Nayak and Misra *Financial Innovation* (2018) 4:21
https://doi.org/10.1186/s40854-018-0104-2

Financial Innovation

RESEARCH Open Access

Estimating stock closing indices using a GA-weighted condensed polynomial neural network

Sarat Chandra Nayak[1*] and Bijan Bihari Misra[2]

* Correspondence: saratnayak234@gmail.com; sarat_silicon@yahoo.co.in
[1]Department of Computer Science and Engineering, CMR College of Engineering &Technology (Autonomous), 501401, Hyderabad, India
Full list of author information is available at the end of the article

Abstract

Accurate forecasting of changes in stock market indices can provide financial managers and individual investors with strategically valuable information. However, predicting the closing prices of stock indices remains a challenging task because stock price movements are characterized by high volatility and nonlinearity. This paper proposes a novel condensed polynomial neural network (CPNN) for the task of forecasting stock closing price indices. We developed a model that uses partial descriptions (PDs) and is limited to only two layers for the PNN architecture. The outputs of these PDs along with the original features are fed to a single output neuron, and the synaptic weight values and biases of the CPNN are optimized by a genetic algorithm. The proposed model was evaluated by predicting the next day's closing price of five fast-growing stock indices: the BSE, DJIA, NASDAQ, FTSE, and TAIEX. In comparative testing, the proposed model proved its ability to provide closing price predictions with superior accuracy. Further, the Deibold-Mariano test justified the statistical significance of the model, establishing that this approach can be adopted as a competent financial forecasting tool.

Keywords: Stock market forecasting, Polynomial neural network, Partial description, Genetic algorithm, Multilayer perceptron

Introduction

Stock index forecasting is the process of making predictions about the future performance of a stock market index based on existing stock market behavior. Over the last few decades, stock index modeling and forecasting has been an important and challenging task for researchers in both financial engineering and mathematical economics. Stock market behavior is very much like a random walk process, and the serial correlations are economically and statistically insignificant. Stock market forecasting is regarded as a difficult and intricate undertaking in financial time-series forecasting because of the uncertainties involved in the movement of the markets, the highly volatile nature of the markets, nonlinearities, discontinuities, the movement of other stock markets, political influences, and the psychology of individuals, along with many other macro economic factors (Abdoh and Jouhare 1996; Oh and Kim 2002; Wang 2003). Studies of stock price prediction such as (Huang et al. 2008; Liu et al. 2009) have employed various economic factors, including oil prices, exchange rates, interest rates, stock price indices in other countries, and domestic/

global economic situations. All these factors have proven to be important elements influencing the markets. As increasing amounts of money are invested in the stock market by inexperienced investors, institutions, brokers, and speculators, there is an increased tendency for investors to become anxious about the future trends of stock prices. Consequently, an effective and more accurate forecasting model is needed to predict stock market behavior. If the direction of the market can be predicted successfully, investors may find better guidance, and the financial rewards could be substantial.

In recent years many new methods for modeling and forecasting the stock market have been developed. Despite these efforts, the forecasting accuracy of these models remains an issue in stock market research. To address this challenge, we developed an efficient model for stock market forecasting that proposes a condensed polynomial neural network (PNN) architecture for predicting stock index closing prices. The model includes partial descriptions (PDs) and is limited to only two layers for the PNN architecture. The outputs of these PDs, along with the original features, are fed to the output layer, which has one neuron. The weight vectors and biases of the CPNN are explored by a GA.

The remainder of this paper is organized as follow: Section "Literature Review" provides a review of the developments in the literature of this field. Section "Model development" describes the architecture details of the forecasting model. Our experimental results are presented and analyzed in Section "Experimental results and analysis", and Section 5offers our concluding remarks along with our proposals for the direction of future research.

Literature review

For many decades, linear models served as the basis of traditional statistical forecasting in financial engineering. However, because of the presence of noise and non-linearity in the financial time series, such traditional methods seldom proved effective. In comparison, nonlinear dynamics proposes that past prices help to determine future prices in a financial time series, but not in a straight forward way. The relationship between past prices and future prices is nonlinear, and this non-linearity implies that past price changes can have wide ranging effects on future prices. Several statistical techniques have been used extensively for stock market prediction (Ravichandran et al. 2007). Among these approaches, moving averages (MA), auto-regressive integrated moving averages (ARIMA), auto-regressive conditional heteroscedasticity (ARCH), and generalized ARCH (GARCH) have received wide acceptance and have been used successfully in various engineering, economic, and social applications. For example, an ARCH-M model augmented by an information diffusion indicator was proposed in (Xie and Wang 2015) for U.S. stock return forecasting. However, since these models were developed to address certain types of problems, they lacked the ability to capture the non-linearity of other types of time series.

The Box-Jenkins method using an auto regressive moving average (ARMA) linear model was applied extensively in many areas of time series forecasting (Box and Jenkins 1976). The combination of the ARIMA-GARCH model was suggested for predicting the movement of selected stocks in India (Narendra Babu and Eswara Reddy 2015). Separately, the GARCH model showed its superior capability in modeling and forecasting exchange rate volatility (Abdullah et al. 2017). Huang and Kou (2014) proposed a kernel entropy manifold learning approach to measure the relationship between two financial data points. They claimed improved accuracy not only for financial warnings, but also for the criteria for explaining and predicting stock market volatility. Huang et al. proposed a non linear

manifold learning technique for early warnings in financial markets (Huang et al. 2017). A wavelet-based approach for co-movement analysis of Asian stock markets against the FTSE 100 and S&P 500 was proposed in (Yilmaz and Unal 2016). A multi-criteria decision-based approach for financial risk analysis was offered in (Kou et al. 2014), where the authors evaluated six popular clustering algorithms and eleven cluster validity indices over three real-world financial data sets.

The last two decades have seen tremendous development in soft computing, including artificial neural networks (ANNs), evolutionary algorithms, and fuzzy systems. This improvement in computational intelligence capabilities enhanced the modeling of complex, dynamic, and multivariate non linear systems. Soft computing methodologies were applied successfully to data classification, financial forecasting, credit scoring, portfolio management, risk level evaluation, and other areas, producing improved performance. The advantage of applying an ANN to stock market forecasting is that this approach incorporates prior knowledge in the ANN to improve prediction accuracy. Use of ANNs also allows adaptive adjustment to the models and nonlinear descriptions of the problems.

ANNs have been applied successfully in financial engineering, and they have gained wide acceptance because of their superior learning and approximation capabilities. When the mapping from the input to the output contains both, regularities and exceptions, the use of an ANN is considered an effective modeling approach. ANNs have the ability to deal with complex problems of structural instability. Neural networks (NNs) are analogous to nonparametric, non-linear regression models. Their novelty lies in their ability to model non-linear processes with few (if any) a priori assumption about the nature of the generating process. This characteristic is particularly useful in financial engineering applications where much is assumed and little is known about the nature of the processes that determine asset prices.

It has been demonstrated that ANNs can learn highly non-linear models, have effective learning algorithms, can handle noisy data, and are able to use inputs of various kinds. Particularly, the multilayer perceptron architecture mostly applied as a forecasting model and found to be similar to other complex non-linear models based on exponential GARCH processes (Bollerslev 1986; Campbell et al. 1997). Among the earliest investigations, in1990 Kimoto et al. used a modular neural network to learn the relationships among various market factors (Kimoto et al. 1990). They used several learning algorithms and forecasting methods to develop a prediction system for the Tokyo Stock Exchange Prices Indexes (TOPIX).The correlation coefficients produced by their model was found to be much higher than those produced by using a multiple regression method. The researchers in (Trippi and DeSieno 1992) combined the outputs of individual networks using logical operators to produce a set of composite rules. They demonstrated that their best composite synthesized rule set system achieved a higher gain than obtained by previous research.

ANNs are used extensively in financial applications (Harvey et al. 2000; McGrath 2002; Kumar and Bhattacharya 2006). In 2005, Cao et al. (2005) used an ANN model to predict stock price movements for firms traded on the Shanghai stock exchange. The authors compared the predictive power using linear models to the predictive power of the uni-variate and multivariate ANN models. Their results showed that ANN models outperformed the linear models. These results were statistically significant across the sample firms, and indicated that NN models are useful for stock price prediction. Leigh et al. used ANN models and linear regression models to predict the

New York Stock Exchange Composite Index (Leigh et al. 2005). Their results were robust and informative as to the role of trading volume in the stock market. Chen et al. predicted the direction of return of the market index of the Taiwan stock exchange using a probabilistic neural network model (Chen et al. 2003). Then they compared the results to the generalized methods of moments (GMM) with a Kalman filter. In addition, researchers have used combinations of multiple neural networks as ensemble methods for improving prediction accuracy (Lahmiri 2018a; Lahmiri and Boukadoum 2015). A combination of individual models under series and parallel strategies was proposed by Khashei and Hajirahimi for financial time series (Khashei and Hajirahimi 2017). Their empirical results indicated that the series combination strategy produced more accurate hybrid models for financial time series forecasting.

