ARTICLE IN PRESS
Journal of King Saud University – Computer and Information Sciences (2015) **xxx**, xxx–xxx

King Saud University
Journal of King Saud University – Computer and Information Sciences
www.ksu.edu.sa
www.sciencedirect.com

Forecasting financial time series using a low complexity recurrent neural network and evolutionary learning approach

Ajit Kumar Rout [a], **P.K. Dash** [b,*], **Rajashree Dash** [b], **Ranjeeta Bisoi** [b]

[a] *G.M.R. Institute of Technology, Rajam, Andhra Pradesh, India*
[b] *S.O.A. University, Bhubaneswar, India*

Received 14 March 2015; revised 24 May 2015; accepted 3 June 2015

KEYWORDS

Low complexity FLANN models;
Recurrent computationally efficient FLANN;
Differential Evolution;
Hybrid Moderate Random Search PSO

Abstract The paper presents a low complexity recurrent Functional Link Artificial Neural Network for predicting the financial time series data like the stock market indices over a time frame varying from 1 day ahead to 1 month ahead. Although different types of basis functions have been used for low complexity neural networks earlier for stock market prediction, a comparative study is needed to choose the optimal combinations of these for a reasonably accurate forecast. Further several evolutionary learning methods like the Particle Swarm Optimization (PSO) and modified version of its new variant (HMRPSO), and the Differential Evolution (DE) are adopted here to find the optimal weights for the recurrent computationally efficient functional link neural network (RCEFLANN) using a combination of linear and hyperbolic tangent basis functions. The performance of the recurrent computationally efficient FLANN model is compared with that of low complexity neural networks using the Trigonometric, Chebyshev, Laguerre, Legendre, and tangent hyperbolic basis functions in predicting stock prices of Bombay Stock Exchange data and Standard & Poor's 500 data sets using different evolutionary methods and has been presented in this paper and the results clearly reveal that the recurrent FLANN model trained with the DE outperforms all other FLANN models similarly trained.

1. Introduction

Financial time series data are more complicated than other statistical data due to the long term trends, cyclical variations, seasonal variations and irregular movements. Predicting such highly fluctuating and irregular data is usually subject to large errors. So developing more realistic models for predicting financial time series data to extract meaningful statistics from it, more effectively and accurately is a great interest of research

* Corresponding author.
E-mail address: pkdash.india@gmail.com (P.K. Dash).
Peer review under responsibility of King Saud University.

Production and hosting by Elsevier

http://dx.doi.org/10.1016/j.jksuci.2015.06.002

in financial data mining. The traditional statistical models used for financial forecasting were simple, but suffered from several shortcomings due to the nonlinearity of data. Hence researchers have developed more efficient and accurate soft computing methods like Artificial Neural Network (ANN); Fuzzy Information Systems (FIS), Support Vector Machine (SVM), Rough Set theory etc. for financial forecasting. Various ANN based methods like Multi Layer Perception (MLP) Network, Radial Basis Function Neural Network (RBFNN), Wavelet Neural Network (WNN), Local Linear Wavelet Neural Network (LLWNN), Recurrent Neural Network (RNN) and Functional Link Artificial Neural Network (FLANN) are extensively used for stock market prediction due to their inherent capabilities to identify complex nonlinear relationship present in the time series data based on historical data and to approximate any nonlinear function to a high degree of accuracy. The use of ANN to predict the behavior and tendencies of stocks has demonstrated itself to be a viable alternative to existing conventional techniques (Andrade de Oliveira and Nobre, 2011; Naeini et al., 2010; Song et al., 2007; Lee and Chen, 2007; Ma et al., 2010).

A system of time series data analysis has been proposed in Kozarzewski (2010) for predicting the future values, based on wavelets preprocessing and neural networks clustering that has been tested as a tool for supporting stock market investment decisions and shows good prediction accuracy of the method. MLP neural networks are mostly used by the researchers for its inherent capabilities to approximate any non-linear function to a high degree of accuracy (Lin and Feng, 2010; Tahersima et al., 2011). But these models suffer from slow convergence, local minimum, over fitting, have high computational cost and need large number of iterations for its training due to the availability of hidden layer. To overcome these limitations, a different kind of ANN i.e. Functional Link ANN (Proposed by Pao (1989)) having a single layer architecture with no hidden layers has been developed. The mathematical expression and computational calculation of a FLANN structure is evaluated as per MLP. But it possesses a higher rate of convergence and lesser computational load than those of a MLP structure (Majhi et al., 2005; Chakravarty and Dash, 2009). A wide variety of FLANNs with functional expansion using orthogonal trigonometric functions (Dehuri et al., 2012; Mili and Hamdi, 2012; Patra et al., 2009), using Chebyshev polynomial (Mishra et al., 2009; Jiang et al., 2012; Li et al., 2012), using Laguerre polynomial (Chandra et al., 2009) and using Legendre orthogonal polynomial (Nanda et al., 2011; George and Panda, 2012; Rodriguez, 2009; Das and Satapathy, 2011; Patra and Bornand, 2010) has been discussed in the literature. The well known Back Propagation algorithm is commonly used to update the weights of FLANN. In Yogi et al. (2010), a novel method using PSO for training trigonometric FLANN has been discussed for equalization of digital communication channels.

In this paper, the detailed architecture and mathematical modeling of various polynomial and trigonometric FLANNs have been described along with a new computationally efficient and robust FLANN, and its recurrent version. It is well known that the recurrent neural networks (RNNs) usually provide a smaller architecture than most of the nonrecursive neural networks like MLP, RBFNN, etc. Also their feedback properties make them dynamic and more efficient to model nonlinear systems accurately which are imperative for nonlinear prediction and time series forecasting. Many of the Autoregressive Moving Average (ARMA) processes have been accurately modeled by RNNs for nonlinear dynamic system identification. One of the familiar approaches of training the RNNs is the Real-Time Recurrent Learning (RTRL) (Ampolucci et al., 1999), which has problems of stability and slow convergence. In nonlinear time series forecasting problems it gets trapped in local minima and cannot guarantee to find global minima. On the other hand, evolutionary learning techniques such as Differential Evolution, particle swarm optimization, genetic algorithm, bacteria foraging, etc. have been applied to time series forecasting successively. DE is found to be efficient among them and outperforms other evolutionary algorithms since it is simpler to apply and involves less computation with less function parameters to be optimized as compared to other algorithms. DE is chosen because it is a simple but powerful global optimization method and converges faster than PSO. A comparative study between Differential Evolution (DE) and Particle Swarm Optimization (PSO) in the training and testing of feed-forward neural network for the prediction of daily stock market prices has shown that DE provides a faster convergence speed and better accuracy than PSO algorithm in the prediction of fluctuated time series (Abdual-Salam et al., 2010). Differential Evolution based FLANN has also shown its superiority over Back Propagation based Trigonometric FLANN in Indian Stock Market prediction (Hatem and Mustafa, 2012). The convergence speed is also faster to find a best global solution by escaping from local minima even for multiple optimal solutions.

Thus, in this paper various evolutionary learning methods like PSO, HMRPSO, DE for improving the performance of different types of FLANN models have been discussed. Comparing the performance of various FLANN models for predicting stock prices of Bombay Stock Exchange data and Standard & Poor's 500 data set, it has been tried to find out the best FLANN among them. The rest of the paper is organized as follows. In Sections 2 and 3, the detailed architecture of various FLANNS and various evolutionary learning algorithms for training has been described. The simulation study for demonstrating the prediction performance of different FLANNS has been carried out in Section 4. This section also provides a comparative result of training and testing of different FLANNS using PSO, HMRPSO, and DE (Mohapatra et al., 2012; Qin et al., 2008; Wang et al., 2011) based learning for predicting financial time series data. Finally conclusions are drawn in Section 5.

2. Architecture of low complexity neural network models

The FLANN originally proposed by Pao in 1992 is a single layer single neuron architecture, having two components: Functional expansion component and Learning component. The functional block helps to introduce nonlinearity by expanding the input space to a higher dimensional space through a basis function without using any hidden layers like MLP structure. The mathematical expression and computational calculation of a FLANN structure is same as MLP. But it possesses a higher rate of convergence and lesser computational cost than those of a MLP structure. A wider application of FLANN models for solving non linear problems like channel equalization, non linear dynamic system identification, electric load forecasting, prediction of earthquake, and financial forecasting has demonstrated its viability, robustness and ease of computation. The functional expansion block comprises either a trigonometric block or a polynomial block. Further the polynomial block can be expressed in terms of Chebyshev, Laguerre, or Legendre basis functions. Trigonometric FLANN (TRFLANN) is a single layer neural network in which the original input pattern in a lower dimensional space is expanded to a higher dimensional space using orthogonal trigonometric functions (Chakravarty and Dash, 2009; Dehuri et al., 2012; Mili and Hamdi, 2012). With the order m any n dimensional input pattern $X = [x_1,\ x_2 \ldots x_n]^T$ is expanded to a p dimensional pattern TX by Trigonometric functional expansion as $TX = [1,\ TF_1(x_1),\ TF_2(x_1) \ldots TF_m(x_1),\ TF_1(x_2),\ TF_2(x_2) \ldots TF_m(x_2) \ldots TF_1(x_n),\ TF_2(x_n) \ldots TF_m(x_n)]^T$ where $p = m * n + 1$. Each x_i in input pattern is expanded using trigonometric functions with order m as $\{1 \sin(\pi * x_i) \cos(\pi * x_i) \sin(2\pi * x_i) \cos(2\pi * x_i) \ldots \sin(m\pi * x_i) \cos(m\pi * x_i)\}$.

Trigonometric function:

$$TF_0(x) = 1, TF_1 = \cos \pi x, TF_2 = \sin \pi x, TF_3 = \cos 2\pi x,$$
$$TF_4 = \sin 2\pi x, TF_5 = \cos 3\pi x, TF_6 = \sin 3\pi x \tag{1}$$

Chebyshev polynomial:

The recursive formula to generate higher order Chebyshev polynomials is given by

$$\begin{aligned} &Ch_{p+1}(x) = 2xCh_p(x) - Ch_{p-1}x, \\ &Ch_0(x) = 1, Ch_1(x) = x, Ch_2 = 2x^2 - 1, Ch_3 = 4x^3 - 3x, \\ &Ch_4(x) = 8x^4 - 8x^2 + 1, Ch_5(x) = 16x^5 - 20x^3 + 5x, \\ &Ch_6 = 32x^6 - 48x^4 + 18x^2 - 1. \end{aligned} \tag{2}$$

Laguerre polynomials:

The recursive formula to generate higher order Laguerre polynomials is given by

$$\begin{aligned} &L_{p+1}(x) = \tfrac{1}{p+1}[(2p+1)L_p(x) - pL_{p-1}(x)] \\ &La_0(x) = 1, La_1(x) = 1 - x, La_2 = 0.5x^2 - 2x + 1, \\ &La_3 = -x^3/6 + 3x^2/2 - 3x + 1, \\ &La_4(x) = x^4/24 - 2x^3/3 + 3x^2 - 4x + 1, \\ &La_5(x) = -x^5/120 + 5x^4/24 - 5x^3/3 + 5x^2 - 5x + 1 \end{aligned} \tag{3}$$

Legendre polynomials:

The Legendre polynomials are denoted by $Le_p(X)$, where p is the order and $-1 < x < 1$ is the argument of the polynomial. It constitutes a set of orthogonal polynomials as solutions to the differential equation:

$$\frac{d}{dx}\left[(1 - x^2)\frac{dy}{dx}\right] + p(p+1)y = 0$$

The zeroth and the first order Legendre polynomials are respectively given by, $Le_0(x) = 1$ and $Le_1(x) = x$. The higher order polynomials are

$$\begin{aligned} &Le_2(x) = 3x^2/2 - 1/2, Le_3(x) = 5x^3/2 - 3x/2, \\ &Le_4 = 35x^4/8 - 15x^2/4 + 3/8 \\ &Le_5(x) = 63x^5/8 - 35x^3/4 + 15x/8, \\ &Le_6 = 231x^6/16 - 315x^4/16 + 105x^2/16 - 5/16 \end{aligned} \tag{4}$$

The predicted sample $x_{(t+k)}$ can be represented as a weighted sum of nonlinear polynomial arrays, $P_i(x_j)$. The inherent nonlinearities in the polynomials attempt to accommodate the nonlinear causal relation of the future sample with the samples prior to it.

