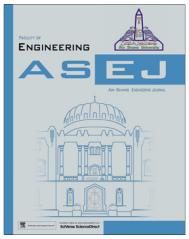
Ain Shams Engineering Journal (2017) 8, 371–390

Ain Shams University

Ain Shams Engineering Journal

www.elsevier.com/locate/asej
www.sciencedirect.com



ELECTRICAL ENGINEERING

# Artificial chemical reaction optimization of neural networks for efficient prediction of stock market indices

S.C. Nayak [a,*], B.B. Misra [b], H.S. Behera [a]

[a] Department of Computer Science Engineering & Information Technology, Veer Surendra Sai University of Technology, Burla 768018, India
[b] Department of Information Technology, Silicon Institute of Technology, Bhubaneswar 751024, India

Received 23 February 2015; revised 11 June 2015; accepted 23 July 2015
Available online 3 October 2015

KEYWORDS

Artificial neural network;
Financial forecasting;
Time series prediction;
Chemical reaction optimization;
Multilayer perceptron

**Abstract** The underlying system models of time series prediction are complex and not known a priori, hence, accurate and unbiased estimation cannot be always achieved using well known linear techniques. The estimation process requires more advanced prediction algorithms, such as multilayer perceptrons (MLPs). This paper presents an artificial chemical reaction neural network (ACRNN), which uses artificial chemical reaction optimization (ACRO) to train the MLP models for forecasting the stock market indices. The underlying motivation for using ACRO is the ability to overcome the issues of convergence, parameter setting and overfitting and to accurately forecast financial time series data even when the underlying system processes are typically nonlinear. Historical data of seven different stock indices have been collected for 15 years to test the performance of the ACRNN approach. After extensive experimentation, it is observed that the ACRNN technique demonstrates significant improvements in prediction accuracy over the MLP approach.


## 1. Introduction

Forecasting the stock market remains a challenging task for researchers. The influence of uncertainty in the stock market increases the degree of difficulties to a great extent in proper prediction of the index prices. Such difficulties in prediction of stock market prices arise due to its nonlinearities, highly volatile nature, discontinuities, movement of other stock markets, political influences and other macro-economical factors and even individual psychology [1–3]. Further various economic factors such as oil prices, exchange rates, interest rates, stock price indices in other countries, and domestic/global

* Corresponding author. Tel.: +91 9438420234.
E-mail addresses: sarat_silicon@yahoo.co.in, saratnayak234@gmail.com (S.C. Nayak), misrabijan@gmail.com (B.B. Misra), hsbehera_india@yahoo.com (H.S. Behera).
Peer review under responsibility of Ain Shams University.

Production and hosting by Elsevier

http://dx.doi.org/10.1016/j.asej.2015.07.015

economic situations, affect the movement of a stock market. These factors have been employed on the study of stock price prediction and found to be important elements for influencing the market [4,5].

A stock market behaves very much like a random walk process and their serial correlation is economically and statistically significant. Nonlinear dynamics proposes that in financial time series past prices help to determine future prices, but not in a straightforward way. The relationship between past prices and future prices is nonlinear, and this nonlinearity implies that past price change can have wide ranging effects on future prices.

As more and more money is being invested in the stock market by common investors, brokers and speculators, investors get anxious about the future trend of the stock market. If the direction of the market is successfully predicted, the investors may be better guided and also monetary rewards will be substantial. Hence an effective and accurate forecasting model is necessary in order to predict the stock market behavior. In recent years, many new methods for modeling and forecasting the stock market have been developed.

Mainly there are two broad categories of forecasting models used, i.e. linear and nonlinear models. For many decades linear models have been the basis of traditional statistical forecasting models in financial engineering. Several statistical techniques have been used extensively for stock market prediction [6]. Among those statistical techniques employed in this regard, moving averages (MAs), auto-regressive integrated moving average (ARIMA), auto-regressive heteroscedastic (ARCH), and generalized ARCH (GARCH) have received wide acceptance. MA is a type of finite impulse response filter used to analyze a set of data points by creating a series of averages of different subsets of the whole data set in the stock market. This is used to smooth out the short-term fluctuations with the help of time series analysis and highlight long term trends. These models may not capture the non-linearity of other types of time series, being developed for specific types of problems.

The Box–Jenkins method using autoregressive moving average (ARMA); linear models have extensively been used in many areas of time series forecasting [7]. These linear models have been successfully applied to different engineering, economic and social applications. These models do not possess the capacity to capture the high degree of non-linearity associated with financial time series. The presence of noise and nonlinearity in the financial time series makes these traditional methods ineffective to adapt toward nonlinear models. The popular nonlinear models used for financial forecasting include artificial neural networks, support vector machine, Bayesian Networks, and fuzzy system models. Among these frequently adopted methods, artificial neural networks have drawn significant attention from several researchers in the field of stock market behavior forecasting.

During the last two decades there has been tremendous development in the field of soft computing methodology which includes artificial neural network (ANN), evolutionary algorithms, and fuzzy systems. This improvement in computational intelligence capabilities has enhanced the modeling of complex, dynamic and multivariate nonlinear systems. These soft computing methodologies have been applied successfully to the areas of data classification, financial forecasting, credit scoring, portfolio management, risk level evaluation, etc. and found to produce significant results. ANNs are software constructs designed to mimic the way the human brain learns. The neural network can imitate the process of human behavior and solve nonlinear problems, which have made it popular and are widely used in calculating and predicting complicated systems. The quality of non-linear mapping achieved in ANN is difficult with the conventional approaches. It has the capability of dealing with complex problems of structural instability. They are analogous to nonparametric, non-linear regression models. Their novelty lies in their ability to model non-linear processes with few a priori assumptions about the nature of the generating process. The neural networks have the ability to discover non-linear relationships in the input data set without a priori assumption of the knowledge of relation between the input and the output. ANNs are found to be good universal approximator which can approximate any continuous function to desired accuracy. ANNs are considered to be an effective modeling procedure when the mapping from the input to the output contains both regularities and exceptions which is the way stock market behaves. It also allows the adaptive adjustment to the model and nonlinear description of the problems. These advantages of ANN attract researchers to develop ANN based forecasting models to the area of stock market prediction. These forecasting models incorporate prior knowledge in ANN to improve the prediction accuracy. Neural networks are also extensively used in medical applications such as image/signal processing [8], pattern and statistical classifiers [9] and for modeling the dynamic nature of biological systems. This is particularly useful in financial engineering applications where much is assumed and little is known about the nature of the processes determining asset prices. ANN is relatively a recent method for business forecasting and has been successfully applied to wide range of forecasting problems such as exchange rate, credit scoring, business failure, bankruptcy, interest rate, stock return, stock market index, portfolio management and option & future prices. ANNs have been successfully applied in financial engineering and gained wide acceptance due to their better learning abilities and approximation capabilities. Gradient based methods are one of the most widely used error minimization methods used to train back propagation based ANN models. Back propagation algorithm is a classical domain dependent technique for supervised training. It works by measuring the output error calculating the gradient of this error, and then adjusting the ANN weights and biases in the descending gradient direction. Back propagation based ANNs are very popular methods to predict stock market with better calculation, spreading abilities and stronger nonlinear mapping ability. But the stock market deals not only with nonlinearity but also with chaos, and it is a dynamic system related to time. Therefore the network for prediction is a dynamic system. Back propagation neural networks, particularly the multilayer perceptron (MLP) have many shortcomings such as the slow learning rate, larger memory size, easy to get into local minimum, bigger randomicity and so on, which affects the prediction accuracy of the stock price. These shortcomings force researchers toward developing hybrid models by combining linear and nonlinear models. These hybrid models that have been developed by many researchers combining nonlinear models such as ANN and evolutionary soft computing techniques such as swarm optimization, genetic algorithm and other nature and bio-inspired search techniques, have come up with better performance.

ANN and its hybridization with other soft computing techniques have been successfully applied to the potential corporate finance applications and found to be appropriate. Over the decades, a number of forecasting models based on soft computing techniques such as ANN [10,11], fuzzy logic and its hybridization [12,13], Genetic Algorithm (GA) based ANN [14] have been applied to the stock index forecasting. Several nature-inspired population-based algorithms such as GA, particle swarm optimization (PSO), differential evolution (DE), and evolutionary algorithm (EA) have shown their promising ability as learning algorithms utilized for forecasting purposes. However, their performance may vary from one stock market to another. A single forecasting model may not be suitable for the different types of stock markets in different countries. According to the "no free lunch theorem" there is no single state of the art constraint handling technique, which can outperform all others on every problem [15]. Hence, choosing a suitable optimization technique for solving a particular problem involves a numerous trial and error method. The efficiency of these optimization techniques is characterized by tuning the parameters. For better convergence of the algorithm, suitable fine-tuned parameters are required. In order to search the global optimum solution, the algorithm requires appropriate selection of parameters which makes the use of algorithm difficult. Hence, an optimization technique requiring less parameters, small number of computations as well as good approximation capability will be the choice for better forecasting accuracy. These facts motivated us to develop a hybrid forecasting model in order to fill the need of an effective and efficient model.

The objective of this research work was to develop a forecasting model which can be applied to global stock market data. In this regard seven fast growing stock index databases such as BSE, DJIA, NASDAQ, TAIEX, FTSE, S&P 500 and LSE have been considered for experimentation. In order to fill up the vacancy of a good training algorithm, a natural chemical reaction inspired metaheuristic has been chosen for optimizing the parameters of a MLP and the model is termed as ACRNN. The prediction of short term (one-day-ahead), medium (one-week-ahead) and long term (one-month-ahead) closing prices of the abovementioned data sets has been carried out. These datasets consist of the daily closing prices for the period of 01 January 2000–31 December 2014. The sliding window technique has been used to select the training pattern for the network instead of dividing the whole data set into training and test pattern. Unlike previous research works, instead of normalizing the whole data set, we normalize the current training data. Also, for each current training pattern, a previously optimized weight set has been utilized adaptively and hence there is a significant reduction in training time.

The rest of the paper is organized as follows. Section 2 covers work related to stock index forecasting. Section 3 describes background of artificial chemical reaction optimization. Section 4 describes the architecture of the proposed forecasting models. Section 5 presents the results and analysis of the experiments conducted. Finally Section 6 gives the concluding remarks followed by a list of references.

## 2. Related work

This section explores some of the previous research attempts on financial time series forecasting using linear, nonlinear and hybrid techniques. Liu et al. [5] investigated the influence of specification of return distribution on the performance of volatility forecasting using two Generalized Auto Regressive Conditional Heteroskedasticity (GARCH) models i.e. GARCH-N and GARCH-SGED. Their empirical results from Shanghai and Shenzhen composite stock indices indicated that the GARCH-SGED model is more accurate and superior to the GARCH-N model in forecasting the volatility of China stock market. Ling-Ming and Shang-Wu Yu [16] adopted the GM (1, 1) model to predict the rates of return of nine major index futures in the American and Eurasian market and compared the performance with GARCH/TGARCH. Their results findings revealed that the latter models perform better than the former in terms of forecasting capabilities.

The ANNs have recently been applied to many areas such as data mining, stock market analysis, medical and many other fields. It is observed that MLP has been adopted as the most frequently used ANN by the researchers for the task of forecasting. An MLP contains one or more hidden layer, and each layer can contain more than one neurons. The input pattern is applied to the input layer of the network and it propagates the signals through the network from one layer to other till it reaches the output layer. During the forward phase, the synaptic weights of the networks are fixed. In the backward phase, the weights are adjusted in accordance with the error correction rule popularly called as back propagation learning rule. Some forecasting applications of MLP are financial time series forecasting [17], market trend analysis [18], macroeconomic data forecasting [19], stock exchange movement [20], railway traffic forecasting [21], airline passenger traffic forecasting [22], maritime traffic forecasting [23], electric load forecasting [24] and air pollution forecasting [25]. Though MLP is the most widely and frequently used technique, it suffers from slow and nonconvergence. Calderon and Cheh [26] argued that the standard MLP network is subject to problems of local minima. Again there is no formal guideline how to develop a network for the MLP technique [27]. It is suggested that to overcome the local minima problem, more nodes may be added to the hidden layers. But multiple hidden layers with large number of neurons in each layer make the network computationally inefficient. Finding an optimal structure of the network of MLP technique leads to a combinatorial problem. Defining a feasible architecture and parameters for MLP is very often a matter of trial and error which is also computationally very expensive. From the study of the existing literature on stock market index forecasting, it is observed that improved forecasting accuracy and adopting models with less computational complexities are important areas of present day research in the stock market. Aiming with better forecasting accuracies, researchers moved toward adopting hybrid ANN models with large number of evolutionary searching algorithms.

