Neurocomputing 156 (2015) 68–78

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

# Forecasting stock market indexes using principle component analysis and stochastic time effective neural networks

Jie Wang*, Jun Wang*

Institute of Financial Mathematics and Financial Engineering, School of Science, Beijing Jiaotong University, Beijing 100044, PR China

ARTICLE INFO

*Article history:*
Received 25 July 2014
Received in revised form
18 October 2014
Accepted 30 December 2014
Communicated by P. Zhang
Available online 9 January 2015

*Keywords:*
Forecast
Stochastic time effective neural network
Principal component analysis
Financial time series model

ABSTRACT

Financial market dynamics forecasting has long been a focus of economic research. A stochastic time effective function neural network (STNN) with principal component analysis (PCA) developed for financial time series prediction is presented in the present work. In the training modeling, we first use the approach of PCA to extract the principal components from the input data, then integrate the STNN model to perform the financial price series prediction. By taking the proposed model compared with the traditional backpropagation neural network (BPNN), PCA-BPNN and STNN, the empirical analysis shows that the forecasting results of the proposed neural network display a better performance in financial time series forecasting. Further, the empirical research is performed in testing the predictive effects of SSE, HS300, S&P500 and DJIA in the established model, and the corresponding statistical comparisons of the above market indices are also exhibited.

## 1. Introduction

Financial price time series prediction has recently garnered significant interest among investors and professional analysts. Artificial neural network (ANN) is one of the technologies that have made great progress in the study of stock market dynamics. Neural networks can provide models for a large class of natural and artificial phenomena that are difficult to handle using classical parametric techniques [1–5]. Usually stock prices can be seen as a random time sequence with noise, and a number of analysis methods have utilized artificial neural networks to predict stock price trends [6–11]. Artificial neural networks have good self-learning ability, a strong anti-jamming capability, and have been widely used in the financial fields such as stock prices, profits, exchange rate and risk analysis and prediction [12–14].

To improve predicting precision, various network architectures and learning algorithms have been developed in the literature [15–19]. The backpropagation neural network (BPNN) is a neural network training algorithm for financial forecasting, which has powerful problem-solving ability. Multilayer perceptron (MLP) is one of the most prevalent neural networks, which has the capability of complex mapping between inputs and outputs that makes it possible to approximate nonlinear function [19,20]. In the present work, we apply MLP with backpropagation algorithm and stochastic time strength function to develop a stock price volatility forecasting model. In the real financial markets, the investing environments as well as the fluctuation behaviors of the markets are not invariant. If all the data are used to train the network equivalently, the expressing ability of network is likely to be curtailed for policymakers concerning about the current regular system. For example, see the Chinese stock markets in 2007, the data in the training set should be time-variant, reflecting the different behavior patterns of the markets at different times. If only the recent data are selected, a lot of useful information (which the early data hold) will be lost. In this research, a stochastic time effective neural network (STNN) and the corresponding learning algorithm are presented. For this improved network model, each of historical data is given a weight depending on the time at which it occurs. The degree of impact of historical data on the market is expressed by a stochastic process [17–19], where a drift function and the Brownian motion are introduced in the time strength function in order to make the model have the effect of random movement while maintaining the original trend.

This paper presents an improved method which integrates the principal component analysis (PCA) into a stochastic time strength neural network (STNN) for forecasting financial time series, called PCA-STNN model. The approach of PCA-STNN is to extract the principal components (PCs) from the input data according to the PCA method, and use PCs as the input of STNN model which can eliminate redundancies of original information and remove the correlation between the inputs [21–28]. In order to display that the PCA-STNN model outperforms the PCA-BPNN model, the BPNN model and the STNN model in forecasting the fluctuations of stock markets, we compare the forecasting performance of the above four forecasting models by selecting the data of the global stock indices,

* Corresponding author.
*E-mail address:* wangjiebjtu@yeah.net (J. Wang).

http://dx.doi.org/10.1016/j.neucom.2014.12.084
0925-2312/© 2015 Elsevier B.V. All rights reserved.

including Shanghai Stock Exchange (SSE) Composite Index, Hong Kong Hang Seng 300 Index (HS300), Dow Jones Industrial Average Index (DJIA), and Standard & Poor's 500 Index (S&P 500).

