Hindawi Publishing Corporation
Advances in Artificial Neural Systems
Volume 2011, Article ID 814769, 11 pages
doi:10.1155/2011/814769

Research Article

Stock Price Prediction Based on Procedural Neural Networks

Jiuzhen Liang, Wei Song, and Mei Wang

Department of Electrical Engineering, Jiangnan University, Wuxi 214122, China

Correspondence should be addressed to Jiuzhen Liang, jz.liang@yahoo.com.cn

Received 11 January 2011; Revised 28 March 2011; Accepted 6 April 2011

Academic Editor: Songcan Chen

We present a spatiotemporal model, namely, procedural neural networks for stock price prediction. Compared with some successful traditional models on simulating stock market, such as BNN (backpropagation neural networks, HMM (hidden Markov model) and SVM (support vector machine)), the procedural neural network model processes both spacial and temporal information synchronously without slide time window, which is typically used in the well-known recurrent neural networks. Two different structures of procedural neural networks are constructed for modeling multidimensional time series problems. Learning algorithms for training the models and sustained improvement of learning are presented and discussed. Experiments on Yahoo stock market of the past decade years are implemented, and simulation results are compared by PNN, BNN, HMM, and SVM.

1. Introduction

From the beginning of time it has been human's common goal to make life easier and richer. The prevailing notion in society is that wealth brings comfort and luxury, so it is not surprising that there has been so much work done on ways to predict the markets. From the day stock was born, the movement of prediction has been the focus of interset for years since it can yield significant profits. There are several motivations for trying to predict stock market prices. The most basic of these is financial gain. Any system that can consistently pick winners and losers in the dynamic market place would make the owner of the system very wealthy. Thus, many individuals including researchers, investment professionals, and average investors are continually looking for this superior system which will yield them high returns. There is a second motivation in the research and financial communities. It has been proposed in the efficient market hypothesis (EMH) that markets are efficient in that opportunities for profit are discovered so quickly that they cease to be opportunities [1]. The EMH effectively states that no system can continually beat the market because if this system becomes public, everyone will use it, thus negating its potential gain.

It is a practically interesting and challenging topic to predict the trends of a stock price. Fundamental and technical analyses are the first two methods used to forecast stock prices. Various technical, fundamental, and statistical indicators have been proposed and used with varying results. However, no one technique or combination of techniques has been successful enough to consistently "beat the market". With the development of neural networks, researchers and investors are hoping that the market mysteries can be unraveled. Although it is not an easy job due to its nonlinearity and uncertainty, many trials using various methods have been proposed, for example, artificial neural networks [2], fuzzy logic [3], evolutionary algorithms [4], statistic learning [5], Bayesian belief networks [6], hidden Markov model [7], granular computing [8], fractal geometry [9], and wavelet analysis [10].

Recently, a novel model named procedural neural networks (PNNs) was proposed to deal with spatiotemporal data modeling problems, especially for time series with huge data of multidimension [11]. Different from the traditional multilayer backpropagation neural network (BNNs), the data in PNN are accumulated along the time axis before or after combining the contribution of the space components. While collecting these data, different components do not

have to be sampled simultaneously, but in the same intervals [12]. In this way, these time series problems subjected to synchronous sampling in all dimensions can be simulated by PNN. Moreover, the dimensional scale of input for PNN does not increase, while in the recurrent BNN a fix slide time window, which makes the dimensional scale large, is usually chosen to deal with time series data [2]. As a result, the complexity of PNNs is intuitively decreased both in the scale of model and in the time cost of training. Intrinsically, PNN differs from BNN in the way of mathematic mapping. BNN tries to map an n-dimensional point to another point in an m-dimensional space, while PNN tends to transfer an n-dimensional function to an m-dimensional point [13]. Varying from the previous work, this paper focuses on discussion of two kinds of quite different structures of PNNs and their application to prediction of stock market.

The rest of this paper is organized as follows. In Section 2, some general theories and analysis of stock markets are mentioned and some typical models are introduced for stock price prediction. In Section 3, the procedural neural network model is described in detail. In Section 4, the learning algorithm is constructed for training procedural neural networks, and the computational complexity of the algorithm is discussed. In Section 5, several experimental results are provided. Finally, Section 6 concludes this paper.

2. Typical Models for Stock Price Prediction

Stock markets are not perfect but pretty tough! Stock markets are filled with a certain energy and excitement. Such excitement and emotion is brought by the prospect of making a "buck," or by buying and selling in the hope of getting rich. Many people buy and sell shares in a rush to make huge fortunes (usually huge losses). Only through knowing future information about a particular market that nobody else knows can you hope to be able to make a definite profit [8]. Research and the idea of stock market efficiency have been extensively studied in the past 40 years. Many of the reported anomalies could be the result of mismeasurements and the failure to incorporate time-varying risks and returns as well as the cost of information [14]. In today's information age, there seems to be too much information out there and many opportunities to get overloaded or just plainly confused. The key is to decide when certain ideas are valid in which context and also to decide on what you believe. The same goes when it comes to investing on shares.

With the development of stock chart software, you may backtest stock trading strategies, create stock trading systems, view, buy, and sell signals on the charts, and do a lot more. Today's charting packages have up to hundreds of predefined indices for you to define your own in analyzing stock markets [1]. Given the number of possible choices, which indexes do you use? Your choice depends on what you are attempting to model. If you are after daily changes in the stock market, then use daily figures of the all-ordinaries index. If you are after long-term trends, then use long-term indexes like the ten-year bond yield. Even if you use an indicator that is a good representation of the total market, it is still no guarantee of producing a successful result. In summary, the success depends heavily on the tool, or the model that you use [8].

2.1. Statistic Methods: Hidden Markov Model and Bayes Networks. Markov models [15, 16] are widely used to model sequential processes and have achieved many practical successes in areas such as web log mining, computational biology, speech recognition, natural language processing, robotics, and fault diagnosis. The first-order Markov model contains a single variable, the state, and specifies the probability of each state and of transiting from one state to another. Hidden Markov models (HMMs) [17] contain two variables, that is, the (hidden) state and the observation. In addition to the transition probabilities, HMMs specify the probability of making each observation in each state. Because the number of parameters of a first-order Markov model is quadratic in the number of states (and higher for higher-order models), learning Markov models is feasible only in relatively small state spaces. Such requirement makes them unsuitable for many data mining applications, which are concerned with very large state spaces.

