Efficient Stock Price Prediction using a Self Evolving Recurrent Neuro Fuzzy Inference System Optimized through a Modified Differential Harmony Search Technique

3.1 Introduction

Developing more realistic models to predict highly fluctuating and irregular stock price values more effectively and accurately is a big challenge for researchers in the field of financial data mining. Being time series stock prices are highly dynamic and volatile in nature. Efficient prediction of such dynamic time series values can be formulated as a function of getting output from the previous recorded input, output or both. Neural network and fuzzy neural network are the two conventionally adopted models to address temporal problems involving dynamic systems. According to their architecture, again these networks can be categorized as feed-forward or recurrent type. Feed-forward networks are capable of approximating any continuous function, but they involve static mapping. Again in feed forward network the number of lag inputs and outputs are known in advance, but the precise order of the dynamic system is generally unknown. Moreover, the delayed values result in a larger network size with an increased input dimension. On the other hand, unlike feed forward networks, the superior capabilities like dynamics and the ability to memorize information for later use make recurrent network as a popular choice for identifying dynamic systems. The closed loop systems obtained by introducing recurrent paths in the network helps to capture the dynamic behavior of the system. Many recurrent structures have been proposed in literature for financial time series prediction [14-16].

In past few years Adaptive Neuro Fuzzy Inference system (ANFIS) modeled by incorporating the low level learning and computational power of neural networks with the high level human like thinking and reasoning of fuzzy systems has grown as another popular research topic in financial time series prediction [20-24]. Consequent part is one of the key points in the design of ANFIS structure. The consequent part may be a minimum fuzzy implication as in Mamdani type, may be a linear combination of input variables like Takagi–Sugeno–Kang (TSK)-type or may include nonlinear combination
of input variables as in [25-28]. All the fuzzy neural networks (FNN) are again of type feed forward FNN or recurrent FNN. Feed forward fuzzy neural networks record only the static input output correlation, making them unsuitable for the solution of dynamic problems. Contrary to the conventional feed forward fuzzy neural networks, the feedback topology of recurrent fuzzy neural networks enhances their ability to memorize information and handling the dynamic systems more effectively. Recurrent fuzzy neural network can be designed by containing a set of recurrent paths connecting all the fuzzy rules or by containing recursive feedback loops from a given fuzzy rule. Several types of recurrent fuzzy neural networks have been cited in literature for different application fields [29-36].

This chapter elaborates the mechanism of a new Self Evolving Recurrent Neuro Fuzzy Inference System (SERNFIS) optimized through a modified differential harmony search technique for efficient prediction of highly fluctuating and irregular financial time series data like stock market indices. The network is modeled including the first order TSK type fuzzy if then rules with two types of feedback loops. The recurrent structure in the proposed model comes from locally feeding the firing strength of the fuzzy rule back to itself and by including few time delay components at the output layer. Another recurrent functional link artificial neural network (RCEFLANN) model is also presented for a comparative study. All the parameters of the model including the antecedent, consequent and feedback loop parameters are refined using a modified differential harmony search technique. Modified differential harmony search is a new version of original DHS algorithm, in which the current to best mutation strategy has been applied in the pitch adjustment operation and the controlling parameters of the algorithm have been adapted by using the JADE [109] approach. The performance of the proposed Self Evolving Recurrent Neuro Fuzzy Inference System (SERNFIS) has been validated by applying it for prediction of stock closing prices of two different stock indices.

3.2 Models used for Stock Price Forecasting

This section delineates the detailed mechanism of two recurrent networks applied for forecasting the dynamic stock price values.
3.2.1 Self Evolving Recurrent Neuro Fuzzy Inference System

Self Evolving Recurrent Neuro Fuzzy Inference System (SERNFIS) is a TSK type fuzzy inference system with two type feedback path. The recurrent structure of the proposed network is incorporated by internal local feedbacks i.e. by feeding the each rule firing strength to it and by including few one step delayed output samples as a recurrent vector in the output layer. The recurrent structure in SERNFIS exhibits some kind of memory that helps to capture the highly fluctuating and dynamic behavior of the time series data. The consequent of each recurrent fuzzy rule is of first order TSK-type.

Fig. 3.1 Architecture of proposed Self Evolving Recurrent Neuro Fuzzy Inference System (SERNFIS) model

The architecture of the Self Evolving Recurrent Neuro Fuzzy Inference System (SERNFIS) is a seven layer network as shown in Fig. 3.1. The mathematical functions of each layer are described as follows.
Layer 1 (Input Layer):

Each node in this layer is an input node specifying a crisp value for each input variable. The nodes only transmit the input values to the next layer directly. No computation is done in this layer and no weights are needed to be adjusted.

Layer 2 (Fuzzification Layer):

This layer performs the fuzzification of the input values through a suitable membership function. With the fuzzy set $F_{kl}$, $R$ number of rules, Gaussian membership function the grade of membership of the input (premise variable) $x^t_l$ in $F_{kl}$ is specified as follows:

$$F_{kl}(x^t_l) = \exp \left( -\frac{1}{2} \left( \frac{x^t_l - c_{kl}}{a_{kl}} \right)^2 \right) \quad (3.1)$$

where $c_{kl}$ is the center and $a_{kl}$ is the spread of $k^{th}$ rule membership function corresponding to the $l^{th}$ premise variable.

