Mining of Stock Market Prices using Machine Learning and Evolutionary Computing Techniques

Report submitted to the department of

Computer Science and Engineering

of

International Institue of Information Technology, Bhubaneswar

in partial fulfilment of the requirements

for the degree of

Bachelor of Technology

by

Reeva Mishra

(Roll- B114028)

under the supervision of

Prof. Puspanjali Mohapatra

Department of Computer Science and Engineering

International Institute of Information Technology, Bhubaneswar

Bhubaneswar- 751003, India

2018

Declaration

I declare that the work presented in this thesis titled **Mining of Stock Market Prices using Machine Learning and Evolutionary Computing Techniques**, submitted to the Department of Computer Science and Engineering, IIIT Bhubaneswar, for the award of the Bachelor of Technology in Computer Science and Engineering, is my original work. I have not plagiarized or submitted the same work for the award of any other degree. In case this declaration is found incorrect, I accept that my degree may be unconditionally withdrawn.

Reeva Mishra
B114028

Dept of CSE
IIIT Bhubaneswar
Date:

Department of Computer Science and Engineering
International Institute of Information Technology
Bhubaneswar-751003, India. `www.iiit-bh.ac.in`

May 9, 2018

Certificate

This is to certify that the work in the thesis entitled **Mining of Stock Market Prices using Machine Learning and Evolutionary Computing Techniques** by ***Reeva Mishra*, B114028** is a record of an original research work carried out by her under my supervision and guidance in partial fulfilment of the requirements for the award of the degree of *Technology* in *Computer Science and Engineering*. Neither this thesis nor any part of it has been submitted for any degree or academic award elsewhere.

Prof. Puspanjali Mohapatra
Project Supervisor
Dept. of CSE, IIIT Bhubaneswar

The Coordinator
Dept. of CSE, IIIT Bhubaneswar

External Supervisor

Acknowledgment

I am thankful to various nearby and worldwide associates who have helped towards moulding this thesis. In the beginning, I might want to express my true thanks to Prof. Pushpanjali Mohapatra for her recommendation throughout my thesis work. As my supervisor, she has always encouraged me to stay concentrated on accomplishing my objective. Her perceptions and remarks helped me to build the general bearing of the research and to push ahead with research in profundity. She has helped me significantly and been a source of information. I am also thankful to all the professors of the department for their support. My true thanks to my all friends and to everyone who has provided me with kind words, a welcome ear, new plans, valuable feedback, or their invaluable time, I am truly indebted. Last, but not the least, I would like to dedicate this thesis to my family, for their love, patience, and understanding.

Reeva Mishra

B114028

Dept of CSE

IIIT Bhubaneswar

Date:

Abstract

This thesis centralizes on stock market prediction as one of the popular areas of data mining tasks. Financial time series data is often found to be very volatile, noisy, chaotic, non-linear and high dimensional. The domain of stock market prediction has gained popularity because of the increasing amount of money being invested in the stock markets. The potential of the stock market is huge. Money invested in it can result in large gains or incur huge losses. Due to this reason, businessmen, investors and industrialists are worried. They are in a need of a way that could provide them quantitative and qualitative information before taking a critical decision. Thus, the need for stock market prediction was found.

The extraction of patterns and trends from financial time series assists the businessmen and investors in taking a wise decision about the right time to buy, hold or sell stocks. To this end in view, this thesis has sought to use Machine Intelligence and Evolutionary Techniques for designing different models and optimization techniques to improve the accuracy of the prediction, thus providing reliable decisions. There are various statistical models for training neural networks but their inability to handle non-linear pattern hidden in time series data motivated us to go for a machine learning approach. Machine learning approach integrates the learning capability of a neural network and thinking capability of fuzzy model. Few popular stock datasets are experimented with various models and algorithms, hybridised and compared to find accurate results for stock datasets.