Dynamic Bayesian networks (DBNs) generalize Markov models by allowing states to have an internal structure [18]. In a DBN, a state is represented by a set of variables, which can depend on each other and on variables in previous states. If the dependency structure is sufficiently sparse, it is possible to successfully learn and reason in much larger state spaces than using Markov models. However, DBNs are still restricted by the assumption that all states are described by the same variables with the same dependencies. To many applications, states naturally fall into different classes and each class is described by a different set of variables.

Recently, there has been a considerable interest in the applications of regime switching models driven by a hidden Markov chain to various financial problems. For an overview of the hidden Markov chain and its financial applications, see the work of Elliott et al. in [19], of Elliott and Kopp [20], and of Aggoun and Elliott in [21]. Some works on the use of the hidden Markov chain in finance include Buffington and Elliott [22, 23] for pricing European and American options, of Ghezzi and Piccardi for stock valuation [24], and of Elliott et al. [25] for option valuation in an incomplete market. Most of the literature concerns the pricing of options under a continuous-time Markov-modulated process, while Hassan et al. [26] propose and implement a fusion model by combining the hidden Markov Model (HMM), artificial neural networks (ANNs) and Genetic Algorithms (GAs) to forecast financial market behavior.

2.2. Fractal and Dynamic Methods: Fractal Geometric Chaos. The chaos theory [27] assumes that the return dynamics are not normally distributed and more complex approaches have to be used to study these time series. In fact, the Fractal Market Hypothesis assumes that the return dynamics are not dependent of the investors' attitudes and represent the result of the interaction of traders who, frequently, adopt different investment styles. The studies proposed in literature to analyze and predict stock price dynamics

assume that, by looking at the past, one may collect useful information to understand the price formation mechanism. The initial approaches proposed in literature, the so-called technical analysis, assume that the price dynamics could be approximated with linear trends and could be analyzed using a standard mathematical or graphical approach [28]. The high number of factors that are likely to influence the stock market dynamics makes this assumption incorrect and calls for the definition of more complex approaches that may succeed in studying these multiple relationships [29].

The nonlinear models are a heterogeneous set of econometric approaches that allow higher predictability levels, but not all the approaches may be easily applied to real data [30]. Deterministic chaos represents the best trade-off to establish fixed rules in order to link future dynamics to past results of a time series without imposing excessively simple assumptions [31]. In essence, chaos is a nonlinear deterministic process that looks random [32] because it is the result of an irregular oscillatory process influenced by an initial condition and characterized by an irregular periodicity [33]. The chaos theory assumes that complex dynamics may be explained if they are considered as a combination of more simple trends that are easy to understand [34]: the higher the number of breakdowns, the higher the probability of identifying a few previously known basic profiles [35]. Chaotic trends may be studied considering some significant points that represent attractors or deflectors for the time series being analyzed and the periodicity that exists in the relevant data. To improve the prediction accuracy of complex multivariate chaotic time series, recently, a scheme has been proposed based on multivariate local polynomial fitting with the optimal kernel function [36], which combines the advantages of traditional local, weighted, multivariate prediction methods.

2.3. Soft Computing: Neural Networks, Fuzzy Logic, and Genetic Algorithms. Apparently, White (1988) was the first to use backpropagation neural networks (BNNs) for market forecasting [1]. He was curious about whether BNNs could be used to extract nonlinear regularities from economic time series and thereby decode previously undetected regularities in asset price movements, such as fluctuations of common stock prices. White found that his training results were overoptimistic, being the result of overfitting or of learning evanescent features. Since then, it has been well established that fusing the soft computing (SC) technologies, for example, BNN, fuzzy logic (FL), and genetic algorithms (GAs), may significantly improve the analysis (Jain and Martin 1999 [37], Abraham et al. 2001 [38]). There are two main reasons for this. First, these technologies are mostly complementary and synergistic. They are complementary which follows from the observations that BNNs used for learning and curve fitting, FL is used to deal with imprecision and uncertainty, and GAs [39] are used for search and optimization. Second, as Zadeh (1992) [40] pointed out, merging these technologies allows for the exploitation of a tolerance for imprecision, uncertainty, and partial truth to achieve tractability, robustness, and low solution cost. Market forecasting and trading rules have numerous facets with potential applications for hybrids of the SC technologies. Given this potential and the impetus on SC during the last decade, it is not surprising that a number of SC studies have focused on market forecasting and trading applications. As an example, Kuo et al. [3] use a genetic-algorithm-based fuzzy neural network to measure the qualitative effects on the stock price. A recent work introduced the generalized regression neural network (GRNN) which is used in various prediction and forecasting tasks [41]. Due to robustness and flexibility of modeling algorithms, neurocomputational models are expected to outperform traditional statistical techniques such as regression and ARIMA in forecasting stock exchange price movements.

2.4. Machine Learning. Applications of machine learning (ML) to stock market analysis include portfolio optimization, investment strategy determination, and market risk Analysis. Duerson et al. [42] focus on the problem of investment strategy determination through the use of reinforcement learning techniques. Four techniques, two based on recurrent reinforcement learning (RLL) and two based on Q-learning, were utilized. Q-learning produced results that consistently beat buy- and- hold strategies on several technology stocks, whereas the RRL methods were often inconsistent and required further investigation. The behavior of recurrent reinforcement learner needs further analysis. The technical justification seems to be most rigorous in the literature for this method it seems to deceive that simpler methods produced more consistent results. It has been observed that the performance of turning point indicator is generally better during short runs of trading which shows the validity of charting analysis techniques as used by professional stock traders.

Support vector machine (SVM) is a very specific learning algorithm characterized by the capacity control of the decision function, the use of the kernel functions, and the sparsity of the solution. Huang et al. [43] investigate the predictability of financial movement direction with SVM by forecasting the week movement direction of NIKKEI 225 index. SVM is a promising tool for financial forecasting. As demonstrated in their empirical analysis, SVM seems to be superior to other individual classification methods in forecasting weekly movement direction. This is a clear message for financial forecasters and traders, which can lead to a capital gain. However, it has been known that each method has its own strengths and weaknesses. The weakness of one method can be balanced by combining the strengths of another by achieving a systematic effect. The combining model performs best among all the forecasting methods.

For time series predictions, SVM is utilized as a regression function. But while preparing samples for SVM, all functions, which are dispersed in a certain interval of time, have to be transferred to spacial vectors. So it is essential that SVM still perform functions that map static vectors from one space to another. PNN combines the spatial and temporal information together; namely, neurons process information both from space and time simultaneously. In [44], the author proposed an extended model, named support function machine (SFM), in which each component of the vector is a time function and applied to predict stock price.