Majhi et al. [28] have used a Bacterial Foraging Optimization (BFO) and Adaptive BFO (ABFO) for stock market index prediction and observed that both BFO and ABFO models are computationally more efficient, prediction wise more accurate and show faster convergence compared to other evolutionary computing models such as GA and PSO based models. Some research work successfully applied Adaptive Neuro-Fuzzy Inference System (ANFIS) and reveals that ANFIS provides a promising alternative for stock market prediction and can be a useful tool for economists and practitioners dealing with the forecasting of the stock price index return [29]. A new

hybrid iterative evolutionary learning algorithm which adopts the capabilities of NN, fuzzy as well as GA has been proposed by Yu and Zhang [30]. They used GA and Gradient Descent (GD) learning algorithm alternatively in an iterative manner to adjust the parameters and their results indicate that hybrid iterative evolutionary learning is more powerful than separate sequential training algorithm. Hassan and Nath [31] have successfully applied the Hidden Markov Model (HMM) for predicting future events and observe that HMM is encouraging and offers a new paradigm for stock market forecasting.

Aboueldahab and Fakhreldin [32] proposed a new hybrid GA/PSO model with perturbation term inspired by the passive congregation biological mechanism to overcome the problem of local search restriction in standard hybrid models. In [33], research conducted to evaluate the effectiveness of neural network models which are known to be dynamic and effective in stock market predictions. The models under analysis were, a MLP, a dynamic ANN (DAN2) and the hybrid neural networks which use generalized autoregressive conditional heteroscedasticity (GARCH). In [34], the authors proposed some neuro-genetic models for stock index prediction implemented on the Indian stock market for six years historical data and observed that in most of the financial year, the FLANN-GA model outperforms the ANNGD and ANNGA in terms of predictability. This shows how the rapid growth of evolutionary optimization techniques during the last few decades has increased the adaptability of hybrid forecasting models.

Optimization is one of the cornerstones in science and engineering. Most of the problems can be formulated in the form of optimization ranging from power generation scheduling in electrical engineering [35], DNA sequencing in biomedical science [36], to stock market trend prediction [37]. In the recent past, the field of nature-inspired optimization techniques has grown incredibly fast. These algorithms are usually general-purpose and population-based. They are normally referred to as evolutionary algorithms because many of them are motivated by biological evolution. In a broad sense, evolutionary algorithms cover those which vary a group of solutions in iterations based on some nature-inspired operations. Examples include GA, memetic algorithm (MA), ant colony optimization (ACO), particle swarm optimization (PSO), differential evolution (DE), and harmony search (HS). Many of them are inspired by biological processes, varying in scale from the genetic level, e.g. GA, MA, and DE, to the creature level, e.g. ACO and PSO. Unlike the others, HS is motivated by the phenomenon of human activities in composing music. These algorithms are successful in solving many different kinds of optimization problems.

Support Vector Machines (SVMs) with a new kernel function called Gaussian Radial Basis Polynomials Function (GRPF) have been proposed by Zanaty [38] for improving the classification accuracy. A comparative analysis of SVMs versus the Multilayer Perception for data classifications is presented to verify the effectiveness of their proposed kernel function. Comparing the classification accuracy of the SVM to the MLP learning algorithms, it is observed that support vector machines with the proposed new kernel function accomplish better accuracy than multilayer networks, especially in high dimension data sets. An Effective Differential Evolution (EDE) algorithm for solving real parameter optimization problems over continuous domain has been introduced by Mohamed et al. [39]. They proposed a new mutation rule based on the best and the worst individuals among the entire population of a particular generation. The mutation rule is combined with the basic mutation strategy through a linear decreasing probability rule. The comparison results between the EDE and several classical differential evolution methods indicate that the proposed EDE algorithm is competitive with, and in some cases superior to, others. SVM, Least Square SVM (LSSVM) and Relevance Vector Machine (RVM) were employed for prediction of liquefaction susceptibility of soil and produced best performance for prediction [40].

Similarly, many researchers studied the structure and parameter design of RBFNN, which help to understand its dynamic behavior and improve generalization ability [41–43]. The RBFNN has been widely applied to nonlinear time series prediction [44], and stock market forecasting [45]. However it has limitations with regard to convergence speed and forecasting accuracy. To circumvent this problem, some attempts have been made to develop hybrid models using RBFNN and evolutionary algorithms. The authors in [46] introduced the artificial fish swarm algorithm to optimize the radial basis functions and applied it to forecast the stock indices of the Shanghai Stock Exchange. The performance of their model was found to be superior to GA, PSO, ARIMA, back propagation and SVM. Zhang and He [47] optimized RBF with GA to forecast nonlinear time series, and Feng and Zhao [48] used RBFNN optimized by SVM to forecast electricity loads successfully.

Artificial chemical reaction optimization (ACRO) is one of the recently established meta-heuristics for optimization proposed by Albert Lam and Li [49] in 2012. It is an evolutionary optimization technique inspired by the nature of chemical reaction. In a short period of time, ACRO has been applied to solve many problems successfully, outperforming many existing evolutionary algorithms in most of the test cases. There are few applications of ACRO to multiple-sequence alignment, data mining, classification rule discovery and some benchmark functions and the efficiency has been demonstrated [50,51]. This optimization method does not need a local search method to refine the search and includes both local and global search capability. Unlike other optimization techniques, ACRO does not require many parameters that must be specified at beginning and only defining number of initial reactants is enough for implementation. As the initial reactants are distributed over feasible global search region, optimal solutions can be obtained with little iteration and hence significant reduction in computational time is achieved. The ACRO has been successfully used to solve many complex problems in recent years and found to be outperforming many other evolutionary population based algorithms. A complete guideline to help the readers implement ACRO for their optimization problem can be found in the tutorial introduced in [52]. The tutorial summarizes the basic characteristics as well as applications reported in the literature. A real coded version of chemical reaction optimization (RCCRO) has been proposed in [53] to solve the continuous optimization problems. Also, they proposed an adaptive scheme for RCCRO for performance improvement. The performance of RCCRO has been compared with a large number of techniques experimented on a set of standard continuous benchmark functions. The results show the suitability of ACRO solving problems in the continuous domain. Chemical reaction optimization was also successfully applied for population transition peer-to-peer live streaming [54]. They employed chemical reaction optimization

to maximize the probability of universal streaming by manipulating population transition probability matrix. Their simulation results show that ACRO outperforms many commonly used strategies for this problem. Minimizing the number of coding links of a network for a given target transmission rate to improve the network efficiency is a NP-hard problem. Pan et al. in [55] adopted chemical reaction optimization to develop an algorithm to solve this complex problem and found that the ACRO based framework outperforms other existing algorithms. The ACRO also has been successfully applied to grid scheduling problem [56] and task scheduling problem [57]. The authors compared the efficiency of several versions of chemical reaction optimization with other algorithms such as genetic algorithm, simulated annealing, threshold accepting and particle swarm optimization and found chemical reaction optimization to be superior. ACRO was also used to replace the back propagation based training of ANN for classification problem [58]. The simulation results show that the ACRO based ANN outperforms other optimization techniques such as GA, and SVM. Allocating available channels to the unlicensed users in order to maximize the utility of radio channels are a complex task. An algorithm based on ACRO has been developed by the authors in [59] to solve this radio spectrum allocation problem which outperforms other evolutionary techniques. Two different types of chemical reaction optimization, named canonical CRO and super molecule based CRO (S-CRO) have been proposed in [60] for the problem of stock portfolio selection and suggested that S-CRO is promising in handling the stock portfolio optimization problem. A special type of higher order neural network, called as Pi-Sigma neural network (PSNN) has been trained with ACRO forms a novel CRO-PSNN model by the authors in [61]. The model performance has been tested with various benchmark data sets. The performance of their proposed model is found superior to PSNN, GA-PSNN and PSO-PSNN. A new chemical reaction optimization with greedy strategy algorithm (CROG) has been proposed in [62] to solve the 0–1 knapsack problem. The article designed a new repair function integrating a greedy strategy and random selection is used to repair the infeasible solutions. The experimental results show the superiority of CROG over GA, ACO and quantum-inspired evolutionary algorithm.

From the above literature studies, it is found that, there are few applications of ACRO toward data mining. Also, there is a lacking of ACRO applications in the domain of financial forecasting. Motivated by the capability of ACRO, only one application was found toward forecasting BSE stock prices [63]. Also, MLP is the most frequently used ANN model for financial time series forecasting. In this study we aimed to fill this research gap through the incorporation of ACRO with ANN to forecast the movements of seven fast growing global stock indices.

## 3. Artificial chemical reaction optimization

ACRO is a population based metaheuristic inspired by natural chemical reaction proposed by Lam and Li [49]. The concept loosely couples mathematical optimization techniques with properties of chemical reactions. A chemical reaction is a natural process of transforming the unstable chemical substances (reactants/molecules) to the stable ones. A chemical reaction starts with some unstable molecules with excessive energy. The molecules interact with each other through a sequence of elementary reactions producing some intermediate chemical products. At the end, they are converted to those with minimum energy to support their existence. The energy associated with a molecule is called as enthalpy (which can be considered as fitness function in case of minimization problem) and/or entropy (which can be considered as fitness function in case of maximization problem). During a chemical reaction this energy changes with the change in intra-molecular structure of a reactant and becomes stable at one point. Most of the reactions can occur in both forward and backward directions, i.e. reversible reaction. These reactions may be monomolecular or bimolecular depending on the number of reactants taking part in the reaction. This property is embedded in ACRO to solve optimization problems. ACRO algorithm begins with set of initial reactants in a solution. Then reactants are consumed and produced via chemical reactions. Algorithm is terminated when the termination criterion is met similar to the state when no more reactions can take place (the solution becomes inert solution). According to the above concept, the ACRO algorithm consists of the following steps:

**Step 1:** Initialize parameters.
**Step 2:** Set the initial reactants and evaluate enthalpy.
**Step 3:** Apply chemical reactions on reactants.
**Step 4:** Update and select reactants.
**Step 5:**Go to step 3 if termination criterion not satisfied.
**Step 6:** Output reactant with best enthalpy.

### 3.1. Problem and parameter initialization

The optimization problem is specified as follows: Minimize $f(x)$ subject to $x_j \in X_j = 1, 2, \ldots, N$, where $f(x)$ is an objective function; $x$ is the set of each decision variable $x_j$; $N$ is the number of decision variables, $X_j$ is the set of the possible range of values for each decision variable, that is $x_j^{\min}$ and $x_j^{\max}$ are the lower and upper bound of the $j$th decision parameter respectively for real-values encoding. The problem may require different types of encoding such as binary, real, and permutation. The initial phase assigns values to the following parameters such as representation type, length of reactant/molecules (number of atoms in the molecule), total number of reactants in the reactant pool (*ReacNum*), and termination criteria.

### 3.2. Setting the initial reactants, evaluation and update

In this step initial reactants are evenly initialized in the feasible search space. Uniform population method [64–66] proposed for initial population generating can be used for creating initial reactants. In general, all vectors in a space can be obtained as a linear combination of elements of the base set. If one of elements in the base set is absent, then the dimension corresponding to this element may be vanished. That is why, it is important that initial reactants must contain reactants which must hold each element of the base set. By considering regularity case and base set, the initial reactants must be regular and also hold the base set. Generating initial reactants based on divide-and-generate paradigm is a method to generate

reactants of good quality. The uniform population method which can be used to generate the initial reactant pool is defined by Algorithm 1.