In this thesis, different types of forecasting methods have been discussed. The different types of optimization techniques Differential Evolution (DE), Particle Swarm Optimization (PSO), Whale Optimization Algorithm (WOA), Jaya and Harmony Search (HS) have been compared and was found that WOA was the superior of all. Further, an integrated training algorithm has been proposed i.e hybrid Jaya-HS. This optimization technique was first tested with a simple

Functional Link Artificial Neural Network Model (FLANN) and was found that this technique outperformed simple Jaya and HS optimization techniques. The various models and optimization techniques have been evaluated and compare by performance measures as Mean Square (MSE) and Mean Absolute Percentage Error (MAPE)

To verify the efficiency of the proposed optimization technique, it has also been tested on a more complex model i.e Locally Recurrent Neuro-Fuzzy Information System(LRNFIS). From the simulation study, we found out that LRNFIS model performed far better than the simple CFLANN model with the proposed hybrid Jaya-HS. Thus, the accuracy of the proposed model was verified on three popular datasets BSE500, DJIA and NASDAQ

Keywords: *FLANN, DE, PSO, WOA, Jaya, HS, LRNFIS, Deep Neural Net*

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Basic Concepts of Time Series Modeling

Time series modeling is a dynamic research area. The main goal of time series modeling is to carefully study the past time series records, observe the pattern and design a model that inherits the features of the time series data. The model is further used to predict the future by understanding the past. The more features the model inherits of the time series data, the better forecast it will be able to make. Thus, many researchers have put all their time and effort to develop effective models to improve the accuracy of the forecast. [1]

There can be 4 components of time series modeling. They are as follows:

- Trend: Any time series data undergoes a cycle with 3 major phases- increase, decrease or stagnate. In a bigger picture, we can say it is a long-term movement in a time series [2].

- Cyclical Variations: These are medium-term changes in the series [2]. They are often observed in economic and financial time series and extend over a long period of time, usually for a few years. A popular example of this kind of variation is a business cycle that consists of 4 phases: expansion, peak, recession and trough.

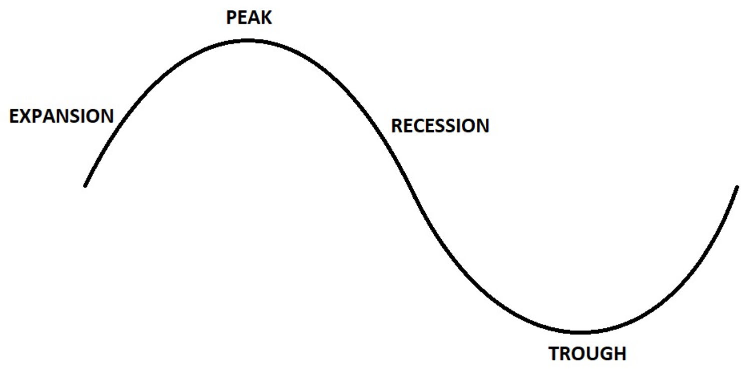


Figure 1.1: The four phases of the business cycle

- Seasonal Variations: They are mostly fluctuations in a time series within a year during the season. These type of variations are often caused by climate and weather conditions, habits of people, etc.
- Irregular or Random Variations: These are caused by unpredictable incidents and they do not repeat in a particular pattern. These type of variations are often caused by incidents like natural calamities or war.

Time Series forecasting can be used for prediction of the stock market, rainfall, temperature and Forex currency rates.

1.2 Examples of Time Series Modeling

Time Series forecasting has been frequently performed in diverse domains such as business, economics, industry and science, etc. Depending on the types of the data, different ways of representation are used to study the patterns. However, the basic pattern to represent any time series data is represented by a graph where the Y-axis represent the pattern in the time series data and X-axis represents the corresponding time.

