) for S&P500.

Transaction cost	**Measure**	**Ex post**	**Benchmark**	**single-EMD2FNN**	**multi-EMD2FNN**
Yes	AAR	1.66	-0.02	0.11	**0.14**
	MD	0.01	0.19	0.17	**0.05**
	SR	7.17	0.04	0.58	**1.30**
	AAR/MD	166.0	-0.11	0.65	**2.80**
	Trade Counts	59	1	16	23

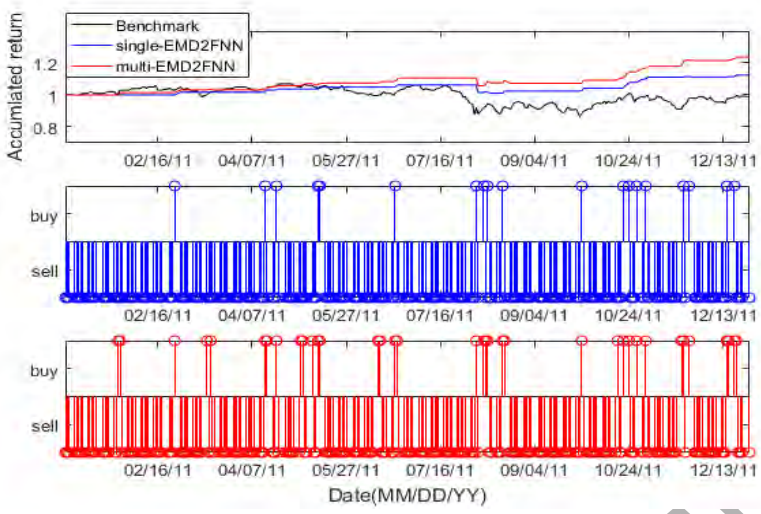

Figure 15: Top: Time evolution of the performance of the strategy in the test (out-of-sample) period based on the predictions of the EMD2FNN-based models on S&P 500 index without taking transaction cost into account. Middle: buy and sell signals of the single-EMD2FNN-based trading strategy. Bottom: buy and sell signals of the multi-EMD2FNN-based trading strategy.

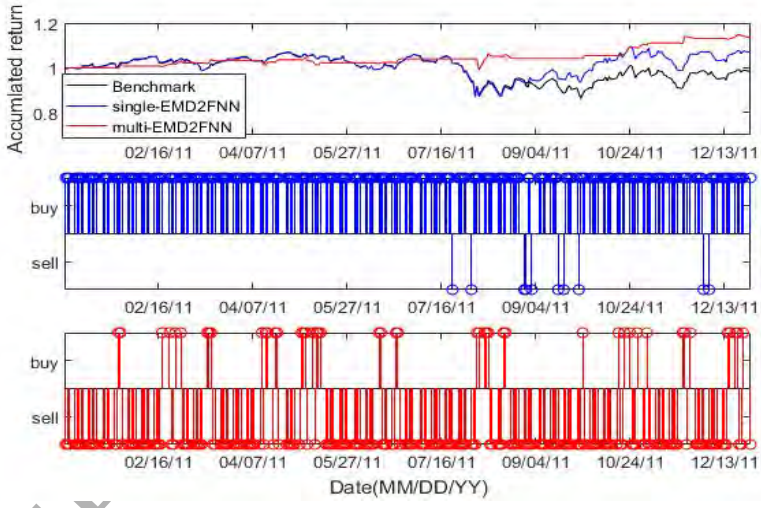

Figure 16: Top: Time evolution of the performance of the strategy in the test (out-of-sample) period based on the predictions of the EMD2FNN-based models on S&P 500 index with taking transaction cost into account. Middle: buy and sell signals of the single-EMD2FNN-based trading strategy. Bottom: buy and sell signals of the multi-EMD2FNN-based trading strategy.

Figures 16 and 15 show the time evolution of the performance of the strategy in the test (out-of-sample) period based on the predictions of the EMD2FNN models for two cases: taking or not taking the transaction cost into consideration for the S&P 500, in comparison with the buy-and-hold strategy (that reproduces the AAR of the indices). The top panels of these three figures present the normalized cumulative compounded returns of the buy-and-hold (black curve), single-EMD2FNN-based strategy (blue curve) and multi-EMD2FNN-based strategy (red curve) respectively. The initial value of return is set as 1, and thus the initial volume equals to 1 divided by the price at that base day. In the following days, we update the volume according to the return divided by the close price one day before. In a sense, we compound the returns. The other two panels of each of these three figures show the corresponding buy or sell signals of the single-EMD2FNN-based trading strategy (middle panel), and multi-EMD2FNN-based trading strategy (bottom panel).