Using trigonometric expansion blocks and a sample index k, the following relations are obtained:

$$u(k)^{Trigono} = \delta_{(0)}(k) + \begin{bmatrix} \delta_{(1,1)}(k) \\ \delta_{(2,1)}(k) \\ \vdots \\ \vdots \\ \delta_{(m-1,1)}(k) \\ \delta_{(m,1)}(k) \end{bmatrix}^T \begin{bmatrix} TF_1(x_1(k)) \\ TF_2(x_1(k)) \\ \vdots \\ \vdots \\ TF_{m-1}(x_1(k)) \\ TF_m(x_1(k)) \end{bmatrix} + \ldots\ldots$$
$$+ \begin{bmatrix} \delta_{(1,n)}(k) \\ \delta_{(2,n)}(k) \\ \vdots \\ \vdots \\ \delta_{(m-1,n)}(k) \\ \delta_{(m,n)}(k) \end{bmatrix}^T \begin{bmatrix} TF_1(x_n(k)) \\ TF_2(x_n(k)) \\ \vdots \\ \vdots \\ TF_{m-1}(x_n(k)) \\ TF_m(x_n(k)) \end{bmatrix} \tag{5}$$

and this FLANN is named as TRFLANN.

Using Chebyshev polynomials, the CHFLANN output is obtained as

$$u(k)^{Chebyshev} = \alpha_{(0)}(k) + \begin{bmatrix} \alpha_{(1,1)}(k) \\ \alpha_{(2,1)}(k) \\ \vdots \\ \vdots \\ \alpha_{(m-1,1)}(k) \\ \alpha_{(m,1)}(k) \end{bmatrix}^T \begin{bmatrix} C_1(x_1(k)) \\ C_2(x_1(k)) \\ \vdots \\ \vdots \\ C_{m-1}(x_1(k)) \\ C_m(x_1(k)) \end{bmatrix} + \ldots\ldots$$
$$+ \begin{bmatrix} \alpha_{(1,n)}(k) \\ \alpha_{(2,n)(k)} \\ \vdots \\ \vdots \\ \alpha_{(m-1,n)}(k) \\ \alpha_{(m,n)}(k) \end{bmatrix}^T \begin{bmatrix} C_1(x_n(k)) \\ C_2(x_n(k)) \\ \vdots \\ \vdots \\ C_{m-1}(x_n(k)) \\ C_m(x_n(k)) \end{bmatrix} \tag{6}$$

Using Laguerre polynomials, the LAGFLANN output is found as

$$u(k)^{Laguerre} = \beta_{(0)}(k) + \begin{bmatrix} \beta_{(1,1)}(k) \\ \beta_{(2,1)}(k) \\ \vdots \\ \vdots \\ \beta_{(m-1,1)}(k) \\ \beta_{(m,1)}(k) \end{bmatrix}^T \begin{bmatrix} La_1(x_1(k)) \\ La_2(x_1(k)) \\ \vdots \\ \vdots \\ La_{m-1}(x_1(k)) \\ La_m(x_1(k)) \end{bmatrix} + \ldots\ldots$$
$$+ \begin{bmatrix} \beta_{(1,n)}(k) \\ \beta_{(2,n)}(k) \\ \vdots \\ \vdots \\ \beta_{(m-1,n)}(k) \\ \beta_{(m,n)}(k) \end{bmatrix}^T \begin{bmatrix} La_1(x_n(k)) \\ La_2(x_n(k)) \\ \vdots \\ \vdots \\ La_{m-1}(x_n(k)) \\ La_m(x_n(k)) \end{bmatrix} \tag{7}$$

Using Legendre polynomials, Eq. (8) gives the LEG-FLANN output

$$u(k)^{Legendre} = \gamma_{(0)}(k) + \begin{bmatrix} \gamma_{(1,1)}(k) \\ \gamma_{(2,1)}(k) \\ \vdots \\ \vdots \\ \gamma_{(m-1,1)}(k) \\ \gamma_{(m,1)}(k) \end{bmatrix}^T \begin{bmatrix} Le_1(x_1(k)) \\ Le_2(x_1(k)) \\ \vdots \\ \vdots \\ Le_{m-1}(x_1(k)) \\ Le_m(x_1(k)) \end{bmatrix} + \ldots\ldots + \begin{bmatrix} \gamma_{(1,n)}(k) \\ \gamma_{(2,n)}(k) \\ \vdots \\ \vdots \\ \gamma_{(m-1,n)}(k) \\ \gamma_{(m,n)}(k) \end{bmatrix}^T \begin{bmatrix} Le_1(x_n(k)) \\ Le_2(x_n(k)) \\ \vdots \\ \vdots \\ Le_{m-1}(x_n(k)) \\ Le_m(x_n(k)) \end{bmatrix} \quad (8)$$

Finally the output from each of the FLANN model is passed through an activation block to give the output as

$$\left.\begin{aligned} & y(k) = \rho S(u(k)^{Trigono}) \\ & \text{or } y(k) = \rho S(u(k)^{Chebyshev}), \\ & \text{or } y(k) = \rho S(u(k)^{Laguerre}), \\ & \text{or } y(k) = \rho S(u(k)^{Legendre}) \end{aligned}\right\} \quad (9)$$

where ρ controls the output magnitude and S is a nonlinear function given by

$$S(u(k)) = \frac{2}{1 + e^{-2u(k)}} - 1 \quad (10)$$

To obtain the optimal δ, α, β and γ values, an error minimization algorithm can be used.

2.1. Computationally efficient FLANN (CEFLANN)

Computationally efficient FLANN is a single layer ANN that uses trigonometric basis functions for functional expansion. Unlike earlier FLANNS, where each input in the input pattern is expanded through a set of nonlinear functions, here all the inputs of the input pattern passes through a few set of nonlinear functions to produce the expanded input pattern; the new FLANN comprises only a few functional blocks of nonlinear functions for the inputs and thereby result in a high dimensional input space for the neural network. This new architecture of FLANN has much less computational requirement and possesses high convergence speed. Fig. 1 depicts the single layer computationally efficient FLANN architecture.

In this architecture a cascaded FIR element and a functional expansion block are used for the neural network. The output of this network is obtained as

$$y(k)^{cef} = \rho S(u(k))\{W_1(k)X(k) + W_2(k).FE(k)\}, \quad (11)$$

and $X(k) = [x_1(k), x_2(k)\ldots\ldots, x_n(k)]^T$

$$FE(k) = [FE_1(k), FE_1(k), FE_1(k), \ldots\ldots, FE_p(k)]^T, \quad (12)$$

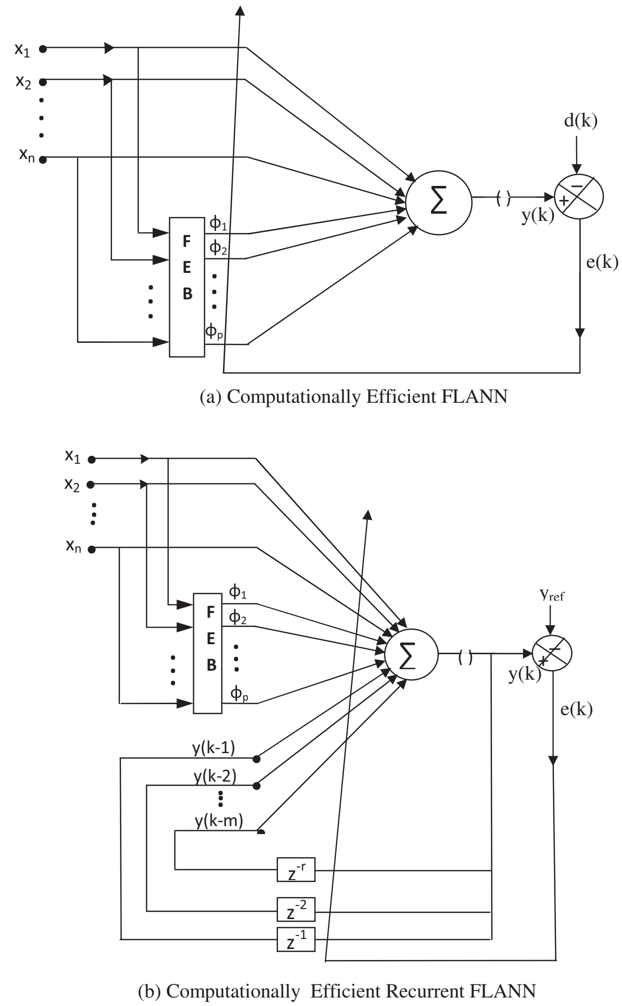


Figure 1 Computationally efficient FLANN models.

and

$$FE_i = \tanh\left(a_{i0} + \sum_{j=1}^{n} a_{ij}x_j\right), i = 1, 2, \ldots\ldots, p \quad (13)$$

where W_1 and W_2 are weight vectors for the linear part and functional expansion part, and p is the total number of expansions with n number of inputs; S' is the derivative of the activation function S.

2.2. Adaptation of weights of the FLANN models

The weights and associated parameters of the four FLANN models and the CEFLANN model are updated at the end of each experiment by computing the error between the desired output and the estimated output. The error at the kth time step of the Lth experiment is expressed as $e(k) = y_{des}(k) - y(k)$, and the y_{des} is the desired FLANN output. The cost function is

$$E(k) = \frac{1}{2}e^2(k) \quad (14)$$

Using gradient descent algorithm, the weights $W^{Tri}, W^{Che}, W^{Lag}, W^{Lege}$ of Trigonometric, Chebyshev, Laguerre, and Legendre FLANN model, respectively are updated as

Please cite this article in press as: Rout, A.K. et al., Forecasting financial time series using a low complexity recurrent neural network and evolutionary learning approach. Journal of King Saud University – Computer and Information Sciences (2015), http://dx.doi.org/10.1016/j.jksuci.2015.06.002

$$W^{FL}(k) = W^{FL}(k-1) + \eta\rho S'(u(k))(X^{FL}(k))' + \vartheta(W^{FL}(k-1) - W^{FL}(k-2)) \quad (15)$$

where η and ϑ are the learning rate and momentum terms, and W^{FL} is the weight vector for the corresponding FLANN model; and X^{FL} stands for the functional expansion terms of the input. With trigonometric expansions X^{FL} becomes

$$X^{FL} = [1\phi_1\phi_2\phi_3\ldots\ldots\phi_{mn-2}\phi_{mn-1}\phi_{mn}]^T = [1x_1\cos\pi x_1\sin\pi x_1\ldots\ldots x_n\cos\pi x_n\sin\pi x_n]^T$$

with total number of terms being equal to $mn+1$. For the first four FLANN models, the weight vector W^{FL} is given by

$$W^{Tri} = [\delta_{(0)}, \delta_{(1,1)}, \ldots, \delta_{(m,n)}]^T,\ W^{Che} = [\alpha_{(0)}, \alpha_{(1,1)}, \ldots, \alpha_{(m,n)}]^T$$
$$W^{Leg} = [\beta_{(0)}, \beta_{(1,1)}, \ldots, \beta_{(m,n)}]^T,\ W^{Lege} = [\gamma_{(0)}, \gamma_{(1,1)}, \ldots, \gamma_{(m,n)}]^T \quad (16)$$