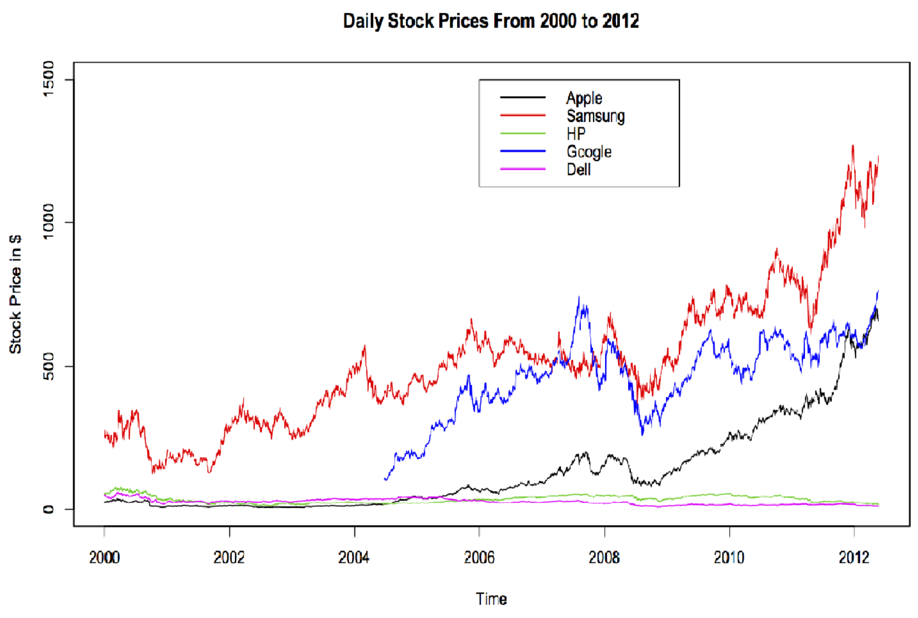


Figure 1.2: Daily Stock Prices from 2000-2012

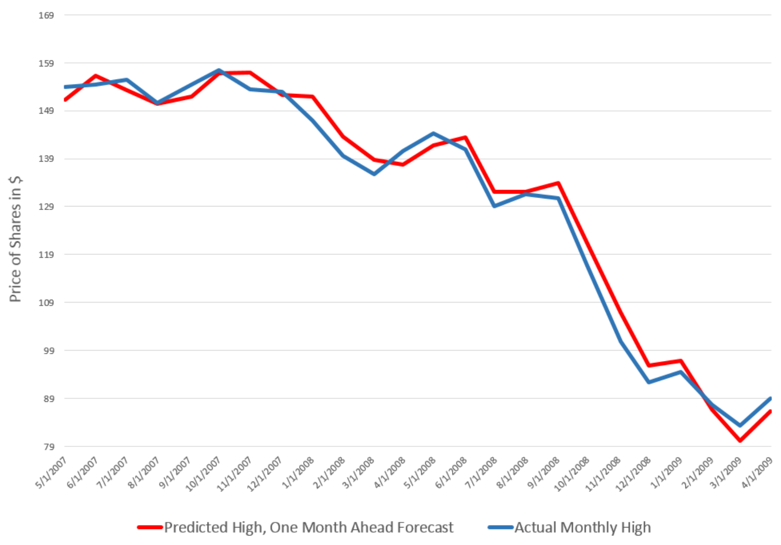


Figure 1.3: SPY One-Month ahead forecasts of High Prices with Actuals

1.3 Problem Statement

Stock Market is such an investment area where one can either earn huge gains or incur a heavy loss. Due to high volatile nature of the stocks, investors and businessmen often get confused about the right actions to take.

Financial forecasting or Stock Market prediction, in specific, has gained a lot of popularity because of the huge commercial benefits it has to offer. It has attracted industries, businessmen and investors. Large amounts of money are being invested in the stock market these days, which make the investors anxious about the future trends. A wrong decision can result in a huge loss. So, a well evaluated and predicted data can help in taking wise decisions of when to buy, hold or sell the stocks. It can help in making strategic decisions and taking precautionary measure.

Thus, the big question arises can financial data be really predicted? Financial Time series has high volatility [3]and is very difficult to predict [4]. It possesses one of the noisiest and non-stationary signals. It is affected by quantitative and qualitative factors like firm′s policies, investor′s decisions and actions of other stock markets. Although it is very difficult, it is not impossible.