Neuro-genetic hybrid networks have gained wide application for nonlinear forecasting because of their broad adaptive and learning abilities (Kwon and Moon 2007). The most widely used type of neural network s a back propagation neural network (BPNN), but it has many short comings such as low learning rate, long computation time, and a tendency to be stuck at the local minimum. Radial basis function (RBF) neural networks are also popular for predicting the stock market. This type of network has better calculation and spreading abilities, and stronger nonlinear mapping ability (Guangxu 2005).

Hybrid iterative evolutionary learning algorithms were shown to be more effective than conventional algorithms in terms of learning accuracy and prediction accuracy (Yu and Zhang 2005). Many researchers have adopted neural network models that are trained by genetic algorithms (GAs) (Nayak et al. 2012). Hybrid models that combine nonlinear models demonstrated better accuracies. Many researchers have used variations of this approach, for example by combining an ANN with evolutionary soft computing techniques such as particle swarm optimization (PSO), GAs, and other nature and bio-inspired search techniques. Compared to other evolutionary computing models, Gas and PSOs are most popular. Recently, Nayak et al. (Nayak et al. 2017) proposed the application of a GA for choosing the optimal parameters of ANN-based models. Here, the authors employed the hybrid model to explore virtual data positions in a financial time series, incorporating them to enhance the forecasting accuracy. Similarly, PSOs have been utilized in combination with ANNs for stock price forecasting (Lahmiri 2018b; Lahmiri 2016), and to train quantile regression neural networks for predicting financial time series volatility (Pradeepkumar and Ravi 2017). GAs have shown promising ability to search the optimal parameters of higher order neural networks for financial time series forecasting (Nayak et al. 2016a; Nayak et al. 2018; Nayak et al. 2016b).

From the literature, we can observe that use of a multilayer perceptron (MLP) has been adapted by researchers as the most promising and frequently used forecasting approach. An MLP contains more than one hidden layer, and each layer can contain more than one neuron. The input pattern is applied to the input layer of the network, and its effect propagates through the network layer by layer. During the forward phase, the synaptic weights of the networks are fixed. In the backward phase, the weights are adjusted in accordance with the error correction rule. MLPs use this algorithm, known as back propagation, for learning. While MLPs are popular, they have two well-known shortcomings: they suffer from slow convergence, and they tend to stick in local minima. The research by Calderon and Cheh in 2002 argued that the standard MLP network is subject to problems of local

minima (Calderon and Cheh 2002). Moreover, there is no formal method for deriving an MLP network for a given classification task (Swicegood and Clark 2001). To overcome the local minima, a greater number of nodes must be added to the hidden layers. However, increasing the hidden layers and adding more neurons in each layer contribute to increased computational complexity of the network. Hence, there is no direct method for finding an optimal MLP structure for solving a problem. The re fining process may suffer from long computational time because of iterative testing of various architectural parameters to adopt the most successful network architecture.

Based on our review of the existing literature on stock market index forecasting, we observed that important areas of present day research in stock market forecasting include improving forecasting accuracy while adapting models to have less computational complexity. Many of the latest evolutionary computation models have been applied for this purpose (Chakravarty and Dash 2012; Rout et al. 2013). Defining optimal architecture and parameters for an MLP is a matter of trial and error, which is computationally very expensive. Given the black-box nature and computational over load of this approach, we concluded that this focus diverts researchers' attention from other more simple and efficient models.

In1971, Ivahnenko (1971) suggested a PNN based on the group method of data handling (GMDH).The GMDH is aimed at identifying the functional structure of a model hidden within the empirical data. The main idea behind the evolution of the GMDH is the use of feed-forward networks based on short-term polynomial transfer functions whose coefficients are obtained using regression combined with emulation of the self-organizing activity behind neural network learning (Farlow 1984). Prior research demonstrated that the GMDH is the best optimal simplified model because it is simpler in structure than a traditional neural model, with higher accuracy for inaccurate, small, or noisy data sets (Ketabchi et al. 2010). AGMDH-type neural network based on a GA was used to predict the stock price index of the petro chemical industry in Iran (Shaverdi et al. 2012). The results obtained by using a GMDH-type neural network were excellent, and provided a high level of performance in stock price prediction. Use of PNNs was suggested and applied successfully for pattern and data classification tasks (Misra et al. 2006a; Misra et al. 2006b). The general approach is based on an evolutionary strategy in which the PNN generates the population or the layers of neurons/PDs, and selects and trains those PDs that provide the best classification. During learning, the PNN model grows with a new population of neurons and increased number of layers until a predefined criterion is met. Consequently, the complexity of the network increases (Ivahnenko 1971; Misra et al. 2006a; Misra et al. 2006b). However, such models can be described comprehensively by a set of short-term polynomials, thereby developing a PNN classifier. The coefficients of a PNN can be estimated by least square fit.

In summary, the forecasting accuracy of these models is still an issue in stock market research. In response, we propose a condensed PNN architecture for prediction of stock closing prices. We developed the partial descriptions (PDs) and set a limit of only two layers for the PNN architecture. The outputs of these PDs along with the original features were fed to the output layer having one neuron. The weight vectors

and biases of the CPNN were explored by a GA. The remainder of this paper explains and evaluates the proposed model.

Model development

This section describes briefly the architecture of three intelligent neural forecasting models considered in this research for the task of predicting one-day-ahead closing prices of major stock markets. The first model is based on the well-known MLP, the second is a radial basis function neural network (RBFNN-based) forecasting model, and the third is the proposed condensed polynomial neural network (CPNN). The MLP is trained with gradient descent as well as a genetic algorithm, hence constructing two separate models (MLP-GD and MLP-GA). Similarly, the CPNN model is first trained with the gradient descent method and a genetic algorithm separately, forming two forecasting models: the CPNN-GD and CPNN-GA forecasting models. The popular RBFNN architecture is described as well.

Multilayer perceptron

Multilayer perceptrons are among of the most widely implemented neural network topologies. An MLP is capable of approximating arbitrary functions in terms of mapping abilities. The feed forward neural network model considered here consists of one hidden layer only. The architecture of the MLP model is presented in Fig. 1. The MLP performs a functional mapping from the input space to the output space. The model discussed contains a single hidden layer, and there are m neurons in this layer. Since there are n input values in an input vector, the number of neurons in the input layer is equal to n. The first layer corresponds to the problem's input variables, with one node for each input variable. The second layer is useful in capturing non-linear relationships among variables. This model consists of a single output unit to estimate one-day-ahead closing prices. The neurons in the input layer use a linear transfer function, and the neurons in the hidden layer and output layer use a sigmoid function as presented in Eq.(1).

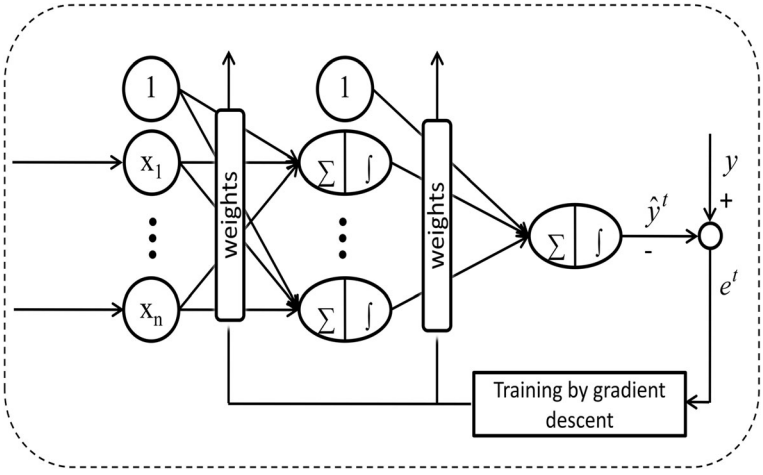

Fig. 1 Multilayer perceptron based forecasting model

$$y_{out} = \frac{1}{1 + e^{-\lambda y_{in}}} \tag{1}$$

Where 퓸표퓶퓶 is the output of the neuron, λ is the sigmoidal gain, and 퓸푗푗 is the input to the neuron. At each neuron j in the hidden layer, the weighted output z is calculated as in Eq.2.

$$z = f\left(B_j + \sum_{i=1}^{n} V_{ij} * X_i\right) \tag{2}$$

Where 푋푖 is the 푖th input vector, 푉푖푗 is the synaptic weight value between the 푖thinput neuron and 푗thhidden neuron, and 퐵푗 is the bias value. The output y at the single output neuron is calculated as in Eq.3.

$$y = f\left(B_0 + \sum_{j=1}^{m} W_j * z\right) \tag{3}$$

Where 푊푗 is the synaptic weight from the j^{th}hidden neuron to the output neuron, z is the weighted sum calculated as in Eq.2, and 퐵$_0$ is the output bias. This output y is compared to the desired output, the error is calculated, and then the error is propagated back. The weight and other parameter values are adjusted by the gradient descent rule for minimal error signal generation. Because of the gradient descent neural network learning, this approach is characterized by problems such as slow convergence and getting trapped in local minima, both of which affect the prediction capabilities of the model.