3. Procedural Neural Network Models

3.1. Procedural Neuron Model. The invention of procedural neuron provides an alternative modeling strategy to simulate time series problems which are related to some procedures [11]. This model also offers an approach to study dynamic characteristics in classification or regression problems with a great deal of spatiotemporal data. The procedural neuron differs from the traditional artificial neuron it combines the spacial and temporal information together. In this way, neurons are endowed with spacial and temporal characteristics simultaneously. The weights, which connect neurons, are usually variable, that is, functions of time. The neurons are expected to be timeaccumulating, which will not be inspired before a period of time long enough by input accumulation. Compared with traditional artificial neurons, the procedural neurons can simulate the biology neurons physiologically better. Moreover, many problems in real life can be reduced to a procedure, for example, agricultural planting, industrial producing, and chemical reacting. However, mostly it is impracticable to stimulate such procedures in the traditional ways by constructing some mathematical or physical equations.

A typical procedural neuron can accept a series of inputs with multiple-dimensions, and there is only one corresponding output. In a procedural neuron, the aggregation operation is involved with not only the assembly of multiinputs in space, but also the accumulation in time domain. So the procedural neuron is the extension of the time region from the traditional neuron. The traditional neuron can be regarded as a special case of the procedure neuron. In the structure of a procedural neuron, the continuity of time is shown in Figure 1, in which, $X(t) = [x_1(t), x_2(t), \ldots, x_n(t)]^T$ is the input function vector of the procedure neuron, $W(t) = [w_1(t), w_2(t), \ldots, w_n(t)]^T$ is the weight function (or weight function vector) in range $[t_1, t_2]$, and f is the activation function, such as linear function, and Sigmoid-Gaussian function. There are two forms of accumulation, time first (the left in Figure 1) and space first (the right in Figure 1).

Mathematically, the static output of the procedural neuron in Figure 1 can be written as follows:

$$y = f\left(\int_{t_1}^{t_2} W(t)^T X(t)\mathrm{d}t\right). \tag{1}$$

In detail, the corresponding component form of (1) is

$$y = f\left(\int_{t_1}^{t_2} \sum_{i=1}^{n} w_i(t)x_i(t)\mathrm{d}t\right) = f\left(\sum_{i=1}^{n} \int_{t_1}^{t_2} w_i(t)x_i(t)\mathrm{d}t\right). \tag{2}$$

The procedural neuron in Figure 1 is valuable in a sense only in theory, because most neural networks are constructed to solve discrete problems. In the case of discrete time, the procedural neuron takes the form as in Figure 2 or Figure 3, in which input data has been sampled along time axis and appears as a matrix $\{x_{ij}\}_{n\times T}$.

Similar to (2), the output of the discrete system in Figure 2 (time first) can be written as

$$y = \sum_{i=1}^{n} v_i f\left(\sum_{j=1}^{T} w_{ij}x_{ij} + w_{i0}\right) + v_0 \tag{3}$$

and for Figure 3 (space first), it turns out to be

$$y = \sum_{j=1}^{T} v_j f\left(\sum_{i=1}^{n} w_{ij}x_{ij} + w_{0j}\right) + v_0, \tag{4}$$

where w_{i0}, w_{0j}, v_0 are thresholds of the neurons, respectively.

3.2. Procedural Neural Network Models. Since the first model of procedural neural networks (PNNs) was proposed [45], several literatures have been concerned this topic, including topological structure constructing [11], computational ability estimating [46], learning algorithm [12], and time series application [47]. Recently, the author has proposed various structures of PNN (e.g., the functional PNN [13], the complex number PNN [48], the segment PNN [49], and the SFM [44]). Generally, all these models try to simulate spatiotemporal problems, especially for problems with large amount of data. To solve time series problems, constructing a suitable structure of PNN is the first step which is valuable in practice. We can design various structures of PNN based on the procedural neuron model as we just mentioned, for example, the procedural perception, the multilayer PNN, and the feedback PNN. This paper focuses on discussing two forms of PNN, which are composed of the time-first neuron and the space-first neuron. Equations (3) and (4) are two typical perceptions of PNN in which the input functions are multidimensional time series, while the output is a static scalar. It is not difficult to extend these models to the case of multiinput and multioutput PNN (called procedural perception) and the corresponding expressions are as follows:

$$y_k = \sum_{i=1}^{n} v_{ki} f\left(\sum_{j=1}^{T} w_{ij}x_{ij} - w_{i0}\right) - v_{k0}, \tag{5}$$

$$y_k = \sum_{j=1}^{T} v_{jk} f\left(\sum_{i=1}^{n} w_{ij}x_{ij} - w_{0j}\right) - v_{0k}, \tag{6}$$

where y_k is the kth component of output vector $Y = [y_1, y_2, \ldots, y_m]^T$. Commonly, these two equations can be rewritten in the following uniforms:

$$\begin{aligned} y_k &= \sum_{i=0}^{n} v_{ki} f\left(\sum_{j=0}^{T} w_{ij}x_{ij}\right), \\ y_k &= \sum_{j=0}^{T} v_{jk} f\left(\sum_{i=0}^{n} w_{ij}x_{ij}\right), \end{aligned} \tag{7}$$

where $x_{0j} = 1$, $x_{i0} = 1$, $f(\sum_{j=0}^{T} w_{0j}x_{0j}) = 1$, and $f(\sum_{i=0}^{n} w_{i0}x_{i0}) = 1$.

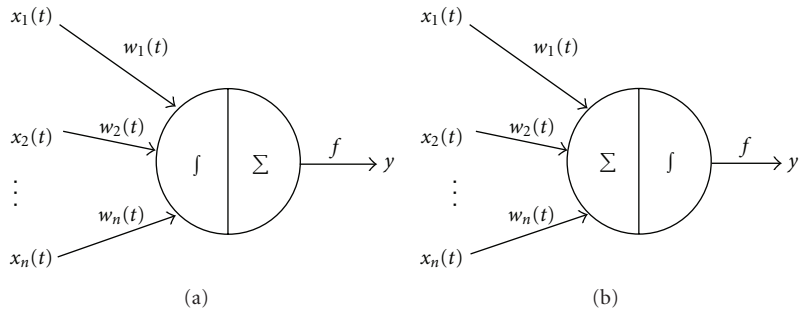

Figure 1: Procedural neurons with continuous time input functions.