Layer 3 (Spatial Firing Layer):

Each node in this layer functions as a spatial rule node corresponding to each fuzzy rule. The spatial firing strength of the $k^{th}$ rule, assuming the product T-norm of the antecedent fuzzy sets is computed as follows:

$$u_k(x^t) = \prod_{l=1}^{d} F_{kl}(x^t_l) = \prod_{l=1}^{d} \exp \left( -\frac{1}{2} \left( \frac{x^t_l - c_{kl}}{a_{kl}} \right)^2 \right) \quad (3.2)$$

Layer 4 (Temporal Firing Layer):

This layer contains the recurrent rule node introducing the self evolving internal feedback loop to the network structure. The output of the recurrent rule node is a temporal firing strength $W_k$ which depends not only on the current spatial firing strength $u_k$, but also on the one step delayed temporal firing strength $W_k^{t-1}$. At any instant $t$ the temporal firing strength of the recurrent rule node can be produced as a linear combination of the current spatial firing strength and the one step delayed temporal firing strength as follows:

$$W_k^t = \alpha_k \times u_k(x^t) + (1 - \alpha_k) \times W_k^{t-1} \quad (3.3)$$
Where \(0 \leq \alpha_k \leq 1\) represents the internal recurrent feedback weight. These weights specify the contribution ratio of current spatial and delayed temporal firing strength towards network outputs.

**Layer 5 (Normalization Layer):**

Each node output in this layer is a normalized temporal firing strength, which is obtained from the ratio of the \(i\)th rule’s temporal firing strength to the sum of all rules’ temporal firing strengths as follows:

\[
\overline{W_k^t} = \frac{W_k^t}{\sum_{k=1}^{r} W_k^t}
\]

**Layer 6 (Consequent Layer):**

The consequent layer is a layer of adaptive nodes which provides a weighted sum of the original input pattern as the output of a linear function. The coefficients of the linear function are adapted by using a suitable error function. The output of the consequent layer at instant \(t\) is specified as:

\[
f_k^t = a_{k0} + \sum_{l=1}^{d} a_{kl} x_l^t
\]

**Layer 7 (Output Layer):**

Node in this layer not only takes the output of layer 5 as input, but also includes a recurrent vector of one step delayed output values as the input. The recurrent vector at the output layer will help to increase the ability of the network in capturing the dynamic behavior of stock time series data by taking in to account the past outputs. Hence the overall output of the SERNFIS is calculated as the summation of the rule consequences and the weighted one step delay outputs as follows:

\[
y^t = \sum_{k=1}^{R} \overline{W_k^t} f_k^t + \sum_{j=1}^{r} \beta_j y^{t-j}
\]

Where \(\beta_j\) represents the weight corresponding to the \(j^{th}\) delayed output, \(y^{t-j}\) represents the \(j^{th}\) delayed output for \(j=1, 2, ..., r\).
The performance of the proposed SERNFIS model mainly depends on fine tuning of the antecedent, consequence, internal feedback weights and the weights corresponding to the delayed outputs of recurrent vector.

### 3.2.2 Recurrent Computationally Efficient Functional Link Artificial Neural Network

Recurrent Computationally Efficient Functional Link Artificial Neural Network (RCEFLANN) is a single hidden layer ANN having three components: Functional expansion component (FEB), Delay component (DB) and a Learning component. The functional expansion component contains p number of trigonometric blocks for functional expansion. Each trigonometric block contains a linear combination function and a highly nonlinear \( \tanh() \) function. The delay component includes a recurrent vector including r number of one step delayed output samples as input to the learning component. Learning component is the single neuron present at the output layer. The structure of the RCEFLANN is shown in Fig. 3.2. RCEFLANN is an extension of the CEFLANN network explained in section 2.2.2. With the order p, r no of output delays, any d dimensional input pattern \( X = [x_1, x_2, \ldots, x_d]^T \) is expanded to a L dimensional pattern \( RCX \) as

\[
RCX(t) = [CX^T(t), R^T(t)]
\]

\[
= [x_1, x_2, \ldots, x_d, o_1, o_2, \ldots, o_p, y_{t-1}, y_{t-2}, \ldots, y_{t-r}]^T
\]

So the input vector contains a total number of \( d + p + r \) inputs comprising \( d \) inputs from the stock closing price indices, \( p \) inputs from the functional expansion, and \( r \) inputs from the delayed outputs. The recurrent vector including \( r \) number of one step delayed output samples is represented as

\[
R^T(t) = [y_{t-1}, y_{t-2}, \ldots, y_{t-r}]^T
\]

With the order \( p \) any d dimensional input pattern \( X = [x_1, x_2, \ldots, x_d]^T \) is expanded to a m dimensional pattern \( CX \) by Trigonometric functional expansion as

\[
CX = [cx_1, cx_2, \ldots, cx_d, cx_{d+1}, cx_{d+2}, \ldots, cx_m]^T = [x_1, x_2, \ldots, x_d, cx_{d+1}, cx_{d+2}, \ldots, cx_m]^T.
\]

For each order in \( p \), the weighted sum of the components of the original input pattern is passed through a hyperbolic tangent (\( \tanh() \)) non linear function to produce an output \( o \) which is stored in \( cx_l \) (with \( d+1 \leq l \leq m \)). Each \( o_i \) is obtained using the following formula.
Where \( a_{ij} \) is the associated parameter. 

\[
o_i = \tanh(a_{i0} + \sum_{i=1, j=1}^{i=p, j=d} a_{ij} \times x_j)
\] 

(3.8)

After expansion, weight is initialized for each expanded unit and delayed unit and then the weighted sum of the components of the enhanced input pattern produces the output \( y \) at any instant \( t \) using the following equation.

\[
y' = \sum_{i=1}^{L} w'_i RCX'_i \]

(3.9)

Fig. 3.2 Architecture of Recurrent Computationally Efficient FLANN (RCEFLANN) model

3.3 Models Parameter Estimation using Modified Differential Harmony Search

The effectiveness of the proposed SERNFIS depends on proper estimation of the unknown parameters like the parameters of the antecedent, the consequent parts of the
fuzzy if-then rules and the weights of the feedback loops, minimizing the error of prediction. Since few decades evolutionary algorithms act as excellent global optimizers for real parameter optimization problems. Hence here the parameters of the proposed model have been estimated through a modified differential harmony search technique.