1.4 Motivation

Many researchers have put their brains into financial stock market prediction and have got successful results. Many models have been designed and the research is still continuing to get the best out of the best predictions. There are three schools of theories regarding the prediction that motivates us to put our time and effort into the **Stock Market Prediction** [3] [5].

According to the first school of theory, no investor can achieve above average trading advantages based on historical and present information [6]. The second theory believes that there exists a fundamental correlation between macro-economic factors and financial conditions, and fluctuations in stock prices. The third theory states that the technical analysts try to observe trends in stock prices by taking into account past stock time series prices and volume data. [6]

1.5 Stock Datasets

There are many stock market indices out of which the most popular ones are as follows:

- **Dataset 1:** The SP BSE500 (The Standard Poor's Bombay Stock Exchange 500) is a stock market index established in December 2010 on Bombay Stock Exchange. In the year 2013, SP Dow Jones Indices and the Bombay Stock Exchange established their partnership to calculate and licence BSE suite of indices. The first index created by the partnership was SP BSE500 Shariah Index.
- **Dataset 2:** The DJIA (Dow Jones Industrial Average) is a stock market index that was founded in the year 1885. Initially, it evaluated how the largest 30, publicly owned companies based in the United States traded during a standard trading session in the stock market.
- **Dataset 3:** The NASDAQ (National Association of Securities Dealers Automated Quotations) is a stock market index that was founded in 1971 by National Association of Securities Dealers (NASD).

1.6 Summary

The time series is very chaotic and non-deterministic in nature. We can never predict with certainty about what will happen in future. However, we can observe the past and current trends to make a better-consolidated prediction. The features of the past and current time series data can be inculcated into a proper model to make the model understand the nature of the time series data. Further, it can be used for future forecasting and simulation.

This technique is specifically helpful where there is no satisfactory explanatory model. This can help us make a close prediction of the future.

The time series forecasting has applications in many fields like the prediction of stock data, discharge and rainfall, temperature fluctuations over a year and Forex currency rates. For example, if the stock closing price is high today, the price tomorrow is like to be high tomorrow as well.

Financial time series forecasting is in high demand owing to the huge commercial benefits it has to offer. It has become very popular among the businessmen and investors. With the help of this prediction, they become efficient at making good decisions about the right time to buy, hold or sell stocks. The three schools of theory about the predictions motivate us to work on models to predict the stock prices.

Chapter 2

Related Work

2.1 Introduction

Prediction of the stock market is an important problem in finance. Artificial Neural Networks(ANNs) have been incorporated in various financial problems such as stock exchange index prediction and bankruptcy prediction. ANNs basically mimic the learning capability of the human brain.

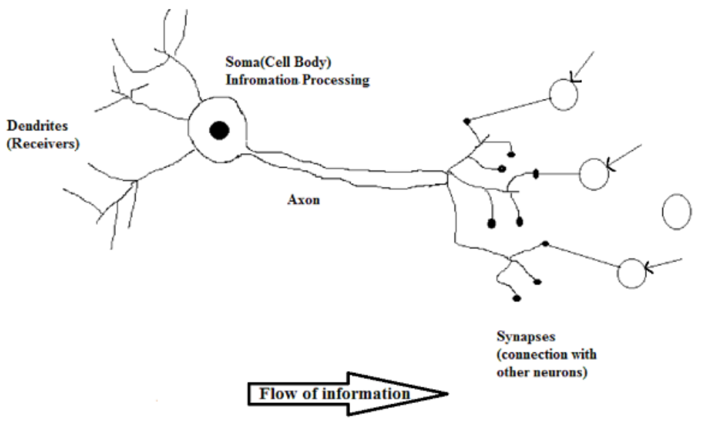


Figure 2.1: The flow of information in a biological neuron

The fundamental element of a neural network is a neuron. A neuron consists of three parts: Dendrites, Soma and Axon. The dendrites are the tree-like structures, whose each branch is connected to one neuron. The Axon is a cylindrical structure which transmits the signals from one neuron to another. At the end of the axon, the contact to the next dendrite is made through a synapse.