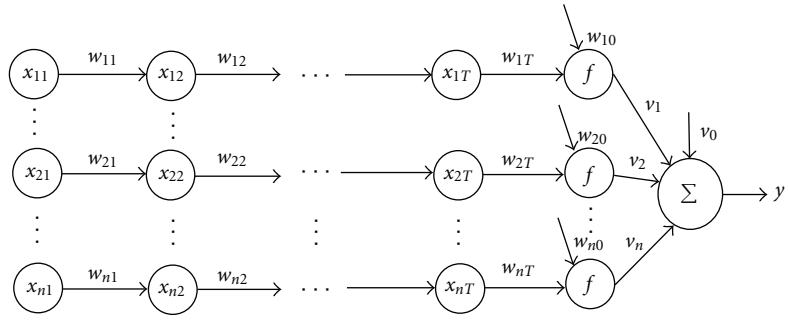

Figure 2: Procedural neurons for the case of discrete time (time first).

For the case of PNN with hidden layers, the structure is very like a multilayer feed-forward neural networks except the input neurons which can accept time series. PNN model with one hidden layer is shown in Figure 4 (time first), in which f_i denotes transfer function of the ith procedural neuron as in Figure 2.

Also in Figure 4, the other parts of the neural networks are quite similar to the traditional multilayer neural networks. Considering the case of PNN with one hidden layer and time-first accumulation, we give the following expression:

$$y_k = \sum_{h=0}^{H} u_{kh} g\left(\sum_{i=0}^{n} v_{ih} f\left(\sum_{j=0}^{T} w_{ij} x_{ij}\right)\right), \tag{8}$$

where g is a transfer function from the hidden nodes to the output layer and and u_{kh} is the weight connecting the hidden and the output nodes. H is the number of nodes in the hidden layer.

3.3. Some Basic Properties. Now, let us consider the computational ability of PNN, which means what kinds of problems PNN can simulate and how PNNs realize them. Comparing with BNN, PNN tries to map an n-dimensional function to an m-dimensional vector. So PNN is a functional function which is defined in the functional space. For the sake of convenience, let us focus on classification problems. PNN tries to classify the given functions or time series. In a spatiotemporal domain, we face much more complicated issues those that in Euclid space and need much more work to do. However, based on functional analysis, it is possible to make the approach on functional classification and we can extend some classic results on BNN to the case of PNN. In [12] we have discussed some theoretical problems on PNN, such as continuity, Lipschitz condition, computational ability and functional approximation. But there are still some issues which remain unknown, for example, how to define the distance between two functions, how to define the margin between two classes of functions, how to evaluate the complexity of functional functions for given time series problem, and so forth.

Suppose $X(t), \widetilde{X}(t) \in \mathbb{G}^n[a,b]$, where $\mathbb{G}^n[a,b]$ is an n-dimension functional space in interval $[a,b]$, endowed with the following metric distance:

$$\left\|X(t) - \widetilde{X}(t)\right\|_p = \left(\sum_{i=1}^{n} \int_a^b |x_i(t) - \widetilde{x}_i(t)|^p \mathrm{d}t\right)^{1/p}, \tag{9}$$

where $p > 0$. Also assume $Y \in \mathbb{R}^m$, $\mathbb{R}^m$ is an m-dimensional Euclid space. We define $\mathbb{F} : \mathbb{G}^n[a,b] \to \mathbb{R}^m$ to be a special function set, which maps an functional space to an Euclid space. For any $F_1, F_2 \in \mathbb{F}$ and

$$Y_i = F_i(X(t)), \quad i = 1, 2 \tag{10}$$

considering the metric distance

$$\|Y_1 - Y_2\|_p = \|F_1(X(t)) - F_2(X(t))\|_p, \tag{11}$$

what we are interested in is that (11) satisfies the following condition:

$$\|Y_1 - Y_2\|_p = \|F_1(X(t)) - F_2(X(t))\|_p = L\left\|X(t) - \widetilde{X}(t)\right\|_p, \tag{12}$$

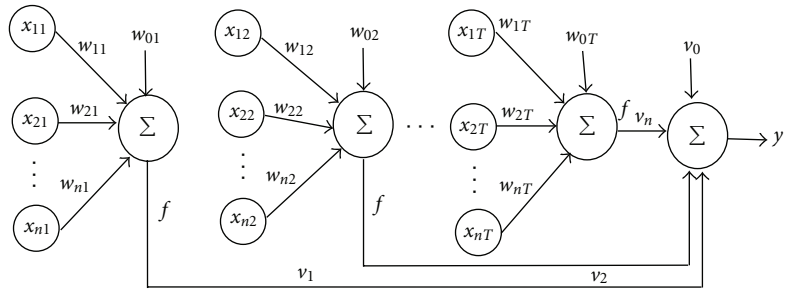

Figure 3: Procedural neurons for for the case of discrete time (space first).

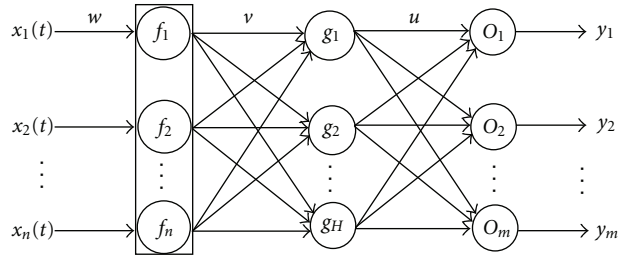

Figure 4: Procedural neural networks with one hidden layer (time first).

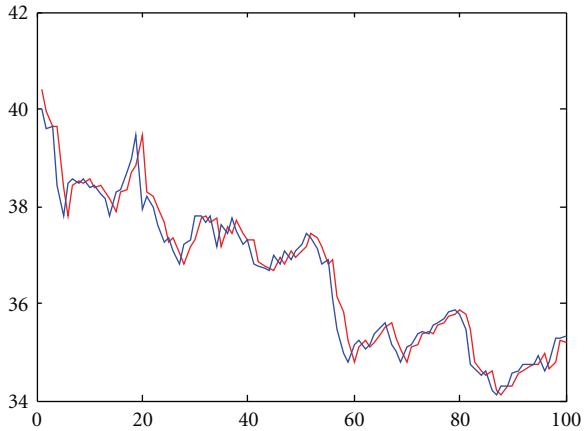

Figure 5: PNN prediction versus actual open price.