Differential Harmony Search (DHS) is a variant of harmony search (HS) technique that uses the mutation scheme of Differential Evolution (DE) technique in the pitch adjustment operation for harmony improvisation process. [101-105]. The detailed steps of DHS algorithm for parameter optimization is discussed in chapter 2. The performance of DHS algorithm is dependent on five parameters, including the size of harmony memory (HMS), the harmony memory consideration rate (HMCR), the maximum number of iterations (NI), the pitch adjustment rate (PAR) and the Scaling factor (F). Initially harmony vectors are randomly generated and stored in a harmony memory (HM). Then a new candidate harmony is created from the HM by applying the steps of memory consideration, pitch adjustment, and random selection. Finally, the HM is updated by repeatedly replacing the worst harmony vector by a new harmony with better fitness value. The above process is repeated until the termination criterion is satisfied.

Like other evolutionary algorithms, the performance of DHS algorithms depends on the settings of the control parameters: harmony memory consideration rate (HMCR), the pitch adjustment rate (PAR) and the scaling factor F. The parameter HMCR keeps a balance between exploration and exploitation and takes value between 0 and 1. The further adjustment of a harmony after memory consideration step is determined by the value of PAR that can be visualized as a local search. Scaling factor F controls the result of mutation. A number of mutation strategies like rand1, rand2, best1, best2 and current to best are available in literature. Existing DHS uses the rand1 approach of mutation strategy. As the optimal settings of these parameters are problem-dependent, so it is often necessary to tune the control parameters in order to achieve the desired results.

In this chapter, the original DHS algorithm has been modified by using the JADE [108, 109] approach of parameter updation scheme and using the current to best mutation strategy in the pitch adjustment operation. Initially the HMCR, PAR and F value associated with each individual are generated according to a normal/ Cauchy
distribution with means $\mu_{HMCR}, \mu_{PAR}, \mu_F$. At the end of each generation, the values of $\mu_{HMCR}, \mu_{PAR}, \mu_F$ are updated according to the $HMCR$, $PAR$, $F$ values resulted in the generation of the successful trial vector in that generation. As the search progresses, $\mu_{HMCR}, \mu_{PAR}, \mu_F$ should gradually approach the optimal values for the given problem.

Steps of DHS based learning are as follows:

1. **Step 1:** Initialize the parameters ($HMS, \mu_{HMCR}, \mu_{PAR}, \mu_F$ and $NI$).
2. **Step 2:** Initialize the harmony memory $HM$ randomly specifying the unknown parameters of the model comprising the antecedent, consequent parameter values and weights of the feedback loops according to the harmony memory size.
3. **Step 3:** Set $t=0$.
4. **Step 4:** Find the fitness function value of each harmony vector in $HM$ i.e. the root mean square error of the actual and predicted output value obtained.
5. **Step 5:** Generate the $HMCR$, $PAR$ and the mutation scaling factor $F$ using Normal and Cauchy distributions having mean value $\mu_{HMCR}, \mu_{PAR}, \mu_F$ as follows:

   $$HMCR_i = randn_i(\mu_{HMCR}, 0.1)$$
   $$PAR_i = randn_i(\mu_{PAR}, 0.1)$$
   $$F_i = randc_i(\mu_F, 0.1)$$

6. **Step 6:** Improvise a new harmony ($X_{new}$) from $HM$ as follows:
   
   **Step 6.1:** for ($j=1$ to $n$) do
   
   **Step 6.2:** if ($rand (0, 1) < HMCR$) then $X_{new}(j) = X_a(j)$ where $a \in (1, 2, ..., HMS)$
   
   **Step 6.3:** if ($rand (0, 1) < PAR$) then
   
   $$X_{new}(j) = X_{new}(j) + F \times (X_{best}(j) - X_{new}(j)) + F \times (X_b(j) - X_c(j))$$
   
   where $b, c \in (1, 2, ..., HMS)$
   
   **Step 6.4:** else $X_{new}(j) = LB(j) + rand (0, 1) \times (UB(j) - LB(j))$

7. **Step 7:** if $f(X_{new})$ is better than the $f(worst)$ update the $HM$ as $X_{worst} = X_{new}$

8. **Step 8:** Update the mean $HMCR$, $PAR$ and scaling factor as follows:
\[ \mu_{HMCR} = (1-c)\mu_{HMCR} + c \times \text{mean}_A(S_{HMCR}) \]
\[ \mu_{PAR} = (1-c)\mu_{PAR} + c \times \text{mean}_A(S_{PAR}) \]
\[ \mu_F = (1-c)\mu_F + c \times \text{mean}_L(S_F) \]  

(3.12)

Where \( c \in (0,1) \) is a positive integer; \( S_{HMCR}, S_{PAR} \) and \( S_F \) denote all successful HMCR, PAR and mutation scaling factor \( F \) respectively; \( \text{mean}_A \) and \( \text{mean}_L \) represent the arithmetic and Lehmer mean respectively. The Lehmer mean is given by

\[ \text{mean}_L(S_F) = \frac{\sum_{i=1}^{[S_F]} F_i^2}{\sum_{i=1}^{[S_F]} F_i} \]  

(3.13)

Step 9: Set \( t=t+1 \).

Step 10: Repeat steps 4 to 9 until \( t=NI \) is reached.

Step 11: Save the Best harmony vector \( X_{best} \) in the HM to represent the parameters of the model and use the model for testing.

### 3.4 Empirical Study

In this study the performance of the proposed Self Evolving Recurrent Neuro Fuzzy Inference System (SERNFIS) trained using a modified differential harmony search technique is validated by applying it on two stock indices i.e. BSE SENSEX of Bombay stock exchange and Standard’s & Poor’s 500 (S&P500) of US stock market.

