Int. j. inf. tecnol. (December 2021) 13:2277–2285
https://doi.org/10.1007/s41870-019-00382-y

ORIGINAL RESEARCH

A novel approach to predict stock market price using radial basis function network

Rajesh Kumar[1] · **Shefali Srivastava**[2] · **Anuli Dass**[3] · **Smriti Srivastava**[3]

Received: 28 March 2019 / Accepted: 1 November 2019 / Published online: 7 November 2019

Abstract In the financial sector, the sales price forecasting is a hot issue. Since the indices associated with the stock are nonlinear and are affected by various internal and external factors, they are very difficult to model and pose a difficult problem to be solved by the researchers. This paper is devoted in designing an intelligent prediction model based on the radial basis function network (RBFN). To tune its parameters a learning algorithm is developed using the back-propagation (BP) method. The performance of the proposed method is also compared with that of the multi-layered feed-forward neural network (MLFFNN) containing only single hidden layer and the results obtained from the simulation study indicate that the performance of RBFN is better as compared to the MLFFNN model.

Keywords Radial basis function network · Multilayer feed forward neural network · Stock market data · Prediction · Back-propagation

✉ Rajesh Kumar
rajeshmahindru23@gmail.com; rajesh.kumar23@thapar.edu

Shefali Srivastava
shefali9625@gmail.com

Anuli Dass
anulidass@gmail.com

Smriti Srivastava
smriti.nsit@gmail.com

[1] Department of Electrical and Instrumentation Engineering, Thapar Institute of Engineering and Technology (Deemed to be University), Patiala 147004, India

[2] Division of Information Technology, Netaji Subhas Institute of Technology, New Delhi 110078, India

[3] Division of Instrumentation and Control, Netaji Subhas Institute of Technology, New Delhi 110078, India

1 Introduction

The economic strength of any country can be judged from the stock market growth in that country. To have a steady financial market the stability of stock market is very crucial. In the fields of finance, mathematics, economy, engineering the prediction of stock market price is a central issue [1]. The challenge is to predict what should be the right time for the investors to buy or sell. The risk associated with the stock market is high since investors invest huge amount of their money in it. This demand for the development of accurate methods that can aid in predicting the stock prices. The procedure to develop such methods is not straightforward since the indices associated with the stock market prices are highly nonlinear, time varying, dynamic, chaotic and nonparametric in nature [2]. These prices are also affected by some of the qualitative macro-economic factors such as political events, investor's expectations, prices movement of other stocks, psychology of investors etc. [2]. The interest of the investor is to have some methods by which he can predict the stock trend, price or its index. They are dependent on it as these methods provide guard against the market risks. The government requires these methods for monitoring the fluctuations in the market. Considerable effort has been applied by the research community in building various algorithms and models and has achieved good results in predicting the prices of the stock. Basically, the stock market forecasting comes under two categories: in the first category the models are based in the theory of statistics. The second category involves the application of intelligent techniques such as artificial neural network (ANN) [3], particle swarm optimization (PSO) [4], and support vector machine (SVM) [5] etc. The results reported in the literature indicate that these intelligent techniques are much better than the

conventional techniques especially in short term forecasting [6]. The stock price also shows dramatic variation in response to various complex factors, hence these intelligent techniques require continuous improvement in their part so that they can handle such variations [7]. Neural networks posses the ability to approximate the intrinsic mathematical relationship existing between the given set of input–output nonlinear data. In this paper we have used two variants of neural networks: RBFN and MLFFNN. Both are nonlinear in nature. The difference between the two is with respect to the functioning of hidden neurons. The other difference is that RBFN contains only one hidden layer whereas MLFFNN may contain more than one hidden layer. Also, the weights connecting the external inputs to the hidden neurons in RBFN are of unity value which is not the case in MLFFNN. Both these techniques along with other intelligent techniques such as fuzzy systems has been successfully applied in various areas [8–11].

1.1 Contributions of the paper

1. Proposed RBFN based stock price prediction model.
2. Compared the performance of RBFN based prediction model with that of the MLFFNN based model.
3. Recursive parameter optimization equations are obtained by implementing the Back-propagation algorithm.

The organization of the remaining paper is described as follows: Sect. 2 contains the discussion on the related research. Section 3 contains the mathematical description of the RBFN model and its defining equations. In Sect. 4 recursive update equations are obtained by implementing the back propagation algorithm. Section 5 contains the simulation results. Here, the performance of RBFN and MLFFNN based prediction models are tested and compared. In Sect. 6 the conclusion of the paper is given.

2 Related work

In Ref. [12], the authors have applied and compared the performances of the multi-layer perceptron (MLP), dynamic artificial neural network (DANN) and the hybrid neural networks for predicting the market values. The performance is evaluated using the two indicators: mean square error (MSE) and mean absolute deviate (MAD).

In Ref. [13], the authors have applied Probabilistic Neural Network (PNN) to model and predict the direction of return on market index of the Taiwan stock exchange. They have used the historical data to train the network. Furthermore, they have also compared the performance of the PNN with that of generalized methods of moments-Kalman filter and random walk forecasting models and found it superior.

In Ref. [14], the authors have considered both qualitative (political effect) and quantitative aspects in developing the prediction model. The model consists of genetic algorithm based fuzzy neural network (GAFNN) which is used to develop the inference rules for the knowledge based system. This system measures the qualitative factor. Further, the proposed method also integrates the ANN to handle the quantitative factors. The proposed method is the successfully applied on the Taiwan stock market.