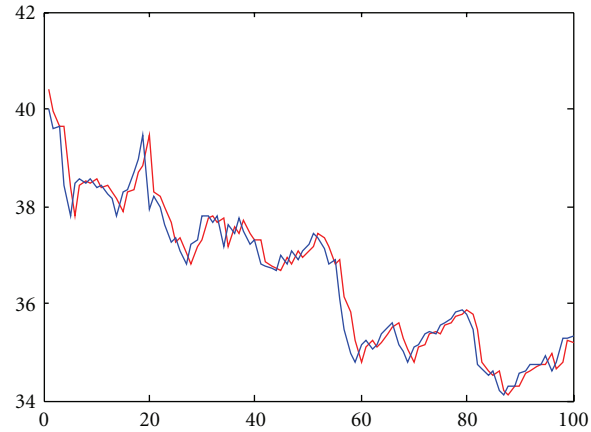

Figure 6: PNN prediction versus actual high price.

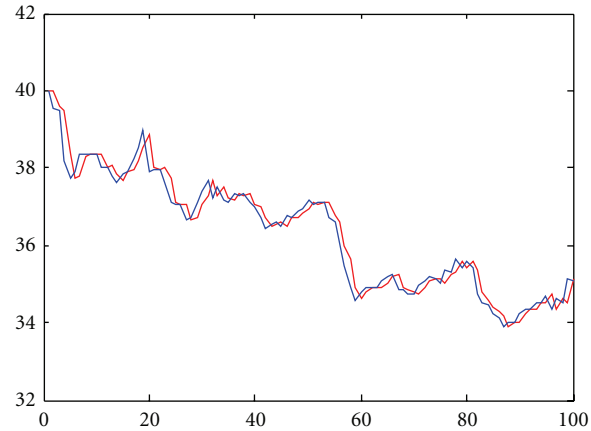

Figure 7: PNN prediction versus actual low price.

where L is the so-called Lipschitz constant. If function $F \in \mathbb{F}$ satisfies (12), we say F is Lipschitzed. It is trivial to prove the following theorem [46].

Theorem 1. *PNNs defined in* (1)–(6) *are Lipschitzed if the transfer functions* f *(and* g *in* (8)*) are Lipschitzed.*

Actually, we have proved the following results in [12, 46].

Theorem 2. *For any functional function* $F \in \mathbb{F}$ *satisfying Lipschitz condition as in* (12)*, there is a procedural neural networks* P *in the form of* (8) *which satisfies*

$$\|F - P\|_p = \|F(X(t)) - P(X(t))\|_p < \varepsilon, \qquad (13)$$

where ε *is an arbitrary small positive real number.*

4. Learning Algorithm and Complexity Analysis

4.1. Preparing Samples for Training PNN. To prepare samples for training PNN, the purpose focuses on how to make the data series into pieces, and each piece of data forms a sample. For different problems, there are different ways in organizing samples. In the case of stock markets, weekdata (five-day-data) naturally form a relative independent data which is a group of daily data. Sometimes we take such data from several weeks as a sample (which is a big sample) if these

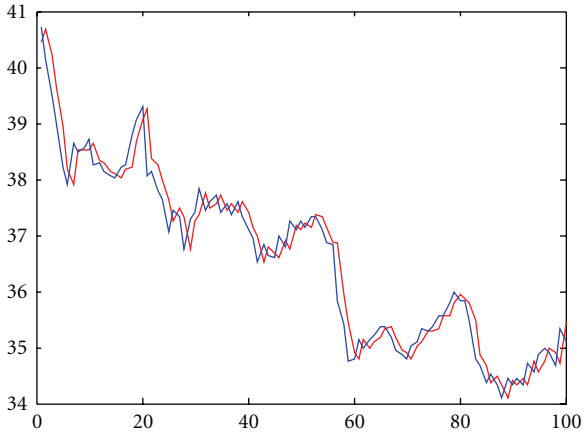

Figure 8: PNN prediction versus actual close price.

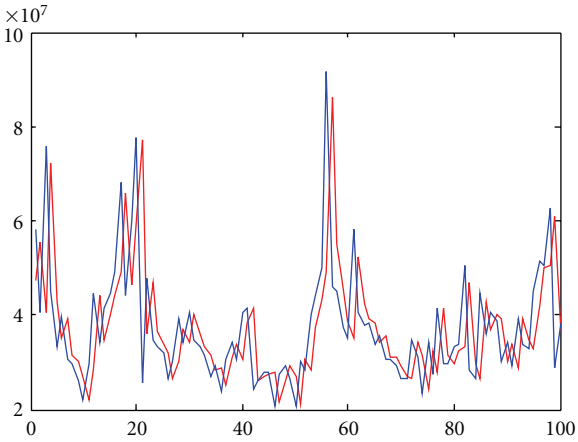

Figure 9: PNN prediction versus actual value of volume.

data are relatively dependent, for example, one-month data or one-season data.

For each sample, data is discrete and appears as a matrix as follows:

$$X = \left(x_{ij}\right)_{n\times T}, \tag{14}$$

where n is the dimension of the data in space and T is the number of sampling on time axis. To construct a sample, typically we choose $Z = [z_1, z_2, \ldots, z_m]$ as the corresponding output of the model, while input is X, and (X, Z) is the normal form of a sample. Compared with samples for training BNN where both input and output are vectors, here X in 15 is a matrix and Z is a vector. So during the training, PNN tries to map a matrix to a vector, namely, $P : X \rightarrow Z$ or $Z = P(X)$.

4.2. Learning Algorithm. Supposing (X, Z) is a sample for training the PNN, there are various forms of error functions which devote to minimize the difference between the output of PNN and the desire output of Z. One of them, a typical

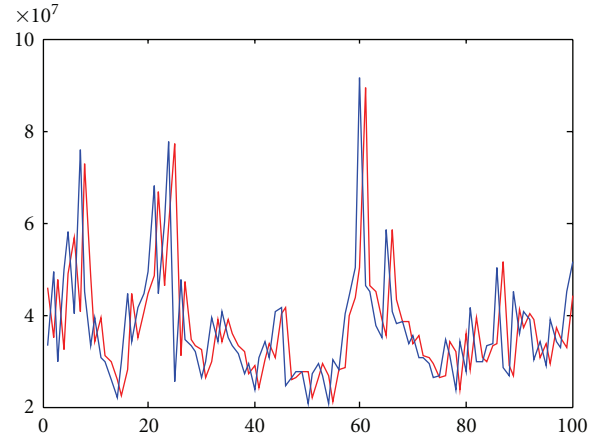

Figure 10: BNN prediction versus actual value of volume.

form, is the minimum square error (MSE) represented as follows:

$$E = \frac{1}{2}\sum_{k=1}^{m}(y_k - z_k)^2. \tag{15}$$

In the case of PNN with one hidden layer (see e.g., (8)), the MSE becomes

$$E = \frac{1}{2}\sum_{k=1}^{m}\left(\sum_{h=0}^{H} u_{kh} g\left(\sum_{i=0}^{n} v_{ih} f\left(\sum_{j=0}^{T} w_{ij} x_{ij}\right)\right) - z_k\right)^2. \tag{16}$$