However, the genetic algorithm performs a search over the whole solution space, finds the optimal solution relatively easily, and does not require continuous differentiable objective functions. The problem of finding an optimal parameter set to train the MLP can be regarded as a search problem in the space of all possible parameters. The parameter set includes the weight set between the input-hidden layers, the weight set between the hidden-output layers, and the bias value. This search can be performed by applying a genetic algorithm. The chromosome representation for the GA is shown in Fig. 2.

The chromosomes of the GA represent the weight and bias values for a set of MLP models. Input data along with the chromosome values are fed to the set of MLP models. The fitness is obtained from the absolute difference between the target y and the estimated output $\hat{y}$. As the fitness value of an individual decreases, the GA considers the individual to be a better fit for the next generation. We used a binary encoding scheme for the experimental portion of our work. The weight values between the input and the hidden layer neuron are represented as 푉$_{11}$ to 푉푚푛. Weight values between the hidden and output layer are represented by 푊$_1$

Weight Values								Bias Values	
Input and Hidden Layer				Hidden and Output Layer				hidden	output
V_{11}	V_{12}	...	V_{nm}	W_1	W_2	...	W_m	$B_1 \ldots \ldots B_m$	B_0

Fig. 2 GA Chromosome representation for MLP

to 뀖퓏. The bias values of the hidden layer and output layer are represented by 뽭$_1$ and 뽭$_0$, respectively.

Radial basis functional neural network

The RBFNN-based forecasting model is shown by Fig. 3. For the input layer, the number of input neurons is determined based on the input signals that connect the network to the environment. The hidden layer consists of a set of kernels that carry out a nonlinear transformation from the input space to the hidden space. Two parameters, the center and the width, are associated with each RBF node. The center is determined during RBF training. Some of the commonly used kernel functions are the Gaussian function, cubic function, linear function, and generalized multi-quadratic function, among others. We used the Gaussian function as represented in Eq.4.

$$\varnothing_i(x) = exp\left(-\frac{\|x-\mu_i\|^2}{2\sigma_i{}^2}\right) \tag{4}$$

Where ‖⋯‖ represents the Euclidean norm, 픍 is the input vector, 휇$_{퓏}$ is the center, 휍$_{퓏}$ is the spread, and ∅(픍) represents the output of the 퓏th hidden node. The output of the RBF network is calculated as in Eq.5.

$$\hat{y} = f(x) = \sum_{k=1}^{N} w_k \varnothing_k(\|x-c_k\|) \tag{5}$$

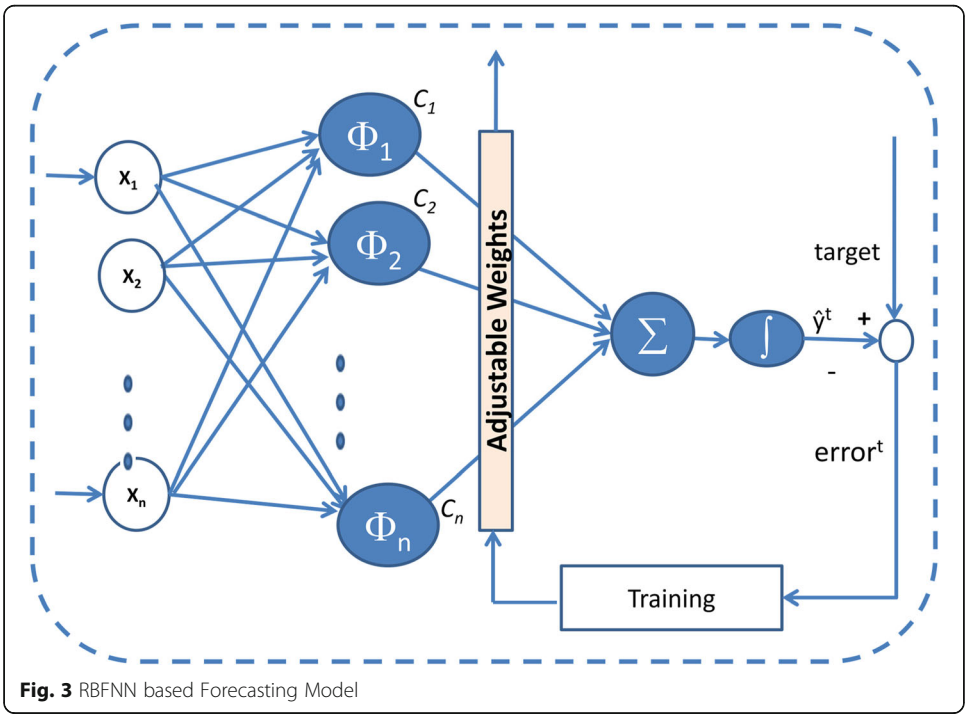

Fig. 3 RBFNN based Forecasting Model

where $\hat{y}$ is the network output, 픆 is an input vector signal, 픝 = [픝$_1$,픝$_2$,⋯,픝픢] 픧 is the weight vector in the output layer, N is the number of hidden neurons,∅픨(·) is the basis function, k is the bandwidth of the basis function, 픆 is the input vector, 뫔픨 = (뫔픨$_1$,뫔픨$_2$,⋯,뫔픨픪) 픧 is the center vector for 픨th node, and m is the number of input features.

Proposed CPNN-GA model

The proposed model develops PDs for two layers i.e., there are two hidden layers. The input layer is fed with the original input vector. The first hidden layer develops PDs with all pair combination of input features, which generates polynomials of degree 2. The PDs generated in the first hidden layer are utilized to develop the PDs in the second hidden layer. Each PD tries to approximate the input-output relationship of the data set. The optimal numbers of PDs yielding better performance are selected on an experimental basis with the hope of getting an improved result in subsequent layers. The optimal sets of PDs along with the original features are given as input to the neuron at the output layer. The weight vectors and biases are optimized by the GA. The proposed model can be represented as in Fig. 4.

The PDs in the first hidden layer are constructed by each possible pair of independent input features. If there are m input features, the number of PDs becomes m_{C_2}. These PDs are utilized for the construction of PDs in the second hidden layer. The PDs in the second hidden layer are constructed with a polynomial of degree 4. The following algorithm can discover the index of the input features for each PD.

Algorithm 1: PD generation

```
1.  Let layers = l.
2.  Let k = 1.
3.  For i = 1 to m-1
4.    For j = i+1 to m
5.      Then PD_k^l receives input
6.      from the features p = i and q = j;
7.        k  = k+1;
8.    End
9.  End
```

These networks come with a high level of flexibility, as each PD can have a different number of input variables and can exploit a different order of polynomial (linear, quadratic, cubic, etc.). Unlike neural networks whose topologies commonly are fixed prior to all detailed (parametric) learning, the CPNN architecture considered in this research is not fixed in advance. Instead, it becomes fully optimized both structurally and parametrically. The high-level procedure of the CPNN through which the weight and biases are optimized by the GA can be described as follows.