In Ref. [15], the authors have proposed an improved model for C-fuzzy decision trees. The proposed model is based on the fuzzy time series. The improvements introduced in the proposed method involves the inclusion of new stopping criteria (which improved the computational time) and the use of fuzzy clustering with weight distance which is calculated using the information gain. The proposed method is also compared with the conventional models and its performance is found to be superior.

In Ref. [16], the authors have applied the adaptive neuro-fuzzy inference system (ANFIS) based controller to control the stock market process model. The proposed model involves the use of a membership function of the Gaussian shaped instead of bell and triangular since the Gaussian provided the better accuracy than the other two. Further, different combinations of 15 past stock prices are evaluated and the best set among them is then applied as the inputs to the ANFIS model. The proposed method is trained and validated using the well-developed stock markets-the Athens and the New York Stock Exchange (NYSE). The performance of the proposed model is found to be better than the other considered conventional methods.

In Ref. [17], the authors have used several ANN's to forecast daily NASDAQ stock exchange rate. To train the networks, they have applied Back-Propagation algorithm. The training and testing data set consists of 70 and 29 days of samples respectively. The inputs applied to the networks consist of short-term historical stock prices as well as the day of week.

In Ref. [18], the authors have applied the ANN to predict the return price of the Japanese Nikkei 225 index. Initially they have applied back-propagation algorithm to tune the weights of the ANN. The proposed algorithm suffered from the local convergence problem. So to avoid it and to further improve the ANN prediction accuracy the authors have replaced Back-Propagation with the hybrid combination of GA and simulated annealing (SA).

In Ref. [19], the authors have explored the techniques of deep learning for the prediction of stock price. In particular, they have exploited the potential of deep learning method in extracting raw data (in large amount) features

and do not require any previous knowledge regarding the predictors.

In Ref. [20], the authors have applied the combination of genetic network programming (GNP), MLP and time series models to predict the stock return. They have used evolutionary algorithms in parallel to ARMA in order to improve the accuracy of the daily return prediction. The MLP is used to classify the data and also the various time series models to forecast the stock return. The proposed method is tested on both turbulent markets and trending markets.

In Ref. [21], the authors have proposed a novel architecture of generative adversarial network (GAN) which consists of two components: MLP (used as a discriminator) and the long short-term memory (LSTM) (applied as an generator) for predicting the stock's closing price. The proposed method has given much better performance than the other models based deep learning and machine learning in the prediction of closing price of stock.

3 Mathematical structure of RBFN model

Figure 1 shows the structural view of the RBFN model. In this figure, vector X denotes an input vector containing m number of signals, that is $X = (x_1, x_2, \ldots, x_m)$. The neurons implementing the hidden layer are also known as radial centers and in this paper we have used Gaussian radial basis function as an activation for them [22]. There also exists few more choices for the activation function but among then the Gaussian function is more intuitive in a sense that it produces appreciable output if the input vector lies in its vicinity and vice-versa. Thus, each radial center represents one of the input vector and any ith radial center is denoted by $C_i(k)$.

In case of RBFN, the input to hidden layer connections are non-adjustable and are kept constant to a unity value

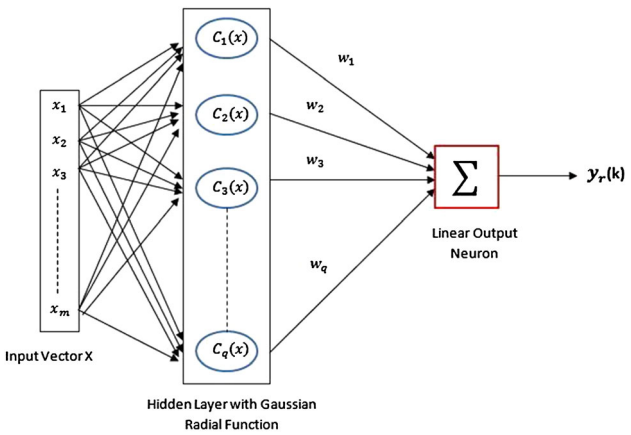

Fig. 1 Structure of RBFN

[23]. For the prediction of SISO and MISO type of time series the RBFN contains only single neuron in its output layer. Its induced field is nothing but the weighted sum of the Gaussian function outputs of the radial centers. The output layer weights connecting the radial centers to the output layer neuron is denoted by the vector $W = [w_1, w_2, w_3, \ldots, w_q)$. The total count of radial centers is denoted by q where the $q \ll m$ to avoid over-fitting problem [24]. The RBFN output can be obtained as follows:

$$y_r(k) = \sum_{i=1}^{q} \psi_i(k) W_i(k). \tag{1}$$

Symbol $y_r(k)$ denotes the RBFN output at any kth time instant. The output of any $C_i(k)$ radial center is given by the following mathematical equation

$$\psi_i(k) = \psi \|X(k) - C_i(k)\|, \tag{2}$$

where the Gaussian function includes the calculation of Euclidean distance present between the input vector and the radial center $C_i(k)$ and is denoted by the following notation $\|X(k) - C_i(k)\|$. The Gaussian function, $\psi(.)$, has the following definition [24]

$$\psi(t) = \exp\left(\frac{-t^2}{2\sigma^2}\right) \tag{3}$$

where $t = \|X(k) - C_i(k)\|$ and the vector $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_q)$ is used to represent the width associated with each of the radial center.