For the CEFLANN model

$$W_1(k) = W_1(k-1) = \rho S'(u(k))e(k)X^T$$
$$W_2(k) = W_2(k-1) = \rho S'(u(k))e(k)FE^T \quad (17)$$

and the coefficients of the ith functional block are obtained as

$$a_{i0}(k) = a_{i0}(k-1) + \rho W_{2,i}S'(u(k))\sec^2 h\left(a_{i0} + \sum_{j=1}^{n} a_{ij}x_j\right) \quad (18)$$

$$a_{ij}(k) = a_{ij}(k-1) + \rho W_{2,i}S'(u(k))\sec^2 h\left(a_{i0} + \sum_{j=1}^{n} a_{ij}x_j\right)x_j \quad (19)$$

2.3. Computationally efficient recurrent FLANN (RCEFLANN)

In conventional FLANN models since no output lagging terms are used there is no correlation between the training samples. However, when lagged output samples are used as inputs, a correlation exists between the training samples and hence an incremental learning procedure is required for the adjustment of weights. A recurrent version of this FLANN is shown in Fig. 1, where one step delayed output samples are fed to the input to provide a more accurate forecasting scenario. As shown in Fig. 1 the input vector contains a total number $m+n+p$ inputs comprising n inputs from the delayed outputs, m inputs from the stock closing price indices, p inputs from the functional expansion. Thus the input vector is obtained as

$$V(k) = [R^T(k), X^T(k), FE^T(k)] \quad (20)$$

which is written in an expanded form as

$$V(k) = [y_1(k-1), y(k-2), \ldots, y_1(k-n), x_1, x_2\ldots, x_m, \phi_1.\phi_2\ldots, \phi_p] \quad (21)$$

and for a low complexity expansion ϕ_i takes the form

$$\phi_i = \tanh(w_{i0} + w_{i1}x_1 + w_{i2}x_2 + \ldots + w_{im}),\ i = 1, 2, \ldots, p$$

Thus the output is obtained as

$$y(k) = \gamma S(u^{cef}(k)) = \gamma S(W_1(k)R(k)) + W_2(k)X(k) + W_3(k).FE(k) \quad (22)$$

where

$$W_1 = [a_1a_2a_3\ldots a_n]^T, W_2 = [b_1b_2a_3\ldots b_m]^T, W_3 = [c_1c_2c_3\ldots c_p]^T \quad (23)$$

The most common gradient based algorithms used for online training of recurrent neural networks are BP algorithms and real-time recurrent learning (RTRL) (Ampolucci et al., 1999).

The RTRL algorithm is shown in the following steps:

Using the same cost function for the recurrent FLANN model, for a particular weight $w(k)$, the change of weight is obtained as

$$\Delta w(k) = -\eta\frac{\partial e(k)}{\partial w(k)} = \eta e(k)S'(u^{cef}(k))\frac{\partial y(k)}{\partial w(k)} \quad (24)$$

where η is the learning rate.

The partial derivative of the above Eq. (24) is obtained in a modified form for the RTRL algorithm as

$$\frac{\partial y(i)}{\partial a_k} = y(i-k) + \sum_{j=1}^{n} a_j\frac{\partial y(i-j)}{\partial a_k},\ k = 1, 2, \ldots, n$$
$$\frac{\partial y(i)}{\partial b_k} = x_k + \sum_{j=1}^{n} a_j\frac{\partial y(i-j)}{\partial b_k},\ k = 1, 2, \ldots, m$$
$$\frac{\partial y(i)}{\partial c_k} = \phi_k + \sum_{j=1}^{n} a_j\frac{\partial y(i-j)}{\partial c_k},\ k = 1, 2, \ldots, p$$
$$\frac{\partial y(i)}{\partial w_{k0}} = c_k\sec^2\phi_k + \sum_{j=1}^{n} a_j\frac{\partial y(i-j)}{\partial w_{k0}},\ k = 1, 2, \ldots, p$$
$$\frac{\partial y(i)}{\partial w_{kr}} = c_kx_k\sec^2\phi_k + \sum_{j=1}^{n} a_j\frac{\partial y(i-j)}{\partial w_{kr}},\ k = 1, 2, \ldots, p,\ r = 1, 2, \ldots, p \quad (25)$$

The weight adjustment formulas are, therefore, obtained as (with k taking values n, m, and p for the weights of the recurrent, input, functional expansion parts, respectively):

$$a_k(i) = a_k(i-1) + \eta e(k)S'(u^{cef}(k))\frac{\partial y(i)}{\partial a_k}$$
$$b_k(i) = b_k(i-1) + \eta e(k)S'(u^{cef}(k))\frac{\partial y(i)}{\partial b_k}$$
$$c_k(i) = c_k(i-1) + \eta e(k)S'(u^{cef}(k))\frac{\partial y(i)}{\partial c_k}$$
$$w_{kr}(i) = w_{kr}(i-1) + \eta e(k)S'(u^{cef}(k))\frac{\partial y(i)}{\partial w_{kr}} \quad (26)$$

where $u_j(k) = \sum_{j=1}^{p+q+r} w_j(k)v_j$, and $\delta_{i,j}$ is a Kronecker delta equal to 1 when $j = k$ and 0 otherwise.

Further if the learning rate is kept small, the weights do not change rapidly and hence

$$\frac{\partial y(i-1)}{\partial a_k} \approx \frac{\partial y(i-1)}{\partial a_k(i-1)} \quad (27)$$

Denoting the gradient

$$\pi_k(i) = \frac{\partial y(i)}{\partial w_k}$$

and the gradient values in successive iterations are generated assuming

$$\pi_k(0) = 0 \quad (28)$$

In both these algorithms gradient descent based on first order derivatives is used to update the synaptic weights of the network. However, both these approaches, exhibit slow convergence rates because of the small learning rates required, and most often they become trapped to local minima. To avoid the common drawbacks of back propagation algorithm and to increase accuracy, different methods have been proposed that include additional momentum method, self-adaptive learning rate adjustment method, and various search algorithms like GA, PSO, DE algorithm in the training step of the neural network to optimize the parameters of the network like the network weights and the number of hidden units in the hidden layer. In Section 3, the various evolutionary learning methods like Particle Swarm Optimization, Differential Evolution and Hybrid Moderate Random Search PSO have been described for training various FLANN models. Evolutionary algorithms act as excellent global optimizers for real parameter problems.

3. Evolutionary learning methods

The problem described can be formulated as an objective function for error minimization using the equation below.

$$ERROR = \sqrt{\sum_{k=1}^{N}\left(\frac{d_{(k)} - y_{(k)}}{\sqrt{N}}\right)^2} \tag{29}$$

where prediction is done k days ahead and $y(k)$ is represented as a function of weights and the prior values of the time series:

$$y_{(k)}^{Trigono} = f\left(\underbrace{\delta_{(0)}, \delta_{(1,1)}, \ldots, \delta_{(m,n)}}_{\text{variables}}, \underbrace{x_{(t)}, \ldots, x_{(k-n+1)}}_{\text{constants}}\right) \tag{30}$$

$$y_{(k)}^{Chebyshev} = f\left(\underbrace{\alpha_{(0)}, \alpha_{(1,1)}, \ldots, \alpha_{(m,n)}}_{\text{variables}}, \underbrace{x_{(t)}, \ldots, x_{(k-n+1)}}_{\text{constants}}\right) \tag{31}$$

$$y_{(k)}^{Laguerre} = f\left(\underbrace{\beta_{(0)}, \beta_{(1,1)}, \ldots, \beta_{(m,n)}}_{\text{variables}}, \underbrace{x_{(t)}, \ldots, x_{(k-n+1)}}_{\text{constants}}\right) \tag{32}$$

$$y_{(k)}^{Legendre} = f\left(\underbrace{\gamma_{(0)}, \gamma_{(1,1)}, \ldots, \gamma_{(m,n)}}_{\text{variables}}, \underbrace{x_{(t)}, \ldots, x_{(k-n+1)}}_{\text{constants}}\right) \tag{33}$$

The dimension of the problem to be solved by the evolutionary algorithm would be $(m + n + 1)$.

The objective function for optimization is chosen as

$$J = \frac{1}{1 + ERROR^2} \tag{34}$$

3.1. DE based learning

Differential Evolution (DE) is a population-based stochastic function optimizer, which uses a rather greedy and less stochastic approach for problem solving in comparison to classical evolutionary algorithms, such as genetic algorithms, evolutionary programing, and PSO. DE combines simple arithmetical operators with the classical operators of recombination, mutation, and selection to evolve from a randomly generated starting population to a final solution. Here, a self-adaptive strategy for the control parameters of DE like F and Cr is adopted to improve the robustness of the DE algorithm. The pseudo code for DE implementation is given below:

3.1.1. Pseudo code for DE implementation

Input: population size Np, No. of variables to be optimized (Dimension D), initial scaling and mutation parameters F, F_1, F_2, and Cr. G = total number of generations, X = target vector, Strategy candidate pool: "DE/rand/2"

Population $P_g = \{X_{(i)}^g, \ldots\ldots, X_{(Np)}^g\}$, $X_{(i)}^g = \{x_{(i,1)}^g, x_{(i,2)}^g, \ldots, x_{(i,j)}^g, \ldots, x_{(i,D)}^g\}$ uniformly distributed.

1. While stopping criterion is not satisfied **do**
2. for $i = 1$ to Np

$$F_1 = F_{L1} + (F_{U1} - F_{L1}) \times rand(0,1)$$
$$F2 = F_{L2} + (F_{U2} - F_{L2}) \times rand(0,1) \tag{35}$$

Generate the mutant vector

$$V_{(i)}^g = \{v_{(i,1)}^g, v_{(i,2)}^g, \ldots, v_{(i,j)}^g, \ldots, v_{(i,D)}^g\}$$
$$V_{(i)}^g = X_{(\mu)}^g + F_1 \times \left(X_{(\delta)}^g - X_{(\gamma)}^g\right) + F_2 \times \left(X_{(\xi)}^g - X_{(\eta)}^g\right) \tag{36}$$

The indices, $\mu, \delta, \gamma, \xi, \eta \in [1,\text{Np}]$**end for**

3. for $i = 1$ to Np
$j_{rand} = rndinit(1, D)$
for $j = 1$ to D

The crossover rate is adapted as

$$Cr = Cr_L + (Cr_U - Cr_L) \times rand3 \tag{37}$$

where $rand3$ is a random number between [0,1].
Generate trial vector $U_{(i)}^g$ using the target vector

$$U_{(i)}^g = \{u_{(i,1)}^g, u_{(i,2)}^g, \ldots, u_{(i,j)}^g, \ldots, u_{(i,D)}^g\}$$

$$u_{(i,j)}^g = \begin{Bmatrix} v_{(i,j)}^g & if(rand_{(i,j)} \leqslant Cr)\, or\, j = j_{rand} \\ x_{(i,j)}^g & otherwise \end{Bmatrix} \tag{38}$$

end for
end for

4. for $i = 1$ to NpEvaluate the objective function of the trial vector

$$U_{(i)}^g = \{u_{(i,1)}^g, u_{(i,2)}^g, \ldots, u_{(i,j)}^g, \ldots, u_{(i,D)}^g\} \tag{39}$$