**Algorithm 1: Generate Initial Reactants**

/*$R$ is the reactants set; $I$ is the indices set and $I_e$ is the enlarged indices set*/
1-Create two reactants such as one of them $R$[1] contains all upper bounds for variables and the other $R$[2] contains all lower bounds for variables.
2-Index ← 3
3-$k$ ← 2
4-**while** $R$ is not saturated do
Let $i_e$ be an element of $I_e$ and each $i_e$ are enlarged with bit value and this bit value corresponds to part.
$i$ ← 1
**while** $R$ is not saturated and all reactants are not generated for a specific value of $k$ (and $r \leqslant 2^k - 2$) **do**
$i$ is a $k$-bit number and $i_e$ corresponds to the enlarged value of $i$. Each bit of $I$ is enlarged up to length of corresponding part of $R$[0] and $R$[1].
**for** $j$ ← 1 to $n$ **do**
**if** $j$th bit of $i_e$ is 1 then
$j$th value of $R$[Index] is equal to $R$[1]*$r$
**else**
$J$th value of R[Index] is equal to $R$[2]*$r$
**end if**
$r$ is a random real number in interval [0, 1]
**end for**
Index ← Index + 1
$i$ ← $i$+1
**end while**
$k$ ← $k$+1
**end while**

Different chemical reactions are applied on the reactants to generate new reactants. If the newly generated reactants give a better function value, the new reactant set is included and the worse reactant is excluded similar to reversible chemical reactions. The reactant with the best enthalpy value is used as the optimal solution for the forecasting model.

The enthalpies of the reactant pool are considered as the fitness of the reactants. The less the value of enthalpy of a reactant there is more chance to select it for the next iteration. The evaluation of enthalpy of the reactant pool can be calculated by Algorithm 2.

**Algorithm 2: Enthalpy (R)**

Evaluate enthalpy of each reactant in R

### 3.3. Applying chemical reactions

Different chemical reactions are applied as searching operators similar to crossover and mutation operator in genetic algorithm. Based on the number of reactants take part in a chemical reaction, the reaction may be divided into two categories: monomolecular (one reactant takes part in reaction) or bimolecular (two reactants take part in chemical reaction). The monomolecular reactions (*Redox1* and *Decomposition*) assist in intensification while the bimolecular reactions (*Synthesis, Redox2* and *Displacement*) can give the effect of diversification. ACRO does not attempt to capture every detail of a chemical reaction; rather it loosely couples chemical reaction with optimization technique. The binary encoded chemical reactions are discussed in the following subsections.

#### 3.3.1. Synthesis reaction

Non-matching bits of two reactants are determined. Then, one bit from the non-matching bit of the first reactant and one bit from the non-matching bit of the second reactant are consecutively selected to form a new reactant. For an example the synthesis reaction for binary encoding is shown in Fig. 1. Let the two reactants are represented as $Reactant_1$ and $Reactant_2$ with 8 bits of size each. The non-matching bit positions are found at 1, 2, 6 and 8 positions, and are highlighted with red color. As described, the 1st bit position of the *New Reactant* takes value from $Reactant_1$ and 2nd takes value from $Reactant_2$. Similarly the 6th position takes value from $Reactant_1$ and 8th takes value from $Reactant_2$. The matching bit positions of $Reactant_1$ and $Reactant_2$ remain unchanged in the *New Reactant*. The pseudo code of synthesis reaction is presented by Algorithm 3.

**Algorithm 3: Synthesis Reaction** ($r_1$, $r_2$, R, data)

```
newReac = R(r1);
Flag = 0;
for i = 0 : Length(R(r1)) − 1
  if R(r1,i) ! = R(r2,i) AND Flag = 0
    newReac(i) = R(r1,i);
    Flag = 1;
  else if R(r1,i) ! = R(r2,i) AND Flag = 1
    newReac(i) = R(r2,i);
    Flag = 0;
  end
end
ent = EvaluateEnthalpy (newReac, data)
if (ent < Enthalpy (r1))
  Replace R(r1) by newReac
  Enthalpy (r1) = ent;
end
if (ent < Enthalpy(r2))
  Replace R(r2) by newReac
  Enthalpy (r2) = ent;
end
```

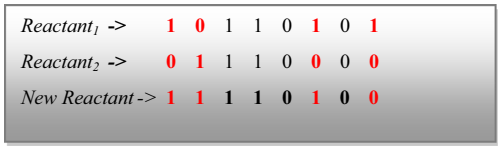



**Figure 1** Synthesis reaction for binary encoding.

### 3.3.2. Displacement reaction

In the displacement reaction, two new reactants are generated by considering two old reactants. A random binary mask is generated of the size of the reactants. Let the *i*th bit of the new reactants to be generated. Then, if *i*th bit of the mask is 1, *i*th bit of new reactant1 is copy of *i*th bit of reactant2, and *i*th bit of new reactant2 is copy of *i*th bit of reactant1. Otherwise *i*th bit of new reactant1 is copy of *i*th bit of reactant1, and *i*th bit of new reactant2 is copy of *i*th bit of reactant2. For an example the displacement reaction for binary encoding is shown in Fig. 2. As shown in the figure, the bit value of mask is 1 at the positions 1, 2, 4 and 7, new reactant1 takes value from reactant2 for these positions and new reactant2 takes value from reactant1 for these positions. For other positions, new reactant1 takes value from reactant1 and new reactant2 takes value from reactant2. The pseudo code of synthesis reaction is described by Algorithm 4.

**Algorithm 4: Displacement Reaction** ($r_1$, $r_2$, R, data)

```
mask = random binary string of length R(r1);
for i = 0 : Length(R(r1)) – 1
    if mask = 1
    newReac1(i) = R(r2, i);
    newReac2(i) = R(r1, i);
  else
    newReac1(i) = R(r1, i);
    newReac2(i) = R(r2, i);
  end
end
ent1 = EvaluateEnthalpy (newReac1, data)
ent2 = EvaluateEnthalpy (newReac2, data)
if ent1 < Enthalpy(r1)
    Replace R(r1) by newReac1
    Enthalpy (r1) = ent1;
end
if ent2 < Enthalpy(r2)
    Replace R(r2) by newReac2
    Enthalpy (r2) = ent2;
end
```

### 3.3.3. Redox1 reaction

A randomly selected bit is changed from one to zero or vice versa. The *Redox1* reaction for binary encoded reactant is shown in Fig. 3. Let the randomly selected bit position is 4 which is highlighted with red color. The corresponding bit value at 4th position of the *New Reactant* is changed from 1 to 0. The other bit positions remain same as in the *Reactant*. The pseudo code of synthesis reaction is described by Algorithm 5.

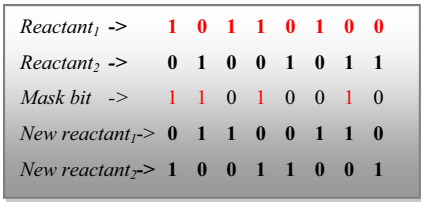



**Figure 2** Displacement reaction for binary encoding.

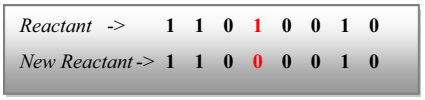



**Figure 3** Redox1 reaction for binary encoding.

**Algorithm 5: Redox1 Reaction** ($r$, R, data)

```
r1 = one random integer from [0, length(R(r)) – 1];
newReac = R(r);
  if newReac(r1) = 1
    newReac(r1) = 0;
  else
    newReac(r1) = 1;
  end
ent = EvaluateEnthalpy (newReac, data)
if (ent < Enthalpy (r))
  Replace R(r) by newReac
  Enthalpy (r) = ent;
end
```

### 3.3.4. Redox2. reaction

Example of the Redox2 reaction for two binary encoded reactants is shown in Fig. 4. Two indices of reactant are randomly generated and the bits of the reactants between the two indices are exchanged to produce two new reactants. Let the two random indices are 3 and 6. The bit values between position 3 and 6 of the *New Reactant*$_1$ receive bit values from index3 to index6 from *Reactant*$_2$ and the rest of the bit values from *Reactant*$_1$ and *New Reactant*$_2$ receives bit values from index 3 to index 6 from *Reactant*$_1$ and the rest of the bit values from *Reactant*$_2$. The pseudo code of synthesis reaction is described by Algorithm 6.

**Algorithm 6: Redox2 Reaction** ($rno_1$, $rno_2$, R, data)

```
r1 = one random integer from [0, length(R(rno1)) – 1];
r2 = one random integer from [0, length(R(rno2)) – 1];
newReac1 = R(rno1);
newReac2 = R(rno2);
newReac1(r1 : r2) = R(rno2) (r1 : r2);
newReac2(r1 : r2) = R(rno1) (r1 : r2);

ent1 = EvaluateEnthalpy (newReac1, data)
ent2 = EvaluateEnthalpy (newReac2, data)

if ent1 < Enthalpy(rno1)
  Replace R(rno1) by newReac1
  Enthalpy (rno1) = ent1;
end
if ent2 < Enthalpy(rno2)
  Replace R(rno2) by newReac2
  Enthalpy (rno2) = ent2;
end
```

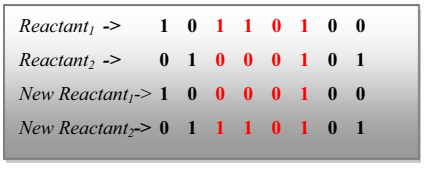



**Figure 4** Redox2 reaction for binary encoding.

### 3.3.5. Decomposition reaction

Two random indices of the reactant string are generated and the bits between those indices are flipped. Example of the decomposition reaction for binary encoding is shown in Fig. 5. Let the two random indices selected are 4 and 6, then the bit values from index4 to index6 of the *New Reactant* are flipped and rest of bit values remain same as in the old *Reactant*. The pseudo code of synthesis reaction is described by Algorithm 7.

**Algorithm 7: Decomposition Reaction** (*r, R, data*)

```
r1 = one random integer from [0, length(R(r)) − 1];
r2 = one random integer from [0, length(R(r)) − 1];
newReac = R(r)
for i = r1 : r2
  if Reac(i) = 1
    newReac(i) = 0;
  else
    newReac(i) = 1;
  end
end
ent = EvaluateEnthalpy (newReac, data)
if (ent < Enthalpy (r))
  Replace R(r) by newReac
  Enthalpy (r) = ent;
end
```

### 3.4. Termination criterion check

The ACRO terminates when the termination criterion (e.g. maximum number of iterations or minimal error signal) has been met. Otherwise chemical operators are applied, enthalpies are evaluated and the reactants are updated repeatedly.

## 4. Methodology

This section describes the basics of the different neural network based forecasting models used for this experimentation such as the gradient descent based MLP, RBFNN and the proposed ACRNN forecasting model.

### 4.1. Gradient descent based MLP

The multilayer perceptron has been considered as capable of approximating any arbitrary functions to expected level of accuracy. MLP is one of the most widely implemented neural network topologies in different fields of research. In this experiment the MLP is trained with a gradient descent based back propagation algorithm. The back propagation rule propagates the errors through the network and allows adaptation of the hidden neurons. The error correction learning in this case is

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| *Reactant* -> | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| *New Reactant* -> | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |

**Figure 5** Decomposition reaction for binary encoding.

supervised learning, i.e. the desired response for the system must be presented at the output neuron. The feed forward neural network model considered here consists of one hidden layer only. The architecture of the MLP model used here is presented in Fig. 6.

This model consists of a single output unit to estimate the closing index prices. The neurons in the input layer use a linear transfer function, and the neurons in the hidden layer and output layer use sigmoidal function as follows:

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

where $y_{out}$ is the output of the neuron, $\lambda$ is the sigmoidal gain and $y_{in}$ is the input to the neuron. Let there be $m$ neurons in the hidden layer. Since there are $n$ input values in an input vector, the number of neurons in the input layer is equal to $n$. The first layer corresponds to the problem input variables with one node for each input variable. The second layer is useful in capturing non-linear relationships among variables. At each neuron $j$ in the hidden layer, the weighted output $z$ is calculated using Eq. (2):

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

where $X_i$ is the $i^{th}$ input vector, $V_{ij}$ is the synaptic weight value between $i^{th}$ input neuron and $j^{th}$ hidden neuron and $B_j$ is the bias value and $f$ is sigmoidal activation function. The output $y$ at the single output neuron is calculated using Eq. (3):

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

where $W_j$ is the synaptic weight from $j$th hidden neuron to output neuron, $z_j$ is the output of the $j$th hidden neuron, and $B_0$ is the output bias. This output y is compared to the desired output and the error is calculated by using Eq. (4):

$$e_i = |t_i - y_i| \tag{4}$$

where $e_i$ is the error signal, $t_i$ is the target signal for $i$th training pattern and $y_i$ is the estimated output for $i$th pattern.

This error is propagated back to train the MLP model. The weight and other parameter values are adjusted by the gradient descent rule for minimal error signal generation. Because of the gradient descent neural network learning, they are characterized with problems such as slow convergence and getting trapped to local minima. Therefore possibilities are there, that it may affect the prediction capabilities of the model.