Algorithm 2: CPNN training

1. *DETERMINE the number of closing prices in the input vector (normalized input values).*
   a. *Calculate first layer PDs.*
   b. *Calculate second layer PDs.*
2. *RANDOMLY INITIALIZE the weights between hidden and output layer.*
3. *ESIMATE MAPE at the output neuron.*
   a. *Calculate the weighted sum along with bias values at the output neuron.*
   b. *Apply sigmoid activation and find estimated output.*
   c. *Supply desired output, compare with estimated and calculate error signal.*
4. *CHECK WHETHER the stopping criterion is met or not, if not then go to the step 5 otherwise stop.*
5. *UPDATE THE WEIGHTS by GA and go to step 3.*

GAs are well-liked for global search optimization tasks that involve a population of potential solutions in the form of chromosomes. A GA will attempt to locate the best solution through the process of artificial evolution. GAs are based on biological evolutionary theory, and they are used to solve optimization problems that work by encoding the parameters instead of using the parameters directly. The process consists of the following repeated artificial genetic operations: evaluation, selection, crossover, and mutation. The weights and other parameters are optimized by the GA, and then used to train the network. The fitness of the best and average individual in each generation increases towards a global optimum.

Employing this method, the proposed model first defines a network structure with a fixed number of inputs, and a single output as shown in Fig. 1. Second, the model employs the GA to find the optimal weight and bias vectors, as it is capable of searching a

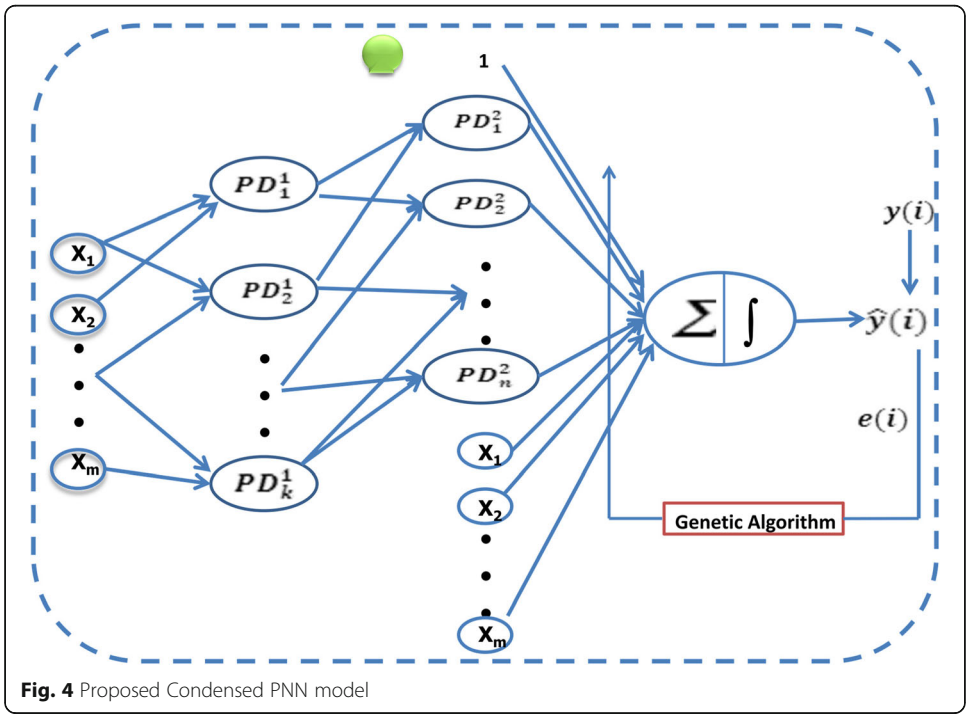

Fig. 4 Proposed Condensed PNN model

large search space. The hybrid of the neural network and GA can select the optimal weight sets as well as the bias value for prediction. The major steps of the GA-based CPNN model can be summarized as follows.

Algorithm 3: GA based CPNN forecasting

1. *Set the training data*
2. *Find all combination of feature values in the input vector*
3. *Map the input patterns*
   Map each pattern from the lower dimension to higher dimension, i.e. create PDs according to the polynomial basis functions of degree 2 (Layer 1)
   Create PDs according to polynomial basis functions of degree 4 (Layer 2)
4. *Random initialization of search spaces, i.e. populations.*
   Initialize each search space, i.e. chromosome with values from the domain [0, 1].
5. *While (termination criteria not met),*
   For each chromosome in the search space
   Calculate the weighted sum and feed as an input to the node of output layer
   Present the desired output, calculate the error signal and accumulate it
   Fitness of the chromosome is equal to the accumulated error signal
   End
   Apply crossover operator.
   Apply mutation operator.
   Select better fit solutions.
   End
6. *Present the test data*
   Calculated the weighted sum and calculate the error value.
7. *Repeat the steps 1-6 for all training and test patterns, calculate the total error signals*

Experimental results and analysis

This section explains the experimental portion of our work, including the preparation of input data, the simulated parameters for the two forecasting models, and the results from the models.

Preparation of input data

For this experiment, we considered the daily closing prices of a major stock index of each of the five fastest growing stock exchanges from January through December 2014. Table 1 provides further details of the data set under consideration. The historical data were collected from https://in.finance.yahoo.com/, an openly available source.

The sliding window technique was used to select the training and test patterns for the forecasting models. The daily closing prices of a stock were represented as a financial time series. A window of fixed size was moved over the series by one step each time. In each move, a new pattern was formed that could be used as an input vector.

Table 1 Stock Indices (daily closing price) considered for experiment

Index	Period	No. of data
BSE	2-Jan-2014 to 31-Dec-2014	248
DJIA	3-Jan-2014 to 31-Dec-2014	249
NASDAQ	3-Jan-2014 to 31-Dec-2014	250
FTSE	3-Jan-2014 to 31-Dec-2014	252
TAIEX	2-Jan-2014 to 28-Dec-2014	248

The size of a window can be decided experimentally. The number of closing prices included by the window was represented as the bed length (window size), and the number of times the window moved to generate a training set was treated as the training length. Each time the sliding window moved one step ahead, the data for one closing price was dropped from the beginning and the data for one new closing price was included at the end. Therefore, two consecutive training sets produced minimal change in the nonlinear behavior of the input-output mapping. For this experiment, a sliding window took only five values for the input layer, and only three patterns were presented to build a model. The training and test patterns generated for one-day-ahead forecasting using this sliding window technique are presented below. Here the bed length (window size) is represented as *blen,* and the training length is given as *l*. In general, the training data with window size = *blen* and training length *l* is:

$$\begin{array}{cccccc} \multicolumn{4}{c}{Training\ data} & & Training\ target \\ x(i) & x(i+1) & \cdots & x(i+blen) & \vdots & x(i+blen+1) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x(i+l) & x(i+l+1) & \cdots & x(i+l+blen) & \vdots & x(i+l+blen+1) \end{array}$$

The test data is shown below:

$$\begin{array}{cccccc} \multicolumn{4}{c}{Test\ \ data} & & Test\ \ t\ arget \\ x(i+l+1) & x(i+l+2) & \cdots & x(i+l+blen+1) & \vdots & x(i+l+blen+2) \end{array}$$

For preprocessing the raw daily closing prices, the prices must be normalized first because the neural models can process normalized values robustly for learning and generalization. Researchers have tried various data normalization techniques, and the sigmoid method was found most suitable (Nayak et al. 2014). For our work, the original closing prices were normalized using a sigmoid data normalization formula as given by Eq.6. Each window treated as a training set was normalized separately.

$$x = \frac{1}{1+e^{-\left(\frac{x_i - x_{min}}{x_{max} - x_{min}}\right)}} \tag{6}$$

Where 푥푛표푟푚 is the normalized price, 푥푖 is the current day closing price, and 푥푚푎푥 and 푥푚푖푛 are the maximum and minimum prices contained within the window, respectively. The record to be tested was also normalized using Eq. 6, but its value was not used for deriving the 푥푚푎푥 and 푥푚푖푛 values i.e., the target value could reside outside [푥푚푎푥, 푥푚푖푛]. Then the normalized data were used to form a training bed for the network model.

Performance metrics

For this research, we used four metrics for evaluation: the MAPE, POCID, ARV, and Theil's U. The mean absolute percentage error (MAPE) is a performance metric that allows comparative measurement of prediction accuracy across experiments using the data for different stocks. The formula for the MAPE is represented as shown in Eq.7.

$$MAPE = \frac{1}{N}\sum_{i=1}^{N}\frac{|x_i - \hat{x}_i|}{xi} \times 100\% \tag{7}$$

Here, $x_i$is the actual closing price, and $\hat{x}_i$is the estimated price (after de-normalization of closing prices).