4 Update rules for RBFN based prediction model

Figure 2 shown the implementation of RBFN based prediction model. The parameters associated with the RBFN will be sequentially updated during the application of learning algorithm. Their updation will be controlled by the recursive equations that are obtained using the BP method. In this method we first define a cost/objective function whose value needs to be minimized by adjusting the values of the RBFN parameters. Mean square error (MSE) is

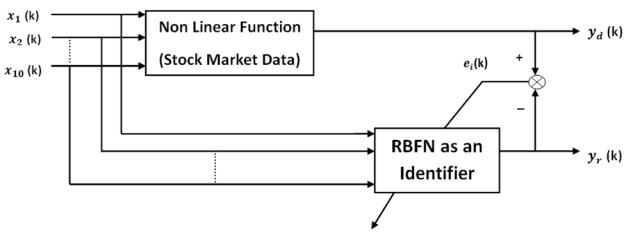

Fig. 2 General prediction model based on RBFN

considered as an objective function. Let this objective function is denoted by $E_o(k)$ and it is defined as follows:

$$E_o(k) = \frac{1}{2}(y_d(k) - y_r(k))^2, \tag{4}$$

where the actual stock price value is denoted by $y_d(k)$ and the one predicted by RBFN is $y_r(k)$. Now taking partial derivative of $E_o(k)$ one by one with respect to each of the RBFN parameters we will obtain their corresponding update equations. Let us first find update equation for $C_{ij}(k)$ where $i = 1$ to m, $j = 1$ to q. Applying the chain rule we will get

$$\frac{\partial E_o(k)}{\partial C_{ij}(k)} = \frac{\partial E_o(k)}{\partial y_r(k)} \times \frac{\partial y_r(k)}{\partial \psi_j(k)} \times \frac{\partial \psi_j(k)}{\partial C_{ij}(k)} \tag{5}$$

or

$$\frac{\partial E_o(k)}{\partial C_{ij}(k)} = \frac{\partial E_o(k)}{\partial y_r(k)} \times \frac{\partial y_r(k)}{\partial \psi_j(k)} \times \frac{\partial \psi_j(k)}{\partial t_j(k)} \times \frac{\partial t_j(k)}{\partial C_{ij}(k)}. \tag{6}$$

After evaluating the expression for the individual partial derivatives in Eq. 6 and then adding momentum term (for expediting the learning of RBFN) we will get the following update equation for the radial centers

$$C_{ij}(k+1) = C_{ij}(k) + \Delta C_{ij} + \alpha \Delta C_{ij}(k-1), \tag{7}$$

here $\Delta C_{ij} = \eta e_i(k) W_j(k) \frac{\psi_j(k)}{\sigma_j^2(k)} (X_j(k) - C_{ij}(k))$ and η represents the learning rate parameter and its value is set anywhere in the interval (0, 1) and $\alpha \Delta C_{ij}(k-1)$ is the momentum term added in the update equation where, α, is known as a momentum constant. Its value is also selected from the 0 and 1 range. By doing the similar analysis we can obtain the update equations for the output layer weights as follows

$$\frac{\partial E_o(k)}{\partial W_j(k)} = \left(\frac{\partial E_o(k)}{\partial y_r(k)} \times \frac{\partial y_r(k)}{\partial W_j(k)} \right). \tag{8}$$

Hence each weight element present in the following vector $W = [w_1, w_2, w_3, \ldots, w_q]$ will be updated as follows

$$W_j(k+1) = W_j(k) + \Delta W_j + \alpha \Delta W_j(k-1), \tag{9}$$

where $\Delta w_j = \eta e_i(k) \psi_j(k)$. Finally, the update equations for the widths associated with the radial centers are computed as follows

$$\frac{\partial E_o(k)}{\partial \sigma_j(k)} = \left(\frac{\partial E_o(k)}{\partial y_r(k)} \times \frac{\partial y_r(k)}{\partial \psi_j(k)} \times \frac{\partial \psi_j(k)}{\partial \sigma_j(k)} \right). \tag{10}$$

So, every width element in the vector $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_q)$ will be updated as follows

$$\sigma_j(k+1) = \sigma_j(k) + \Delta\sigma + \alpha \Delta\sigma(k-1), \tag{11}$$

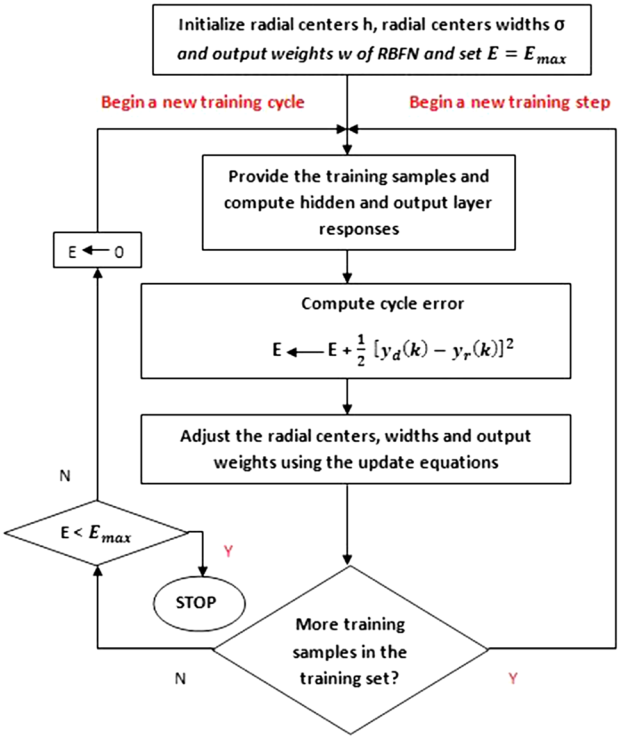

Fig. 3 Steps to train the RBFN prediction model

where $\Delta\sigma = \eta e_i(k) W_j(k) \frac{\psi_j(k) t_j^2(k)}{\sigma_j^3(k)}$. A flowchart is shown in Fig. 3 which describes the training steps involved in the RBFN prediction model.