### 4.2. Radial basis function neural network (RBFNN)

Radial basis function (RBF) network can be used for approximating functions and recognizing patterns. The RBF network is a two layered network. In RBF network, each hidden unit of hidden layer implements a radial activation function and each output neuron of output layer implements a weighted sum of hidden units' output. This network is a special class of neural network in which the activation of a hidden neuron is determined by the distance between the input vector and a prototype vector. Prototype vectors refer to centers of clusters formed by the patterns or vectors in the input space. The interconnection between the hidden and output layer is made

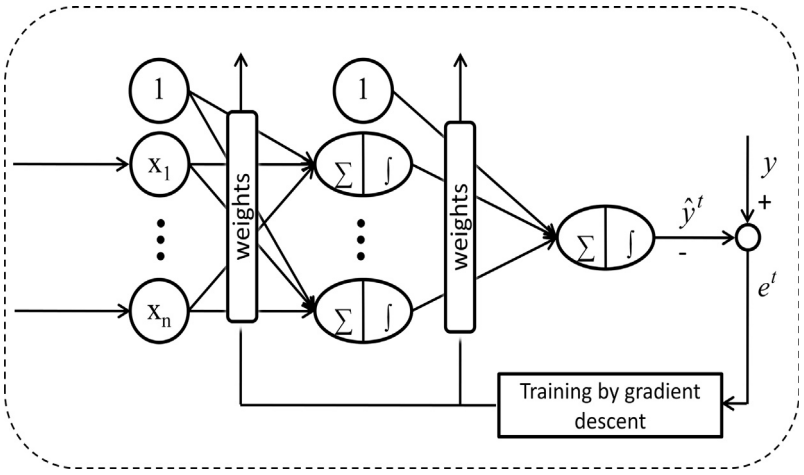



**Figure 6** Architecture of gradient descent based MLP forecasting model.

through weighted connections $W_i$. The output layer, a summation unit, supplies the response of the network to the outside world. The basic structure of a RBF neural network is shown in Fig. 7.

As shown in the figure, in the input layer, the number of input neurons is determined based on the input signals that connect the network to the environment. The hidden layer consists of a set of kernel units that carry out a nonlinear transformation from the input space to the hidden space. Two parameters, the center and the width are associated with each RBF node. The centers are determined during RBF training. The problem of selecting suitable number of basis functions is an important issue for RBFN. The number of basis functions controls the approximation and the generalization ability of RBF network. Some of the commonly used kernel functions are the Gaussian function, cubic function, linear function, and generalized multiquadratic function. We used the Gaussian function which is represented in Eq. (5).

$$\varnothing_i(x) = \exp\left(-\frac{\|x-\mu_i\|^2}{2\sigma_i^2}\right) \tag{5}$$

where $\|\cdots\|$ represents the Euclidean norm, $x$ is the input vector, $\mu_i$ is the center, $\sigma_i$ is the spread and $\varnothing_i(x)$ represents the output of the $i^{th}$ hidden node.

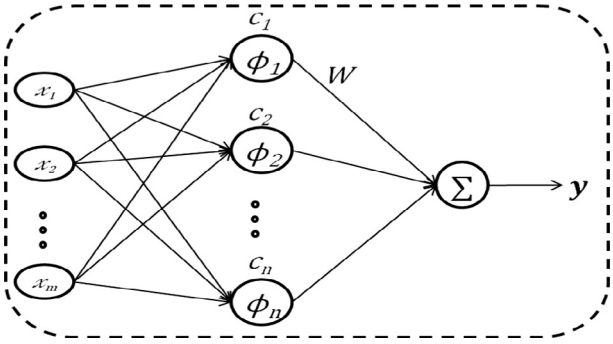



**Figure 7** Architecture of RBFNN based forecasting model.

The output of the RBF network is calculated as in Eq. (6):

$$y = f(x) = \sum_{k=1}^{N} w_k \varnothing_k(\|x - c_k\|) \tag{6}$$

where $y$ is the network output, $x$ is an input vector signal, $w = [w_1, w_2, \cdots, w_N]^T$ is the weight vector in the output layer, $N$ is the number of hidden neurons, $\varnothing_k(\cdot)$ is the basis function, $k$ is the bandwidth of the basis function, $x$ is the input vector and $c_k = (c_{k1}, c_{k2}, \cdots, c_{km})^T$ is the center vector for $k^{th}$ node, $m$ is the number of input.

### 4.3. Proposed ACRNN forecasting model

This model employs the MLP as the base architecture as described in the previous sub-section. The efficient search capability of ACRO is incorporated with the generalization capability of MLP, hence synergies the forecasting accuracies. The architecture for ACRNN is presented in Fig. 8. Each reactant represents the candidate solution for the model which comprises weight and bias vectors.

#### 4.3.1. Reactant encoding

Binary strings are used to represent the reactants of ACRO and each reactant represents a weight and bias set for the MLP model. The length of the reactant is $n*m+m*1+m+1 = m(n+2)+1$, where: $n$ is the number of neurons in the input layer, and $m$ is the number of neurons in the hidden layer. The output layer has one neuron to estimate the closing price value predicted for each input pattern presented to the network. There are $m$ number of bias values to the hidden layer and one bias to the output neuron. The reactant representation for ACRNN is shown in Fig. 9.

#### 4.3.2. Enthalpy of a reactant

Each reactant represents a potential solution (weight and bias vector) for the MLP based forecasting model. A reactant is associated with some enthalpy (minimum prediction error

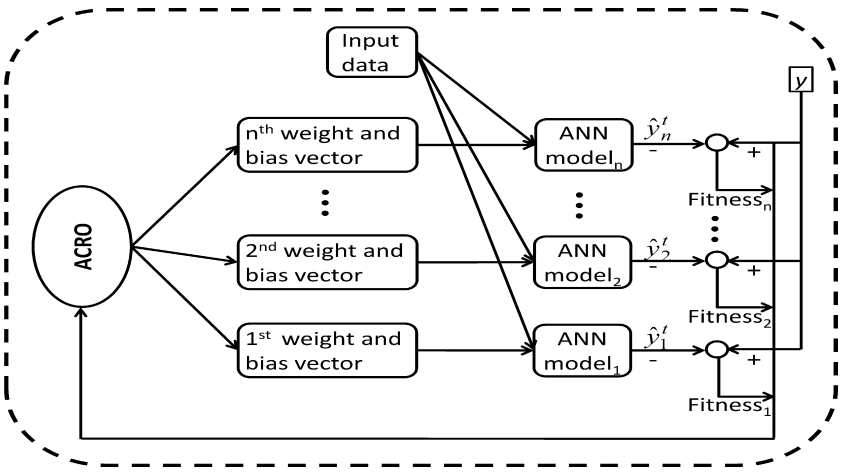



**Figure 8** Architecture of ACRNN forecasting model.

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

**Figure 9** Reactant representation for ACRNN model.

signal in this case), which can be considered as the fitness value of that reactant. The Mean Absolute Error (MAE) value is treated as the enthalpy of a reactant and represented as Eq. (7):

$$\text{MAE} = \frac{\sum_{i=1}^{N} |Actual_i - Estimated_i|}{N} \tag{7}$$

where $N$ is the number of input patterns in the training set.

### 4.3.3. ACRNN training

The ACRO algorithm, as described in Section 3 is employed here to search the optimal weight and bias set for the ANN model. The fitness of the best and average individual in each iteration moves toward the global optimum. The ACRO performs a search over the whole solution space, finds the optimal solution relatively easily, and does not require a continuous differentiable objective function as case of gradient based optimization techniques. The problem of finding an optimal parameter set to train the model could be seen as a search problem into the space of all possible parameters. The parameter set includes the weight set between the input-hidden layers, weight set between hidden-output layers, and the bias value. This search is performed here by employing the ACRO algorithm. The reactants of ACRNN represent the weight and bias values for a set of ANN models. Input data along with the reactant values are fed to the set of ANN models. The fitness is obtained from the absolute difference between the target y and the estimated output ŷ. The less the fitness value (enthalpy) of an individual, ACRO considers it better fit. A binary encoding scheme for ACRO has been used for this experimental work. The weight values between input and hidden layer neuron are represented as $V_{11}$ to $V_{nm}$. Weight values between hidden and output layer are represented by $W_1$ to $W_m$. The bias values to the hidden and output layer are represented by $B_1$ and $B_0$ respectively. The high level training for ACRNN model is presented as Algorithm 8.

**Algorithm 8: ACRNN**

1. Generate Initial Reactants /*Use algorithm1*/
2. Set *MaxIter* /* initially set maximum iteration number with a bigger value for first training */
3. Select the next **trainData** and **testData** and normalize
4. Enthalpy = **EvaluateEnthalpy (R, trainData)**
5. **R** = **ApplyChemicalReaction (R, ReacNum, MaxIter, trainData)** /*use Algorithm 9*/
6. **BestReac** = Choose the reactant from **R** with minimum enthalpy
7. *ent* = **EvaluateEnthalpy (BestReac, testData)** /*use algorithm 10*/
8. store *ent* as error for the respective **testData**
9. set *MaxIter* /* set maximum iteration number with a small value for subsequent training*/
10. Repeat step 3–8 till end of trainData and testData sets

The different chemical operators applied on the reactants have been discussed in the previous section. The pseudo code for the ApplyChemicalReaction procedure is presented by Algorithm 9.

**Algorithm 9: ApplyChemicalReaction** (*R, ReacNum, MaxIter*)

```
Set ItrNum = 0
While (ItrNum < MaxIter) do
  for i = 0 to ReacNum-1 do
    Mi = R (i, :)
    /*Apply all reactions over the reactant Mi */
      Get rand1 randomly in interval [0, 1]
        if rand1 < 0.5 then
          Get rand2 randomly in interval [0, 1]
            if rand2 < 0.5 then
              Decomposition (Mi)
            else
              Redox1 (Mi)
            end
        else
          Select another molecule M (Mi ≠ M)
          Get rand3 randomly in interval [0, 1]
        if rand3 < 0.33 then
            Synthesis (Mi, M)
        else if rand3 < 0.66 then
              Displacement (Mi, M)
            else
              Redox2 (Mi, M)
            end
          end
        end for
        ItrNum = ItrNum+1
end while
```

The enthalpy calculation for each reactant in the reactant set R can be calculated by the procedure as described by Algorithm 10. As earlier mentioned, we used binary encoded reactants for ACRO.

**Algorithm 10: EvaluateEnthalpy** (R, DataSet)

```
1. Weight = bin2Dec(R)/* Obtain the decimal equivalent of each
binary encoded reactant */
2. Enthalpy = 0;
3. for i = 1: number of reactant in R
4.    for each input–output pair in the DataSet
5.       Estimate the output of MLP model for the input pattern
and Weight.
6.          Compare the desired output with estimated value and
obtain error.
7.          Enthalpy (i) = Enthalpy (i) + |error|.
8.    end
9. end
```

The overall design of adaptive ACRNN forecasting model can be visualized by the flowchart as presented in Fig. 10. The reactants are first initialized by uniform population generation method as described by algorithm 1. The training and testing patterns are generated by the sliding window method from the original financial time series, details discussed in Section 5. Initially to train the ACRNN model more iterations are required; therefore, a larger value is assigned to the MaxIter for the first instance of training; subsequently, as the optimized set of reactants are used, the MaxIter size is significantly reduced. The train and test data are then normalized as detailed in Section 5. Then the binary encoded reactants are converted to decimal equivalent and are assigned as the connection weight and bias vector of the neural model. Enthalpies of the ACRNN models are evaluated by presenting the training data to the network. The different chemical reactions are applied to each reactant on the basis of a probability value and new reactants are formed. The reaction can be monomolecular or bimolecular. If the reaction type is bimolecular, another reactant is chosen other than the current reactant. The enthalpy of current reactant(s) is compared with that of new reactants and updated accordingly. The reaction mechanisms as well as the reactant update process are discussed in algorithm 3 to algorithm 7 in Section 3.3. This process is continued till reaching the maximum iteration value. This completes the training phase of the ACRNN model. The reactant with minimum enthalpy is selected and used for test. The enthalpy generated from this reactant is stored as the error signal for the corresponding test data. Then the next training and test patterns are generated by moving the sliding window one step ahead. For the subsequent training patterns the maximum iteration value is fixed to a small number and the above process is repeated for all training and testing patterns in the data set.