The objective function is obtained from Eq. (34) as

$$f(U_{(i)}^g) = J$$
$$\text{If } f(U_{(i)}^g) < f(X_{(i)}^g), \text{ Then } X_{(i)}^{g+1} = U_{(i)}^g, f(X_{(i)}^{g+1}) = f(U_{(i)}^g)$$
$$\text{If } f(U_{(i)}^g) < f(X_{(best)}^g), \text{Then } X_{(best)}^g = U_{(i)}^g, f(X_{(best)}^g) = f(U_{(i)}^g) \tag{40}$$

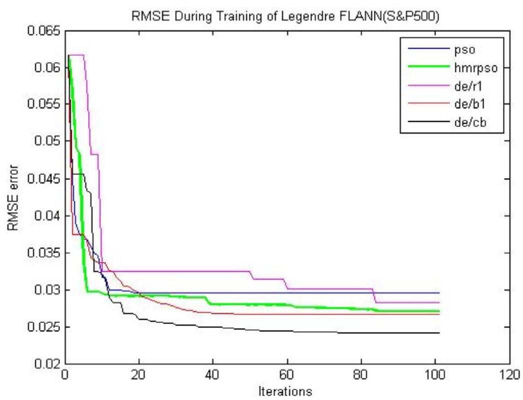
end ifend forg = g + 1

5. **end** While

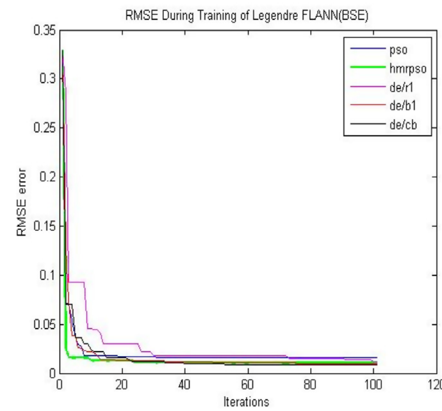
3.2. Adaptive HMRPSO based learning

In the conventional PSO algorithm the particles are initialized randomly and updated afterward according to:

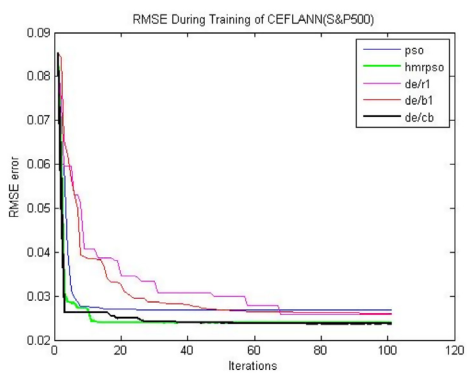
Please cite this article in press as: Rout, A.K. et al., Forecasting financial time series using a low complexity recurrent neural network and evolutionary learning approach. Journal of King Saud University – Computer and Information Sciences (2015), http://dx.doi.org/10.1016/j.jksuci.2015.06.002



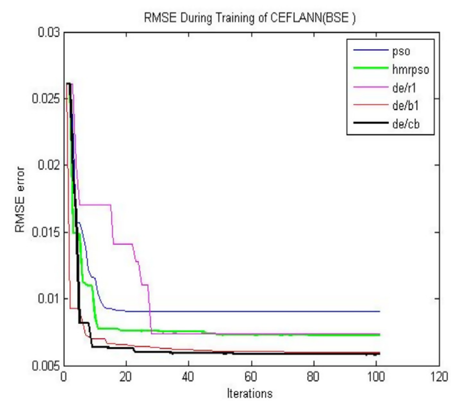
(a) RMSE error comparison of S&P500 data using evolutionary algorithms and LEFLANN



(b) RMSE error comparison of BSE data using evolutionary algorithms and LEFLANN



(c) RMSE error comparison of S&P500 data using different evolutionary algorithms and CEFLANN



(d) RMSE error comparison of BSE data using different evolutionary algorithms and CEFLANN

Figure 2 RMSE errors of Legender and computationally efficient FLANNs for S&P500 and BSE stock market indices using different Evolutionary algorithms.

$$\begin{aligned} x_{ij}^{k+1} &= x_{ij}^{k} + V_{ij}^{k+1} \\ V_{ij}^{k+1} &= \omega V_{ij}^{k} + c_1 r_1 \left(pbest_{ij}^{k} - x_{ij}^{k}\right) + c_2 r_2 \left(gbest_{j}^{k} - x_{ij}^{k}\right) \end{aligned} \tag{41}$$

where w, c_1, c_2 are inertia, cognitive and social acceleration constants respectively, r_1 and r_2 are random numbers within [0,1]. *pbest* is the best solution of the particle achieved so far and indicates the tendency of the individual particles to replicate their corresponding past behaviors that have been successful. *gbest* is the global best solution so far, which indicates the tendency of the particles to follow the success of others (Lu, 2011; Sanjeevi et al., 2011; Ampolucci et al., 1999). HMRPSO is a new PSO technique using the moderate random search strategy to enhance the ability of particles to explore the solution spaces more effectively and a new mutation strategy to find the global optimum (Gao and Xu, 2011). This new method improves the convergence rate for the particles by focusing their search in valuable search space regions. The mutation operator is a combination of both global and local mutation. The global mutation enhances the global search ability of MRPSO, when the difference of a particle's position between two successive iterations on each dimension has decreased to a given threshold value. The momentum for this particle to roam through the solution space is maintained by resetting the particle in a random position. The local mutation operator enhance the particles to search precisely in the local area of the *pbest* found so far, which compensates the weaker local search ability, which results from the use of a global mutation operator. Like PSO it does not require any velocity update equation. Using MRS strategy the position update equation is as follows:

$$x_{ij}^{k+1} = p_j + \alpha\gamma\left(mbest_{ij} - x_{ij}^{k}\right) \tag{42}$$

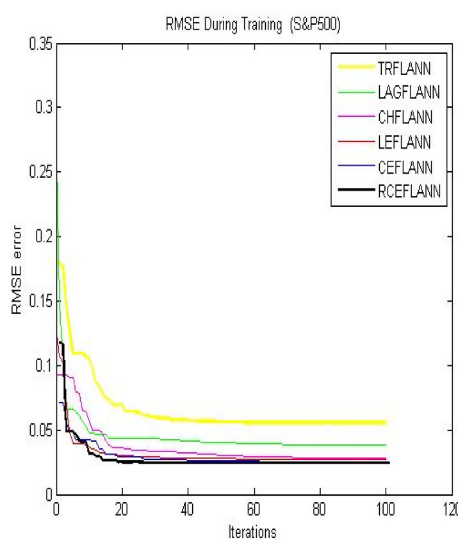
where

$$mbest = \sum_{i=1}^{np} \frac{pbest_i}{np} \tag{43}$$

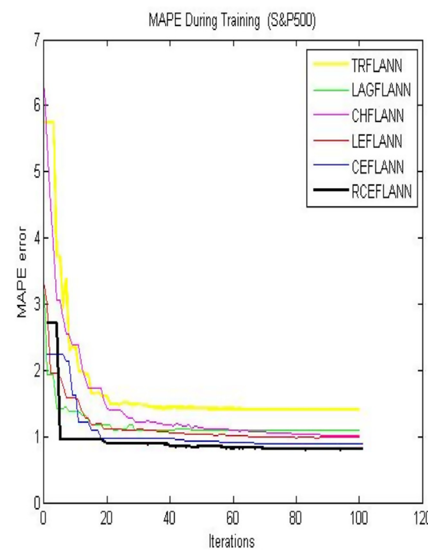
$$p_j = r1 \times pbest_{ij} + (1 - r1)gbest_j,\ \gamma = (r2 - r3)/r4 \tag{44}$$

The attractor p is the main moving directions of particles and r_1, r_2, r_3 are the uniformly distributed random variables within [0, 1], where as $r4$ is a random variable within [−1, 1].

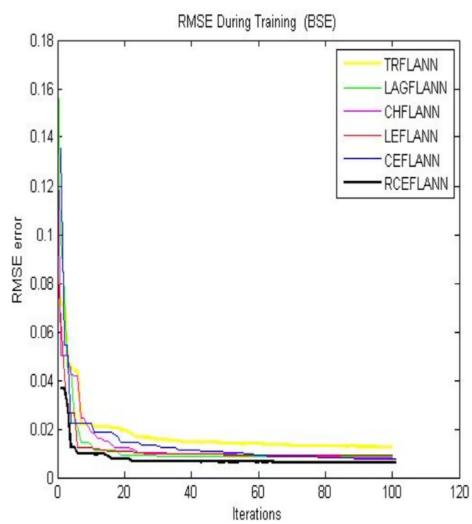
Please cite this article in press as: Rout, A.K. et al., Forecasting financial time series using a low complexity recurrent neural network and evolutionary learning approach. Journal of King Saud University – Computer and Information Sciences (2015), http://dx.doi.org/10.1016/j.jksuci.2015.06.002



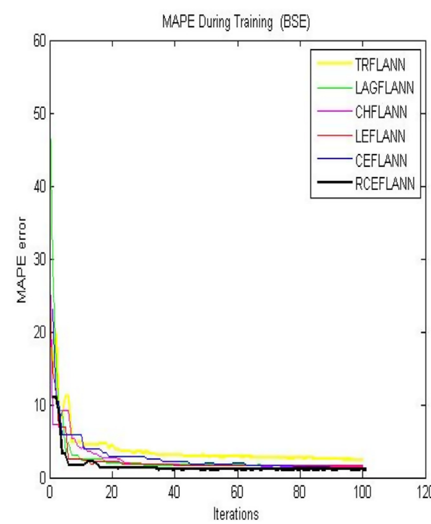
(a) RMSE error comparison of S&P500 data for different FLANNs using DE



(b) MAPE error comparison of BSE data using different FLANNs using DE



(a) RMSE error comparison of BSE data using different FLANNs and DE



(b) MAPE error comparison of BSE data using different FLANNs and DE

Figure 3 RMSE and MAPE Errors for different FLANNS and DE.