5. Concluding Remarks

This paper introduces a novel method for predicting the stock prices by integrating EMD, FM and the neural network together. EMD is used to decompose the original nonlinear and non-stationary time series into IMFs that can be considered quasi-stationary. The produced IMFs are used as inputs to the neural network, which incorporates the FM strategy to exploit the nonlinear interactions between features extracted at different time scales. The resulting EMD2FNN can handle not only the nonlinearity but also the influence among time scales, which are two aspects that the existing methods face challenges. The numerical experiments demonstrated that the integrated methods have significant advantages in improving the prediction accuracies, measured by MAE, RMSE and MAPE. Furthermore, the profitability is computed with a simple long-short trading strategy to examine the trading performance of the proposed models by taking transaction cost into account or not. The performances, measured by AAR, MD, SR and AAR/MD, are also found economically significant.

This study is another example that demonstrates the power of combining EMD with neural network to achieve outstanding performance in stock price predictions. In fact, we believe that EMD can be used in conjunction with other statistic or artificial intelligent strategies to form general methods for predicting, especially for the time series that is nonlinear and non-stationary in nature. Various combinations in algorithm design can be investigated in the future to tackle problems emerging from different applications.

References

References

Armano G., Marchesi M. & Murru A. (2005). A hybrid genetic-neural architecture for stock indexes forecasting. *Information Sciences*, *170*, 3–33.

Arjo D. (2009). Statistical models: Theory and practice. *Biometrics*, *48*, 315–315.

Amodei D. , Anubhai R., Battenberg E., Case C., Casper J. , Catanzaro B., Chen J., Chrzanowski M., Coates A. & Diamos G. (2015). Deep speech 2: End-to-end speech recognition in English and Mandarin. *Computer Science*.

Bi N., Sun Q., Huang D., Yang Z. & Huang J. (2007). Robust image watermarking based on multiband wavelets and empirical mode decomposition. *IEEE Transactions on Image Processing A Publication of the IEEE Signal Processing Society*, *16*, 1956.

Bahdanau D., Cho K. & Bengio Y. (2014). Neural machine translation by jointly learning to align and translate. *Computer Science*.

Bayer I., He X., Kanagal B. & Rendle S. (2017). A generic coordinate descent framework for learning from implicit feedback. In *International Conference on World Wide Web*, 1341–1350.

Chen A. S., Leung M. T. & Daouk H. (2003). Application of neural networks to an emerging financial market: Forecasting and trading the taiwan stock index. *Computers and Operations Research*, *30*, 901–923.

Chen Q., Huang N., Riemenschneider S. & Xu Y. (2006). A b-spline approach for empirical mode decompositions. *Advances in Computational Mathematics*, *24*, 171–195.

Chang P. C., Wang D. & Zhou C. (2012). A novel model by evolving partially connected neural network for stock price trend forecasting. *Expert Systems With Applications*, *39*, 611–620.

Chen C. F., Lai M. C. & Yeh C. C. (2012). Forecasting tourism demand based on empirical mode decomposition and neural network. *Knowledge-Based Systems*, *26*, 281–287.

Clevert D. A., Unterthiner T. & Hochreiter S. (2015). Fast and accurate deep network learning by exponential linear units (ELUs). *Computer Science*.

Chen T., He X. & Kan M. Y. (2016). Context-aware image tweet modelling and recommendation. In *ACM on Multimedia Conference*, 1018–1027.

Delechelle E., Lemoine J. & Niang O. (2005). Empirical mode decomposition: an analytical approach for sifting process. *IEEE Signal Processing Letters*, *12*, 764–767.

Diop E. H. S., Alexandre R. & Boudraa A. O. (2010). Analysis of intrinsic mode functions: A PDE approach. *IEEE Signal Processing Letters*, *17*, 398–401.

Daubechies I., Lu J. & Wu H. T. (2011). Synchrosqueezed wavelet transforms: An empirical mode decomposition-like tool. *Applied and Computational Harmonic Analysis*, *30*, 243–261.

Ding Y. & Selesnick I. W. (2013). Sparse frequency analysis with sparse-derivative instantaneous amplitude and phase functions. *Computer Science*.

Enke D. & Thawornwong S. (2005). The use of data mining and neural networks for forecasting stock market returns. *Expert Systems with Applications*, *29*, 927–940.

Franses P. H. & Ghijsels H. (1999). Additive outliers, GARCH and forecasting volatility. *International Journal of Forecasting*, *15*, 1–9.

Flandrin P., Rilling G. & Goncalves P. (2004). Empirical mode decomposition as a filter bank. *IEEE Signal Processing Letters*, *11*, 112–114.

Han J. & Moraga C. (1995). The influence of the sigmoid function parameters on the speed of backpropagation learning. In *International Workshop on Artificial Neural Networks: From Natural To Artificial Neural Computation*, 195–201.