## 5. Results and discussion

To ascertain the performance of the suggested model, the daily closing price of different stock markets across the globe such as BSE, DJIA, NASDAQ, TAIEX, FTSE, S&P 500 and LSE are considered for this study. Further to establish that the suggested model is unbiased and it can work for different types of trend in different economic/political scenario without much deviation in the capabilities of prediction, the daily closing prices are used for a period from 1st January 2000 to 31st December 2014.

Again to significantly reduce the computation time consumed for training to a significant extent the following steps are taken:

I. Minimal data are used for training i.e. few input neuron with minimal patterns presented in each epoch.
II. Adaptive models are used.

Very often to train a model a large number of patterns are presented with large number of epochs to train the model for prediction. In some cases about 2/3rd data are presented as training set and the rest are used for testing. Here, though the objective is to design a generalized model, but very often it fails to track the financial market trend in general. Further the number of neurons in the input layer is also kept quite high, which also adds to the computation time. For this experiment, a sliding window concept is used which takes only five values for the input layer and only three patterns are presented

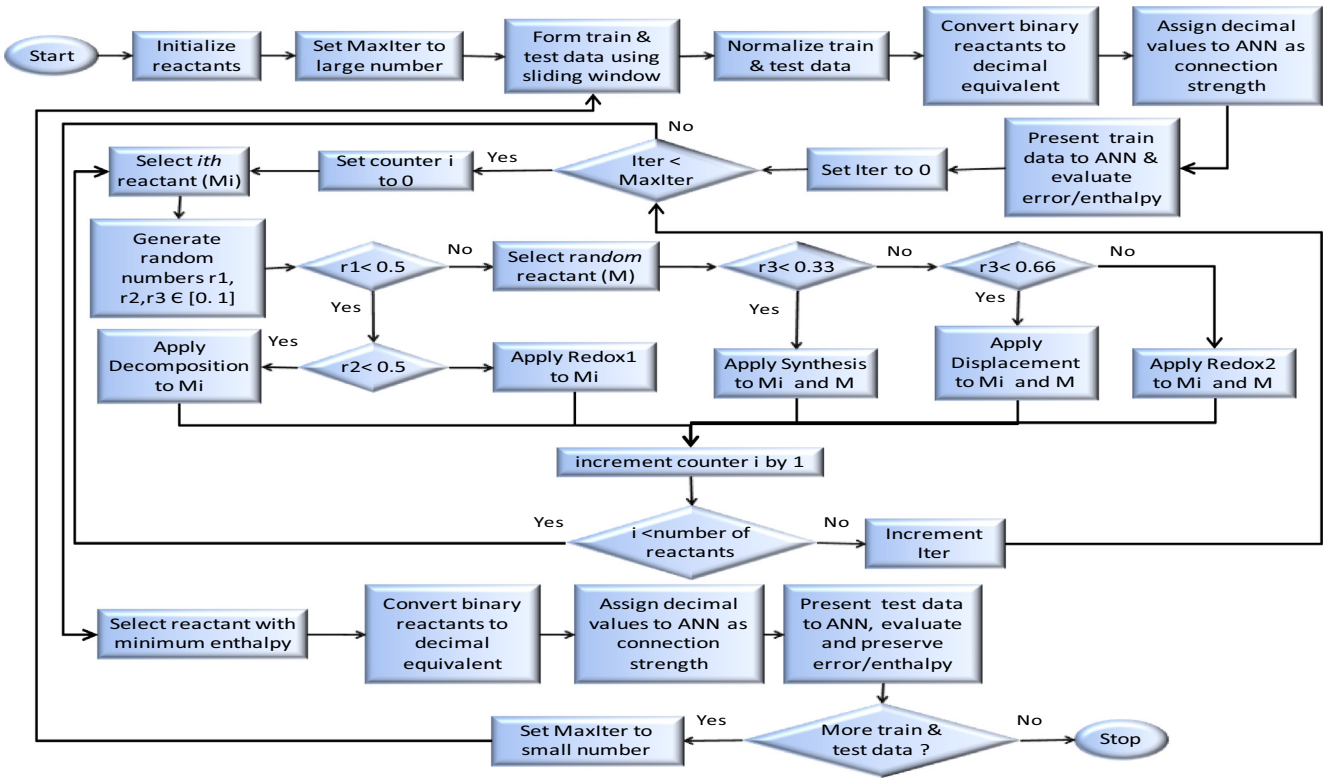



**Figure 10** Flow control of the adaptive ACRNN forecasting model.

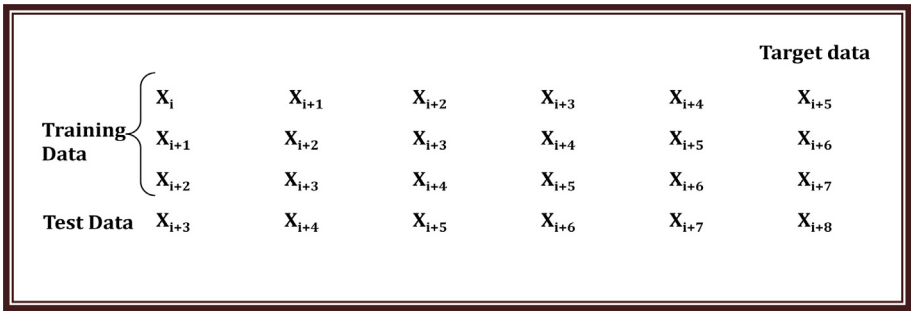



**Figure 11** Training window generation for short term prediction.

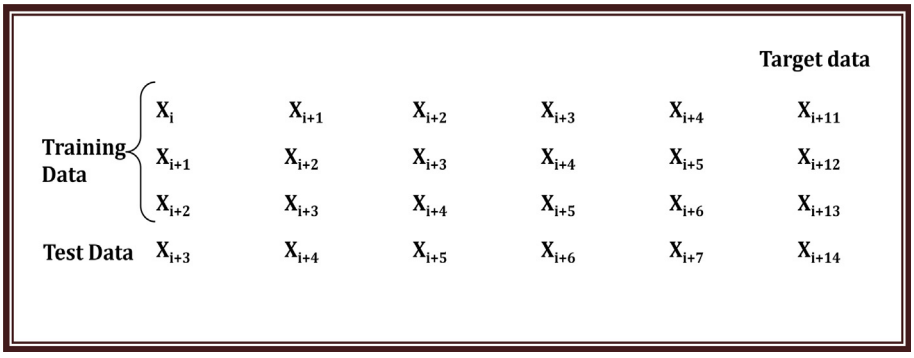



**Figure 12** Training window generation for medium term prediction.

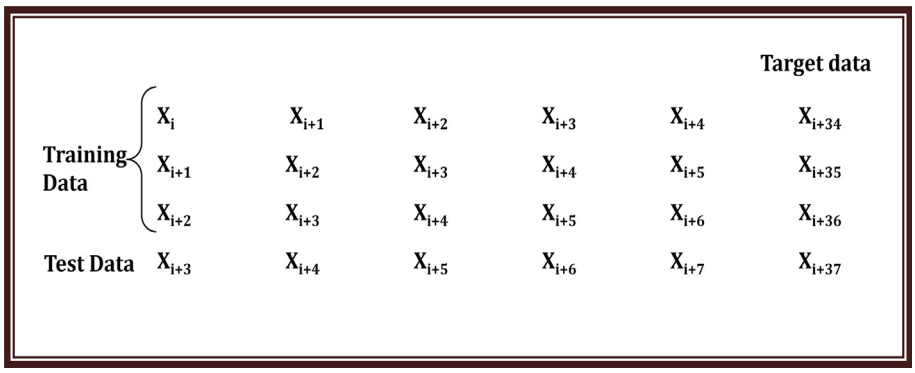



**Figure 13** Training window generation for long term prediction.

to build a model as shown in Figs. 11–13 for short term (one-day-ahead), medium term (one-week-ahead) and long term (one-month-ahead) predictions. Three patterns per epoch keep the computation time per epoch significantly low.

It is a fact that as each time the sliding window moves one step ahead, closing price data at the beginning is dropped and one new closing price data at the end is included. Therefore two consecutive training sets possibly possess minimal change in the nonlinear behavior of the input–output mapping. To incorporate the minor change in input–output mapping in the new model, the optimized weight set of the predecessor model for the same dataset is considered and minimal epochs help in capturing the change in nonlinear mapping.

Further considering the requirements of the neural network, the input data needs to be normalized. Here Sigmoidal function is used for this purpose as in Eq. (8).

$$x_{norm} = \frac{1}{1 + e^{-\lambda x_i}} \tag{8}$$

where $x_{\text{norm}}$ is the normalized price, $x_i$ is the current day closing price, $\lambda = 1/x_{\max}$ and $x_{\max}$ is the maximum price of the respective training set.

To establish the performance of the proposed model, few other forecasting models are considered here in this experimental study such as MLP, RBFNN, and Multiple Linear Regression (MLR). The normalized training sets and test sets presented to the proposed model are also presented to these three models to obtain the respective error in forecasting and these errors in forecasting are compared. The less the error the better is the performance of the model considered. The parameters considered for experimentation of different models are presented in Table 1.

Depending on the number of working days in a year, the total data obtained for each index may be different; accordingly, different numbers of training and test sets are generated for the 15 year period considered for this experiment. Table 2 shows the number of training sets generated for each dataset and for each prediction category.

The error signals obtained from all the data sets out of MLP, MLR, RBFNN and ACRNN for one-day-ahead, one-week-ahead and one-month-ahead forecasting are presented in Tables 3–5 respectively.

In one-day-ahead forecasting, it is observed that the results of ACRNN are much better in comparison with MLP, RBFNN, and MLR technique in case of all the stock index data considered. Further significantly low standard deviation values of ACRNN also depict that the variation in forecasting is also low and the proposed model is consistent in its performance. The average results are made bold which are later used for further evaluation of performance.

**Table 1** Simulated parameters for all forecasting models.

| Parameter | ACRNN | MLP | RBFNN | MLR |
|---|---|---|---|---|
| Learning rate ($\alpha$) | NA | 0.3 | NA | NA |
| Momentum factor ($\mu$) | NA | 0.5 | NA | NA |
| No. of iteration for 1st training set | 100 | 500 | NA | NA |
| No. of iteration for subsequent training | 5 | 20 | NA | NA |
| Reactant number | 50 | NA | NA | NA |
| Center selection | NA | NA | Randomly sampled from the training set | NA |
| Kernel function/ activation | Sigmoid | Sigmoid | Gaussian | Sigmoid |
| Weight updated/found | ACRO | Gradient descent | Pseudo inverse | Pseudo inverse |

**Table 2** Number of windows generated for each category of prediction.

| Stock indices | One-day-ahead | One-week-ahead | One-month-ahead |
|---|---|---|---|
| BSE | 3738 | 3730 | 3707 |
| DJIA | 3775 | 3767 | 3744 |
| NASDAQ | 3771 | 3763 | 3740 |
| FTSE | 3904 | 3896 | 3873 |
| TAIEX | 3706 | 3698 | 3675 |
| S&P 500 | 3771 | 3763 | 3741 |
| LSE | 3498 | 3490 | 3468 |