The inter-neuron signal transmission at synapse usually takes place through a chemical diffusion. An incoming impulse signal at a synapse is either excitatory or inhibitory, which results in helping or hindering effect. The firing condition is caused only when the excitatory signal exceeds the inhibitory signal by a certain amount in a short time span called the latent summation. The excitatory signals have a positive weight whereas an inhibitory signal has a negative weight. So, we can say a neuron undergoes firing only when the total summation of the synapses in the period of latent summation exceeds the threshold.

The mostly widely used ANNs in the prediction field are multi-layer perceptron (MLPs), which is a feedforward network with a single hidden layer. It consists of 3 layers, viz, input, hidden and an output layer. There may be more than one hidden layer [6]. The three-layer feed forward ANN architecture can be depicted as follows.

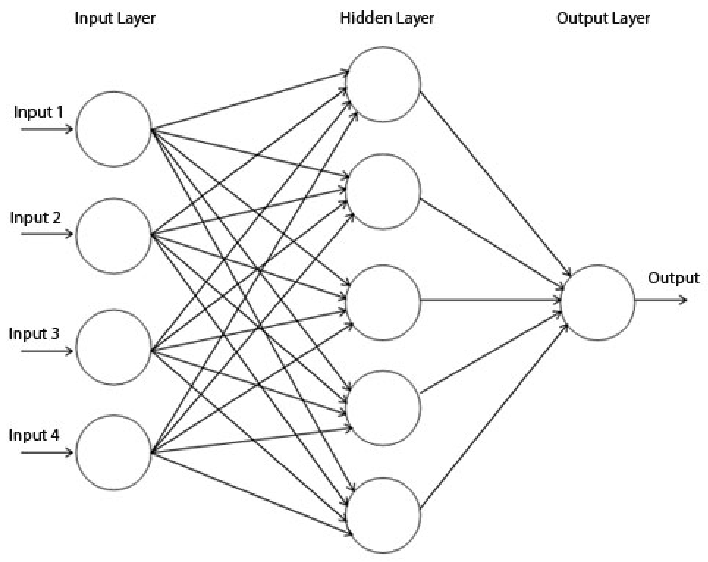


Figure 2.2: The three-layer feed forward ANN architecture

The weights associated with each node represent the connection strength i.e the influence of the node. Positive weights represent reinforcement, whereas negative weights represent inhibition. [1] The output of the model is computed by the mathematical expression:

$$y_t = \alpha_0 + \sum_{i=1}^{q} \alpha_j g(\beta_{0j} + \sum_{i=1}^{p} (\beta_{ij} y_{t-i}) + \epsilon_t, \forall t \tag{2.1}$$

where $y_{t-i}(i = 1, 2, ..., p)$ are the p inputs and y_t is the output. The integers p and q are the numbers of input and hidden nodes respectively. $\alpha_j(0, 1, 2, ..., q)$ and $\beta_{ij}(0, 1, 2, ..., p)$ for *j=(0,1,2,...,q)* are the connection weights and ϵ_t is the random stock. α_0 and β_{0j} are the bias terms. Usually, a tan hyperbolic function is applied as the non-linear activation function. Other activation functions, such as linear, sigmoid, Gaussian, etc can also be used. [1]

2.2 Normalization

In a stock data, the closing prices and volume vary within a large range. To rescale the data within the range of [0,1], we normalize the data. These normalized values are fed as inputs to the ANNs. First, the data are obtained from *www.finance.yahoo.com* and *www.bseindia.com.* Then, the inputs are normalized within a range based on the activation function. The most regular activation functions used are sigmoid and tanh functions. So, the data is normalized within a range of [0,1] using the formula:

$$X_{norm} = \frac{X_{orig} - X_{min}}{X_{max} - X_{min}} \tag{2.2}$$

where X_{norm} represents normalized value, X_{orig} represents current closing price and X_{min} and X_{max} are the minimum and maximum prices of the stock data respectively. [7]

2.3 Methods of Forecasting

As discussed in chapter 1, the main goal of time series forecasting is to rigorously study the past and current time series data to develop an appropriate model which portrays the inherent structure of the time series. This model is further used to predict the future values. Time series forecasting thus is concluded as understanding the past and the present to estimate the future.