## 2. Methodology

### 2.1. Stochastic time effective neural network (STNN)

Artificial neural network has been extensively used as a method to forecast financial market behaviors. The backpropagation algorithm has emerged as one of the most widely used learning procedures for multilayer networks [27–29]. In Fig. 1, a three-layer multi-input BPNN model is exhibited, the corresponding structure is $m \times n \times 1$, where $m$ is the number of inputs, $n$ is the number of neurons in the hidden layer and one output unit. Let $x_{it}$ $(i=1,2,\dots,m)$ denote the set of input vector of neurons at time $t$, and $y_{t+1}$ denote the output of the network at time $t+1$. Between the inputs and the output, there is a layer of processing units called hidden units. Let $z_{jt}$ $(j=1,2,\dots,n)$ denote the output of hidden layer neurons at time $t$, $w_{ij}$ is the weight that connects the node $i$ in the input layer neurons to the node $j$ in the hidden layer, $v_j$ is the weight that connects the node $j$ in the hidden layer neurons to the node in the output layer. Hidden layer stage is as follows: The input of all neurons in the hidden layer is given by

$$\mathrm{net}_{jt} = \sum_{i=1}^{n} w_{ij} x_{it} - \theta_j, \quad i=1,2,\dots,n. \tag{1}$$

The output of hidden neuron is given by

$$z_{jt} = f_H(\mathrm{net}_{jt}) = f_H\left(\sum_{i=1}^{n} w_{ij} x_{it} - \theta_j\right), \quad i=1,2,\dots,n \tag{2}$$

where $\theta_j$ is the threshold of neuron in hidden layer. The sigmoid function in hidden layer is selected as the activation function: $f_H(x) = 1/(1+e^{-x})$. The output of the hidden layer is given as follows:

$$y_{t+1} = f_T\left(\sum_{j=1}^{m} v_j z_{jt} - \theta_T\right) \tag{3}$$

where $\theta_j$ is the threshold of neuron in output layer and $f_T(x)$ is an identity map as the activation function.

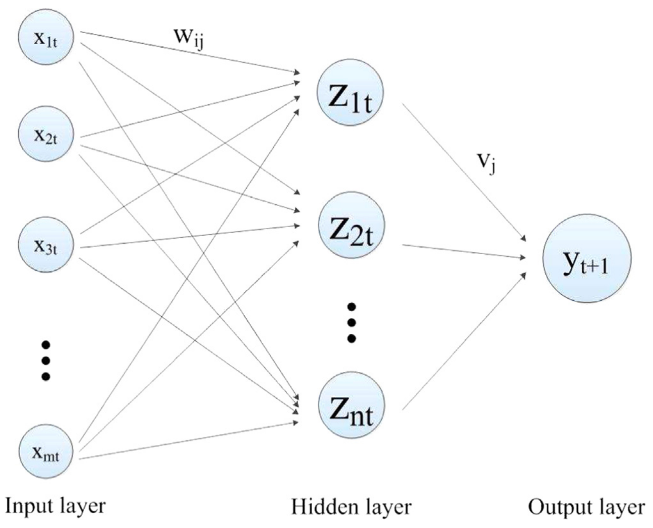

**Fig. 1.** Three-layer neural network with one-output.

### 2.2. Predicting algorithm with a stochastic time effective function

The backpropagation algorithm is a supervised learning algorithm which minimizes the global error $E$ by using the gradient descent method. For the STNN model, we assume that the error of the output is given by $\varepsilon_{t_n} = d_{t_n} - y_{t_n}$ and the error of the sample $n$ is defined as