Referring to the deepest gradient descent algorithm, we have the following formulas which contribute to compute the update forms of weights in PNN

$$\begin{aligned}
\frac{\partial E}{\partial u_{kh}} &= (y_k - z_k) g\left(\sum_{i=0}^{n} v_{ih} f\left(\sum_{j=0}^{T} w_{ij} x_{ij}\right)\right),\\
\frac{\partial E}{\partial v_{ih}} &= \sum_{k=1}^{m}(y_k - z_k) u_{kh} g'\left(\sum_{i=0}^{n} v_{ih} f\left(\sum_{j=0}^{T} w_{ij} x_{ij}\right)\right)\\
&\quad \times f\left(\sum_{j=0}^{T} w_{ij} x_{ij}\right),\\
\frac{\partial E}{\partial w_{ij}} &= \sum_{k=1}^{m}(y_k - z_k)\sum_{h=0}^{H} u_{kh} g'\left(\sum_{i=0}^{n} v_{ih} f\left(\sum_{j=0}^{T} w_{ij} x_{ij}\right)\right)\\
&\quad \times f'\left(\sum_{j=0}^{T} w_{ij} x_{ij}\right) x_{ij}.
\end{aligned} \tag{17}$$

In the case of the procedural perception (see e.g., (3)), the corresponding error function is

$$E = \frac{1}{2}\sum_{k=1}^{m}\left(v_{ki} f\left(\sum_{i=0}^{n} w_{ij} x_{ij}\right) - z_k\right)^2. \tag{18}$$

Table 1: Ten records of a stock price list.

Date	Open	High	Low	Close	Volume
01/03/2000	153.00	153.69	149.19	150.00	22069800
01/04/2000	147.25	148.00	144.00	144.00	22121400
01/05/2000	143.75	147.00	142.56	143.75	27292800
01/06/2000	143.13	146.94	142.63	145.67	19873200
01/07/2000	148.00	151.88	147.00	151.31	20141400
01/10/2000	152.69	154.06	151.13	151.25	15226500
01/11/2000	151.00	152.69	150.63	151.50	15123000
01/12/2000	151.06	153.25	150.56	152.00	18342300
01/13/2000	153.13	154.94	153.00	153.75	14953500
01/14/2000	153.38	154.63	149.56	151.00	18480300

The updating weights are reduced to

$$\frac{\partial E}{\partial v_{ki}} = (y_k - z_k) f\left(\sum_{j=0}^{T} w_{ij} x_{ij}\right),$$
$$\frac{\partial E}{\partial w_{ij}} = \sum_{k=1}^{m} (y_k - z_k) v_{ki} f'\left(\sum_{j=0}^{T} w_{ij} x_{ij}\right) x_{ij}. \tag{19}$$

The final forms of updating weights corresponding to (17) can be described as

$$u_{kh}(t+1) = u_{kh}(t) - \alpha(t)\frac{\partial E(t)}{\partial u_{kh}(t)},$$
$$v_{ih}(t+1) = v_{ih}(t) - \beta(t)\frac{\partial E(t)}{\partial v_{ih}(t)}, \tag{20}$$
$$w_{ij}(t+1) = w_{ij}(t) - \gamma(t)\frac{\partial E(t)}{\partial w_{ij}(t)},$$

where $\alpha(t)$, $\beta(t)$, and $\gamma(t)$ are learning speed factors and can be optimized during learning. Usually $\alpha(t), \beta(t), \gamma(t) \in [0, 1]$ are chosen as constants, for example, $\alpha(t) = \alpha$, $\beta(t) = \beta$ and $\gamma(t) = \gamma$.

4.3. Computational Complexity Analysis. Training PNN is quite similar to training BNN. Both of them refer to the deepest gradient descent algorithm. In the view of computation, training PNN with one hidden layer costs the same amount of time as training BNN with two hidden layers, in which one of the hidden layers in BNN takes T as the number of nodes.

Of course, training PNN with the deepest gradient descent algorithm cannot avoid the issue of minimal value of the error function. It has a similar problem with training BNN. Therefore, training PNN remains NP-hard in computational complexity as with training BNN. However, there are still some improved aspects compared with BNN, such as reducing the dimension of input space, processing time series and spacial information synthetically, and endowing different weights with respect to time and space.

5. Experimental Results and Comparisons

The stock data comes from Yahoo finance web site [50]. Historical chart data and daily updates were provided by Commodity Systems, Inc. (CSI). International historical chart data and daily updates were provided by Hemscott Americas. Fundamental company data provided by Capital IQ. Quotes and other information supplied by independent providers can be identified on the Yahoo Finance partner page. Data in one week or five days composes a sample for PNN. For each sample there are five observation fields including the open price, the highest price, the lowest price, the closing price, and the stock volume. Table 1 lists ten-day records of Yahoo stock from 01/03/2000 to 01/14/2000.

It is the experimental work to fix the parameters of PNN for a given data set. Here we choose a group of parameters in training a time-first PNN with one hidden layer, for example, the input node number $n = 5$, the time sampling with $T = 1, 2, 3, 4, 5$, the hidden node number $H = 1$, and the transfer functions $f(x) = g(x) = (1 + \exp(-x))^{-1}$.

Also the parameters for training algorithm are important experimentally. Here we suggest a group of such parameters so that the readers can repeat the experiment easily, for example, the size of training set $N = 25$, the precision of difference between two error functions in the succeeded loop $\varepsilon = 0.0001$, and the learning rate $\alpha = \beta = \gamma = 0.7$. Experiments show that for most of the test samples 10 loops of training are enough for the given ε. Here, 1000 samples are selected as the test set, and for each test sample the nearest $T + 25$ samples in date are used as the training set.

From Figure 5 to Figure 10, the open, high, low, and close price and the volume are plotted in which both predictive and actual values are given. The blue lines denote the actual prices and the red lines stand for the predictive values.

In our experiment,we compare PNN with BNN which is a spacial case of PNN when $T = 1$. The methods of HMM, SVM are also compared.

Here is a simple method to evaluate the model for predictions. If the actual price increases or decreases in the next day and the prediction is also of the same increases or decreases, we say the model "hit" the point, otherwise the model "miss" the point. The percent of the "hits" points to the total points is named hit rate. In Tables 2 and 3, the hit

Table 2: Hit-rate comparison of PNN (time first) versus BNN and SVM.