5 Simulation results

In this section, performance of RBFN model is tested and compared with that of MLFFNN prediction model. In both these models, same number of hidden nodes is taken which is equal to 20. The value of η and α are taken to be 0.025 and 0.067 respectively. Total number of inputs to prediction models are 10 and are denoted by input vector $X = \{x_1(k), x_2(k), \ldots, x_{10}(k)\}$. Total 100 input–output samples are used for training purpose [25]. To evaluate the performances of the models we have used average mean square error (AMSE) as a performance indicator and is defined as follows:

$$AMSE = \frac{1}{N} \sum_{i=1}^{N} (y_d(k) - y_r(k))^2, \tag{12}$$

where N represents the total number of training samples. Further, to get effective results from the prediction models the experimental data requires some pre-processing. In our paper we have performed the normalization of the input–output data (in the [0, 1] range) using the following formula:

Table 1 Input–output data samples [25]

x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	yd
0.239	0.227	8.307	2.82	− 0.498	0.87	− 3.281	− 2.562	− 2.523	− 3.255	34.512
0.279	0.246	16.771	5.222	− 0.25	2.588	1.696	− 4.568	0.848	− 0.502	24.39
0.319	0.272	21.253	5.819	− 0.418	0.841	− 0.834	− 3.255	4.352	0.756	24.187
0.346	0.27	19.867	4.654	− 0.42	− 0.834	− 0.841	− 0.502	2.588	0.338	17.662
0.351	0.303	19.447	7.848	0.084	3.364	− 2.545	0.756	3.364	3.626	− 2.441
0.269	0.268	14.32	5.809	0.253	− 1.627	0.87	0.338	0.834	1.682	7.444
0.25	0.267	11.157	5.527	0.084	0	2.588	3.626	1.682	1.597	2.481
0.256	0.336	12.787	11.263	0.756	5.79	0.841	1.682	4.068	6.672	− 11.728
0.296	0.379	12.453	14.309	1.168	3.127	− 0.834	1.597	9.098	8.739	− 12.889
0.326	0.4	12.088	16.444	1.649	2.274	3.364	6.672	11.58	9.408	− 20.015
0.354	0.35	13.587	14.317	1.379	− 1.483	− 1.687	8.739	3.909	6.32	− 10.534
0.395	0.315	15.971	14.815	1.2	0.752	0	9.408	1.516	5.85	− 17.924
0.388	0.295	19.283	12.767	0.949	− 1.494	5.79	6.32	− 2.224	3.289	− 20.47
0.433	0.294	22.512	16.698	1.41	3.791	3.127	5.85	3.01	5.714	− 16.801
0.466	0.35	28.465	19.689	1.39	2.922	2.274	3.289	5.228	7.311	− 16.324
0.466	0.34	26.015	21.823	1.752	2.129	− 1.483	5.714	9.089	7.71	− 20.848
0.391	0.404	19.058	26.391	2.171	4.17	0.752	7.311	9.496	9.817	− 26.017
0.346	0.478	13.125	30.825	2.051	4.003	− 1.494	7.71	10.646	11.917	− 27.582
0.348	0.401	11.815	29.467	1.651	− 0.641	3.791	9.817	7.644	9.393	− 22.595
0.357	0.427	10.504	28.084	1.342	− 0.646	2.922	11.917	2.668	7.247	− 23.392
0.357	0.361	11.021	22.99	0.906	− 5.198	2.129	9.393	− 6.414	0.76	− 19.191
0.314	0.378	10.674	22.186	0.967	1.371	4.17	7.247	− 4.519	1.163	− 24.341
0.292	0.413	13.067	25.798	1.436	3.381	4.003	0.76	− 0.65	3.102	− 26.161
0.297	0.457	11.83	26.044	1.146	0.654	− 0.641	1.163	5.483	2.6	− 22.092
0.266	0.455	9.733	28.734	1.133	2.599	− 0.646	3.102	6.761	4.087	− 21.533
0.255	0.408	5.862	25.776	0.725	− 1.9	− 5.198	2.6	1.308	1.374	− 20.013
0.264	0.415	7.374	28.491	0.589	2.582	1.371	4.087	3.249	3.383	− 23.914
0.249	0.404	6.214	25.557	0	− 1.888	3.381	1.374	− 1.267	1.431	− 18.602
0.243	0.394	5.066	24.262	0	− 0.641	0.654	3.383	0	0.781	− 20.658
0.223	0.41	2.165	24.554	0.13	0.646	2.599	1.431	− 1.888	1.299	− 19.243
0.227	0.364	2.82	24.102	0.65	0	− 1.9	0.781	0	0.646	− 19.243
0.246	0.318	5.222	17.36	0	− 5.131	2.582	1.299	− 4.519	− 4.519	− 14.875
0.272	0.238	5.819	17.873	− 0.258	0.676	− 1.888	0.646	− 4.49	− 3.625	− 16.79
0.27	0.176	4.654	10.554	− 0.908	− 6.044	− 0.641	− 4.519	− 10.263	− 8.622	− 13.581
0.303	− 0.026	7.848	− 5.226	− 2.482	− 14.296	0.646	− 3.625	− 18.932	− 19.692	0
0.268	0.07	5.809	2.607	− 1.674	8.34	0	− 8.622	− 12.76	− 11.512	− 10.778
0.267	0.053	5.527	− 2.184	− 2.384	− 4.619	− 5.131	− 19.692	− 11.437	− 13.538	− 4.036
0.336	− 0.035	11.263	− 10.837	− 3.001	− 8.878	0.676	− 11.512	− 5.838	− 18.777	3.543
0.379	− 0.044	14.309	− 9.218	− 2.878	1.771	− 6.044	− 13.538	− 11.547	− 14.889	0
0.4	− 0.097	16.444	− 14.666	− 3.556	− 6.092	− 14.296	− 18.777	− 12.914	− 17.127	5.561
0.35	0	14.317	− 5.967	− 2.842	10.195	8.34	− 14.889	5.314	− 6.008	− 3.364
0.315	− 0.07	14.815	− 13.023	− 3.004	− 7.569	− 4.619	− 17.127	− 4.352	− 10.432	5.459
0.295	− 0.114	12.767	− 16.885	− 3.586	− 4.55	− 8.878	− 6.008	− 2.78	− 11.327	16.206
0.294	− 0.026	16.698	− 9.731	− 2.198	8.58	1.771	− 10.432	− 4.205	− 1.556	5.268
0.35	0.026	19.689	− 6.585	− 0.173	3.512	− 6.092	− 11.327	7.279	2.078	2.554
0.34	0.053	21.823	− 9.802	− 1.385	− 3.393	10.195	− 1.556	8.58	0	0.878
0.404	0.035	26.391	− 12.209	− 1.141	− 2.634	− 7.569	2.078	− 2.634	− 1.51	2.705
0.478	0.018	30.825	− 10.641	0	1.803	− 4.55	0	− 4.241	0.266	4.429