The *mbest* term gives step size for the movement of particles and also makes the contribution of all *pBest* to the evolution of particles. The inertia factor α controls the convergence rate of the method. If the absolute value of the difference between x_{ij}^{k+1} and x_{ij}^{k} is smaller than a threshold value T_j, then the current particle can use a mutation operator on dimension j. A suitable value is chosen for threshold, such that neither the algorithm spent more time in mutation operator and less time to conduct MRS in solution space, nor it gets a little chance to do mutation such that the particles will need a relatively long time to escape from the local optima. Again a gradient descent method is used to control the value of T_j using the following formula:

$$\text{If } (F_j = k) \text{ then } F_j = 0 \text{ and } T_j = T_j/m \tag{45}$$

Parameter m controls the decline rate of T_j and k controls the mutation frequency of particles. F_j denotes the number of particles that has used the mutation operator on dimension j. F_j is updated by the following equation:

$$F_j(k) = F_j(k-1) + \sum_{i=1}^{np} b_{ij}^k \tag{46}$$

where

$$b_{ij}^k = \begin{cases} 0 & \text{if } (abs(x_{ij}^k - x_{ij}^{k-1}) > T_j) \\ 1 & \text{if } (abs(x_{ij}^k - x_{ij}^{k-1}) \leqslant T_j) \end{cases} \tag{47}$$

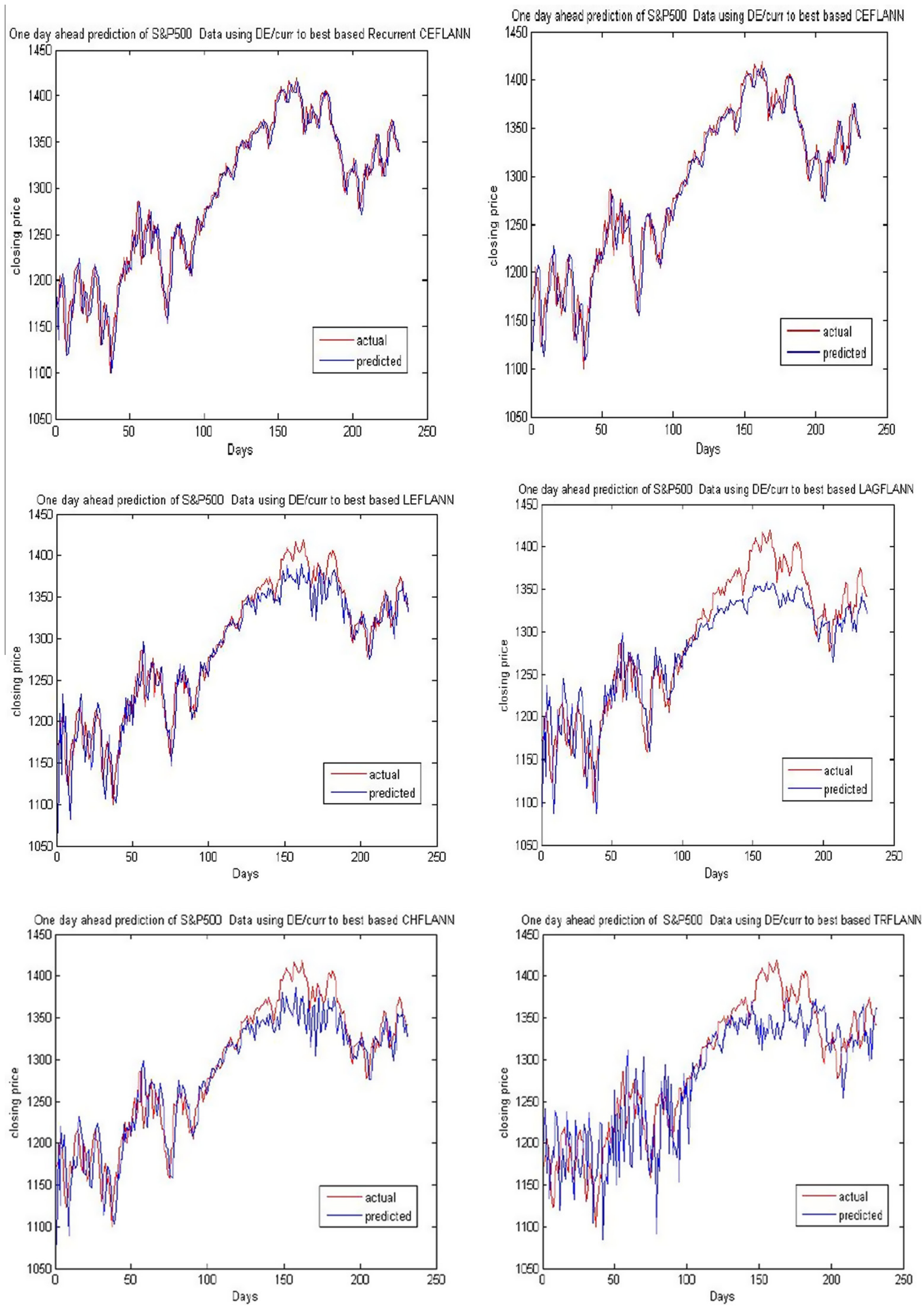


Figure 4 Comparison of actual and predicted S&P500 stock indices with different FLANN models.

F_j is initially set to zero and T_j is initially set to the maximum value taken for the domain of particles position. To enhance the global search ability of the HMRPSO, a global mutation operator is applied using the following formula:

$$mutate(x_{ij}) = (r_5 \times range) \tag{48}$$

r_5 is a random variable within [−1,1] and range is the maximum value set for the domain of particles position. Again the weaker local searching ability caused using the global mutation operator is compensated using a local mutation operator as follows:

$$pbest'_{tj} = pbest_{tj}(1 \pm mut(\sigma)) \tag{49}$$

where t is the index of the *pbest that* gives best achievement among all other *pbest* positions and $mut(\sigma)$ returns a value within (0, 1), which is drawn from a standard Gaussian distribution. finding the variance of the population fitness as

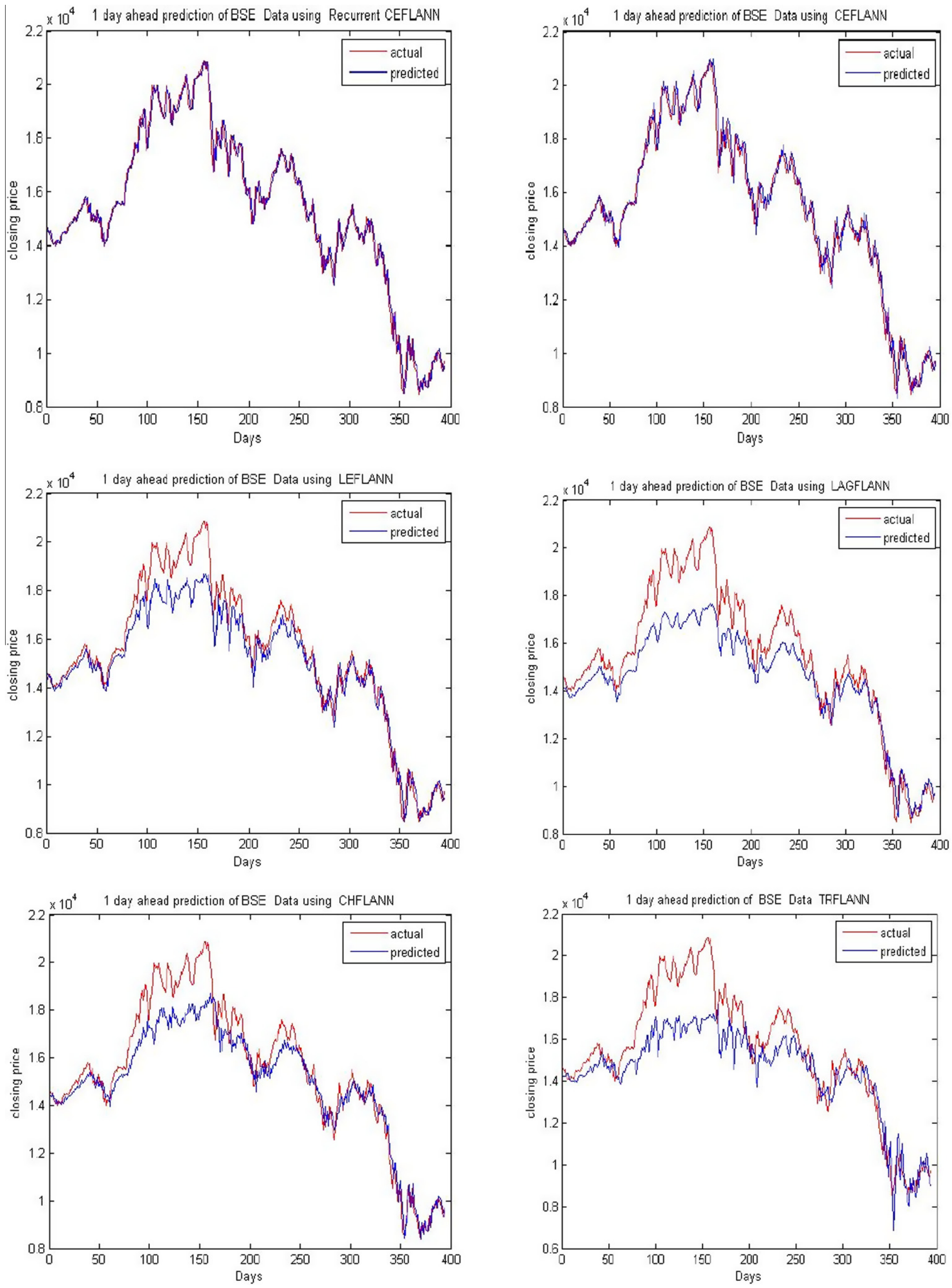


Figure 5 Comparison of actual and predicted BSE stock indices with different FLANN models.

$$\xi = \sqrt{\sum_{i=1}^{Np}\left(\frac{J_i - Javg}{F_0}\right)^2} \tag{50}$$

where $Javg$ is the average fitness of the population of particles in a given generation. F_i is the fitness of the ith particle in the population, M is the total number of particles.

$$F_0 = \{\max(|F_i - Favg|)\}, i = 1, 2, 3 \ldots\ldots M \tag{51}$$

In the equation given above F_n is the normalizing factor, which is used to limit ξ. If ξ is large, the population will be in a random searching mode, while for small ξ or $\xi = 0$, the solution tends toward a premature convergence and will give the local best position of the particles. To circumvent this phenomenon and to obtain gbest solution, the factor α in (42) controls the convergence rate of the HMRPSO, which is similar with inertia weight *used* in the PSO. On the one hand, a larger α value enables particles to have a stronger exploration ability but a less exploitation ability, while on the other hand, smaller α allows particles a more precise exploitation ability.

The factor α in Eq. (42) is updated using a fuzzy rule base and fuzzy membership values of the change in standard deviation $\Delta\xi$ in the following way:

$$\mu_{1\,large}(\xi) = e^{-|\xi|^2}, \mu_{1\,small}(\xi) = 1 - e^{-|\xi|^2} \tag{52}$$

ARTICLE IN PRESS

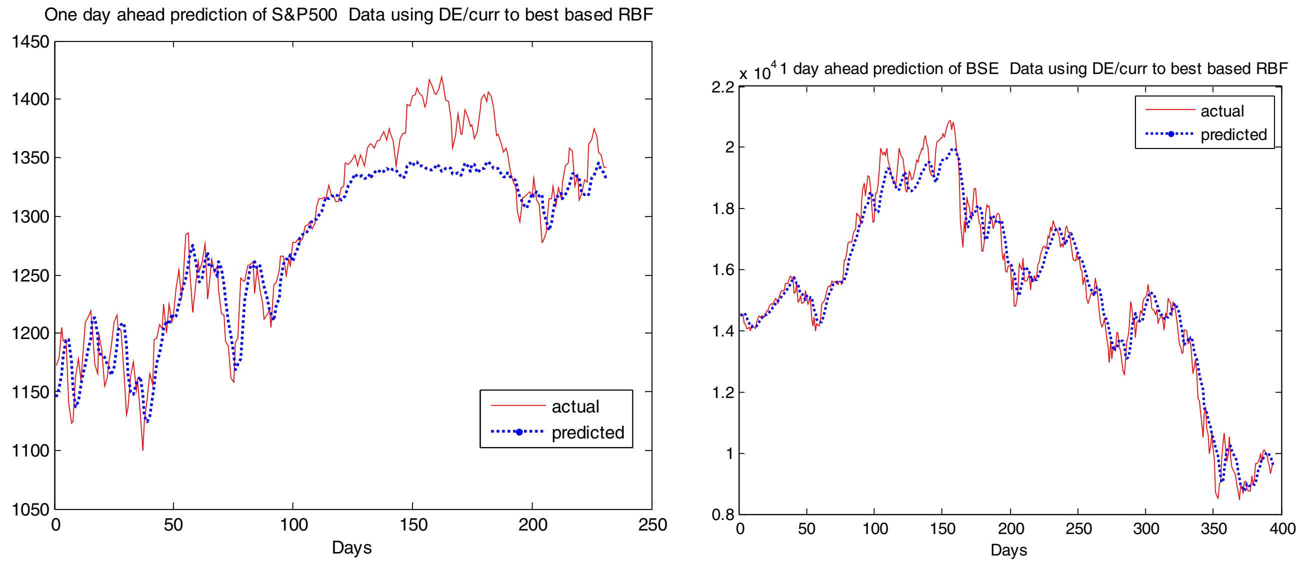


Figure 6 Comparison of actual and predicted S&P500 and BSE stock indices with RBF.