#### 3.4.1 Data set Description

Simulations are carried on 2 different stock indices data set i.e. BSE SENSEX index belongs to Indian stock market and S&P500 belong to US stock market. The total number of samples used in experiment for BSE SENSEX dataset is 491 from 2\(^{nd}\) July 2012 to 11\(^{th}\) July 2014 and for S&P500 are 512 from 2\(^{nd}\) July 2012 to 6\(^{th}\) August 2014. The closing price of the two stock indices is predicted by using the direct method of prediction. The sample dataset comprising the open, low, high and closing stock index prices are passed through a windowing process with a chosen window size and
prediction horizon to set the input and output of the model. To measure the generalization ability of the model data sets are divided into training and testing sets. The training part is used to train the model, while the test part is used to compute the predictions. The details of training and test set of two datasets are given in Table 3.1.

Table 3.1 Data set Description

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Duration</th>
<th>Total no. of samples</th>
<th>Training samples</th>
<th>Testing samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>BSE SENSEX</td>
<td>02-Jul-12 to 11-Jul-14</td>
<td>491</td>
<td>327</td>
<td>164</td>
</tr>
<tr>
<td>S&amp;P500</td>
<td>02-Jul-12 to 06-Aug-14</td>
<td>512</td>
<td>341</td>
<td>171</td>
</tr>
</tbody>
</table>

3.4.2 Input Setup

Initially the performance of the proposed SERNFIS model is observed for short term prediction of two stock indices and its performance is compared with few evolutionary learning algorithms. Then its performance is compared with other popular prediction models available in literature. As in short term prediction few days or weeks of data are taken as input, so initially with a smaller input size the proposed model is compared. For predicting the one day a head closing price using SERNFIS, the number of input layer node is set to 6 to express the past 3 days closing index and three technical indicators. The number of output node is chosen as 1 for expressing the closing index of 6th day. Further the performance of the model is compared with other input size and prediction horizon by varying the number of past closing price values taken in input set with same technical indicators. To improve the performance of the model, all the inputs are scaled between 0 and 1 using the min max normalization as given in equation (2.35).

3.4.3 Technical Indicators

Technical indicators are one of the popular choices of representing the stock price movements. In literature different sets of technical indicators has been used in input set for enhancing the predictive ability of the model. Out of the many technical indicators, 3 indicators commonly used by researchers for stock price prediction problem have been
chosen as input to the proposed network. The details of the technical indicators used in the study are as follows:

**Simple Moving Average (SMA):** It is the simple average of the values by taking a window of the specified period. The n period moving average is defined as:

\[
\text{n period MA} = \frac{1}{n} \sum_{i=1}^{n} cp_i \quad \text{where} \quad cp_i = \text{ith day closing price}
\]

The 5 days moving average value has been used in this study.

**Williams %R:** It is a momentum indicator that measures overbought/oversold levels.

\[
\text{Williams %R} = \frac{\text{Highest high in k period} - \text{Today's closing price}}{\text{Highest high in k period} - \text{Lowest low in k period}} \times -100
\]

For this experiment \(k\) is set to 14.

**Relative Strength Index (RSI):** It calculates the internal strength of the security. RSI is calculated as follows:

\[
\text{RSI} = 100 - \frac{100}{1 + \text{RS}} \\
\text{RS} = \frac{\text{average of k periods gain}}{\text{average of k periods loss}}
\]

For this study the periods have been taken as 14 days.

### 3.4.4 Parameter Setup

The size of parameter space of different models that need to be tuned using evolutionary learning algorithm is given in Table 3.2. For CEFLANN with order 2, input size 6, the number of associated parameters used in functional expansion is 14 and number of weights between expanded pattern and output neuron is 8. Hence total number of unknown parameters need to be tuned by a learning algorithm is 22. With two delay components the parameter space size of RCEFLANN is 24. With a set of 3 fuzzy rules, the total number of antecedent and consequent parameters of the ANFIS model need to be tuned by the learning algorithm is 42. For the SERNFIS model with 3 rules and 2 delay components at output layer the size of parameter space including the antecedent, consequent, local feedback and output delay weights reach to 47.

The controlling parameters of any evolutionary algorithm are normally application oriented. There is no fixed value for it. So initially through a number of
simulations the controlling parameters of the evolutionary algorithms are derived. With a suitable harmony memory size, each harmony i.e. the unknown parameters of the model is initialized randomly according to the specified dimension of parameter space. As the parameter space size of CEFLANN and RCEFLANN is within 25, so the harmony memory size chosen for these two models is 30 and as the parameter space size of other models is within 50, so for them the harmony memory size is set to 60. The RMSE error is taken as the fitness function for all the above learning algorithms. The same harmony memory with maximum iterations 100 has been used for all the evolutionary learning based methods.

Table 3.2 Parameter Space size of different models

<table>
<thead>
<tr>
<th>Model</th>
<th>SERNFIS</th>
<th>RSEFNN-LF</th>
<th>ANFIS</th>
<th>RCEFLANN</th>
<th>CEFLANN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter Space Size</td>
<td>47</td>
<td>45</td>
<td>42</td>
<td>24</td>
<td>22</td>
</tr>
</tbody>
</table>

3.4.5 Performance Evaluation Criteria

The Root Mean Square Error (RMSE), Mean Absolute Percentage Error (MAPE) defined in equation (2.36) and (2.37) of chapter 2 has also been used to compare the performance of the models for predicting the closing price of the BSE SENSEX and S&P500 index in one day advance with different learning algorithms. Along with these two error metrics the Mean Absolute Error (MAE) is also included in this chapter for performance comparison. The MAE is defined as follows:

\[
MAE = \frac{1}{N} \sum_{k=1}^{N} \left| y_k - \hat{y}_k \right|
\]

(3.17)

where \( y_k \) = actual closing price on kth day

\( \hat{y}_k \) = predicted closing price on kth day

N = number of test data.
3.4.6 Significance Test

Instead of comparing the models only with respect to the error metrics values, a significance test i.e. the Superior Predictive Ability (SPA) test is also used to better assess the forecasting performance of the models.