Table 1 continued

x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	yd
0.401	0.105	29.467	− 5.201	0.444	6.2	8.58	− 1.51	5.268	6.012	− 3.336
0.427	0.123	28.084	− 6.897	0.884	− 1.668	3.512	0.266	6.312	3.33	1.696
0.361	0.088	20.99	− 6.978	− 0.088	0	− 3.393	6.012	4.429	3.421	1.696
0.378	0.061	22.186	− 11.766	0.175	− 5.089	− 2.634	3.33	− 6.672	− 2.014	10.724
0.413	0.017	25.798	− 10.993	0.7	0.894	1.803	3.421	− 4.241	− 1.826	6.2
0.457	0.061	26.044	− 5.533	0.522	6.2	6.2	− 2.014	1.696	3.72	0
0.455	0.105	28.734	− 2.484	0.519	3.336	− 1.668	− 1.826	10.724	6.626	− 0.807
0.408	0.149	25.776	− 2.628	0.861	0	0	3.72	9.743	5.717	− 6.457
0.415	0.113	28.491	− 5.094	0.853	− 2.421	− 5.089	6.626	0.834	2.284	− 6.617
0.404	0.174	25.557	− 0.557	1.184	4.963	0.894	5.717	2.421	6.104	− 9.456
0.394	0.148	24.262	− 3.834	0.251	− 3.152	6.2	2.284	− 0.807	2.502	− 6.509
0.41	0.174	24.554	− 1.658	0.667	2.441	3.336	6.104	4.136	4.308	− 3.971
0.364	0.13	24.102	− 1.786	0.663	0	0	2.502	− 0.788	3.621	5.56
0.318	0.052	17.36	− 1.837	1.152	0	− 2.421	4.308	2.441	2.441	3.177
0.238	− 0.009	17.873	− 3.388	0.895	− 1.589	4.963	3.621	− 1.589	− 0.081	4.843
0.176	− 0.026	10.554	− 5.702	0.081	− 2.421	− 3.152	2.441	− 3.971	− 2.579	3.308
− 0.026	− 0.035	− 5.226	− 6.45	− 0.322	− 0.827	2.441	− 0.081	− 4.766	− 3.072	2.502
0.07	− 0.026	2.607	− 9.547	− 0.647	− 3.336	0	− 2.579	− 6.457	− 5.696	5.177
0.053	0.026	− 2.184	− 7.23	− 0.163	2.588	0	− 3.072	− 1.654	− 3.097	5.887
− 0.035	− 0.009	− 10.837	− 8.783	− 0.815	− 1.682	− 1.589	− 5.696	− 2.502	− 3.944	11.976
− 0.004	− 0.026	− 9.218	− 10.32	− 0.657	− 1.711	− 2.421	− 3.097	− 0.863	− 4.963	10.444
− 0.097	− 0.035	− 14.666	− 11.07	− 0.993	− 0.87	− 0.827	− 3.944	− 4.205	− 4.845	10.532
0	− 0.043	− 5.967	− 10.25	− 0.919	0.878	− 3.336	− 4.963	− 1.711	− 3.12	6.962
− 0.07	− 0.061	− 13.023	− 9.414	− 0.843	0.87	2.588	− 4.845	0.87	− 1.446	5.177
− 0.114	− 0.043	− 16.885	− 4.683	− 0.17	5.177	− 1.682	− 3.12	7.024	3.833	0.82
− 0.026	− 0.096	− 9.731	− 6.157	− 0.085	− 1.641	− 1.711	− 1.446	4.352	2.217	1.668
0.026	− 0.148	− 6.585	− 5.234	0.085	0.834	− 0.87	3.833	4.314	2.981	1.654
0.053	− 0.183	− 9.802	− 9.772	− 0.085	− 4.963	0.878	2.217	− 5.742	− 2.046	6.962
0.035	− 0.131	− 12.209	− 10.44	− 0.426	− 0.87	0.87	2.981	− 5.004	− 2.483	7.024
0.018	− 0.044	− 10.641	− 7.255	0.086	3.512	5.177	− 2.046	− 2.481	0.855	5.089
0.105	− 0.052	− 5.201	− 8.78	0.085	− 1.696	− 1.641	− 2.483	0.87	− 0.94	8.628
0.123	− 0.009	− 6.897	− 5.624	0.513	3.451	0.834	0.855	5.268	1.956	5.004
0.088	− 0.017	− 6.978	− 5.607	0.425	0	− 4.963	− 0.94	1.696	1.524	2.502
0.061	0.017	− 11.766	− 2.475	0.677	3.336	− 0.87	1.956	6.902	4.205	− 0.807
0.017	− 0.044	− 10.993	− 5.582	− 0.168	− 3.228	3.512	1.524	0	1.011	1.668
0.061	− 0.035	− 5.533	− 5.549	0	0	− 1.696	4.205	0	1.011	1.668
0.105	0.009	− 2.484	− 3.19	0.168	2.502	3.451	1.011	− 0.807	3.364	1.627
0.149	− 0.017	− 2.628	− 8.692	0.084	− 5.696	0	1.011	− 3.336	− 2.605	6.04
0.113	− 0.061	− 5.094	− 11.001	− 0.084	− 2.588	3.336	3.364	− 5.838	− 5.0046	10.629
0.174	− 0.035	− 0.0557	− 9.393	− 0.252	1.771	− 3.228	− 2.605	− 6.509	− 3.12	7.833
0.148	− 0.026	− 3.834	− 9.369	− 0.084	0	0	− 5.046	− 0.863	− 3.038	10.444
0.174	0.053	− 1.658	− 4.686	0.084	5.222	2.502	− 3.12	7.086	1.939	4.963
0.13	0.0149	− 1.786	4.618	1.096	9.926	− 5.696	− 3.038	15.666	10.842	− 5.267
0.052	0.096	− 1.837	2.158	0.5	− 2.257	− 2.588	1.939	13.055	7.801	− 3.849
− 0.009	0.087	− 3.388	2.069	0.83	0	1.7711	0.842	7.444	6.914	− 3.849
− 0.026	0.052	− 5.702	− 1.911	0.412	− 3.849	0	7.801	− 6.02	2.377	3.203
− 0.035	0	− 6.45	− 3.482	0	− 1.601	5.222	6.914	− 5.389	0.738	4.882
− 0.026	0.009	− 9.547	− 4.275	0.492	− 0.814	9.926	2.377	− 6.159	− 0.571	6.563