Change in standard deviation

$$\Delta\xi(k) = \xi(k) - \xi(k-1) \tag{53}$$

The fuzzy rule base for arriving at a weight change is expressed as

$$\begin{aligned} &\text{R1. If } |\Delta\xi| \text{ is Large Then } \Delta\alpha = \text{rand}_1(\,)\,\Delta\xi \\ &\text{R2. If } |\Delta\xi| \text{ is Small Then } \Delta\alpha = \text{rand}_2(\,)\Delta\alpha \end{aligned} \tag{54}$$

where the membership functions for Large and Small are given by

$$\mu_L(\xi) = |\Delta\xi|;\ \mu_S(\xi) = 1 - |\Delta\xi| \tag{55}$$

where $rand_1(\,)$ and $rand_2(\,)$ are random numbers between 0 and 1, and

$$0 \leqslant |\Delta\xi| \leqslant 1 \tag{56}$$

$$\Delta\alpha = |\Delta\xi|.\ rand_1\ (\,)\ \Delta\xi + rand_2\ (\,)\ \Delta\xi.(1 - |\Delta\xi|). \tag{57}$$

Thus the value of the new weight is obtained as

$$\alpha(k) = \alpha(k-1) + \Delta\alpha \tag{58}$$

3.2.1. Steps of adaptive HMRPSO based learning

Step 1: Expand the input pattern using functional expansion block
Step 2: Initialize the position of each individual within a given maximum value
Step 3: Find the fitness function value of each particle i.e. the error obtained by applying the weights specified in each particle to the expanded input and applying the nonlinear tanh () function at the output unit
Step 4: Initialize the *pbest*, *gbest*, $pbest_t$ positions of the particles
Step 5: Update the particle's position using the updated value of α using Eq. (58)
Step 6: For each dimension j
Step 7: *if* $(abs(x_{ij}^k - x_{ij}^{k-1}) < T_j)$
Step 8: if $(F_j \leqslant k/2)$
Step 9: Apply global mutation operator on x_{ij} using Eq. (48)
Step 10: **else**
Step 11: Apply local mutation operator on best *pbest* position i.e. $pbest_{tj}$
Step 12: **end if**
Step 13: **end if**
Step 14: Update the F_j and T_j using Eqs. (47) and (46)
Step 15: **end for**
Step 16: Update the *pbest* and *gbest* positions by comparing the fitness value of new mutant particles obtained using global mutation with the fitness value of particles of last iteration
Step 17: Update the position of $pbest_t$ and accordingly the *gbest* position by comparing the fitness value of $pbest_t$ obtained using location mutation with its previous fitness value
Step 18: Repeat step 5–17 for each particle in the population until some termination condition is reached, such as predefined number of iterations is reached or the error has satisfied the default precision value
Step 19: Fixed the weight equal to the *gbest* position and use the network for testing

4. Experimental result analysis

In this study the sample data set of Bombay Stock Exchange (BSE) and Standard's &Poor's 500(S&P500) collected from Google finance comprising the daily closing prices has been taken for experiment to test the performance of different FLANN models trained using evolutionary methods. The total no. of samples for BSE is 1200 from 1st January 2004 to 31st December 2008 and for S&P500 are 653 from 4th January 2010 to 12th December 2012. Both the data sets are divided into training and testing sets. For BSE data set; the training set consists of initial 800 patterns and the following 400 data are set for testing and for S&P500 dataset the training set

Please cite this article in press as: Rout, A.K. et al., Forecasting financial time series using a low complexity recurrent neural network and evolutionary learning approach. Journal of King Saud University – Computer and Information Sciences (2015), http://dx.doi.org/10.1016/j.jksuci.2015.06.002

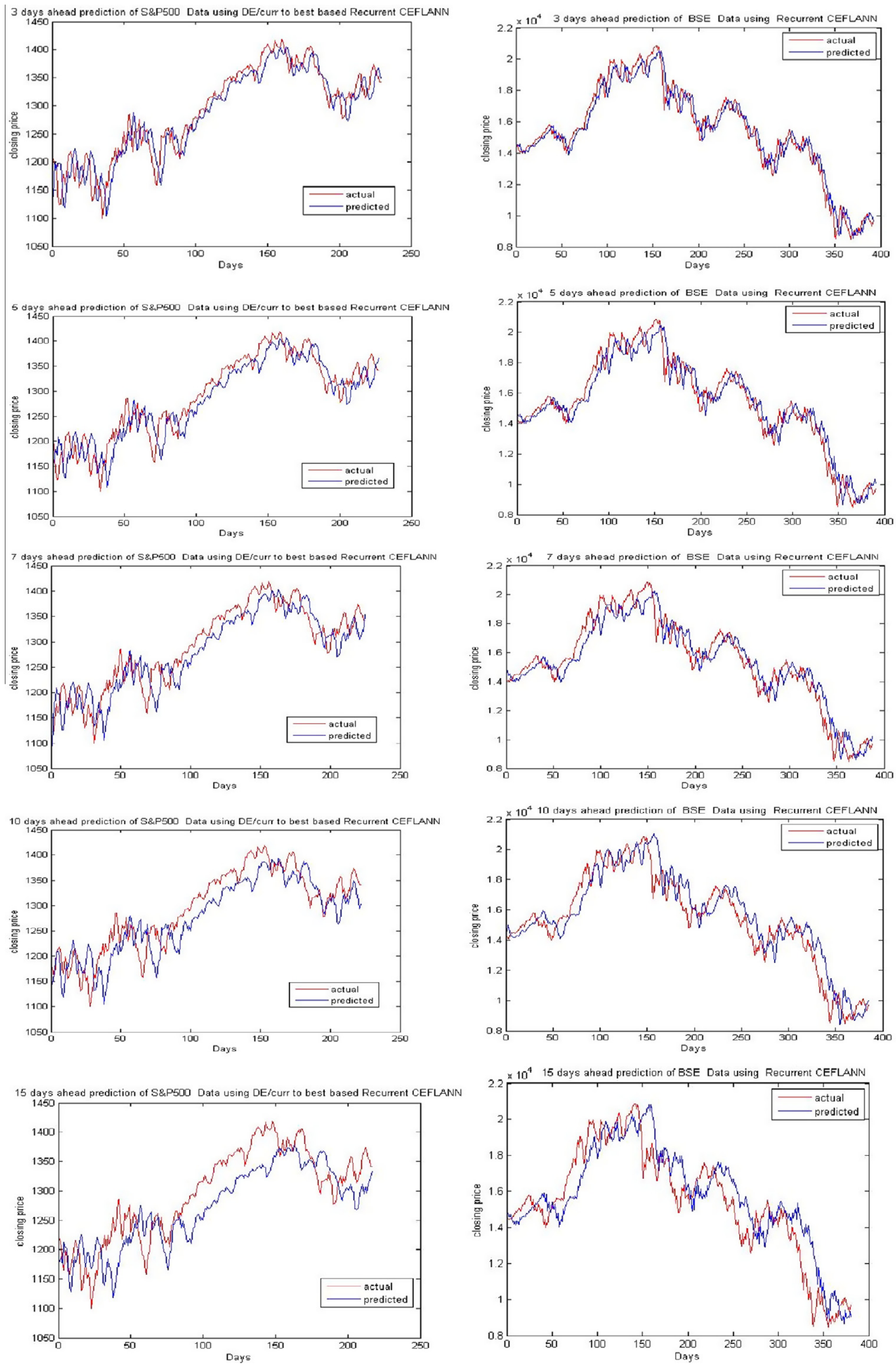


Figure 7 Comparison of actual and predicted S&P500and BSE stock indices with the recurrent CEFLANN model (3–15 days ahead).

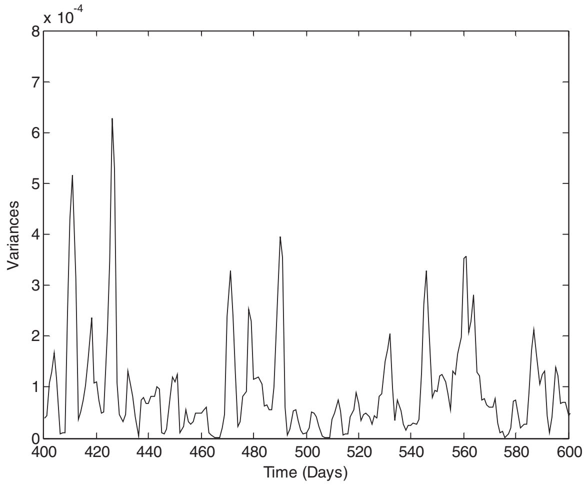
consists of initial 400 patterns leaving the rest for testing. To improve the performance initially all the inputs are scaled between 0 and 1 using the min–max normalization as follows:

$$y = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{59}$$

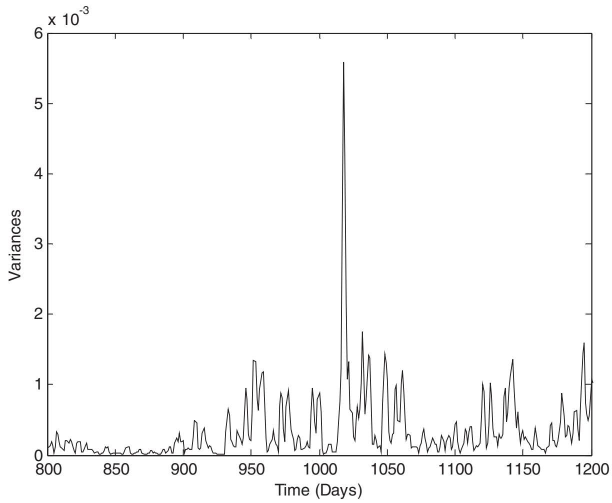
where y = normalized value.

x = value to be normalized
$x_{\min}$ = minimum value of the series to be normalized
$x_{\max}$ = maximum value of the series to be normalized

The Root Mean Square Error (RMSE) and Mean Absolute Percentage Error (MAPE) has been used to compare the performance of different evolutionary FLANNs for predicting



(a) Variances of S&P 500 Stock data from 23rd July 2010 to 6th May 2011



(b) Variances of BSE Stock data from 7th March 2007 to 13th October 2008

Figure 8 Computed variances of the S&P 500 and BSE stock indices.

the closing price of the BSE and S&P500 index in 1 day, 3 days, 5 days, 7 days, 10 days, 15 days advance with different learning algorithms. The RMSE and MAPE are defined as:

$$RMSE = \sqrt{\frac{1}{n}\sum_{k=1}^{n}(y_k - \hat{y}_k)^2} \tag{60}$$

$$MAPE = \frac{1}{n}\sum_{k=1}^{n}\left|\frac{y_k - \hat{y}_k}{y_k}\right| \times 100 \tag{61}$$

where y_k = actual closing price on kth day

$\hat{y}_k$ = predicted closing price on kth day
n = number of test data.