$$E(t_n) = \tfrac{1}{2}\varphi(t_n)(d_{t_n} - y_{t_n})^2 \tag{4}$$

where $t_n$ is the time of the sample $n$ $(n=1,\dots,N)$, $d_{t_n}$ is the actual value, $y_{t_n}$ is the output at time $t_n$, and $\varphi(t_n)$ is the stochastic time effective function which endows each historical data with a weight depending on the time at which it occurs. We define $\varphi(t_n)$ as follows:

$$\varphi_{t_n} = \frac{1}{\beta}\exp\left\{\int_{t_0}^{t_n} \mu(t)\,dt + \int_{t_0}^{t_n} \sigma(t)\,dB(t)\right\} \tag{5}$$

where $\beta(>0)$ is the time strength coefficient, $t_0$ is the time of the newest data in the data training set, and $t_n$ is an arbitrary time point in the data training set. $\mu(t)$ is the drift function, $\sigma(t)$ is the volatility function, and $B(t)$ is the standard Brownian motion.

Intuitively, the drift function is used to model deterministic trends, the volatility function is often used to model a set of unpredictable events occurring during this motion, and Brownian motion is usually thought as random motion of a particle in liquid (where the future motion of the particle at any given time is not dependent on the past). Brownian motion is a continuous-time stochastic process, and it is the limit of or continuous version of random walks. Since Brownian motion's time derivative is everywhere infinite, it is an idealized approximation to actual random physical processes, which always have a finite time scale. We begin with an explicit definition. A Brownian motion is a real-valued, continuous stochastic process $\{Y(t), t \ge 0\}$ on a probability space $(\Omega, \mathcal{F}, \mathbb{P})$, with independent and stationary increments. In detail, (a) continuity: the map $s \mapsto Y(s)$ is continuous $\mathbb{P}$ a.s.; (b) independent increments: if $s \le t$, $Y_t - Y_s$ is independent of $\mathcal{F} = (Y_u, u \le s)$; (c) stationary increments: if $s \le t$, $Y_t - Y_s$ and $Y_{t-s} - Y_0$ have the same probability law. From this definition, if $\{Y(t), t \ge 0\}$ is a Brownian motion, then $Y_t - Y_0$ is a normal random variable with mean $rt$ and variance $\sigma^2 t$, where $r$ and $\sigma$ are constant real numbers. A Brownian motion is standard (we denote it by $B(t)$) if $B(0)=0$ $\mathbb{P}$ a.s., $\mathbb{E}[B(t)]=0$ and $\mathbb{E}[B(t)]^2 = t$. In the above random data-time effective function, the impact of the historical data on the stock market is regarded as a time variable function, the efficiency of the historical data depends on its time. Then the corresponding global error of all the data at each network repeated training set in the output layer is defined as

$$E = \frac{1}{N}\sum_{n=1}^{N} E(t_n) = \frac{1}{2N}\sum_{n=1}^{N} \frac{1}{\beta}\exp\left\{\int_{t_0}^{t_n} \mu(t)\,dt + \int_{t_0}^{t_n} \sigma(t)\,dB(t)\right\}(d_{t_n} - y_{t_n})^2. \tag{6}$$

The main objective of learning algorithm is to minimize the value of cost function $E$ until it reaches the pre-set minimum value $\xi$ by repeated learning. On each repetition, the output is calculated and the global error $E$ is obtained. The gradient of the cost function is given by $\Delta E = \partial E / \partial W$. For the weight nodes in the input layer, the gradient of the connective weight $w_{ij}$ is given by

$$\Delta w_{ij} = -\eta \frac{\partial E(t_n)}{\partial w_{ij}} = \eta \varepsilon_{t_n} v_j \varphi(t_n) f'_H(\mathrm{net}_{jt_n}) x_{it_n} \tag{7}$$

and for the weight nodes in the hidden layer, the gradient of the connective weight $v_j$ is given by

$$\Delta v_j = -\eta \frac{\partial E(t_n)}{\partial v_j} = \eta \varepsilon_{t_n} \varphi(t_n) f_H(\mathrm{net}_{jt_n}) \tag{8}$$