Model	Open price	Close price	High price	Low price	Volume
BNN	0.589	0.547	0.550	0.664	**0.779**
HMM	0.628	0.688	0.638	0.660	0.640
SVM	0.612	0.633	0.666	0.667	0.655
PNN ($T = 2$)	0.634	0.642	0.639	0.679	0.689
PNN ($T = 3$)	0.707	**0.705**	**0.703**	**0.713**	0.676
PNN ($T = 4$)	**0.718**	0.667	0.679	0.697	0.666
PNN ($T = 5$)	0.664	0.665	**0.693**	0.676	0.658

Table 3: Hit-rate comparison of PNN (space first) versus BNN and SVM.

Model	Open price	Close price	High price	Low price	Volume
BNN	0.637	0.550	0.550	0.624	**0.788**
HMM	0.628	0.688	0.638	0.660	0.640
SVM	0.612	0.633	0.666	0.667	0.655
PNN ($T = 2$)	0.713	0.649	**0.749**	0.713	0.704
PNN ($T = 3$)	0.676	0.673	0.704	**0.722**	0.681
PNN ($T = 4$)	**0.718**	0.672	0.676	0.684	0.687
PNN ($T = 5$)	0.690	**0.689**	0.703	0.682	0.665

rates of the models PNN, BNN, HMM, and SVM for the four prices and volume are compared.

Table 2 and 3 show that in the two cases of both time-first and space-first models of PNN, PNN performs higher hit rate than BNN, HMM, and SVM in prediction of daily stock price. Unfortunately, the behavior of daily stock volume seems to be more dependent on the price of the last day and has little relation with the data far from the current day. This phenomenon appears in the performance of Yahoo stock and can be shown experimentally by the last column of Table 2 and 3. It is obvious that the hit rates of the volume decrease when more data of days is involved.

6. Conclusion

Stock markets are too complicated to be successfully predicted and have not been consistent in theory so far (some people say it is not disciplinarian and cannot be predicted). This paper deals with stock market predictions based on a novel model, named procedural neural networks. Two structures of procedural neural networks are proposed: one is time first-model in which time sampling calculation is computed first for the input data, the other is a space-first model in which spacial accumulate is computed first for the multiple-dimensional input data. These two models can well handle various time-space series problems, especially for large scale of data. To verify the efficiency of the improved PNN model, stock price prediction is designed by training PNN. While training PNN, a group of time series data is regarded as one sample. In this way, PNN is flexible in structure to fit the time series problems. Sometimes, the desire outputs of the real-life system depend not only on the last former state of the system, but also on a series of former states. So it is necessary to find the inherent relevance among data and separate the data into samples in an appropriate manner. Computational complexity cannot be ignored in this paper because of its large size of data when we process the time series issues. The training process of PNN is similar to that of BNN, but the input dimensions of PNN are much lower than those of BNN. Moreover, PNN decreases the time for aggregating information from different time segments. Unfortunately, the generalization ability of PNN for new coming data is still a challenging issue.

Acknowledgments

This work is partly supported by CMP Laboratory, Department of Cybernetics, Faculty Electrical Engineering, Czech Technical University. The first author is supported by the Agreement between the Czech Ministry of Education and the Chinese Ministry of Education. Thanks are offered to all CMP members who have discussed with the authors and given them suggestions on theoretical problems and experiments.

References

[1] H. White, "Economic prediction using neural networks: the case of IBM daily stock returns," in *Proceedings of the IEEE International Conference on Neural Networks*, San Diego, Calif, USA, 1988.

[2] E. W. Saad, D. V. Prokhorov, and D. C. Wunsch II, "Comparative study of stock trend prediction using time delay, recurrent and probabilistic neural networks," *IEEE Transactions on Neural Networks*, vol. 9, no. 6, pp. 1456–1470, 1998.

[3] R. J. Kuo, C. H. Chen, and Y. C. Hwang, "An intelligent stock trading decision support system through integration of

genetic algorithm based fuzzy neural network and artificial neural network," *Fuzzy Sets and Systems*, vol. 118, no. 1, pp. 21–45, 2001.

[4] K.-J. Kim and I. Han, "Genetic algorithms approach to feature discretization in artificial neural networks for the prediction of stock price index," *Expert Systems with Applications*, vol. 19, no. 2, pp. 125–132, 2000.

[5] L. Cao and F. E. H. Tay, "Financial forecasting using Support Vector Machines," *Neural Computing and Applications*, vol. 10, no. 2, pp. 184–192, 2001.

[6] R. K. Wolfe, "Turning point identification and Bayesian forecasting of a volatile time series," *Computers and Industrial Engineering*, vol. 15, pp. 378–386, 1988.

[7] M. R. Hassan and B. Nath, "Stock market forecasting using hidden Markov model: a new approach," in *Proceedings of the 5th International Conference on Intelligent Systems Design and Applications*, pp. 192–196, Wroclaw, Poland, September 2005.

[8] Y.-Q. Zhang, S. Akkaladevi, G. Vachtsevanos, and T. Y. Lin, "Granular neural web agents for stock prediction," *Soft Computing*, vol. 6, pp. 406–413, 2002.

[9] W. Dekker, "The fractal geometry of the European eel stock," *ICES Journal of Marine Science*, vol. 57, no. 1, pp. 109–121, 2000.

[10] P. Oswiecimka, J. Kwapien, S. Drozdz, and R. Rak, "Investigating nultifractality of stock market fluctuations using wavelet and detrending fluctuation methods," *Acta Physica Polonica B*, vol. 36, pp. 2447–2457, 2005.

[11] J. Z. Liang, J. Q. Zhou, and X. G. He, "Procedure neural networks with supervised learning," in *Proceedings of the 9th International Conference on Neural Information Processing*, pp. 523–527, orchid country club, singapore, November 2002.

[12] J. Z. Liang and X. G. He, "Function approximation of fuzzy neural network with its research on learning algorithm," Ph.D. dissertation, BeiHang University, 2001.

[13] J. Z. Liang, "Functional procedure neural network," *Dynamic of Continuous Discrete and Impulsive Systems-Series B*, pp. 27–31, 2005.

[14] R. Ball, "The development, accomplishments and limitations of the theory of stock market efficiency," *Managerial Finance*, vol. 20, no. 2, pp. 3–48, 1994.

[15] L. R. Rabiner, "Tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[16] M. R. Hassan, "A combination of hidden Markov model and fuzzy model for stock market forecasting," *Neurocomputing*, vol. 72, no. 16, pp. 3439–3446, 2009.