In order to avoid the adverse effect of very small Stock indices over a long period of time, the AMAPE defined in (62) is adopted and compared for all the learning models:

$$AMAPE = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{y_i - \hat{y}_i}{(1/N)\sum_{i=1}^{N} y_i}\right| \times 100 \tag{62}$$

Before using a single model or a combination of models to predict the future stock market indices, it is assumed that there is a true model or a combination of models for a given time series. In this paper, the variance of forecast errors is used to measure this uncertainty of the neural network model. The smaller the variance, the less uncertain is the model or more accurate is the forecast results. The variance of the forecast errors over a certain span of time is needed and this parameter is obtained as

$$\sigma_E^2 = \frac{1}{ND}\sum_{i=1}^{ND}\left\{\frac{|y_i - \hat{y}_i|}{(1/ND)\sum_{i=1}^{N} y_i} - \sum_{i=1}^{ND}\frac{|y_i - \hat{y}_i|}{(1/ND)\sum_{i=1}^{N} y_i}\right\}^2 \tag{63}$$

Further the Technical indicators seem to predict the future price value by looking at past patterns. A brief explanation of each indicator is mentioned here:

Technical Indicators

(i) Five days Moving Average(MA)Moving average (Eq. (64)) is used to emphasize the direction of a trend and smooth out price and volume fluctuations that can confuse interpretation.

$$MA(n) = \frac{\sum_{i=t-n}^{t} y_i}{n} \tag{64}$$

MA of the n days and y_i the closed price of the i^{th} -day.

(ii) Five days bias (BIAS)The difference between the closing value and moving average line, which uses the stock price nature of returning back to average price to analyze the stock market in (Eq. (65)).

$$\text{BIAS for } n \text{ days}, BIAS(n) = y_i \times \frac{y_i - MA5(n)}{MA(n)}. \tag{65}$$

(iii) Standard Deviation(SD)

$$\mathrm{SD}(\sigma) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \mu)^2} \quad \text{where } \mu = \frac{1}{n}\sum_{i=1}^{n} y_i \tag{66}$$

The no. of input layer node for the FLANN models has been set to 8 to express the closing index of 5 days ago and the technical indicators mentioned above and the number of output node to 1 for expressing the closing index of 6th day. The 8 dimensional input patterns have been expanded to a pattern of dimension 33 using order 4 and the corresponding basis functions for TRFLANN, LAGFLANN, CHFLANN, and LEFLANN. For CEFLANN using two nonlinear tanh() functions for expansion, the input pattern has been expanded to pattern of 10 dimensions. The same model also is used in the case of the recurrent CEFLANN model with three lagged output terms fed into the input. Initially the performance of each FLANN model having the same network structure and same data sets trained using PSO, HMRPSO, and DE with different mutation strategy has been observed. The same population having size 20 with maximum training times $k_{\max} = 100$ has been used for all the evolutionary learning based methods. Each individual in the population has width 33 i.e. equal to the no. of weights for training TRFLANN, CHFLANN, LAGFLANN, LEFLANN

Table 1 Performance comparison of different FLANN models with different evolutionary learning methods for 1-day ahead prediction.

Type of FLANN	Evolutionary learning method	S&P500		BSE	
		RMSE test	MAPE test	RMSE test	MAPE test
Recurrent CEFLANN	PSO	0.0411	1.3876	0.0259	3.1120
	HMRPSO	0.0332	1.0895	0.0210	2.4883
	DE current to best	**0.0308**	**1.0197**	**0.0152**	**1.8180**
CEFLANN	PSO	0.0441	1.5158	0.0282	3.2834
	HMRPSO	0.0343	1.1362	0.0224	2.6905
	DE current to best	**0.0328**	**1.0849**	**0.0184**	**2.2441**
LEFLANN	PSO	0.0479	1.7231	0.0301	3.3398
	HMRPSO	0.0438	1.5243	0.0272	3.0531
	DE current to best	**0.0340**	**1.1594**	**0.0258**	**2.8172**
LAGFLANN	PSO	0.0567	2.0437	0.0622	7.0308
	HMRPSO	0.0527	1.8557	0.0576	6.1226
	DE current to best	**0.0506**	**1.7729**	**0.0568**	**5.9825**
CHFLANN	PSO	0.0545	1.9178	0.0524	5.7065
	HMRPSO	0.04200	1.4166	0.0425	4.3535
	DE current to best	**0.0371**	**1.3099**	**0.0415**	**4.2276**
TRFLANN	PSO	0.1041	3.4783	0.0791	8.3051
	HMRPSO	0.0883	3.0075	0.0699	7.2520
	DE current to best	**0.0738**	**2.5679**	**0.0688**	**7.1153**
Recurrent CEFLANN	RTRL	0.0563	2.579	0.0712	5.862
CEFLANN	Gradient Descent	0.0513	2.456	0.0451	4.685

Bold values are the outputs from the proposed method.

Table 2 Comparison of convergence speed of CEFLANN models with different evolutionary learning methods.

Type of FLANN	Evolutionary learning method	S&P500	BSE
		Time elapsed in sec during training	Time elapsed in sec during training
Recurrent CEFLANN	PSO	66.28	154.26
	HMRPSO	155.85	270.65
	DE current to best	**49.94**	**148.31**
CEFLANN	PSO	40.07	96.57
	HMRPSO	108.17	229.43
	DE current to best	**35.09**	**92.65**

Bold values are the outputs from the proposed method.

Table 3 MAPE errors during testing of S&P500 data set using different evolutionary FLANN models.

Days ahead	TRFLANN	LAGFLANN	CHFLANN	LEFLANN	CEFLANN	RCEFLANN	RBFNN	WNN
1	2.8264	2.1865	1.7922	1.4662	1.1212	0.9958	1.8565	1.9432
3	3.4751	2.3116	2.6464	2.3718	1.7768	1.7672	2.9070	3.112
5	3.0419	2.5902	3.8075	3.1018	2.1164	2.1064	3.1682	3.323
7	4.2072	2.8144	4.8223	4.1813	2.4299	2.4093	3.2899	3.425
10	5.3751	3.0774	6.1739	5.2114	2.6965	2.6951	4.1729	4.0867
15	6.6261	3.4447	6.4565	6.3137	3.2343	3.1429	5.2024	4.9326

structure using evolutionary methods. But the individual has width 28 i.e. equal to the total no. of associated parameters and the weights when used for training the CEFLANN network. For PSO the values of c1 and c2 are set at 1.9 and the inertia factor has linearly decreased from 0.45 to 0.35. For DE the mutation scale has been fixed to $F_{U1} = 0.95, F_{L1} = 0.4$ $F_{U2} = 0.8, F_{L2} = 0.3$, and the crossover rate to $Cr_U = 0.9, Cr_L = 0.6$. For HMRPSO the inertia factor has linearly decreased from 0.4 to 0.1. The RMSE error has taken as the fitness function for all the evolutionary learning algorithms.

Please cite this article in press as: Rout, A.K. et al., Forecasting financial time series using a low complexity recurrent neural network and evolutionary learning approach. Journal of King Saud University – Computer and Information Sciences (2015), http://dx.doi.org/10.1016/j.jksuci.2015.06.002

Table 4 MAPE errors during testing of BSE data set using different evolutionary FLANN models.

Days ahead	TRFLANN	LAGFLANN	CHFLANN	LEFLANN	CEFLANN	RCEFLANN	RBFNN	WNN
1	7.1153	5.9825	4.2276	3.8741	2.2441	1.8180	2.8752	2.924
3	7.8761	6.5576	5.4677	5.2244	4.4055	3.4747	4.8619	4.976
5	8.3101	6.9390	5.4759	5.5157	4.8933	4.6082	5.3918	5.281
7	8.7211	7.2503	7.4708	6.4177	5.2592	5.2263	7.7466	7.612
10	9.8088	7.8726	7.6583	8.3432	6.3007	6.2692	7.8579	7.986
15	10.6614	9.0252	9.3138	9.5982	7.8549	7.8364	9.6480	9.610

Table 5 Variances of the predicted S&P 500 stock indices on specific days.

Stock market	Dates (from 23rd July 2010 to 13th Aug. 2010)	Variances (from 23rd July 2010 to 13th Aug. 2010	Dates (from 29th Oct. 2010 to 19th Nov. 2010)	Variances (from 29th Oct. 2010 to 19th Nov. 2010)	Dates (from 9th Feb. 2011 to 3rd Mar. 2011)	Variances (from 9th Feb. 2011 to 3rd Mar. 2011)
		dvo(400:415) 1.0e−03*		dvo(470:485) 1.0e−03*		dvo(540:555) 1.0e−03*
S&P 500	23-07-2010	0.0399	29-10-2010	0.2379	09-02-2011	0.0245
	26-07-2010	0.0430	01-11-2010	0.3299	10-02-2011	0.0293
	27-07-2010	0.1077	02-11-2010	0.2383	11-02-2011	0.0270
	28-07-2010	0.1302	03-11-2010	0.1290	14-02-2011	0.0372
	29-07-2010	0.1674	04-11-2010	0.0229	15-02-2011	0.1269
	30-07-2010	0.1168	05-11-2010	0.0321	16-02-2011	0.2620
	02-08-2010	0.0089	08-11-2010	0.0830	17-02-2011	0.3279
	03-08-2010	0.0113	09-11-2010	0.0904	18-02-2011	0.2261
	04-08-2010	0.0101	10-11-2010	0.2536	22-02-2011	0.0789
	05-08-2010	0.2518	11-11-2010	0.2301	23-02-2011	0.0947
	06-08-2010	0.4284	12-11-2010	0.1159	24-02-2011	0.0917
	09-08-2010	0.5153	15-11-2010	0.1184	25-02-2011	0.1228
	10-08-2010	0.3136	16-11-2010	0.1187	28-02-2011	0.1240
	11-08-2010	0.0365	17-11-2010	0.1054	01-03-2011	0.1104
	12-08-2010	0.0520	18-11-2010	0.0630	02-03-2011	0.0877
	13-08-2010	0.0738	19-11-2010	0.0663	03-03-2011	0.0557

Fig. 2 shows a comparison of RMSE errors of S&P500 and BSE data sets during training of the two better performing FLANN models like the LEFLANN and CEFLANN using PSO, HMRPSO (adaptive version), DE/current to best methods, etc. From these figures it is quite clear that the RMSE converges to the lowest value for the two FLANN models using DE current to best variant during the training period in comparison to other evolutionary methods like the PSO, and HMRPSO. Comparing the performance of each FLANN using various evolutionary learning methods like PSO, HMRPSO, DE with different mutation strategies, it has been observed that each FLANN trained using DE/current to best mutation provides good result than other learning methods. Thus for producing the final forecast of the stock indices the DE current to best is used to reduce the prediction errors of the stock market indices. The RMSE and MAPE values for the two stock indices like the S&P500 and BSE obtained during training with different models and DE current to best evolutionary approach are shown in Fig. 3. From this figure it is seen that the both the CEFLANN and its recurrent version produce the least RMSE and MAPE errors during training in just 20 iterations. The RMSE and MAPE values for all the FLANN models using different evolutionary training paradigms are shown in Table 1, from which it can be concluded that the recurrent CEFLANN produce the least errors when trained with DE current to best technique. Again to compare the convergence speed of the evolutionary learning algorithms on the CEFLANN models the time elapsed during training is also specified in Table 2. From the table it is also clear that DE method has a faster convergence speed compared to the other two methods.

Figs. 4 and 5 show the actual and predicted closing price values of BSE and S&P500 data sets during testing with different FLANN models, and from this figure it is quite clear that the CEFLANN and recurrent adaptation produce accurate forecasts in comparison to all other FLANN models like the TRFLANN,LAGFLANN, CHFLANN, and LEGFLANN, etc. Also the LAG FLANN and TRFLANN show the worst forecast results for one day ahead stock closing price for both S&P500 and BSE stocks. A comparison of the evolutionary approaches with the well known gradient descent algorithm is also shown in Table 1 showing clearly the superior forecasting performance of the former in comparison to the gradient descent algorithms. Further Tables 3 and 4 show the forecast results from one day ahead to 15 days ahead, in which the RCEFLANN has a MAPE varies for nearly 1–3.15% in comparison with CEFLANN and LEGFLANN and LAGFLANN from nearly 1.15–6%, respectively (Fig. 7).