The second metric used was the prediction of change in direction (POCID), which is particularly important for stock trend forecasting. The POCID may be considered more important than the MAPE, because if the direction of a stock trend can be predicted more accurately, investors may have better guidance that could lead to substantial monetary gain. The POCID can be represented as shown in Eq.8 and Eq.9.

$$POCID = \frac{\sum_{i=1}^{N} Trend_i}{N} * 100 \tag{8}$$

where

$$Trend_i = \begin{cases} 1, & if(x_i - x_{i-1}) \quad (\hat{x}_i - \hat{x}_{i-1}) \quad > 0 \\ 0, & otherwise \end{cases} \tag{9}$$

This measure gives an account of the number of correct directions when predicting the next closing prices in the financial time series. The ideal value of the POCID for a perfect predictor is 100, so the prediction model is shown to be more accurate as the value becomes closer to100.

The third evaluation measure used was the average relative variance (ARV). The ARV can be calculated as in Eq.10.

$$\frac{\sum_{i=1}^{N}(\hat{x}_i - x_i)^2}{\sum_{i=1}^{N}(\hat{x}_i - \overline{X})^2} \tag{10}$$

If the ARV value of the forecasting model is equal to 1, then the performance of the model is the same as calculating the mean of the financial time series. If the ARV value is greater than1, the model is considered to be performing worse than the mean. However, if the ARV value is less than 1, the model is considered to be performing better than simply calculating the mean. Hence, as the value becomes closer to 0, the forecasting model becomes more accurate.

The fourth measure considered in our evaluation of the models was Theil's U. This metric, which compares the performance of the model with a random walk model, can be calculated by using Eq.11.

$$U\ of\ Theil\ =\ \frac{\sum_{i=1}^{N}(x_i-\hat{x}_i)^2}{\sum_{i=1}^{N}(x_i-x_{i+1})^2} \tag{11}$$

If the value of the result is equal to 1, then the model in question provides the same performance as the random walk model. If the result is greater than 1, the model is considered as performing worse compared to a random walk model. The model is performing better than a random walk model if the Theil's U result is less than 1. Hence, a model's performance is considered better as the value comes closer to 0.

Experimental setting

Let the 푋푡ℎ input pattern vector to the model be given by 푋푝 = [(1),(2),...,푋푝(푝)]. Taking a combination of two inputs from the input pattern 푋푝, we get another expanded list. This list can be represented by 푋푝′ = [푋푝(1)*푋푝(2),...,푋푝(1) 푋푝(푝),.....,푋푝(푝 − 1)*푋푝(푝)]. Each element of the expanded vector 푋푝′ is applied to generate PDs for the next layer. The polynomial functions are applied as the basis function to generate PDs for the second layer and let the PDs are represented as $\bar{\bar{X}}_i^T$.

In this experiment the optimal number of input signals was chosen as five. Therefore, the number of PDs generated in the first layer was ten, and in the second layer was 45. The PDs in the first and second layer were of degree 2 and 4 respectively. As mentioned earlier, PDs were developed and limited to two layers of the PNN architecture. Given the input $\overline{X}_k^T$, the model produces an output $\hat{y}(k)$ that acts as an estimate of the desired value. The output of the linear part of the model is computed as shown:

$$y'^{(i)}\ =\ \overline{X}_i^T * W(i)\ +\ b \tag{12}$$

Where b represents the weighted bias input, and (푝) denotes the weight values for the 푝푡ℎ pattern. This output is then passed through a nonlinear function, in this case sigmoid activation, to produce the estimated output $\hat{y}(i)$:

$$\hat{y}(i) = \frac{1}{1+e^{-\lambda y' i}} \tag{13}$$

The error signal (푝) is calculated as the difference between the desired response and the estimated output of the model.

$$e(i) = |y(i)-\hat{y}(i)| \tag{14}$$

The error signal (푘) and the input vector are employed by the weight update algorithm to compute the optimal weight vector. To overcome the difficulties of back propagation, we employed the GA for global search optimization. The network has the ability to learn through training by the GA. During the training, the network is repeatedly presented with the training vector and the weights, and biases are adjusted by the GA until the desired input-output mapping occurs.

The error is calculated by Eq.14, and our objective is to minimize the error function as in Eq.15 with an optimal set of weight vectors.

$$E(i) = \sum_{i=1}^{N} e(i) \tag{15}$$

During the experiment, various possible values for the model parameters were tested, and the best values were recorded. The suitable parameter values obtained during the simulation process were called simulated parameters, and they are presented in Table 2.

We adopted binary encoding for the GA. Each weight and bias value consisted of 17 binary bits. To calculate the weighted sum at the output neuron, the decimal equivalent of the binary chromosome was used, with a randomly initialized population of 50 to 60 genotypes. The GA was run for a maximum 250 to 300 generations for different models. Parents were selected from the population by the elitism method, in which the first 10% of the mating pools were selected from the best parents and the rest were selected by a binary tournament selection method. New offspring were generated from these parents using uniform crossover followed by a mutation operator. In this experiment, the crossover probability varied between 0.5 and 0.6, and the mutation probability was taken as 0.02 to 0.05. In this way the new population that was generated replaced the current population, and the process continued until convergence occurred. The fitness of the best and average individuals in each generation increased toward a global optimum. The uniformity of the individuals increased gradually leading to convergence.

Result analysis

This sub section describes the results obtained from the forecasting models. The models considered were a gradient descent-based MLP (MLP-GD), GA-based MLP (MLP-GA), radial basis functional neural network (RBFNN), CPNN trained with gradient descent (CPNN-GD), and the proposed CPNN trained with a GA (CPNN-GA). The same training and test data sets were supplied as input signals to all the models considered. Each model was simulated 10 times for each training and test data set, and the average error was considered for comparative analysis. Table 3 presents the MAPE, POCID, Theil U, and ARV values generated from all the forecasting models considering all five financial time series.

Table 2 Simulation parameters for MLP-GA and CPNN-GA forecasting model

Parameters	Forecasting models	
	MLP-GA	CPNN-GA
Population size	60	50
Gene Size (bit)	17	17
Crossover probability (C_p)	0.6	0.5
Mutation probability (m_c)	0.02	0.05
Selection method	Elitism	Elitism
Max. no. of generation	300	250

Table 3 Performance of the forecasting models on five stock indices

Stock Index	Error Statistic	Forecasting Models				
		CPNN-GA	CPNN-GD	RBFNN	MLP-GA	MLP-DG
BSE	MAPE	**0.082532**	0.087403	0.151002	0.084635	0.761849
	ARV	**0.011522**	0.015524	0.017282	0.013280	0.127503
	POCID	92.55	88.98	83.25	**94.00**	82.86
	U of Theil	**0.072205**	0.090023	0.250044	0.082372	0.388572
DJIA	MAPE	**0.086603**	0.092601	0.550872	0.100535	0.523583
	ARV	**0.012955**	0.015524	0.041224	0.019282	0.075002
	POCID	**94.35**	89.34	85.74	89.23	81.75
	U of Theil	**0.049713**	0.052285	0.277258	0.058302	0.484922
NASDAQ	MAPE	**0.009263**	0.009975	0.013964	0.011765	0.056570
	ARV	**0.034508**	0.081764	0.087792	0.084112	0.272843
	POCID	**96.57**	92.00	88.15	91.33	82.45
	U of Theil	**0.076244**	**0.076245**	0.100572	0.079990	0.552932
FTSE	MAPE	**0.025923**	0.041221	0.096658	0.027925	0.521705
	ARV	**0.038155**	0.064845	0.500325	0.077359	0.532601
	POCID	**95.75**	94.47	82.35	88.47	80.92
	U of Theil	**0.080005**	0.082545	0.250660	0.105502	0.499725
TAIEX	MAPE	**0.042113**	0.054159	0.083155	0.054211	0.380937
	ARV	**0.025390**	0.062900	0.076225	0.210045	0.278608
	POCID	**95.55**	92.22	85.65	91.73	85.75
	U of Theil	**0.039577**	0.058222	0.448025	0.042275	0.472895

It is clear from Table 3 that the proposed CPNN-GA outperformed the other models for all five data sets. The best error statistic values are highlighted in bold face. For the BSE index, MLP-GA generated the best POCID value, i.e. 94.00%. For the DJIA, FTSE, and TAIEX, the CPNN-GA model performed better for all error statistics compared to the other models. Except for a few cases, the average forecasting performance of the CPNN-GA was found quite satisfactory.