Superior Predictive Ability Test

Superior Predictive Ability (SPA) test of Hansen helps to identify whether any of the competing model is better than the benchmark model. So the null hypothesis is specified as none of the models is better than the benchmark model against the alternative that among the competing models at least one is significantly better than the benchmark. Hansen has applied a supremum over the standardized performances and tests the null hypothesis:

$$H_0: \max_{k=1,...,M} \frac{\mu_k}{\sqrt{\text{var}(n^{1/2}d_k)}} \leq 0$$

(3.18)

Using the statistic:

$$T_{SPA}^n = \max_k \left[ \max_{k=1,...,M} \frac{n^{1/2}d_k}{\sqrt{\text{var}(n^{1/2}d_k)}} , 0 \right]$$

(3.19)

where $\text{var}(n^{1/2}d_k)$ is an estimate of the variance of $\left(n^{1/2}d_k\right)$ obtained by the bootstrap.

The average performance of model $k$ relative to benchmark model is represented as:

$$\bar{d}_k = \frac{1}{n} \sum_{i=1}^{n} d_{k,i}$$

(3.20)

where $d_{k,i} = L_{o,i} - L_{k,i}$

(3.21)

The loss function at time $t$ for the benchmark model is defined by $L_{o,t}$ and $L_{k,t}$ indicates the value of the corresponding loss function for another competing model $k$. $n$ is the number of out of sample data. The distribution of the test statistics under null hypothesis
is approximated by the empirical distribution derived from bootstrap resample based on stationary bootstrap. This is done by defining:

$$Z_{k,b,t}^* = d_{k,b,t}^* - g_i(d_k)$$  \hspace{1cm} (3.22)

for \( b = 1, \ldots, B \), \( t = 1, \ldots, n \), \( i = l, c, u \)

where \( d_{k,b,t}^* \) is the bootstrap performance of model \( k \) relative to benchmark model at time \( t \) and \( g(x) \) is the estimator function defined as follows:

$$g_l(x) = \max(0, x)$$

$$g_c(x) = x \cdot \max \left\{ \frac{\sum_{i=1}^n x_i}{n}, \left( \frac{\log n}{\log \log n} \right) \right\}$$

$$g_u(x) = x$$

Based on these three \( g(x) \) values there are three corresponding SPA test values i.e. SPA \( l \), SPA \( c \) and SPA \( u \). SPA \( l \) represents the lower bound, SPA \( c \) represents the consistent value and SPA \( u \) represents the upper bound of SPA test statistics. Before calculating \( P \) value of SPA test first the SPA statistics for each bootstrap is calculated as follows:

$$T_{SPA}^{n*} = \max \left\{ 0, \max_{k=1, \ldots, M} \left\{ \frac{n^{1/2} - z_{k,b}}{\sqrt{\text{var} \left( n^{1/2} d_k \right)}}, \frac{n}{2} \log \log n \right\} \right\} \text{ for each } b = 1, \ldots, B$$  \hspace{1cm} (3.24)

Then comparing SPA statistics of each bootstrap with original SPA statistics the \( P \) value of SPA test is obtained as follows:

$$P^{SPA} = \sum_{b=1}^B \frac{\left\{ T_{SPA}^{n*} > T_n^{SPA} \right\}}{B} \text{ where } B = \text{total no. of bootstraps}$$  \hspace{1cm} (3.25)

A high \( P^{SPA} \) value indicates that the Null hypothesis cannot be rejected, which means that the benchmark model is not outperformed by the competing models. The detail of SPA test is available in [52, 53, 110, 111].
3.4.7 Experimental Results and Model Outputs

Originally the dataset has been divided into a single train and test set. As the evolutionary algorithms are based on randomness, so performance metrics are observed from ten independent executions of the algorithm with same training and testing sets. The best, average and standard deviation of these 10 runs are reported for all the data sets. Tables 3.3 and 3.4 list the various forecast statistics of SERNFIS model trained using different learning algorithms for one day a head prediction of stock index closing price of BSE SENSEX and S&P500 index respectively. Performance comparison of the proposed SERNFIS model with respect to the RSEFNN-LF [29], ANFIS, RCEFLANN, CEFLANN model with input size 6 and prediction horizon 1 are listed in Tables 3.5 and 3.6, respectively. The one day a head closing price prediction of BSE and S&P500 data set using all the models are shown in Figs. 3.3 to 3.12.

Table 3.3 Performance comparison of SERNFIS with different learning algorithms for one day ahead stock price prediction of BSE SENSEX Data set

<table>
<thead>
<tr>
<th>Learning Algorithm</th>
<th>RMSE Train</th>
<th>RMSE Test</th>
<th>MAPE Test</th>
<th>MAE Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>MDHS</td>
<td>min</td>
<td>0.0125</td>
<td>0.0145</td>
<td>0.6415</td>
</tr>
<tr>
<td></td>
<td>avg</td>
<td>0.0128</td>
<td>0.0232</td>
<td>0.9474</td>
</tr>
<tr>
<td></td>
<td>std</td>
<td>0.0003</td>
<td>0.0137</td>
<td>0.4777</td>
</tr>
<tr>
<td>DHS</td>
<td>min</td>
<td>0.0126</td>
<td>0.0154</td>
<td>0.6728</td>
</tr>
<tr>
<td></td>
<td>avg</td>
<td>0.0128</td>
<td>0.0265</td>
<td>1.1139</td>
</tr>
<tr>
<td></td>
<td>std</td>
<td>0.0002</td>
<td>0.0081</td>
<td>0.3212</td>
</tr>
<tr>
<td>DE</td>
<td>min</td>
<td>0.0126</td>
<td>0.0157</td>
<td>0.6926</td>
</tr>
<tr>
<td></td>
<td>avg</td>
<td>0.0129</td>
<td>0.0300</td>
<td>1.2697</td>
</tr>
<tr>
<td></td>
<td>std</td>
<td>0.0004</td>
<td>0.0113</td>
<td>0.4562</td>
</tr>
<tr>
<td>HS</td>
<td>min</td>
<td>0.0150</td>
<td>0.0166</td>
<td>0.7552</td>
</tr>
<tr>
<td></td>
<td>avg</td>
<td>0.0160</td>
<td>0.0203</td>
<td>0.9115</td>
</tr>
<tr>
<td></td>
<td>std</td>
<td>0.0008</td>
<td>0.0030</td>
<td>0.1451</td>
</tr>
</tbody>
</table>
Table 3.4 Performance comparison of SERNFIS with different learning algorithms for one day ahead stock price prediction of S&P500 Data set