Table 1 continued

x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	yd
0.026	0.044	− 7.23	− 1.177	1.06	3.281	− 2.257	0.738	0.801	1.614	0.794
− 0.009	0.026	− 8.783	2.72	1.291	3.971	0	− 0.571	6.509	4.303	− 1.528
− 0.026	− 0.044	− 10.32	− 0.375	0.956	− 3.056	− 3.849	1.614	4.102	0.158	4.728
− 0.035	− 0.079	− 11.07	− 1.082	0.395	− 0.788	− 1.601	4.303	0	− 1.022	4.766

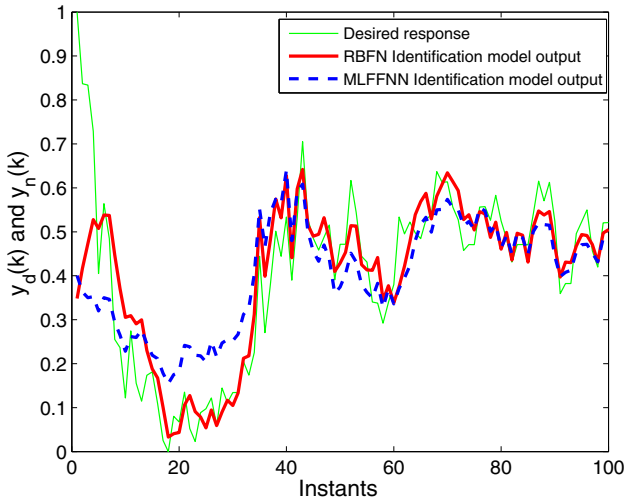

Fig. 4 Prediction models response during initial phase of the training

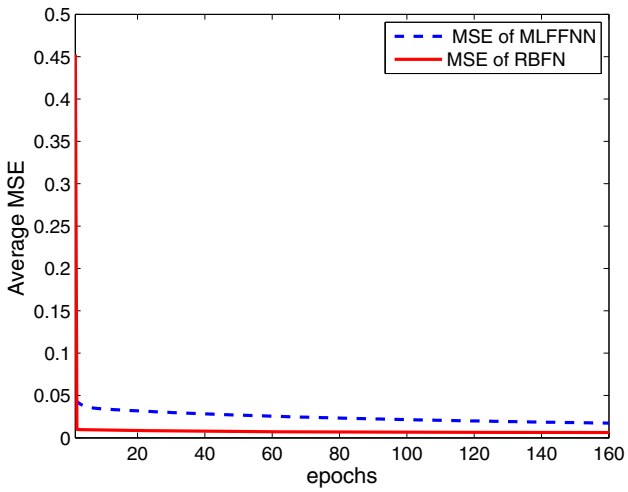

Fig. 5 Average MSE obtained during initial phase of training

$$X_{Nor} = \frac{X - X_{min}}{X_{max} - X_{min}}, \tag{13}$$

where X_{Nor} represents the normalized value of any variable X and X_{max} and X_{min} represents its maximum and minimum value respectively. The data set is shown in the Table 1.