[17] L. S. Kuei, W. S. Yun, and T.P. Ling, "Application of hidden Markov switching moving average model in the stock markets: theory and empirical evidence," *International Review of Economics and Finance*, vol. 18, no. 2, pp. 306–317, 2009.

[18] P. Smyth, D. Heckerman, and M. I. Jordan, "Probabilistic independence networks for hidden Markov probability models," *Neural Computation*, vol. 9, no. 2, pp. 227–269, 1997.

[19] R. Elliott, L. Aggoun, and J. Moore, *Hidden Markov models: estimation and Control*, Springer, Berlin, Germany, 1994.

[20] R. Elliott and P. Kopp, *Mathematics of Financial Markets*, Springer, Berlin, Germany, 2004.

[21] L. Aggoun and R. Elliott, *Measure Theory and Filtering: Introduction and Applications*, Cambridge University Press, Cambridge, UK, 2004.

[22] J. Buffington and R. Elliott, "Regime switching and European options," in *Stochastic Theory and Control: Proceedings of a Workshop held in Lawrence, Kansas*, B. Pasik-Duncan, Ed., pp. 73–81, Springer, Berlin, Germany, 2002.

[23] J. Buffington and R. Elliott, "American options with regime switching," *International Journal of Theoretical and Applied Finance*, vol. 5, pp. 497–514, 2002.

[24] L. L. Ghezzi and C. Piccardi, "Stock valuation along a Markov chain," *Applied Mathematics and Computation*, vol. 141, no. 2, pp. 385–393, 2003.

[25] R. Elliott, L. Chan, and T. Siu, "Option pricing and Esscher transform under regime switching," *Annals of Finance*, vol. 1, no. 4, pp. 423–432, 2005.

[26] M. R. Hassan, B. Nath, and M. Kirley, "A fusion model of HMM, ANN and GA for stock market forecasting," *Expert Systems with Applications*, vol. 33, no. 1, pp. 171–180, 2007.

[27] Y. Wang, K. K. Wong, X. F. Liao, and G. Chen, "A new chaos-based fast image encryption algorithm," *Applied Soft Computing*, vol. 11, no. 1, pp. 514–522, 2011.

[28] M. J. Pring, *Analisi Tecnica dei Mercati Finanziari*, McGraw Hill Italia, Milano, Italy, 2002.

[29] W. C. Clide and C. L. Osler, "Charting: chaos theory in disguise?" *Journal of Futures Markets*, vol. 17, no. 5, pp. 489–514, 1997.

[30] T. Schreimber, "Interdisciplinary application of nonlinear time series methods," *Physics Reports*, vol. 308, pp. 1–64, 1998.

[31] H. O. Peitgen, H. Jurgens, and D. Saupe, *Chaos and Fractals: New Frontiers of Science*, Springer, Berlin, Germany, 2004.

[32] D. A. Hsieh, "Chaos and nonlinear dynamics: application to financial markets," *Journal of Finance*, vol. 46, pp. 1839–1877, 1991.

[33] C. Brown, *Chaos and Catastrophe Theories*, SAGE publications, Thousand Oaks, Calif, USA, 1995.

[34] R. L. Devaney, *Caos e Frattali*, Addison-Wesley Published Company, Milano, Italy, 1990.

[35] "A review of applications in science," in *Gli Oggetti Frattali*, B. B. Mandelbrot, Ed., Giulio Einaudi, Milano, Italy, 1987.

[36] L. Y. Su, "Prediction of multivariate chaotic time series with local polynomial fitting," *Computers and Mathematics with Applications*, vol. 59, no. 2, pp. 737–744, 2010.

[37] L. C. Jain and N. M. Martin, *Fusion of Neural Networks, Fuzzy Sets, and Genetic Algorithms: Industrial Applications*, CRC Press, New York, NY, USA, 1999.

[38] A. Abraham, B. Nath, and P. K. Mahanti, "Hybrid intelligent systems for stock market analysis," in *Proceedings of the International Conference on Computational Science*, V. N. Alexandrov et al., Ed., pp. 337–345, Springer, San Francisco, Calif, USA, May 2001.

[39] D. Y. Chiu and P. J. Chen, "Dynamically exploring internal mechanism of stock market by fuzzy-based support vector machines with high dimension input space and genetic algorithm," *Expert Systems with Applications*, vol. 36, no. 2, pp. 1240–1248, 2009.

[40] L. A. Zadeh, in *Foreword of the Proceedings of the Second International Conference on Fuzzy Logic and Neural Networks*, pp. xiii–xiv, Iizuka, Japan, 1992.

[41] M. M. Mostafa, "Forecasting stock exchange movements using neural networks: empirical evidence from Kuwait," *Expert Systems with Applications*, vol. 37, no. 9, pp. 6302–6309, 2010.

[42] S. Duerson, F. S. Khan, V. Kovalev, and A. H. Malik, "Reinforcement learning in online stock trading systems," http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.83.5299.

[43] W. Huang, Y. Nakamori, and S.-Y. Wang, "Forecasting stock market movement direction with support vector machine," *Computers & Operations Research*, vol. 32, no. 10, pp. 2513–2522, 2005.

[44] J. Z. Liang, "Support functional machines," in *Proceedings of the 8th International Conference on Intelligent Data Engineering*

and Automated Learning, pp. 1–9, Springer, Birmingham, UK, December 2007, LNCS4881.

[45] X. G. He and J. Z. Liang, "Procedure neural networks," in *Proceedings of Conference on Intelligent Information Processing*, Z. Shi, Ed., 16th World Computer Congress 2000, pp. 143–146, Publishing House of Electronics Industry, China, 2000.

[46] X. G. He and J. Z. Liang, "Some theoretic problems of procedure neural network," *Engineering Science in China*, vol. 2, no. 12, pp. 40–44, 2000.

[47] X. G. He, J. Z. Liang, and S. H. Xu, "Training and application of procedure neural network," *Engineering Science in China*, vol. 3, no. 4, pp. 31–45, 2001.

[48] J. Z. Liang and J. M. Han, "Complex number procedure neural networks," in *Proceedings of the 1st International Conference on Natural Computation*, pp. 336–339, Springer, Changsha, China, August 2005, Part I, LNCS3610.

[49] J. Z. Liang and X. H. Wu, "Segment procedure neural networks," in *Proceedings of the IEEE International Conference on Granular Computing*, pp. 526–529, Beijing, China, July 2005.

[50] 2007, http://finance.yahoo.com/q/hp?s=GE&a=00&b=1&c=2007&d=06&e=26&f=2007&g=d.