<table>
<thead>
<tr>
<th>Learning Algorithm</th>
<th>RMSE Train</th>
<th>RMSE Test</th>
<th>MAPE Test</th>
<th>MAE Test</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MDHS</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>min</td>
<td>0.0110</td>
<td>0.0124</td>
<td>0.4556</td>
<td>0.0131</td>
</tr>
<tr>
<td>avg</td>
<td>0.0111</td>
<td>0.0146</td>
<td>0.5503</td>
<td>0.0159</td>
</tr>
<tr>
<td>std</td>
<td>0.0002</td>
<td>0.0018</td>
<td>0.0756</td>
<td>0.0022</td>
</tr>
<tr>
<td><strong>DHS</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>min</td>
<td>0.0116</td>
<td>0.0130</td>
<td>0.4822</td>
<td>0.0139</td>
</tr>
<tr>
<td>avg</td>
<td>0.0124</td>
<td>0.0148</td>
<td>0.5682</td>
<td>0.0164</td>
</tr>
<tr>
<td>std</td>
<td>0.0004</td>
<td>0.0022</td>
<td>0.1056</td>
<td>0.0031</td>
</tr>
<tr>
<td><strong>DE</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>min</td>
<td>0.0114</td>
<td>0.0138</td>
<td>0.5307</td>
<td>0.0153</td>
</tr>
<tr>
<td>avg</td>
<td>0.0115</td>
<td>0.0183</td>
<td>0.6959</td>
<td>0.0203</td>
</tr>
<tr>
<td>std</td>
<td>0.0002</td>
<td>0.0042</td>
<td>0.1588</td>
<td>0.0048</td>
</tr>
<tr>
<td><strong>HS</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>min</td>
<td>0.0132</td>
<td>0.0143</td>
<td>0.5650</td>
<td>0.0164</td>
</tr>
<tr>
<td>avg</td>
<td>0.0140</td>
<td>0.0178</td>
<td>0.6913</td>
<td>0.0200</td>
</tr>
<tr>
<td>std</td>
<td>0.0005</td>
<td>0.0036</td>
<td>0.1437</td>
<td>0.0042</td>
</tr>
</tbody>
</table>

Table 3.5 Performance comparison of Proposed SERNFIS model with other FLANN and ANFIS models for one day ahead stock price prediction of BSE Data set

<table>
<thead>
<tr>
<th>Different Models</th>
<th>RMSE Train</th>
<th>RMSE Test</th>
<th>MAPE Test</th>
<th>MAE Test</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SERNFIS</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>min</td>
<td>0.0125</td>
<td>0.0145</td>
<td>0.6415</td>
<td>0.0155</td>
</tr>
<tr>
<td>avg</td>
<td>0.0128</td>
<td>0.0232</td>
<td>0.9474</td>
<td>0.0233</td>
</tr>
<tr>
<td>std</td>
<td>0.0003</td>
<td>0.0137</td>
<td>0.4777</td>
<td>0.0125</td>
</tr>
<tr>
<td><strong>RSEFNN-LF</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>min</td>
<td>0.0126</td>
<td>0.0152</td>
<td>0.6738</td>
<td>0.0162</td>
</tr>
<tr>
<td>avg</td>
<td>0.0128</td>
<td>0.0400</td>
<td>1.5230</td>
<td>0.0386</td>
</tr>
<tr>
<td>std</td>
<td>0.0003</td>
<td>0.0286</td>
<td>0.9410</td>
<td>0.0248</td>
</tr>
<tr>
<td><strong>ANFIS</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>min</td>
<td>0.0126</td>
<td>0.0157</td>
<td>0.7042</td>
<td>0.0171</td>
</tr>
<tr>
<td>avg</td>
<td>0.0128</td>
<td>0.0307</td>
<td>1.2656</td>
<td>0.0316</td>
</tr>
<tr>
<td>std</td>
<td>0.0002</td>
<td>0.0150</td>
<td>0.5660</td>
<td>0.0148</td>
</tr>
<tr>
<td><strong>RCEFLANN</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>min</td>
<td>0.0151</td>
<td>0.0162</td>
<td>0.7268</td>
<td>0.0176</td>
</tr>
<tr>
<td>avg</td>
<td>0.0161</td>
<td>0.0248</td>
<td>1.1197</td>
<td>0.0275</td>
</tr>
<tr>
<td>std</td>
<td>0.0008</td>
<td>0.0092</td>
<td>0.4328</td>
<td>0.0109</td>
</tr>
<tr>
<td><strong>CEFLANN</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>min</td>
<td>0.0142</td>
<td>0.0220</td>
<td>0.9656</td>
<td>0.0237</td>
</tr>
<tr>
<td>avg</td>
<td>0.0164</td>
<td>0.0384</td>
<td>1.7663</td>
<td>0.0439</td>
</tr>
<tr>
<td>std</td>
<td>0.0031</td>
<td>0.0204</td>
<td>1.1096</td>
<td>0.0274</td>
</tr>
</tbody>
</table>
Instead of comparing the models only with respect to the error metrics values, Hansen’s Superior Predictive Ability (SPA) test has also been used to better assess the forecasting performance of the models. To avoid the data snooping problem, the upper and lower bound of the SPA test over MAPE error has been calculated for the datasets. Table 3.7 and 3.8 presents the p-value of upper and lower bound of SPA test with 3 different size of bootstrap re-samples for the two datasets. In this study individually each model is considered as the benchmark model to evaluate whether a particular model is significantly outperformed by other competing modes. For both the datasets, the p-value of SPA test over MAPE error with different bootstrap size is showing higher value indicating that the other competing models are not giving better performance compared to the proposed model.