To make the comparative study even more specific, the percentage gain in MAPE reduction was calculated as follows, as presented by Fig. 5.

$$MAPE\ gain\ = \frac{(MAPE\ of\ existing\ \ model\ -\ MAPE\ of\ proposed\ \ model)}{MAPE\ of\ existing\ \ model} \times\ 100\% \tag{16}$$

The average MAPE gain over CPNN-GD considering all data sets was 15.70844%, which demonstrates the contribution of the GA. The gain was 57.16491% over RBFNN. Similarly, the average MAPE gain over MLP-GA and MLP-GD was 13.41896% and 88.0456%, respectively. It can be observed easily that the proposed model provided substantially better performance compared to the other models. The error statistic values obtained from CPNN-GD and MLP-GA were found to be closer to that of CPNN-GA, which was not in so in

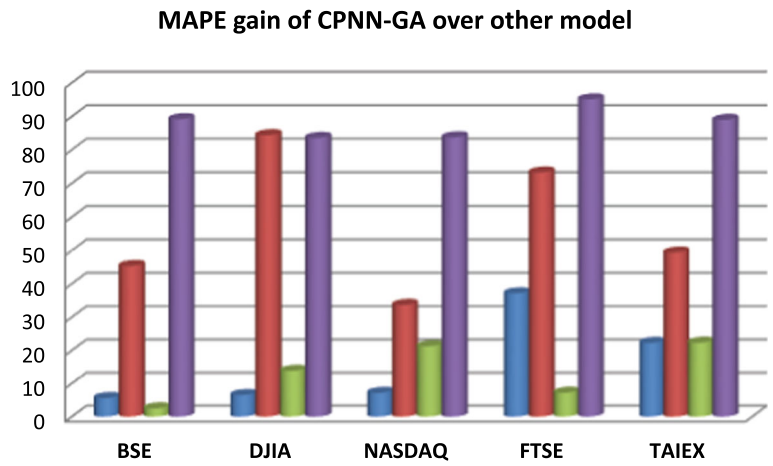

Fig. 5 MAPE gain of CPNN-GA over CPNN-GD, RBFNN, MLP-GA, and MLP-GD considering five stock indices.

the case of RBFNN and MLP-GD. The prediction accuracies of the models were enhanced when GA was adopted to search the optimal model parameters.

For a clearer view of CPNN-GA's performance, the actual prices vs. estimated closing prices are plotted and presented by Figs. 6, 7, 8, 9, 10.

Further comparison of the performance of the models was provided by recording their computation times. The experiments were carried out on a computer system equipped with an Intel ®core™ i3 CPU, 2.27 GHz, with 2.42 GB memory, using MATLAB-2009. The computation times (in seconds) are summarized in Table 4. Comparing the computation times, we can observe that the proposed CPNN-GA forecasting model required the least amount of time, averaging a computation time

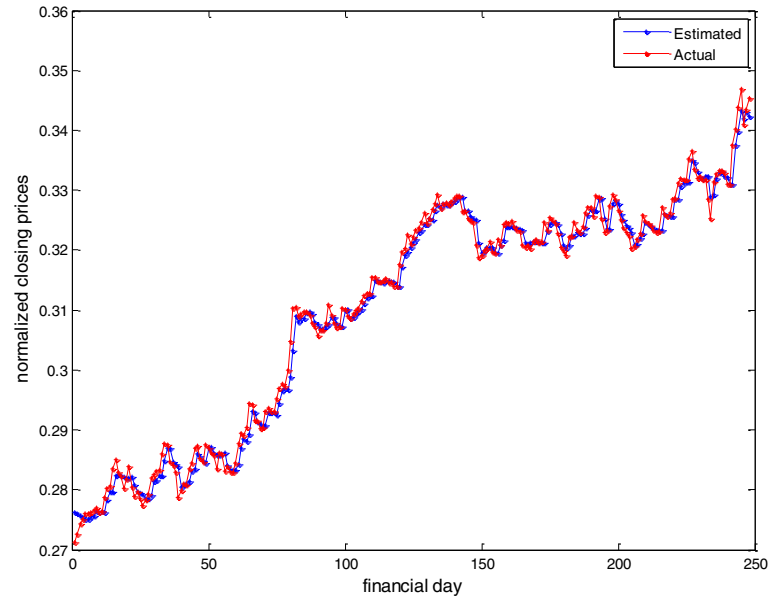

Fig. 6 Actual v/s estimated closing prices by CPNN-GA for BSE indices

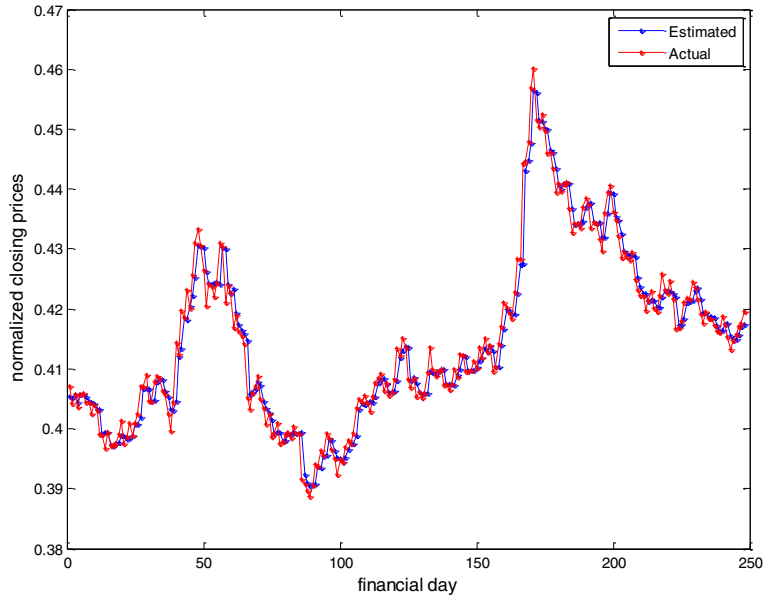

Fig. 7 Actual v/s estimated closing prices by CPNN-GA for DJIA indices

of only 51.692 s for five data sets. Clearly, the GA provided faster convergence than the GD technique.

To discover the exact benefits of the proposed model, we used the Deibold-Mariano (DM) test to determine the statistical significance. The DM test (Diebold and Mariano 1995) is a pair-wise comparison of two or more time series models for forecasting a particular variable of interest. Let the actual time series is $\{y_t;\ t = 1,\ \cdots,\ T\}$ and the two forecasts are $\{\hat{y}_{1t}; t = 1, \cdots, T\}$ and $\{\hat{y}_{2t}; t = 1, \cdots, T\}$.