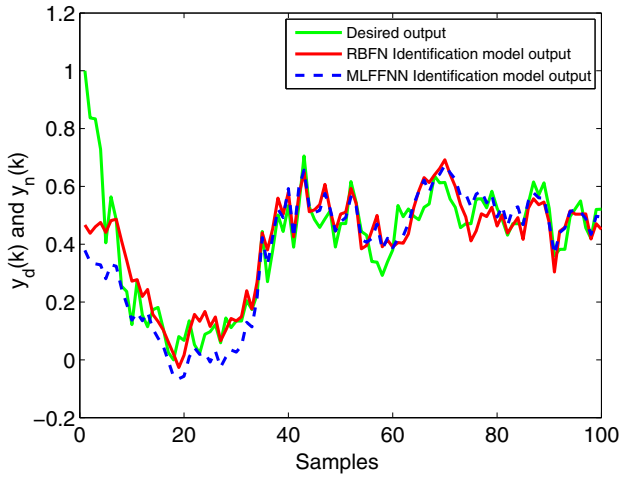

Fig. 6 Prediction models response after sufficient amount of training

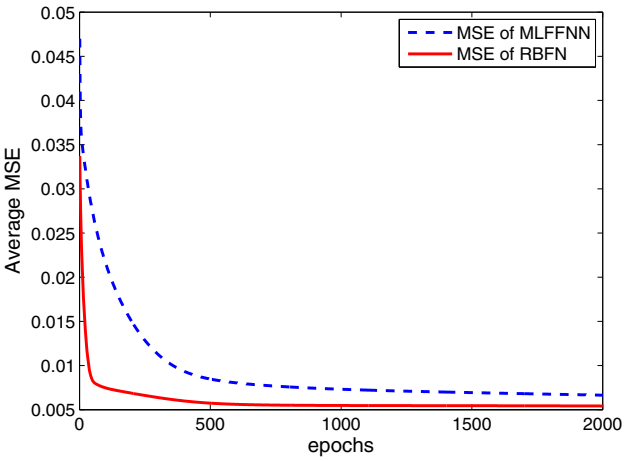

Fig. 7 AMSE obtained during the training

The training was performed in an epoch type manner. In one epoch 100 training samples are presented and the average error obtained for the epoch is saved. Total of 2000 epochs are run for training the RBFN and MLFFNN based prediction models. At the end of the final epoch their prediction curves are plotted and compared. The description of 10 inputs given to the prediction models can be

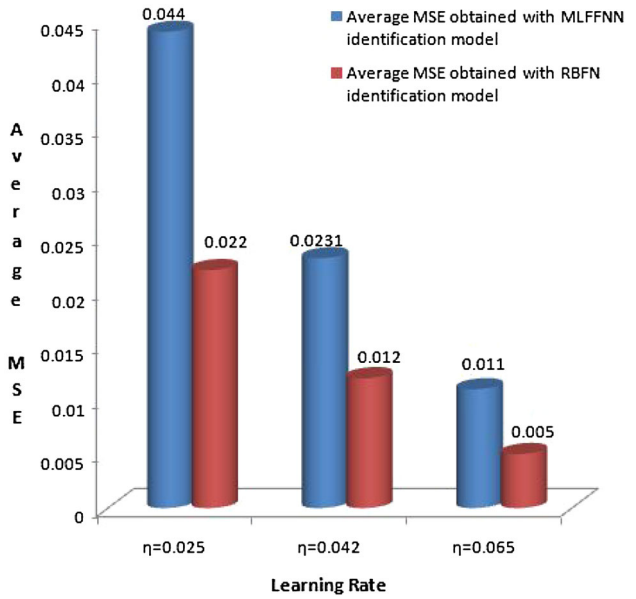

Fig. 8 AMSE obtained during the training with different values of learning rate

found in Ref. [25]. Figure 4 shows the initial stage of training response of prediction models. The response were plotted after the 450th epoch. Note that $y_d(k)$ represents actual stock price and $y_n(k)$ represents stock price predicted by the prediction models. It can be seen that RBFN output is closer to that of desired response as compared to the response obtained with MLFFNN. This shows the superior learning ability of RBFN over MLFFNN.

The corresponding average MSE obtained is shown in Fig. 5. From the plot it can be again seen that as the training progressed, the error reduces at a much faster rate in case of RBFN as compared to that obtained in MLFFNN model training phase.

After a sufficient training was done which was continued till 2000 epochs, the responses of RBFN and MLFFNN prediction models after the last epoch is shown in Fig. 6.

From the figure it can be seen that RBFN prediction model response is closer to the desired values as compared to prediction of stock price obtained with MLFFNN based model. This is further proved from the plot of AMSE which is shown in Fig. 7.