**Table 3** Error signals generated by ACRNN, MLP, RBFNN and MLR forecasting models for one-day-ahead prediction.

| Stock indices | | ACRNN | MLP | RBFNN | MLR |
|---|---|---|---|---|---|
| BSE | Minimum | 0.000001 | 0.000201 | 0.000275 | 0.000004 |
| | Maximum | 0.024535 | 0.028213 | 0.026729 | 0.604622 |
| | Average | **0.007640** | **0.020932** | **0.013704** | **0.032537** |
| | Std. deviation | 0.001821 | 0.007014 | 0.005512 | 0.185269 |
| DJIA | Minimum | 0.000003 | 0.000137 | 0.000030 | 0.000001 |
| | Maximum | 0.020505 | 0.028503 | 0.032611 | 0.726500 |
| | Average | **0.002501** | **0.009359** | **0.005508** | **0.223583** |
| | Std. deviation | 0.002304 | 0.070012 | 0.007596 | 0.185136 |
| NASDAQ | Minimum | 0.000074 | 0.000352 | 0.000008 | 0.000005 |
| | Maximum | 0.0303631 | 0.286615 | 0.041363 | 0.145339 |
| | Average | **0.008478** | **0.042974** | **0.013964** | **0.056570** |
| | Std. Deviation | 0.002806 | 0.012705 | 0.008697 | 0.299219 |
| FTSE | Minimum | 0.000001 | 0.000033 | 0.000089 | 0.000008 |
| | Maximum | 0.028219 | 0.225782 | 0.030006 | 0.279426 |
| | Average | **0.003752** | **0.009671** | **0.008265** | **0.052170** |
| | Std. deviation | 0.003638 | 0.070233 | 0.005559 | 1.384871 |
| TAIEX | Minimum | 0.001166 | 0.000025 | 0.000058 | 0.000003 |
| | Maximum | 0.039852 | 0.286401 | 0.040783 | 0.128334 |
| | Average | **0.006651** | **0.010445** | **0.013155** | **0.380917** |
| | Std. Deviation | 0.003810 | 0.070422 | 0.007903 | 0.211121 |
| S&P 500 | Minimum | 0.000001 | 0.000013 | 0.000074 | 0.000007 |
| | Maximum | 0.018269 | 0.080483 | 0.027109 | 0.332520 |
| | Average | **0.002374** | **0.027202** | **0.010182** | **0.042165** |
| | Std. deviation | 0.002234 | 0.019505 | 0.006013 | 0.621658 |
| LSE | Minimum | 0.000004 | 0.000008 | 0.000266 | 0.000087 |
| | Maximum | 0.032880 | 0.091409 | 0.031715 | 0.123983 |
| | Average | **0.001861** | **0.032194** | **0.011180** | **0.066666** |
| | Std. deviation | 0.002452 | 0.027187 | 0.006602 | 0.422523 |

As expected, the result of one-week-ahead prediction is slightly inferior to that of one-day-ahead prediction. But the results of ACRNN are much better than MLP, RBFNN and MLR for all data sets as before. In some cases the standard deviation values of ACRNN are not good enough as in case of one-day-ahead prediction, but it is better than MLP and MLR models; however RBFNN shows slightly better standard deviation value for one-week-ahead forecasting.

The results of one-month-ahead forecasting are relatively inferior to those of the other two categories. But the prediction accuracy of ACRNN is better than MLP, RBFNN, and MLR models for all the data sets. The standard deviation values of RBFNN are found slightly better than those of ACRNN in majority of cases. However, the standard deviation values of ACRNN are much better than MLP and MLR models.

It can be observed from Tables 3–5 that, the proposed ACRNN forecasting model generates better prediction accuracy in comparison with MLP, RBFNN, and MLR for all data sets in one-day-ahead, one-week-ahead, and one-month-ahead prediction. Further, to find the quantum of gain in accuracy by ACRNN over MLP, RBFNN, and MLR, percentage gain in accuracy is evaluated by Eq. (9):

$$\text{Gain in accuracy} = \frac{(\text{MAE of existing model} - \text{MAE of ACRNN model})}{\text{MAE of existing model}} \times 100\% \tag{9}$$

The percentage gain in accuracy by ACRNN over MLP, RBFNN, and MLR for different data set is presented in Fig. 14 for one-day-ahead, Fig. 14 for one-month-ahead and Fig. 16 for one-month-ahead forecasting, respectively.

From Fig. 14, it can be revealed that the percentage gain in accuracy by ACRNN is significant over all other models and in case of all datasets with a minimum gain of 36.32% in TAIEX data for MLP and maximum gain of 98.88% in case of DJIA data for MLR model.

Fig. 15 shows that in case of one-month-ahead forecasting quite good percentage gain in accuracy is obtained like Fig. 12 with a minimum gain of 31.33% in case of DJIA data for MLP model and maximum gain of 99.08% in case of FTSE data for MLR model.

It can be revealed form Fig. 16 that the percentage gain in accuracy by ACRNN is minimal i.e. 2.25% only in case of NASDAQ data for RBFNN model; otherwise, the gain is moderately good ranging from 27.20% for TAIEX data with MLP model to 98.00% for DJIA data with MLR model.

The average gain in accuracy by ACRNN over MLP, RBFNN, and MLR for seven different stock indices has been presented in Fig. 17. Average percentage gain in accuracy by ACRNN is always maximum over MLR, whereas next best gain in accuracy is over MLP for short term and long term predictions and over RBNN for medium term predictions.

In case of the financial forecasting, prediction of an absolute error is not good enough for the investors or the traders

**Table 4** Error signals generated by ACRNN, MLP, RBFNN and MLR forecasting models for one-week-ahead prediction.

| Stock indices | | ACRNN | MLP | RBFNN | MLR |
|---|---|---|---|---|---|
| BSE | Minimum | 0.000002 | 0.001100 | 0.007929 | 0.001509 |
| | Maximum | 0.252044 | 0.337627 | 0.086901 | 0.258028 |
| | Average | **0.021766** | **0.105256** | **0.051886** | **0.318371** |
| | Std. deviation | 0.059013 | 0.058318 | 0.024600 | 0.661832 |
| DJIA | Minimum | 0.001000 | 0.000572 | 0.000298 | 0.004216 |
| | Maximum | 0.023550 | 0.300406 | 0.099427 | 0.200303 |
| | Average | **0.007752** | **0.011289** | **0.064961** | **0.540421** |
| | Std. deviation | 0.060362 | 0.083350 | 0.025341 | 0.321572 |
| NASDAQ | Minimum | 0.000005 | 0.001003 | 0.000180 | 0.000082 |
| | Maximum | 0.251055 | 0.310421 | 0.107554 | 0.231468 |
| | Average | **0.020818** | **0.100306** | **0.047069** | **0.730785** |
| | Std. deviation | 0.059627 | 0.085170 | 0.024087 | 0.317837 |
| FTSE | Minimum | 0.000215 | 0.000353 | 0.002727 | 0.001048 |
| | Maximum | 0.254404 | 0.333771 | 0.081412 | 0.025583 |
| | Average | **0.007985** | **0.015275** | **0.050017** | **0.873451** |
| | Std. deviation | 0.059643 | 0.082148 | 0.016696 | 0.542483 |
| TAIEX | Minimum | 0.000001 | 0.001003 | 0.000160 | 0.000015 |
| | Maximum | 0.256054 | 0.347520 | 0.086622 | 0.413010 |
| | Average | **0.032732** | **0.110249** | **0.082715** | **0.440463** |
| | Std. Deviation | 0.060122 | 0.082653 | 0.016640 | 0.169659 |
| S&P 500 | Minimum | 0.000001 | 0.000042 | 0.000017 | 0.000012 |
| | Maximum | 0.021263 | 0.081203 | 0.027109 | 0.282520 |
| | Average | **0.008304** | **0.046282** | **0.032152** | **0.049163** |
| | Std. deviation | 0.004123 | 0.014805 | 0.006212 | 0.062165 |
| LSE | Minimum | 0.000010 | 0.000015 | 0.000076 | 0.000087 |
| | Maximum | 0.042180 | 0.091809 | 0.021311 | 0.133933 |
| | Average | **0.006265** | **0.042804** | **0.047115** | **0.076586** |
| | Std. deviation | 0.003473 | 0.027187 | 0.001682 | 0.382525 |

to take a decision. However, prediction of the market trend i.e. the index price will go up or will fall below the current level is rather an important information to facilitate the decision making process. In fact the information on the magnitude of deviation from the current price level adds to this decision process. This metric is called as prediction of change in direction (POCID) and can be represented as in Eq. (10):

$$\text{POCID} = \frac{\sum_{i=1}^{N} Trend_i}{N} * 100 \tag{10}$$

where

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

$x_{i-1}$ is the current index price, $x_i$ is the actual index price for next term, $\hat{x}_{i-1}$ is the current predicted index price, and $\hat{x}_i$ is the predicted index price for the next term.

This measure gives an account of number of correct directions when predicting the next closing prices in the financial time series. The ideal value of POCID for a perfect predictor is closer to 100 and the closer the values to 100 the more accurate is the prediction model. The POCID values for one-day-ahead, one-week-ahead and one-month-ahead are shown in Tables 6–8.

For short term forecasting, the proposed model achieves a maximum POCID value of 93.3% for TAIEX data set and minimum of 75% for NASDAQ. For medium term time horizon, ACRNN has a minimum POCID value of 73.3% and a maximum value of 86.5%. Similarly, in case of long term prediction it achieves POCID values in the range of 72.7–85.3%. It can be observed that in all cases the proposed model achieves a better POCID value as compared with that of other models. Further, to find the gain in POCID by ACRNN over MLP, RBFNN, and MLR, percentage gain in POCID is evaluated by Eq. (11):

$$\text{Gain in POCID} = \frac{(\text{POCID of existing model} - \text{POCID of ACRNN model})}{\text{POCID of existing model}} \times 100\% \tag{11}$$

The percentage gain in POCID by ACRNN over MLP, RBFNN, and MLR for different data set is presented in Fig. 18 for one-day-ahead, Fig. 19 for one-month-ahead and Fig. 20 for one-month-ahead forecasting, respectively.

Form Fig. 18 it can be observed that the proposed model obtains better POCID gain over other forecasting models. As compared to MLP, it has a minimum gain in POCID value of 7.2% in case of S&P 500 and maximum of 42.62% in case of BSE. When compared to RBFNN, it has a POCID gain ranging from 6.02% to 29.58%. Similarly, it has a minimum gain of 12.65% and maximum of 55.5% over MLR model.

From Fig. 19 it can be revealed that, ACRNN has a performance gain in the range of 12% to 40% over MLP, 13.82% to

**Table 5** Error Signals generated by ACRNN, MLP, RBFNN and MLR forecasting models for one-month-ahead prediction.

| Stock indices | | ACRNN | MLP | RBFNN | MLR |
|---|---|---|---|---|---|
| BSE | Minimum | 0.000007 | 0.000013 | 0.005136 | 0.003536 |
| | Maximum | 0.027982 | 0.140161 | 0.156382 | 0.704581 |
| | Average | **0.090426** | **0.131675** | **0.167259** | **0.506175** |
| | Std. deviation | 0.013706 | 0.122531 | 0.043239 | 0.371826 |
| DJIA | Minimum | 0.000002 | 0.000015 | 0.002881 | 0.004542 |
| | Maximum | 0.052850 | 0.413750 | 0.177939 | 0.038143 |
| | Average | **0.013976** | **0.120353** | **0.096865** | **0.701034** |
| | Std. Deviation | 0.070316 | 0.125533 | 0.053799 | 0.655197 |
| NASDAQ | Minimum | 0.000061 | 0.000037 | 0.003017 | 0.002013 |
| | Maximum | 0.028736 | 0.427573 | 0.178962 | 0.834681 |
| | Average | **0.095755** | **0.147532** | **0.097962** | **0.878752** |
| | Std. deviation | 0.062937 | 0.123604 | 0.051237 | 0.078352 |
| FTSE | Minimum | 0.000051 | 0.000215 | 0.000021 | 0.001048 |
| | Maximum | 0.280242 | 0.418102 | 0.131229 | 0.025883 |
| | Average | **0.020963** | **0.130065** | **0.092559** | **0.908325** |
| | Std. deviation | 0.071305 | 0.120527 | 0.037833 | 0.754352 |
| TAIEX | Minimum | 0.000032 | 0.000077 | 0.001451 | 0.000015 |
| | Maximum | 0.024965 | 0.423725 | 0.192830 | 0.813215 |
| | Average | **0.098145** | **0.134824** | **0.137054** | **0.830415** |
| | Std. deviation | 0.070700 | 0.121204 | 0.039764 | 0.925403 |
| S&P 500 | Minimum | 0.000005 | 0.000050 | 0.000057 | 0.000055 |
| | Maximum | 0.022500 | 0.031289 | 0.027220 | 0.281545 |
| | Average | **0.010030** | **0.067284** | **0.059159** | **0.089158** |
| | Std. deviation | 0.004321 | 0.014692 | 0.003718 | 0.052016 |
| LSE | Minimum | 0.000012 | 0.000005 | 0.000416 | 0.000055 |
| | Maximum | 0.042083 | 0.046802 | 0.031357 | 0.150930 |
| | Average | **0.008266** | **0.071002** | **0.071711** | **0.096085** |
| | Std. deviation | 0.004722 | 0.032188 | 0.001283 | 0.384526 |