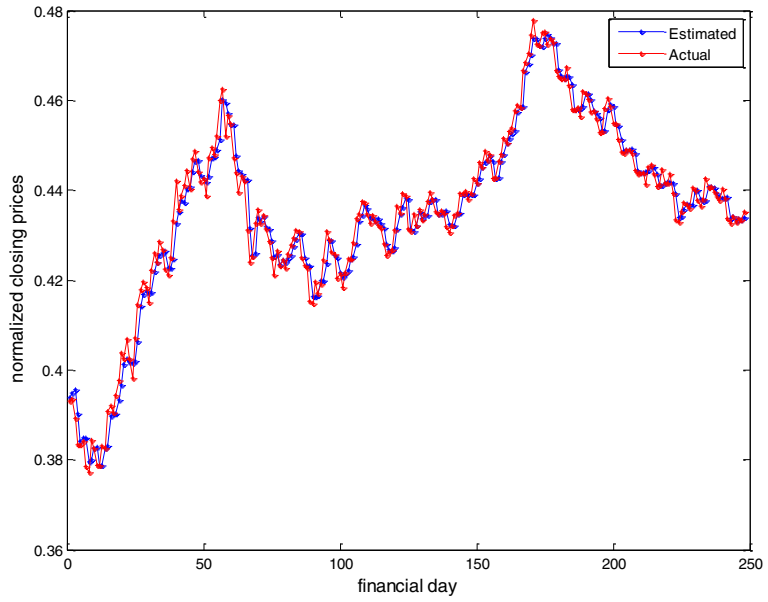

Fig. 8 Actual v/s estimated closing prices by CPNN-GA for NASDAQ indices

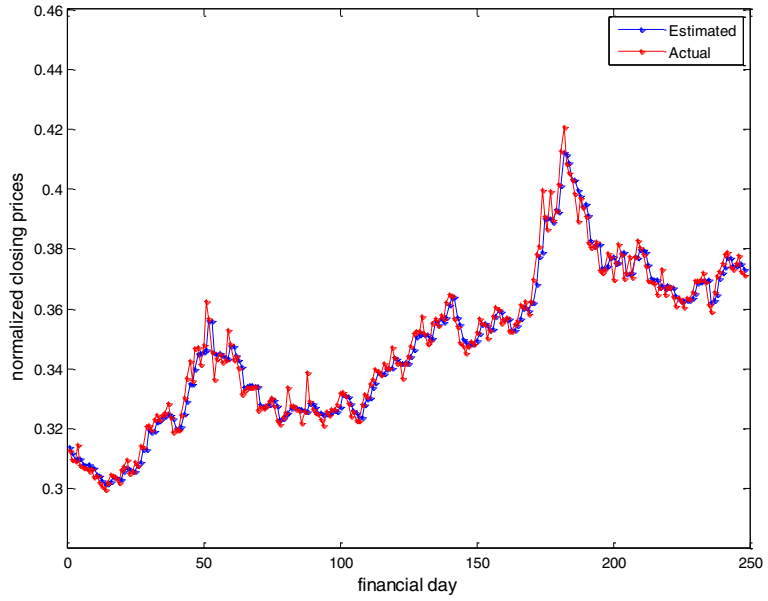

Fig. 9 Actual v/s estimated closing prices by CPNN-GA for FTSE indices

The objective was to test whether the forecasts were equally good or not. Let the forecast errors be defined as $e_{it} = \hat{y}_{it} - y_t, i = 1, 2$. Let the loss function associated with the forecast be defined as $g(e_{it}) = |e^2{}_{it}|$, and let the loss differential between the two forecasts be $d_t = g(e_{1t}) - g(e_{2t})$. The null hypothesis and the alternative are defined as follows:

$H_0 : E(d_t) = 0 \,\forall\, t$, indicating that the two forecasts have the same accuracy

$H_{alt} : E(d_t) \neq 0$, indicating that the two forecasts have different levels of accuracy.

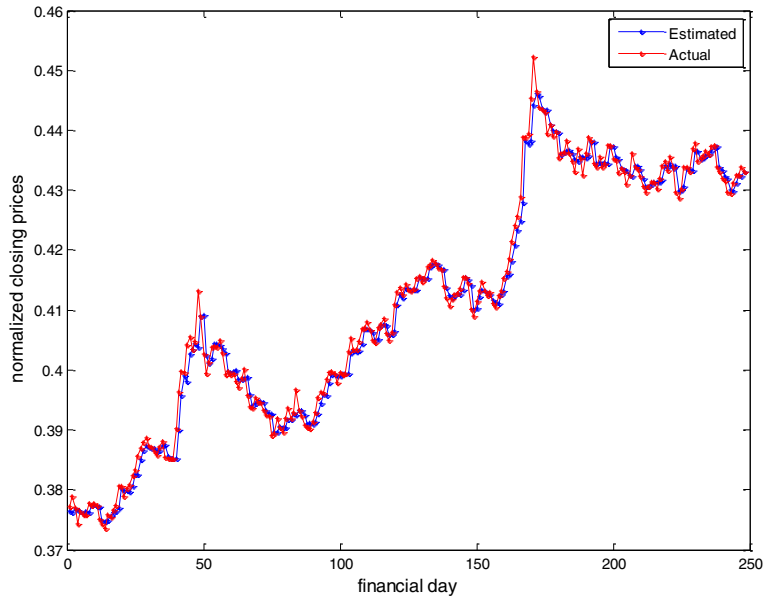

Fig. 10 Actual v/s estimated closing prices by CPNN-GA for TAIEX indices

Table 4 Computation time from all models

Model	Stock Index					Average
	BSE	DJIA	NASDAQ	FTSE	TAIEX	
MLP-GA	65.55	60.45	60.65	57.62	55.95	60.044
CPNN-GD	65.79	64.69	66.25	60.31	62.43	63.894
CPNN-GA	51.33	51.35	51.64	52.57	51.57	51.692
MLP-GD	96.37	92.14	90.65	91.66	92.26	92.616
RBFNN	90.53	82.35	75.66	72.35	87.39	81.656

The DM-statistic defined as:

$$DM = \frac{\overline{d}}{\sqrt{\frac{\hat{\gamma}_d(0)+2\sum_{k=1}^{h-1}\hat{\gamma}_d(k)}{T}}} \tag{17}$$

where, $\overline{d}$ is the sample mean of the loss differential, h is the forecast horizon, and $\hat{\gamma}_d(k)$ is an approximation of the auto-covariance of the loss differential $\gamma_d(k)$ at lag k. The null hypothesis of no difference is rejected if the DM statistic value falls outside the range of -픕$_{훼/2}$ to 픕$_{훼/2}$, i.e. |팻피| > 픕$_{훼/2}$, where 픕$_{훼/2}$ is the upper z-value from the standard normal table corresponding to half of the desired 훼 level of the test. Consider the significance level of the test is 훼 = 0.05. Since this is a two-tailed test, the lower critical z-value corresponding to – 0.025 is – 1.96, and the upper critical z-value corresponding to 0.975 is + 1.96. The computed DM statistic values obtained are summarized in Table 5. The results show that the DM statistics obtained always were outside of the critical range. Hence, the null hypothesis of no difference between the CPNN-GA and other model was rejected.

Conclusion

To provide improved prediction of the closing prices of stock market indices, this paper proposed a novel GA-weighted condensed polynomial neural network (CPNN-GA) model. This model generates PDs for the first and second layers of degree two and four, respectively. A generic algorithm is utilized to select the optimal synaptic weight set and biases of the model. These weight and bias values, along with the input features, are fed to the output neuron. The prediction performance of the proposed model was compared experimentally to the

Table 5 Computed DM statistic values from all models and stock indices

Index	C	CPNN-GD	RBFNN	MLP-GD	MLP-GA
BSE	P	2.2140	1.9809	−2.0245	2.7353
DJIA	N	1.9821	2.4503	3.3005	−4.1782
NASDAQ	N	2.5344	−2.7863	2.4577	2.0016
FTSE	–	−2.5662	3.2655	2.5575	2.0516
TAIEX	G	−2.2427	2.4373	−3.1565	2.0096
	A				

performance of a CPNN-GD, MLP-GD, MLP-GA, and RBFNN model, all of which have been employed for forecasting next days' closing prices of a real stock market. Data for five major fast-growing stock market indices were considered for this work. The experimental results and statistical significance tests proved the superiority of the proposed model over the others, demonstrating that the proposed CPNN-GA model can be considered an efficient and promising forecasting model for the stock market.

Future work may include exploration of the applicability of the proposed model in other domains. Also, other meta-heuristics may be employed to search the optimal parameters for the model.

Abbreviations

ANN: Artificial Neural Network; ARCH: Autoregressive Conditional Heteroscedastic; ARIMA: Autoregressive Integrated Moving Average; ARMA: Autoregressive Moving Average; ARV: Average Relative Variance; BPNN: Back Propagation Neural Network; BSE: Bombay Stock Exchange; CPNN: Based CPNN; CPNN: Condensed Polynomial Neural Network; CPNN-GD: Gradient Descent Based CPNN; GA: Genetic Algorithm; GARCH: Generalized ARCH; GMDH: Group Method of Data Handling; GMM: Generalized Methods of Moments; MA: Moving Average; MAPE: Mean Absolute Percentage of Error; MLP: Multilayer Perceptron; MLP-GA: Genetic Algorithm Based MLP; MLP-GD: Gradient Descent Based MLP; PD: Partial Description; PNN: Polynomial Neural Network; POCID: Prediction Of Change in Direction; PSO: Particle Swarm Optimization; RBFNN: Radial Basis Functional Neural Network; UT: U of Thiel

Acknowledgements

The author would like to thank to the Editor and the reviewers for their valuable comments and constructive suggestions that helped to improve the content of the paper in a large extent.
The first author would like to thank to the management, CMR College of Engineering & Technology, Hyderabad, India, for their continuous encouragement and support.

Availability of data and materials

The historical closing prices for experimentation were collected from the source https://in.finance.yahoo.com/

Authors' contributions

SCN (first author) designed the forecasting model, analyzed and interpreted data, conducted experiments, discussed the results and wrote the article. BBM (second author) conducted the literature study, explored the research area and was a major contributor in writing the manuscript. Both authors read and approved the final manuscript.

Competing interests

The authors declare that they have no competing interests.

Publisher's Note

Author details

[1]Department of Computer Science and Engineering, CMR College of Engineering &Technology (Autonomous), 501401, Hyderabad, India. [2]Department of Information Technology, Silicon Institute of Technology, 751024, Bhubaneswar, India.