It can be seen easily from the MSE plot that error in case of RBFN decreases much rapidly as the training progressed when compared to error obtained with MLFFNN. This shows the superiority of RBFN model over the conventional MLFFNN model. The average MSE obtained when different values of learning rate are considered (keeping momentum constant to a fixed value) is shown in Fig. 8. From the bar graph it can be easily seen that we get lesser value of MSE in case of RBFN prediction model as compared to that obtained with MLFFNN model.

6 Conclusion

This paper presents the comparative analysis of prediction capabilities of two variants of neural networks: RBFN and MLFFNN. Simulation results indicate that the performance of RBFN based prediction model is found to be better than that obtained with MLFFNN model. A highly nonlinear stock market price data is used for the prediction purpose. The parameter update equations are derived using back-propagation algorithm. The prediction models and AMSE plots responses depicts that RBFN have performed much better than that of MLFFNN.

References

1. Budhani N, Jha C, Budhani SK (2014) Prediction of stock market using artificial neural network. In: 2014 International conference on soft computing techniques for engineering and technology (ICSCTET). IEEE, pp 1–8
2. Rout M, Majhi B, Mohapatra UM, Mahapatra R (2012) Stock indices prediction using radial basis function neural network. In: International conference on swarm, evolutionary, and memetic computing. Springer, Berlin, pp 285–293
3. Kumar R, Srivastava S, Gupta J, Mohindru A (2018) Diagonal recurrent neural network based identification of nonlinear dynamical systems with lyapunov stability based adaptive learning rates. Neurocomputing 287:102–117
4. Majhi R, Panda G, Sahoo G, Panda A, Choubey A (2008) Prediction of s&p 500 and DJIA stock indices using particle swarm optimization technique. In: 2008 IEEE world congress on computational intelligence. IEEE, pp 1276–1282
5. Huang W, Nakamori Y, Wang S-Y (2005) Forecasting stock market movement direction with support vector machine. Comput Oper Res 32(10):2513–2522
6. Hill T, O'Connor M, Remus W (1996) Neural network models for time series forecasts. Manag Sci 42(7):1082–1092
7. Shen W, Guo X, Wu C, Wu D (2011) Forecasting stock indices using radial basis function neural networks optimized by artificial fish swarm algorithm. Knowl Based Syst 24(3):378–385
8. Ghose U, Bisht U et al (2019) Tailored feedforward artificial neural network based link prediction. Int J Inf Technol, pp 1–9
9. Koul N, Manvi SS (2019) A proposed model for neural machine translation of Sanskrit into English. Int J Inf Technol, pp 1–7
10. Chhabra S, Singh H (2019) Optimizing design parameters of fuzzy model based COCOMO using genetic algorithms. Int J Inf Technol, pp 1–11
11. Solanki A, Pandey S (2019) Music instrument recognition using deep convolutional neural networks. Int J Inf Technol, pp 1–10
12. Guresen E, Kayakutlu G, Daim TU (2011) Using artificial neural network models in stock market index prediction. Expert Syst Appl 38(8):10Ã‘Â 389–10Ã‘Â 397
13. Chen A-S, Leung MT, Daouk H (2003) Application of neural networks to an emerging financial market: forecasting and trading the Taiwan Stock Index. Comput Oper Res 30(6):901–923
14. Kuo RJ, Chen C, Hwang Y (2001) An intelligent stock trading decision support system through integration of genetic algorithm based fuzzy neural network and artificial neural network. Fuzzy Sets Syst 118(1):21–45

15. Qiu W, Liu X, Wang L (2012) Forecasting shanghai composite index based on fuzzy time series and improved c-fuzzy decision trees. Expert Syst Appl 39(9):7680–7689
16. Atsalakis GS, Valavanis KP (2009) Forecasting stock market short-term trends using a neuro-fuzzy based methodology. Expert Syst Appl 36(7):10696–10707
17. Moghaddam AH, Moghaddam MH, Esfandyari M (2016) Stock market index prediction using artificial neural network. J Econ Finance Admin Sci 21(41):89–93
18. Qiu M, Song Y, Akagi F (2016) Application of artificial neural network for the prediction of stock market returns: the case of the Japanese stock market. Chaos Solitons Fractals 85:1–7
19. Chong E, Han C, Park FC (2017) Deep learning networks for stock market analysis and prediction: methodology, data representations, and case studies. Expert Syst Appl 83:187–205
20. Ramezanian R, Peymanfar A, Ebrahimi SB (2019) An integrated framework of genetic network programming and multi-layer perceptron neural network for prediction of daily stock return: an application in Tehran stock exchange market. Appl Soft Comput, p 105551
21. Zhang K, Zhong G, Dong J, Wang S, Wang Y (2019) Stock market prediction based on generative adversarial network. Procedia Comput Sci 147:400–406
22. Du K-L, Swamy M (2014) Radial basis function networks. In: Neural networks and statistical learning. Springer, pp 299–335
23. Kitayama S, Srirat J, Arakawa M, Yamazaki K (2013) Sequential approximate multi-objective optimization using radial basis function network. Struct Multidiscip Optim 48(3):501–515
24. Behera L, Kar I (2010) Intelligent systems and control principles and applications. Oxford University Press, Oxford
25. Sugeno M, Yasukawa T (1993) A fuzzy-logic-based approach to qualitative modeling. IEEE Trans Fuzzy Syst 1(1):7–31