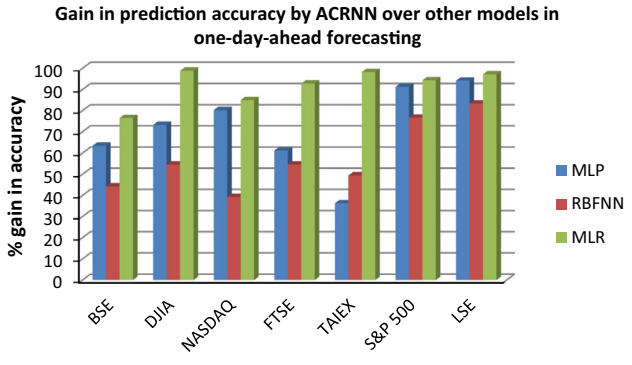



**Figure 14** A comparison of percentage gain in accuracy by ACRNN over MLP, RBFNN, and MLR in short term prediction.

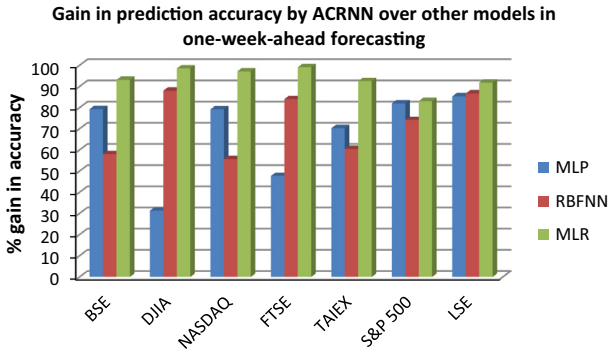



**Figure 15** A comparison of percentage gain in accuracy by ACRNN over MLP, RBFNN, and MLR in medium term prediction.

39.51% over RBFNN and 23.52% to 61.53 over MLR model. From Fig. 20 it can be observed that, ACRNN has a performance gain in the range of 9.52% to 40.99 over MLP, 10.27% to 27.14% over RBFNN and 17.34% to 49.64% over MLR forecasting model. The average % gain in POCID values by ACRNN over other models is shown in Fig. 21.

According to efficient market hypothesis, the stock prices perform a random walk. It is not possible for the market participants and common investors to predict the exact market trend because all future prices do not follow any patterns or trends. As new information occurs randomly, stock price variations are random and accurate prediction is difficult. Hence a model which is able to predict the direction of the market trend correctly along with the price may be beneficial to the investors. It may be observed from Tables 6–8 that, the ACRNN model shows much better POCID value as compared to other models. The POCID metric of the proposed model has the

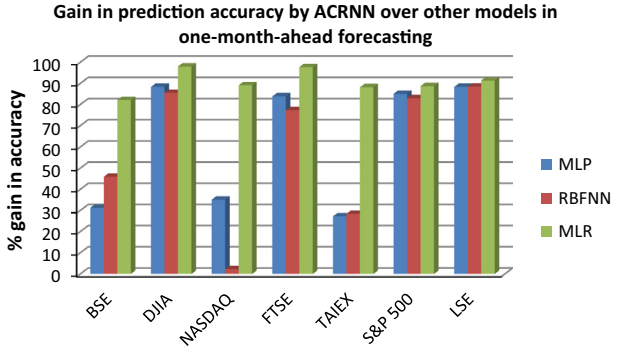



**Figure 16** A comparison of percentage gain in accuracy by ACRNN over MLP, RBFNN, and MLR in long term prediction.

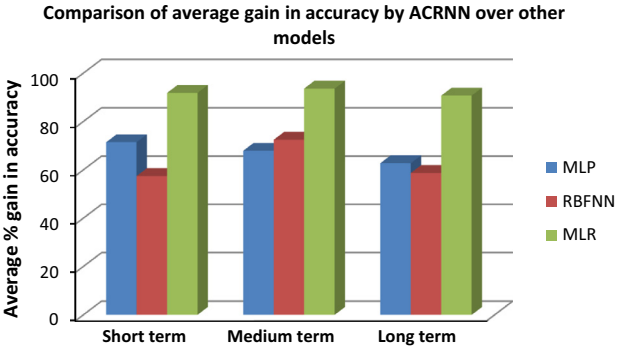



**Figure 17** Comparison of average percentage gain in accuracy by ACRNN over MLP, RBFNN and MLR for one-day-ahead, one-week-ahead and one-month-ahead prediction of stock index values.

**Table 6** Performance comparison of POCID values for one-day-ahead forecasting.

| Stock index | POCID | | | |
|---|---|---|---|---|
| | ACRNN | MLP | RBFNN | MLR |
| BSE | 87 | 61 | 72 | 58 |
| DJIA | 76.5 | 59.5 | 64 | 55 |
| NSDAQ | 75 | 60.2 | 60.5 | 51 |
| FTSE | 83 | 64.5 | 73 | 59.5 |
| TAIEX | 93.3 | 69.5 | 72 | 60 |
| S&P 500 | 89 | 83 | 82.7 | 79 |
| LSE | 88 | 81.5 | 83 | 74 |

**Table 7** Performance comparison of POCID values for one-week-ahead forecasting.

| Stock index | POCID | | | |
|---|---|---|---|---|
| | ACRNN | MLP | RBFNN | MLR |
| BSE | 84 | 60 | 63 | 52 |
| DJIA | 78.5 | 58.7 | 66 | 58 |
| NSDAQ | 73.3 | 60 | 62.7 | 58.2 |
| FTSE | 80.5 | 64.3 | 65 | 56.5 |
| TAIEX | 86.5 | 62.5 | 62 | 60.5 |
| S&P 500 | 84 | 75 | 73.8 | 68 |
| LSE | 85.3 | 73 | 73.5 | 67 |

**Table 8** Performance comparison of POCID values for one-month-ahead forecasting.

| Stock index | POCID | | | |
|---|---|---|---|---|
| | ACRNN | MLP | RBFNN | MLR |
| BSE | 74 | 60.5 | 61.7 | 53 |
| DJIA | 75.5 | 55.9 | 67 | 52.3 |
| NSDAQ | 72.7 | 55 | 60.4 | 52.2 |
| FTSE | 80.1 | 62.5 | 63 | 55 |
| TAIEX | 85.3 | 60.5 | 68.5 | 57 |
| S&P 500 | 83 | 71 | 71 | 69.5 |
| LSE | 80.5 | 73.5 | 73 | 68.6 |

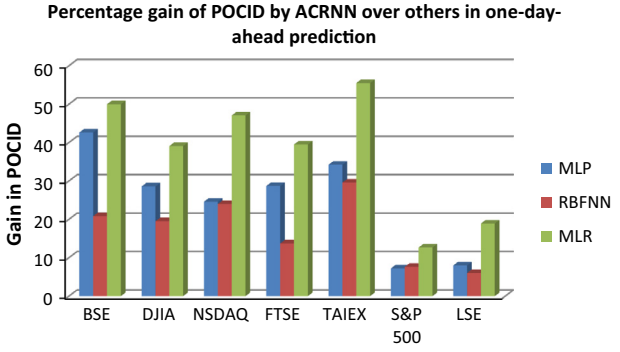



**Figure 18** A comparison of percentage gain in POCID by ACRNN over MLP, RBFNN, and MLR in short term prediction.

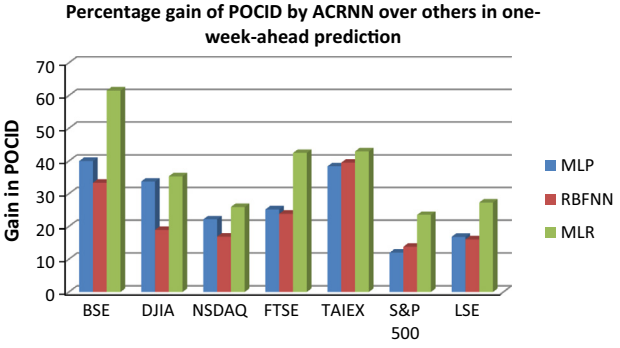



**Figure 19** A comparison of percentage gain in POCID by ACRNN over MLP, RBFNN, and MLR in medium term prediction.

great results. On an average it shows that in eighty percent of the time, decisions were correct when forecasting the market trend. So, the model is claimed to be able to find the correct direction of stock trend whether the market movement is upward or downward. As previously mentioned, the correct prediction of market movement is much beneficial for naïve investors as well as financial managers. They can take proper decisions whether to hold or sell greatly.

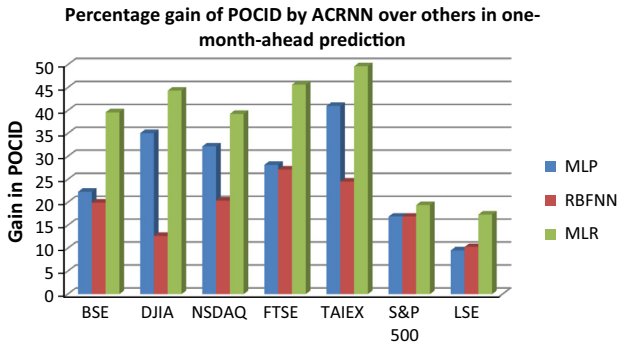



**Figure 20** A comparison of percentage gain in POCID by ACRNN over MLP, RBFNN, and MLR in long term prediction.

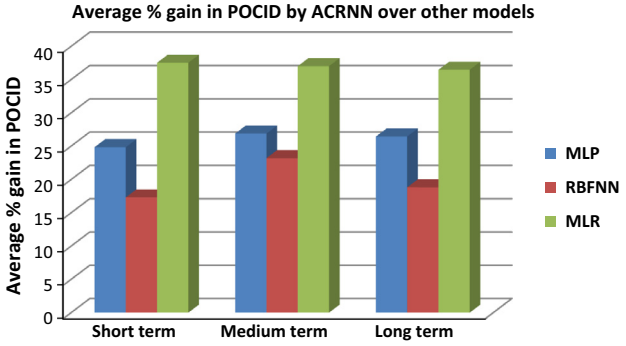



**Figure 21** Comparison of average percentage gain in POCID by ACRNN over MLP, RBFNN and MLR for one-day-ahead, one-week-ahead and one-month-ahead prediction of stock index values.

## 6. Conclusions

To capture the nonlinearity and high volatility in the stock market, this paper proposes an adaptive hybrid artificially chemical reacted neural network for prediction of stock market indices. In the proposed ACRNN approach, the global search capability of a natural chemistry inspired optimization has been incorporated with the high nonlinearity capturing capability of MLP model. The proposed model is tested with seven stock index values such as BSE, DJIA, NASDAQ, TAIEX, FTSE, S&P 500, and LSE. Usually the stock index values are influenced by political and economic scenario of the respective country as well as by the global scenario. To ensure that the proposed model copes up with the changing scenarios, dataset is collected for a period of 15 years and tested. Further, at time a technique developed for short term prediction fails for medium or long term predictions. To overcome such a scenario, here the proposed technique is used for prediction of short term, medium term as well as long term predictions, to ensure that it can work efficiently for different categories of prediction.

To map the activation of the MLP network, Sigmoidal function is used here to normalize the historical data obtained from the different stock markets. The sliding window concept is used here to prepare the data for training and testing. After the first phase of training, the parameters are used adaptively for subsequent training to efficiently reduce the time required for the training phase. Accuracy and POCID values of the proposed model are compared with BPN, RBFNN and MLR and it is found that the proposed model has a significant gain over all other models for all the stock index values considered.

The work may be extended by considering other neural network models and evolutionary optimization techniques. In the future, higher order neural networks may be investigated as they are characterized with good generalization abilities due to higher order terms. Also, hybridization of some robust evolutionary search algorithms may be incorporated to the proposed forecasting model.