Table 3.6 Performance comparison of Proposed SERNFIS model with other FLANN and ANFIS models for one day ahead stock price prediction of S&P500 Data set

<table>
<thead>
<tr>
<th>Different Models</th>
<th>RMSE Train</th>
<th>RMSE Test</th>
<th>MAPE Test</th>
<th>MAE Test</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SERNFIS</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>min</td>
<td>0.0110</td>
<td>0.0124</td>
<td>0.4556</td>
<td>0.0131</td>
</tr>
<tr>
<td>avg</td>
<td>0.0111</td>
<td>0.0146</td>
<td>0.5503</td>
<td>0.0159</td>
</tr>
<tr>
<td>std</td>
<td>0.0002</td>
<td>0.0018</td>
<td>0.0756</td>
<td>0.0022</td>
</tr>
<tr>
<td><strong>RSEFNN-LF</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>min</td>
<td>0.0111</td>
<td>0.0129</td>
<td>0.4822</td>
<td>0.0139</td>
</tr>
<tr>
<td>avg</td>
<td>0.0113</td>
<td>0.0160</td>
<td>0.6021</td>
<td>0.0175</td>
</tr>
<tr>
<td>std</td>
<td>0.0002</td>
<td>0.0044</td>
<td>0.1784</td>
<td>0.0053</td>
</tr>
<tr>
<td><strong>ANFIS</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>min</td>
<td>0.0110</td>
<td>0.0135</td>
<td>0.4936</td>
<td>0.0143</td>
</tr>
<tr>
<td>avg</td>
<td>0.0114</td>
<td>0.0243</td>
<td>0.9443</td>
<td>0.0277</td>
</tr>
<tr>
<td>std</td>
<td>0.0003</td>
<td>0.0100</td>
<td>0.3874</td>
<td>0.0115</td>
</tr>
<tr>
<td><strong>RCEFLANN</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>min</td>
<td>0.0121</td>
<td>0.0136</td>
<td>0.5119</td>
<td>0.0148</td>
</tr>
<tr>
<td>avg</td>
<td>0.0140</td>
<td>0.0148</td>
<td>0.5639</td>
<td>0.0165</td>
</tr>
<tr>
<td>std</td>
<td>0.0018</td>
<td>0.0015</td>
<td>0.0542</td>
<td>0.0015</td>
</tr>
<tr>
<td><strong>CEFLANN</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>min</td>
<td>0.0139</td>
<td>0.0151</td>
<td>0.5613</td>
<td>0.0162</td>
</tr>
<tr>
<td>avg</td>
<td>0.0148</td>
<td>0.0178</td>
<td>0.6889</td>
<td>0.0199</td>
</tr>
<tr>
<td>std</td>
<td>0.0012</td>
<td>0.0044</td>
<td>0.2009</td>
<td>0.0059</td>
</tr>
</tbody>
</table>
Table 3.7 P value of Superior Predictive Ability test over MAPE error for BSE Dataset

<table>
<thead>
<tr>
<th>Benchmark model</th>
<th>1000</th>
<th>2000</th>
<th>3000</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPA_u SPA_1</td>
<td>SPA_u SPA_1 SPA_u SPA_1 SPA_u SPA_1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SERNFIS</td>
<td>0.9670 0.5300</td>
<td>0.9655 0.5340</td>
<td>0.9733 0.5390</td>
</tr>
<tr>
<td>RSEFNN-LF</td>
<td>0.4340 0.1090</td>
<td>0.4290 0.1120</td>
<td>0.4120 0.1190</td>
</tr>
<tr>
<td>ANFIS</td>
<td>0.1110 0.0250</td>
<td>0.0980 0.0270</td>
<td>0.0937 0.0223</td>
</tr>
<tr>
<td>RCEFLANN</td>
<td>0.0520 0.0230</td>
<td>0.0590 0.0275</td>
<td>0.0523 0.0273</td>
</tr>
<tr>
<td>CEFLANN</td>
<td>0.0050 0.0050</td>
<td>0.0025 0.0025</td>
<td>0.0013 0.0013</td>
</tr>
</tbody>
</table>

Table 3.8 P value of Superior Predictive Ability test over MAPE error for S&P500 Dataset

<table>
<thead>
<tr>
<th>Benchmark model</th>
<th>1000</th>
<th>2000</th>
<th>3000</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPA_u SPA_1</td>
<td>SPA_u SPA_1 SPA_u SPA_1 SPA_u SPA_1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SERNFIS</td>
<td>0.9990 0.5860</td>
<td>0.9985 0.6010</td>
<td>0.9990 0.5953</td>
</tr>
<tr>
<td>RSEFNN-LF</td>
<td>0.3100 0.0850</td>
<td>0.2915 0.0665</td>
<td>0.2847 0.0640</td>
</tr>
<tr>
<td>ANFIS</td>
<td>0.1800 0.0690</td>
<td>0.1855 0.0820</td>
<td>0.1933 0.0803</td>
</tr>
<tr>
<td>RCEFLANN</td>
<td>0.0380 0.0270</td>
<td>0.0390 0.0280</td>
<td>0.0317 0.0210</td>
</tr>
<tr>
<td>CEFLANN</td>
<td>0.0050 0.0050</td>
<td>0.0025 0.0025</td>
<td>0.0013 0.0013</td>
</tr>
</tbody>
</table>