Huang N. E., Shen Z., Long S. R., Wu M. C., Shih H. H., Zheng Q., Yen N. C., Chi C. T. & Liu H. H. (1998a). The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series analysis. *Proceedings of the Royal Society A: Mathematical Physical and Engineering Sciences*, *454*, 903–995.

Huang W., Shen Z., Huang N. E. & Yuan C. F. (1998b). Engineering analysis of biological variables: An example of blood pressure over 1 day. *Proceedings of the National Academy of Sciences of the United States of America*, *95*, 4816.

Huang N. E., Shen Z. & Long S. R. (1999). A new view of nonlinear water waves: the Hilbert spectrum. *Annual Review of Fluid Mechanics*, *31*, 417–457.

Hansen J. V. & Nelson R. D. (2002). Data mining of time series using stacked generalizers. *Neurocomputing*, *43*, 173–184.

Hong H., Wang X. & Tao Z. (2009). Local integral mean-based sifting for empirical mode decomposition. *IEEE Signal Processing Letters*, *16*, 841–844.

Hou T. Y., & Shi Z. (2011). Adaptive data analysis via sparse time-frequency representation. *Advances in Adaptive Data Analysis*, *03*, 1–28.

Hinton G., Deng L., Yu D., Dahl G. E., Mohamed A. R., Jaitly N., Senior A., Vanhoucke V., Nguyen P. & Sainath T. N. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, *29*, 82–97.

Hu X., Peng S. & Hwang W. L. (2013). Multicomponent am-fm signal separation and demodulation with null space pursuit. *Signal Image and Video Processing*, *7*, 1093–1102.

He K., Zhang X., Ren S. & Sun J. (2015a). Deep residual learning for image recognition. *arXiv.org*, 770–778.

He K., Zhang X., Ren S. & Sun J. (2015b). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, 1026–1034.

He X. & Chua T. (2017). Neural factorization machines for sparse predictive analytics. *arXiv.org*, 355–364.

He X., Zhang H., Kan M. Y. & Chua T. S. (2017). Fast matrix factorization for online recommendation with implicit feedback. *International Acm Sigir Conference on Research and Development in Information Retrieval ACM*, 2016:549-558.

Jaber A. M., Ismail M. T. & Altaher A. M. (2014). Application of empirical mode decomposition with local linear quantile regression in financial time series forecasting. *The scientific world journal*, *2014*, 708-918.

Juan Y., Zhuang Y., Chin W. S. & Lin C. J. (2016). Field-aware factorization machines for ctr prediction. In *ACM Conference on Recommender Systems*, 43–50.

Kim K. J. & Han I. (2000). Genetic algorithms approach to feature discretization in artificial neural networks for the prediction of stock price index. *Expert Systems with Applications*, *19*, 125–132.

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

Koren Y. (2008). Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 426–434.

Krizhevsky A., Sutskever I. & Hinton G. E. (2012). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, *60*, 2012.

Lin L., Wang Y. & Zhou H. (2009). Iterative filtering as an alternative algorithm for empirical mode decomposition. *Advances in Adaptive Data Analysis*, *1*, 543–560.

Lu C. J. (2010). Integrating independent component analysis-based denoising scheme with neural network for stock price prediction. *Expert Systems with Applications*, *37*, 7056–7064.

Liu H., Chen C., Tian H. Q. & Li Y. F. (2012). A hybrid model for wind speed prediction using empirical mode decomposition and artificial neural networks. *Renewable Energy*, *48*, 545–556.

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

Luong M. T., Pham H. & Manning C. D. (2015). Effective approaches to attention-based neural machine translation. *Computer Science*.

Liu S., Xu L. & Li D. (2016). Multi-scale prediction of water temperature using empirical mode decomposition with back-propagation neural networks. *Computers and Electrical Engineering*, *49*, 1–8.

Makridakis S. (1993). Accuracy measures: theoretical and practical concerns. *International Journal of Forecasting*, *9*, 527–529.

Nunes J. C., Guyot S. & Deléchelle E. (2005). Texture analysis based on local analysis of the bidimensional empirical mode decomposition. *Machine Vision and Applications*, *16*, 177–188.

Oh K. J. & Kim K. J. (2002). Analyzing stock market tick data using piecewise nonlinear model. *Expert Systems with Applications*, *22*, 249–255.

Omidi A., Nourani E. & Jalili M. (2011). Forecasting stock prices using financial data mining and neural network. In *International Conference on Computer Research and Development*, 242–246.

Oberlin T., Meignen S. & Perrier V. (2012). An alternative formulation for the empirical mode decomposition. *IEEE Transactions on Signal Processing*, *60*, 2236–2246.

Oentaryo R. J., Lim E. P., Low J. W., Lo D. & Finegold M. (2014). Predicting response in mobile advertising with hierarchical importance-aware factorization machine. In *ACM International Conference on Web Search and Data Mining*, 123–132.