Tables 3 and 4 provide a comparison of MAPE values with other neural network models like the RBF neural network (Fig. 6), and Wavelet neural network (WNN), when trained with DE/current to best variant. From these tables it is clearly

Table 6 Variances of the predicted BSE stock indices on specific days.

Stock Market	Dates (from 07th Mar. 2007 to 29th Mar. 2007)	Variances (from 07th Mar. 2007 to 29th Mar. 2007)	Dates (from 19th Dec. 2007 to 11th Jan. 2008)	Variances (from 19th Dec. 2007 to 11th Jan. 2008)	Dates (from 29th Jul. 2008 to 20th Aug. 2008)	Variances (from 29th Jul. 2008 to 20th Aug. 2008)
		dvo(800:815) 1.0e−03*		dvo(1000:1015) 1.0e−03*		dvo(1150:1165) 1.0e−03*
	07-03-2007	0.0970	19-12-07	0.8897	29-07-08	0.3438
	08-03-2007	0.1210	20-12-07	0.6140	30-07-08	0.2138
BSE	09-03-2007	0.1382	24-12-07	0.0864	31-07-08	0.2537
	12-03-2007	0.1827	26-12-07	0.0073	01-08-08	0.1802
	13-03-2007	0.1359	27-12-07	0.0321	04-08-08	0.1679
	14-03-2007	0.0343	28-12-07	0.0269	05-08-08	0.1247
	15-03-2007	0.1121	31-12-07	0.0827	06-08-08	0.0861
	16-03-2007	0.3349	01-01-08	0.1456	07-08-08	0.0821
	19-03-2007	0.2909	02-01-08	0.1522	08-08-08	0.2340
	20-03-2007	0.1897	03-01-08	0.1464	11-08-08	0.3754
	21-03-2007	0.1107	04-01-08	0.0482	12-08-08	0.2549
	22-03-2007	0.1031	07-01-08	0.0436	13-08-08	0.1315
	23-03-2007	0.0688	08-01-08	0.0454	14-08-08	0.1107
	26-03-2007	0.2121	09-01-08	0.1863	18-08-08	0.0972
	28-03-2007	0.1963	10-01-08	0.4594	19-08-08	0.0783
	29-03-2007	0.1901	11-01-08	0.7512	20-08-08	0.0732

apparent that the RCFLANN is the simplest neural model which is quite robust and provides superior forecasting performance when trained with DE algorithm in comparison to the well established neural networks like the RBF and the wavelet neural network. After it is observed that the RCEFLANN performs the best in predicting the future stock indices, it is important to calculate the day to day variances and variances over a given time frame. Fig. 8 depicts the variances showing clearly the accuracy of the forecasting model, since their magnitudes are quite small over a large time frame (more than one year). Further the day to day variances shown in Tables 5 and 6 completely validate this argument.

5. Conclusion

Functional Link Artificial Neural Network is a single layer ANN structure, in which the hidden layers are eliminated by transforming the input pattern to a high dimensional space using a set of polynomial or trigonometric basis functions giving rise to a low complexity neural model. Different variants of FLANN can be modeled depending on the type of basis functions used in functional expansion. In this paper, the detailed architecture and mathematical modeling of different FLANNs with polynomial and trigonometric basis functions have been described to predict the stock market return. Further for improving the performance of different FLANNS, a number of evolutionary methods have been discussed to optimize their parameters. Comparing the performance of each FLANN using various evolutionary learning methods like PSO, HMRPSO, DE, etc. with different mutation strategies, it has been observed that each FLANN trained using DE/current to best mutation provides good result in comparison to other learning methods. Again the performance comparison of various evolutionary FLANN models to predict stock prices of Bombay Stock Exchange and Standard & Poor's 500 data set for 1 day, 7 days, 15 days and 30 days ahead, shows that performance of stock price prediction can significantly be enhanced using CEFLANN and recurrent CEFLANN models trained with DE/current to best method in comparison with the well known RBF and WNN models. The recurrent CEFLANN exhibits very low variances and can also be used to predict the volatility of stock indices and the trends for providing trading decisions.

References

Abdual-Salam, Mustafa E., Abdul-Kader, Hatem M., Abdel-Wahed, Waiel F., 2010. Comparative study between differential evolution and particle swarm optimization algorithms in training of feed-forward neural network for stock price prediction. In: 7th International Conference on Informatics and Systems (INFOS), pp. 1–8.

Ampolucci, P., Uncini, A., Piazza, F., Rao, B.D., 1999. On-line learning algorithms for locally recurrent neural networks. IEEE Trans. Neural Networks 10 (2), 253–270.

Andrade de Oliveira, Fagner, Nobre, Cristiane Neri, 2011. The use of artificial neural networks in the analysis and prediction of stock prices. In: IEEE International Conference on Systems, Man and Cybernetics (SMC), pp. 2151–2155.

Chakravarty, S., Dash, P.K., 2009. Forecasting stock market indices using hybrid network. In: World Congress on Nature & Biologically Inspired Computing (NaBIC), pp. 1225–1230.

Chandra, Patra Jagdis, Bornande, C., Meher, P.K., 2009. Laguerre neural network based smart sensors for wireless sensor networks. In: IEEE Conference on Instrumentation and Measurement Technology, pp. 832–837.

Das, K.K., Satapathy, J.K., 2011. Legendre neural network for nonlinear active noise cancellation with nonlinear secondary path. In: International Conference on Multimedia, Signal Processing and Communication Technologies, pp. 40–43.

Dehuri, S., Roy, R., Cho, S.B., Ghosh, A., 2012. An improved swarm optimized functional link artificial neural network (ISO-FLANN) for classification. J. Syst. Software 85, 1333–1345.

Gao, Hao, Xu, Wenbo, 2011. A new particle swarm algorithm and its globally convergent modifications. IEEE Trans. Syst. Man Cybernet. 41 (5), 1334–1351.

George, Nithin V., Panda, Ganapati, 2012. A reduced complexity adaptive Legendre neural network for nonlinear active noise control. In: 19th International Conference on Systems, Signals and Image Processing (IWSSIP), pp. 560–563.

Hatem, A.K., Mustafa, A.S., 2012. Evaluation of differential evaluation and particle swarm optimization algorithms at training of neural network for stock prediction. Int. Arab J. e-Technol. 2 (3), 145–151.

Jiang, L.L., Maskell, D.L., Patra, J.C., 2012. Chebyshev functional link neural network-based modeling and experimental verification for photovoltaic arrays. In: IEEE International Joint Conference on Neural Networks, pp. 1–8.

Kozarzewski, B., 2010. A neural network based time series forecasting system. In: 3rd International Conference on Human System Interaction, Rzeszow, Poland, pp. 59–62.

Lee, Chun-Teh, Chen, Yi-Ping, 2007. The efficacy of neural networks and simple technical indicators in predicting stock markets. In: International Conference on Convergence Information Technology, pp. 2292–2297.

Li, Mingyu, Liu, Jinting, Jiang, Yang, Feng, Wenjiang, 2012. Complex Chebyshev functional link neural network behavioral model for broadband wireless power amplifiers. IEEE Trans. Microw. Theory Tech. 60 (6), 1979–1989.

Lin, QianYu, Feng, ShaoRong, 2010. Stock market forecasting research based on neural network and pattern matching. In: International Conference on E-Business and E-Government, pp. 1940–1943.

Lu, Bo, 2011. A stock prediction method based on PSO and BP hybrid algorithm. In: International Conference on E-Business and E-Government, pp. 1–4.

Ma, Weimin, Wang, Yingying, Dong, Ningfang, 2010. Study on stock price prediction based on BP neural network. In: IEEE International conference on Emergency Management and Management Sciences (ICEMMS), pp. 57–60.

Majhi, B.D., Shalabi, H., Fathi, M., 2005. FLANN based forecasting of S&P 500 index. Inf. Technol. J. 4 (3), 289–292.

Mili, Faissal, Hamdi, Manel, 2012. A hybrid evolutionary functional link artificial neural network for data mining and classification. Int. J. Adv. Comput. Sci. Appl. 3 (8), 89–95.

Mishra, S.K., Panda, G., Meher, S., 2009. Chebyshev functional link artificial neural networks for denoising of image corrupted by salt and pepper noise. Int. J. Recent Trends Eng. 1 (1), 413–417.

Mohapatra, P., Raj, A., Patra, T.K., 2012. Indian stock market prediction using differential evolutionary neural network model. Int. J. Electron. Commun. Comput. Technol. 2 (4), 159–166.

Naeini, Mahdi Pakdaman, Taremian, Hamidreza, Hashemi, Homa Baradaran, 2010. Stock market value prediction using neural networks. In: International Conference on Computer Information Systems and Industrial Management Applications (CISIM), pp. 132–136.

Nanda, Santosh Kumar, Tripathy, Debi Prasad, Mahapatra, S.S., 2011. Application of legendre neural network for air quality prediction. In: The 5th PSU-UNS International Conference on Engineering and Technology (ICET-2011), pp. 267–272.

Pao, Y.-H., 1989. Adaptive Pattern Recognition and Neural Networks. Addison-Wesley Publishing Co., Inc., Reading, MA (US).

Patra, J.C., Bornand, C., 2010. Nonlinear dynamic system identification using Legendre neural network. In: International Joint Conference on Neural Networks (IJCNN), pp. 1–7.

Patra, J.C., Thanh, N.C., Meher, P.K., 2009. Computationally efficient FLANN-based intelligent stock price prediction system. In: Proceedings of International Joint Conference on Neural Networks, pp. 2431–2438.

Qin, A.K., Huang, V.L., Suganthan, P.N., 2008. Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Trans. Evol. Comput. 13 (2), 398–417.

Rodriguez, Nibaldo, 2009. Multiscale Legendre neural network for monthly anchovy catches forecasting. In: Third International Symposium on Intelligent Information Technology Application, pp. 598–601.

Sanjeevi, Sriram G., Naga Nikhila, A., Khan, Thaseem, Sumathi, G., 2011. Hybrid PSO-SA algorithm for training a neural network for classification. Int. J. Comput. Sci. Eng. Appl. 1 (6), 73–83.

Song, Ying, Chen, Zengqiang, Yuan, Zhuzhi, 2007. New chaotic PSO-based neural network predictive control for nonlinear process. IEEE Trans. Neural Networks 18 (2), 595–600.

Tahersima, Hanif, Tahersima, Mohammad Hossein, Fesharaki, Morteza, 2011. Forecasting stock exchange movements using neural networks: A case study. In: International Conference on Future Computer Sciences and Application, pp. 123–126.

Wang, Y., Cai, Z., Zhang, Q., 2011. Differential evolution with composite trial vector generation strategies and control parameters. IEEE Trans. Evol. Comput. 15 (1), 55–66.

Yogi, S, Subhashini, K.R., Satapathy, J.K., 2010. A PSO based functional link artificial neural network training algorithm for equalization of digital communication channels. In: 5th International Conference on Industrial and Information Systems, ICIIS, pp. 107–112.

Please cite this article in press as: Rout, A.K. et al., Forecasting financial time series using a low complexity recurrent neural network and evolutionary learning approach. Journal of King Saud University – Computer and Information Sciences (2015), http://dx.doi.org/10.1016/j.jksuci.2015.06.002