Received: 29 December 2017 Accepted: 23 August 2018
Published online: 06 September 2018

References

Abdoh TH, Jouhare H (1996) The investigation of efficiency of stock price index of T.S.E. J Financ Res 13:11–12
Abdullah SM, Siddiqua S, Siddiquee MSH, Hossain N (2017) Modeling and forecasting exchange rate volatility in Bangladesh using GARCH models: a comparison based on normal and Student's t-error distribution. Financ Innov 3:18. https://doi.org/10.1186/s40854-017-0071-z
Bollerslev T (1986) Generalized autoregressive conditional heteroskedasticicty. J Econ 52:307–327
Box GEP, Jenkins GM (1976) Time series analysis-forecasting and control. Holden-Day Inc., San Francisco
Calderon T, Cheh J (2002) A roadmap for future neural networks research in auditing and risk assessment. Int J Account Inf Syst 3:203–236
Campbell JY, Lo AW, MacKinlay AC (1997) The econometrics of financial markets. Princeton University Press, Princeton
Cao Q, Leggio K, Schniederjans M (2005) A comparison between Fama and French's model and artificial networks in predicting the Chinese stock market. Comput Oper Res 32:2499–2512
Chakravarty S, Dash PK (2012) A PSO based integrated functional link net and interval type-2 fuzzy logic system for predicting stock market indices. Appl Soft Comput 12:931–941
Chen A, Leung M, Daouk H (2003) Application of neural networks to an emerging financial market: forecasting and trading the Taiwan stock index. Comput Oper Res 30:901–923
Diebold FX, Mariano RS (1995) Comparing predictive accuracy. J Bus Econ Stat 13:253–263

Farlow SJ (1984) Self-organizing method in modelling: GMDH-type algorithm. Marcel Dekker, New York
Guangxu Z (2005) RBF.Based time-series forecasting. J Comput Appl 9:2179–2183
Harvey C, Travers K, Costa M (2000) Forecasting emerging market returns using neural networks. Emerg Mark Q 4:43–55
Huang C-J, Yang D-X, Chuang Y-T (2008) Application of wrapper approach and composite classifier to the stock trend prediction. Expert Syst Appl 34:2870–2878
Huang Y, Kou G (2014) A kernel entropy manifold learning approach for financial data analysis. Decis Support Syst 64:31–42
Huang Y, Kou G, Peng Y (2017) Nonlinear manifold learning for early warnings in financial markets. Eur J Oper Res 258(2):692–702
Ivahnenko AG (1971) Polynomial theory of complex systems. IEEE Trans Syst. Man Cybernet SMC-1:364–378
Ketabchi S, Ghanadzadeh H, Ghanadzadeh A, Fallahi S, Ganji M (2010) Estimation of VLE of binary systems (tert-butanol + 2-ethyl-1-hexanol) and (n-butanol + 2-ethyl-1-hexanol) using GMDH-type neural network. J Chem Thermodynamics 42:1352–1355
Khashei M, Hajirahimi Z (2017) Performance evaluation of series and parallel strategies for financial time series forecasting. Financ Innov 3:24. https://doi.org/10.1186/s40854-017-0074-9
Kimoto T, Asakawa K, Yoda M, Takeoka M (1990) Stock market prediction system with modular neural network. Proceedings of the international joint conference on neural networks, San Diego, pp 1–6
Kou G, Peng Y, Wang G (2014) Evaluation of clustering algorithms for financial risk analysis using MCDM methods. Inf Sci 275:1–12
Kumar K, Bhattacharya S (2006) Artificial neural network vs. linear discriminant analysis in credit ratings forecast. Rev Account Finance 5:216–227
Kwon YK, Moon BR (2007) A hybrid neuro genetic approach for stock forecasting. IEEE Trans Neural Netw 18(3):851–864
Lahmiri S (2016) Interest rate next-day variation prediction based on hybrid feedforward neural network, particle swarm optimization, and multiresolution techniques. Physica A: Statistical Mechanics and its Applications 444:388–396
Lahmiri S (2018a) A technical analysis information fusion approach for stock price analysis and modeling. Fluctuation and Noise Letters 17(01):1850007
Lahmiri S (2018b) Minute-ahead stock price forecasting based on singular spectrum analysis and support vector regression. Appl Math Comput 320:444–451
Lahmiri S, Boukadoum M (2015) Intelligent ensemble forecasting system of stock market fluctuations based on symetric and asymetric wavelet functions. Fluctuation and Noise Letters 14(04):1550033
Leigh W, Hightower R, Modani N (2005) Forecasting the New York stock exchange composite index with past price and interest rate on condition of volume spike. Expert Syst Appl 28:1–8
Liu H-C, Lee Y-H, Lee M-C (2009) Forecasting China stock markets volatility via GARCH models under skewed-GED distribution. J Money Investment Bank 7:5–14
McGrath C (2002) Terminator portfolio. Kiplinger's Personal Finance, vol 56, pp 56–57
Misra BB, Satapathy SC, Biswal BN, Dash PK, Panda G (2006a) Pattern classification using polynomial neural networks. In: IEEE Internat. Conf. On cybernetics and intelligent systems (CIS)
Misra BB, Satapathy SC, Hanoon N, Dash PK (2006b) Particle swarm optimized polynomials for data classification. In: Proc. IEEE Internat. Conference on Intelligent Systems Design and Application
Narendra Babu C, Eswara Reddy B (2015) Prediction of selected Indian stock using a partitioning-interpolation based ARIMA-GARCH model. Appl Comput Inf 11:130–143
Nayak, S. C. Misra, B. B. and Behera, H. S. (2012) "Index prediction using neuro-genetic hybrid networks: a comparative analysis of perfermance," International conference on computing communication and application, IEEE Xplore, doi: 10.1109 / ICCCA.2012.6179215
Nayak SC, Misra BB, Behera HS (2016a) Adaptive Hybrid Higher Order Neural Networks for Prediction of Stock Market Behavior, Applied artificial higher order neural networks for control and recognition. IGI Global, USA pp 174–191
Nayak SC, Misra BB, Behera HS (2016b) An Adaptive Second Order Neural Network with Genetic-Algorithm-based Training (ASONN-GA) to Forecast the Closing Prices of the Stock Market. J Appl Metaheuristic Comput (IJAMC) 7(2):39–57 IGI Global
Nayak SC, Misra BB, Behera HS (2017) Efficient financial time series prediction with evolutionary virtual data position exploration. Neural Comput & Applic 1–22
Nayak SC, Misra BB, Behera HS (2018) On Developing and Performance Evaluation of Adaptive Second Order Neural Network With GA-Based Training (ASONN-GA) for Financial Time Series Prediction, Advancements in applied metaheuristic computing. IGI global, pp 231–263
Nayak SC, Misra BB, Behera HS (2014) Impact of data normalization on stock index forecasting. Int J Comp Inf Syst Ind Manage 6:357–369
Oh KJ, Kim K-J (2002) Analyzing stock market tick data using piecewise non linear model. Expert Syst Appl 22:249–255
Pradeepkumar D, Ravi V (2017) Forecasting financial time series volatility using particle swarm optimization trained quantile regression neural network. Appl Soft Comput 58:35–52
Ravichandran KS, Thirunavukarasu P, Nallaswamy R, Babu R (2007) Estimation on return on investment in share market through ANN. J Theor Appl Inf Technol 3:44–54
Rout M, Majhi B, Majhi R, Panda G (2013) Forecasting of currency exchange rates using an adaptive ARMA model with differential evolution based training. J King Saud University Comput Inf Sci 26:7–18 Elsevier
Shaverdi M, Fallahi S, Bashiri V (2012) Prediction of stock price of Iranian petrochemical industry using GMDH-type neural network and genetic algorithm. Appl Math Sci 6(7):319–332
Swicegood P, Clark J (2001) Off-site monitoring systems for prediction bank underperformance: a comparison of neural networks, discriminant analysis, and professional human judgment. Int J Intell Syst Account Finance Manage 10:169–186
Trippi RR, DeSieno D (1992) Trading equity index futures with a neural network. J Portfolio Manangment 19:27–33
Wang Y (2003) Mining stock prices using fuzzy rough set system. Expert Syst Appl 24:13–23
Xie H, Wang S (2015) Risk-return trade-off, information diffusion, and US stock market predictability. Int J Financ Eng 2(04):1550038
Yilmaz A, Unal G (2016) Co-movement analysis of Asian stock markets against FTSE100 and S&P 500: wavelet-based approach. Int J Financ Eng 3(04):1650033
Yu L, Zhang YQ (2005) Evolutionary fuzzy neural networks for hybrid financial prediction. IEEE Trans Syst Man Cybern C Appl Rev 35(2):244–49