Finally the performance of the proposed SERNNFIS model is observed for multistep ahead prediction of closing price with different input size. The error metrics obtained for the two datasets with prediction horizon 1, 5, 10 and input size 6, 8 and 10 are listed in Tables 3.9 and 3.10 respectively. The multistep ahead prediction output with input size 6 has been shown in Figs. 3.13 to 3.16. Again the one day ahead prediction of closing price with input size 8 and 10 are shown in Figs. 3.17 to 3.20.
Fig. 3.3 One day ahead prediction of daily closing price of BSE data using SERNFIS

Fig. 3.4 One day ahead prediction of daily closing price of BSE data using RENFF-LF

Fig. 3.5 One day ahead prediction of daily closing price of BSE data using ANFIS

Fig. 3.6 One day ahead prediction of daily closing price of BSE data using RCEFLANN

Fig. 3.7 One day ahead prediction of daily closing price of BSE data using CEFLANN

Fig. 3.8 One day ahead prediction of daily closing price of S&P500 data using SERNFIS
Fig. 3.9 One day ahead prediction of daily closing price of S&P500 data using RENFF-LF

Fig. 3.10 One day ahead prediction of daily closing price of S&P500 data using ANFIS

Fig. 3.11 One day ahead prediction of daily closing price of S&P500 data using RCEFLANN

Fig. 3.12 One day ahead prediction of daily closing price of S&P500 data using CEFLANN

Fig. 3.13 One day ahead prediction of daily closing price of BSE data using SERNFIS with input size 8

Fig. 3.14 One day ahead prediction of daily closing price of S&P500 data using SERNFIS with input size 8
Fig. 3.15 One day ahead prediction of daily closing price of BSE data using SERNFIS with input size 10

Fig. 3.16 One day ahead prediction of daily closing price of S&P500 data using SERNFIS with input size 10

Fig. 3.17 5 days ahead prediction of closing price of BSE data using SERNFIS with input size 6

Fig. 3.18 10 days ahead prediction of closing price of BSE data using SERNFIS with input size 6

Fig. 3.19 5 days ahead prediction of closing price of S&P500 data using SERNFIS with input size 6

Fig. 3.20 10 days ahead prediction of closing price of S&P500 data using SERNFIS with input size 6
Table 3.9 Performance comparison of proposed SERNFIS model with different input size and prediction horizon over BSE dataset

<table>
<thead>
<tr>
<th>Input size</th>
<th>Prediction horizon</th>
<th>RMSE</th>
<th>MAPE</th>
<th>RMSE</th>
<th>MAPE</th>
<th>RMSE</th>
<th>MAPE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>5</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>0.0145</td>
<td>0.6415</td>
<td>0.0432</td>
<td>1.9968</td>
<td>0.0705</td>
<td>3.1700</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>0.0151</td>
<td>0.6755</td>
<td>0.0527</td>
<td>2.4139</td>
<td>0.0690</td>
<td>2.9075</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>0.0241</td>
<td>1.0326</td>
<td>0.0591</td>
<td>2.6766</td>
<td>0.0740</td>
<td>3.3016</td>
</tr>
</tbody>
</table>

Table 3.10 Performance comparison of proposed SERNFIS model with different input size and prediction horizon over S&P500 dataset

<table>
<thead>
<tr>
<th>Input size</th>
<th>Prediction horizon</th>
<th>RMSE</th>
<th>MAPE</th>
<th>RMSE</th>
<th>MAPE</th>
<th>RMSE</th>
<th>MAPE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>5</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>0.0124</td>
<td>0.4556</td>
<td>0.0286</td>
<td>1.0620</td>
<td>0.0352</td>
<td>1.2200</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>0.0141</td>
<td>0.5281</td>
<td>0.0292</td>
<td>1.04318</td>
<td>0.0343</td>
<td>1.2559</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>0.0138</td>
<td>0.5394</td>
<td>0.0303</td>
<td>1.0993</td>
<td>0.0365</td>
<td>1.2608</td>
</tr>
</tbody>
</table>

3.4.8 Result Analysis

Analyzing the forecast statistics given in the above graphs and tables following inference is drawn:

- A comparison of different learning approaches over SERNFIS model shows clearly the superior forecasting performance of SERNFIS trained using MDHS approach in comparison to other learning algorithms in the table 3.3 and 3.4.
- The global search capability of harmony search is improved by including the mutation operation in pitch adjustment of each harmony and by updating the controlling parameters based on its successful trial values.
- Again from the experimental result analysis it is clearly apparent that the SERNFIS model provides superior forecasting performance when trained with MDHS algorithm in comparison to RSEFNN-LF, ANFIS, RCEFLANN and CEFLANN models in forecasting the stock index closing price.
• From the significance test, it is observed that for both the datasets, the p-value of SPA test over MAPE error with different bootstrap size is showing higher value indicating that the other competing models are not giving better performance compared to the proposed SERNFIS model.

• Finally from Tables 3.9 and 3.10, it is concluded that prediction accuracy is inversely proportional to time horizon, i.e. the prediction accuracy decreases with increase in prediction horizon.

### 3.5 Summary

In this chapter, a Self Evolving Recurrent Neuro Fuzzy Inference System (SERNFIS) is demonstrated for efficient prediction of highly dynamic stock market indices over varying time frames. The recurrent structure is included in the proposed model by two types of feedback loops. One type is inserted at the temporal firing strength layer and another at the output layer. All the parameters of the model including the antecedent, consequent and feedback loop parameters are refined using a modified differential harmony search (MDHS) technique. The MDHS algorithm is another version of DHS algorithm, in which the current to best mutation strategy is applied in pitch adjustment operation and controlling parameters are adapted iteratively according to their previous successful experience. Experimental results obtained by implementing the model on two different stock market indices demonstrate the effectiveness of the proposed model compared to RSEFNN-LF, ANFIS, RCEFLANN and CEFLANN models.