## References

[1] Abdoh TH, Jouhare H. The investigation of efficiency of stock price index of T.S.E. J Financial Res 1996;13:11–2.
[2] Oh KJ, Kim K-J. Analyzing stock market tick data using piecewise non linear model. Expert Syst Appl 2002;22:249–55.
[3] Wang Y. Mining stock prices using fuzzy rough set system. Expert Syst Appl 2003;24:13–23.
[4] Huang C-J, Yang D-X, Chuang Y-T. Application of wrapper approach and composite classifier to the stock trend prediction. Expert Syst. Appl. 2008;34:2870–8.
[5] Liu H-C, Lee Y-H, Lee M-C. Forecasting china stock markets volatility via GARCH models under skewed-GED distribution. J Money Invest Bank 2009:5–14.
[6] Ravichandran KS, Thirunavukarasu P, Nallaswamy R, Babu R. Estimation on return on investment in share market through ANN. J Theor Appl Inform Technol 2007;3:44–54.
[7] Box GEP, Jenkins GM. Time series analysis-forecasting and control. San Francisco: Holden-Day Inc.; 1976.
[8] Miller AS, Blott BH, Hames TK. Review of neural network applications in medical imaging and signal processing. Med Biol Eng Comput 1992:449–64.
[9] Maglaveras N, Stamkopoulas T, Pappas C, Strintzis M. An Adaptive back-propagation neural network for real time ischemia episodes detection Development and performance using the European ST-T databases. IEEE Trans Biomed Eng 1998:805–13.
[10] White H. Economic prediction using neural networks: the case of IBM daily stock returns. Proceedings of the second annual ieee conference on neural networks, 1998; 2: 451–8.
[11] Chiang WC, Urban TL, Baldridge GW. A neural network approach to mutual fund net asset value forecasting. Omega 1996;24(2):205–15.
[12] Romahi Y, Shen Q. Dynamic financial forecasting with automatically induced fuzzy associations. In: Proceedings of the 9th international conference on Fuzzy systems; 2000. p. 493–8.
[13] Abraham A, Nath B, Mahanti PK. Hybrid intelligent systems for stock market analysis. Proceedings of the international conference on computational science. Springer; 2001.
[14] Nayak SC, Misra BB, Behera HS. Stock index prediction with neuro-genetic hybrid techniques. Int J Comput Sci Inform, IJCSI 2012;2(3):27–34.
[15] Wolpert DH, Macready WG. No free lunch theorems for optimization. IEEE Trans Evol Comput 1997;1(1):67–82.
[16] Kung L, Yu S. Prediction of index futures returns and the analysis of financial spillovers-A comparison between GARCH and the grey theorem. Eur. J. Oper. Res. 2008;186:1184–200.
[17] Yu L, Lai Wang SK. A neural-network-based nonlinear meta-modeling approach to financial time series forecasting. Appl Soft Comput 2009;9:563–74.
[18] Aiken M, Bsat M. Forecasting market trends with neural networks. Inform Syst Manage 1999;16:42–9.
[19] Aminian F, Suarez E, Aminian M, Walz D. Forecasting economic data with neural networks. Comput Econ 2006;28:71–88.

[20] Mostafa Mohamed M. Forecasting stock exchange movements using neural networks: Empirical evidence from Kuwait. Expert Syst Appl 2010;37:6302–9.
[21] Zhuo W, Li-Min J, Yong Q, Yan-hui W. Railway passenger traffic volume prediction based on neural network. Appl Artificial Intell 2007;21:1–10.
[22] Nam K, Yi J. Predicting airline passenger volume. J Business Forecast Methods Syst 1997;16:14–7.
[23] Mostafa Mohamed M. Forecasting the Suez Canal traffic: a neural network analysis. Mar Policy Manage 2004;31:139–56.
[24] Darbellay G, Slama M. Forecasting the short-term demand for electricity: do neural networks stand a better chance. Int J Forecast 2000;16:71–83.
[25] Videnova I, Nedialkova D, Dimitrova M, Popova S. Neural networks for air pollution forecasting. Appl Artificial Intell 2006;20:493–506.
[26] Calderon T, Cheh J. A roadmap for future neural networks research in auditing and risk assessment. Int J Account Inform Syst 2002;3:203–36.
[27] Swicegood P, Clark J. Off-site monitoring systems for prediction bank underperformance: a comparison of neural networks, discriminant analysis, and professional human judgment. Int J Intell Syst Account, Finance Manage 2001;10:169–86.
[28] Majhi R, Panda G, Majhi B, Sahoo G. Efficient prediction of stock market indices using adaptive bacterial foraging optimization (ABFO) and BFO based techniques. Expert Syst Appl 2009;36:10097–104.
[29] Boyacioglu MA, Avci D. An adaptive network-based fuzzy inference system (ANFIS) for the prediction of stock market return: the case of the istanbul stock exchange. Expert Syst Appl 2010;37:7908–12.
[30] Yu L, Zhang Y. Evolutionary fuzzy neural networks for hybrid financial prediction. IEEE Trans Syst, Man, Cybernetics – Part C: Appl Rev 2005;35(2) [May 2005].
[31] Hassan MdR, Nath B. Stock market forecasting using hidden markov model: a new approach. Proc IEEE international conference on intelligent systems design and applications. IEEE Press; 2005.
[32] Aboueldahab T, Fakhreldin Md. Prediction of stock market indices using hybrid genetic algorithm/particle swarm optimization with perturbation term. In: International conference on swarm intelligence (ICSI 2011). Cergy, France; June 14–15, 2011.
[33] Gurusen E, Kayakutlu G, Daim TU. Using artificial neural network models in stock market index prediction. Expert Syst Appl 2011;38:10389–97.
[34] Nayak SC, Misra BB, Behera HS. Index prediction using neuro-genetic hybrid networks: a comparative analysis of performance. In: IEEE conference on computing, communication and application, (ICCCA 2012), http://dx.doi.org/10.1109/ICCCA.2012.6179215.
[35] AlRashidi M, EI-Hawary M. A survey of particle swarm optimization applications in electric power system. IEEE Trans Evol Comput 2009;13(4):913–8.
[36] Shin SY, Lee IH, Kim D, Zhang BT. Multiobjective evolutionary optimization of DNA sequences for reliable DNA computing. IEEE Trans Evol Comput 2005;9(2):143–58.
[37] Yu L, Chen H, Wang S, Lai KK. Evolving least square support vector machines for stock market trend mining. IEEE Trans Evol Comput 2009;13(1):87–102.
[38] Zanaty EA. Support vector machines (SVMs) versus multilayer perception (MLP) in data classification. Egyptian Inform J 2012;13:177–83.
[39] Mohamed AW, Sabry HZ, Abd-Elaziz T. Real parameter optimization by an effective differential evolution algorithm. Egyptian Inform J 2013;14:37–53.
[40] Samui P. Vector machine techniques for modeling of seismic liquefaction data. Ain Shams Eng J 2014;5:355–60.
[41] Peng H, Ozaki T, Haggan-Ozaki V, Toyoda Y. A parameter optimization method for the radial basis function type models. IEEE Trans Neural Networks 2003;14(2):432–8.
[42] Karayiannis NB, Randolph-Gips MM. On the construction and training of reformulated radial basis function neural networks. IEEE Trans Neural Networks 2003;14(4):835–46.
[43] Harpham C, Dawson CW. The effect of different basis functions on a radial basis function network for time series prediction: a comparative study. Neuro Comput 2006;69(16):2161–70.
[44] Du H, Zhang N. Time series prediction using evolving radial basis function networks with new encoding scheme. Neurocomputing 2008;71(7–9):1388–400.
[45] Sun B, Li T. Forecasting and identification of stock market based on modified RBF neural network. In: IEEE 17th international conference on industrial engineering and engineering management (IE&EM); 2010. p. 424–7.
[46] Shen W, Guo X, Wu C, Wu D. Forecasting stock indices using radial basis function neural networks optimized by artificial fish swarm algorithm. Knowl-Based Syst 2011;24:378–85.
[47] Zhang Q, He X. RBF network based on genetic algorithm optimization for nonlinear time series prediction. Proc Int Sympos Circuits Syst 2003;5:693–6.
[48] Feng Y, Zhao H. Price forecasting algorithm for coal and electricity based on PSO and RBF neural network. In: IEEE international conference on control and automation; 2009. p. 1365–9.
[49] Lam AYS, Li VOK. Chemical-reaction-inspired metaheuristic for optimization. IEEE Trans Evol Comput 2010;14(3):381–99.
[50] Alatas B. ACROA: artificial chemical reaction optimization algorithm for global optimization. Expert Syst Appl 2011;38:13170–80.
[51] Alatas B. A novel chemistry based metaheuristic optimization method for mining of classification rules. Expert Syst Appl 2012;39:11080–8.
[52] Lam AYS, Li VOK. Chemical reaction optimization: a tutorial. Memtic Comput 2012;4:3–17. http://dx.doi.org/10.1007/s12293-012-0075-1.
[53] Lam AYS, Li VOK, Yu JJQ. Real-coded chemical reaction optimization. IEEE Trans Evol Comput 2012;16(3):339–53.
[54] Lam AYS, Xu J, Li VOK. Chemical reaction optimization for population transition in peer-to-peer live streaming. The In: IEEE congress on evolutionary computation (CEC). Barcelona, Spain; 18–23 July 2010. p. 1–8.
[55] Pan B, Lam AYS, Li VOK. Network coding optimization based on chemical reaction optimization. In: IEEE global telecommunications conference (GLOBECOM 2011); 2011. p. 1–5.
[56] Xu J, Lam AYS, Li VOK. Chemical reaction optimization for grid scheduling problem. In: IEEE international conference on communications (ICC, 2010); 2010. p. 1–5.
[57] Xu J, Lam AYS, Li VOK. Chemical reaction optimization for task scheduling in grid computing. IEEE Trans Parallel Distributed Syst 2011;22(10):1624–31.
[58] Yu JJQ, Lam AYS, Li VOK. Evolutionary artificial neural network based on chemical reaction optimization. In: IEEE congress on evolutionary computation (CEC, 2011); 2011. p. 2083–90.
[59] Lam AYS, Li VOK. Chemical reaction optimization for cognitive radio spectrum allocation. In: IEEE global telecommunications conference (GLOBECOM, 2010); 2010. p. 1–5.
[60] Xu J, Lam AYS, Li VOK. Portfolio selection using chemical reaction optimization. World Acad Sci Eng Technol 2011;5:402–7.
[61] Nayak J, Naik B, Behera HS. A novel chemical reaction optimization based higher order neural network (CRO-HONN) for nonlinear classification. Ain Shams Eng J 2015;6:1069–91.
[62] Truong TK, Li K, Xu Y. Chemical reaction optimization with greedy strategy for the 0–1 knapsack problem. Appl Soft Comput 2013;13:1774–80.

[63] Nayak SC, Misra BB, Behera HS. Hybridizing chemical reaction optimization and artificial neural network for stock future index forecasting. In: International conference on emerging trends and applications in computer science, IEEE, http://dx.doi.org/10.1109/ICETACS.2013.6691409.
[64] Karci A, Arslan A. Uniform population in genetic algorithms IU. J Electr Electr 2002;2(2):495–504.
[65] Karci A, Alatas B. Thinking capability of sapling growing up algorithm iDEAL. Lecture notes in computer science, vol. 4224. Springer-Verlag; 2006. p. 286–393.
[66] Karci A. Theory of sapling growing up algorithm. Lect Notes Comput Sci 2007;31:450–60.

**S.C. Nayak** has received a M.Tech. (Computer Science) degree from Utkal University, Bhubaneswar, India, in 2011. He is pursuing Ph.D. (Engineering) at VSS University of Technology, Burla, India. His area of research interest includes data mining, soft computing, evolutionary computation, and financial time series forecasting. He has published 12 research papers in various journals and conferences of National and International repute.

**Dr. B.B. Misra** is currently working as Dean (Research) at Silicon Institute of Technology, Bhubaneswar, India. He received Bachelor degree in Textiles from Kanpur University, India, in 1984, Master of Technology in Computer Science from Utkal University, Bhubaneswar, India, in 2002 and Ph.D. in Engineering from Biju Patnaik University of Technology, Rourkela, India, in 2011. He has done his Post Doctoral Research during 2013–2014, at AJOU University, South Korea, under the Technology Research Program for Brain Science of Yonsei University. His areas of interests include Data Mining, Sensor Network, Bioinformatics, Evolutionary Computation, Bio-inspired Computation and Computational Intelligence. He has published one book, three book chapters and more than 70 papers in different journals and conferences of National and International repute. He has been a key note speaker and session chair of different national and international conferences. Dr. Misra has more than 30 years of industrial as well as academics experiences.

**Dr. H.S. Behera** has completed his M.E. (Computer Science & Engineering) from National Institute of Technology, Rourkela, India, and Ph.D. (Engineering) from Biju Patnaik University of Technology. His area of interest includes data mining and soft-computing, software engineering, and distributed system. Dr. Behera has more than 50 research publications in National and International journals and conferences. Currently he is working as Reader, Department of Computer Science Engineering & Information Technology, VSS University of Technology, Burla, India. He has more than 18 years of experiences in teaching as well as research.