Peng S., & Hwang W. L. (2008). Adaptive signal decomposition based on local narrow band signals. *IEEE Transactions on Signal Processing*, *56*, 2669–2676.

Peng S. & Hwang W. L. (2010). Null space pursuit: An operator-based approach to adaptive signal separation. *IEEE Transactions on Signal Processing*, *58*, 2475–2483.

Pustelnik N., Borgnat P. & Flandrin P. (2010). A multicomponent proximal algorithm for empirical mode decomposition. In *Signal Processing Conference* (pp. 1880–1884).

Pustelnik N., Borgnat P., & Flandrin P. (2014). Empirical mode decomposition revisited by multicomponent non-smooth convex optimization. *Signal Processing*, *102*, 313–331.

Patel J., Shah S., Thakkar P. & Kotecha K. (2015). Predicting stock market index using fusion of machine learning techniques. *Expert Systems with Applications*, *42*, 2162–2172.

Qian X. Y. & Gao S. (2017). Financial series prediction: Comparison between precision of time series models and machine learning methods. *arXiv.org*.

Rendle S. (2010). Factorization machines. In *ICDM 2010, the IEEE International Conference on Data Mining, Sydney, Australia, 14-17 December*, 995–1000.

Rather A. M., Agarwal A. & Sastry V. N. (2015). Recurrent neural network and a hybrid model for prediction of stock returns. *Expert Systems with Applications*, *42*, 3234–3241.

Sarantis N. (2001). Nonlinearities, cyclical behavior and predictability in stock markets: international evidence. *International Journal of Forecasting*, *17*, 459–482.

Shen L., & Han T. L. (2004). Applying rough sets to market timing decisions. *Decision Support Systems*, *37*, 583–597.

Smith J. S. (2005). The local mean decomposition and its application to eeg perception data. *Journal of the Royal Society Interface*, *2*, 443.

Sekine Y. (2007). A new formulation for empirical mode decomposition based on constrained optimization. *IEEE Signal Processing Letters*, *14*, 932–935.

Szegedy C., Liu W., Jia Y., Sermanet P., Reed S., Anguelov D., Erhan D., Vanhoucke V. & Rabinovich A. (2015). Going deeper with convolutions. In *Computer Vision and Pattern Recognition*, 1–9.

Ture M. & Kurt I. (2006). Comparison of four different time series methods to forecast hepatitis a virus infection. *Expert Systems with Applications*, *31*, 41–46.

Vellido A., Lisboa P. J. G. & Meehan K. (1999). Segmentation of the on-line shopping market using neural networks. *Expert Systems with Applications*, *17*, 303–314.

Wang Y. F. (2002). Predicting stock price using fuzzy grey prediction system. *Expert Systems with Applications*, *22*, 33–38.

Wang Y. F. (2003). Mining stock price using fuzzy rough set system. *Expert Systems with Applications*, *24*, 13–23.

Wang J. Z., Wang J. J., Zhang Z. G. & Guo S. P. (2011). Forecasting stock indices with back propagation neural network. *Expert Systems with Applications*, *38*, 14346–14355.

Wu H. T. (2013). Instantaneous frequency and wave shape functions (i). *Applied & Computational Harmonic Analysis*, *35*, 181–199.

Yang Z., Huang D. & Yang L. (2004). A Novel Pitch Period Detection Algorithm Based on Hilbert-Huang Transform. *Chinese Conference on Advances in Biometric Person Authenticationg*, *1*, 138–146.

Yang Z., Qi D. & Yang L. (2005b). Signal period analysis based on Hilbert-Huang transform and its application to texture analysis. In *International Conference on Image and Graphics*, 430–433.

Yang Z., Yang L. & Qi D. (2006a). *Detection of Spindles in Sleep EEGs Using a Novel Algorithm Based on the Hilbert-Huang Transform*. Birkhäuser Basel.

Yang Z., Yang L., Qi D. & Suen C. Y. (2006b). An EMD-based recognition method for Chinese fonts and styles. *Pattern Recognition Letters*, *27*, 1692–1701.

Yang L., Yang Z., Zhou F. & Yang L. (2014). A novel envelope model based on convex constrained optimization. *Digital Signal Processing*, *29*, 586-593.

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

Zhang X., Lai K. K. & Wang S. Y. (2008). A new approach for crude oil price analysis based on empirical mode decomposition. *Energy Economics*, *30*, 905–918.

Zhou, W.-X., Mu, G.H., Chen, W. & Sornette, D. (2011). Investment strategies used as Spectroscopy of Financial Markets Reveal New Stylized Facts. *PLoS ONE*, *6*, *9*, e24391.

Zhou F., Yang L., Zhou H. & Yang L. (2016). Optimal averages for nonlinear signal decompositions-another alternative for empirical mode decomposition. *Signal Processing*, *121*, 17–29.