2.3.1 Statistical Models

There are various stochastic time series models. One of the popular and most frequently used statistical model is Autoregressive Integrated Moving Average(ARIMA) model. The basic assumption made of the model is that the time series is linear and follows a specific statistical distribution. [1] [8] The subclasses of ARIMA model are:

- Autoregressive (AR): AR model specifies that the output values depend linearly on its respective previous values and an unpredictable stochastic term. It is used to explain time-varying processes in finance, statistical and economics.

- Moving Average (MA): MA model analyses data by creating several average points of subsets of the full dataset. It is also called moving mean(MM) or rolling mean.

- Autoregressive Moving Average (ARMA): ARMA model provides a very weak representation of the stationary stochastic process in form of two polynomials, one for the auto-regression(AR) and the other for the moving average(MA). [1] [8]

- Seasonal Autoregressive Moving Average (SARMA): This model was proposed by Box and Jenkins and is a successful variation of the ARMA model. It is popularly used for seasonal time series forecasting. [1] [9]

Autoregressive Conditional Heteroskedasticity (ARCH) model is a statistical model for time series data. It describes the dependency of the variance of the current error term as a function of the actual sizes of the previous model's error terms. This model is used when the error variance in a time series data follows Autoregressive(AR) model. The Generalized Autoregressive Conditional

Heteroskedasticity (GARCH) model is otherwise used if the time series data follows a Autoregressive Moving Average (ARMA) model [10] [11].

2.3.2 Neural Networks

Recently, the Artificial Neural Networks(ANNs) have gained a lot of popularity over the statistical models because of their ANNs' ability to deal with non-linear patterns hidden within the stock datasets. Thus, the Evolutionary and Soft Computing techniques have been granted to be superior to the statistical models. Few types of ANNs are as follows. [12]

- Radial Basis Function(RBF): It is an artificial neural network that uses radial functions as activation functions. The output is a result of the linear combination of the inputs and corresponding neuron's features [13].
- Multi-Layer Perceptron(MLP): It lies in the class of feedforward artificial neural network and it consists of minimum 3 layers(input, hidden and output layer). There may exist more than one hidden layer. MLP utilizes a supervised learning technique, often Back Propagation(BP). The main advantage of this model over other models is it can distinguish data that is not linearly separable.
- Recurrent Neural Network(RNN): Unlike feedforward network, RNNs are able to use their internal state memory to process the time series data. In this model, the connections between units form a directed graph along a sequence. These models exhibit a dynamic temporal behaviour.
- Time Delay Neural Network(TDNN):It is a multi-layer ANN which is basically used for two major purposes.
  - shift-invariance classification i.e classifies doesn't require segmentation prior to classification.
  - contextual modeling i.e each neuron at every layer not only receives input from activation at the previous layer but also from a pattern of

unit output.

- Local Linear Wavelet Neural Networks (LLWNN): This model is similar to Wavelet Neural Network(WNN) except for the fact that the connection weights between the hidden units and output units are replaced by a local linear model. [9]

2.3.3 Hybrid Neuro-fuzzy System

The hybrid neuro-fuzzy system is like an integrated intelligent system that combines the human-like thinking and analysis power of fuzzy system with the learning capability of the neural networks. It is otherwise called as Neuro-fuzzy System(NFS). The main advantage of this system over others is that it follows a clever approach that not only thinks like a human brain but also incorporates its learning from past and current dataset. By the universal approximation theorem, it is also capable of approximating data with the ability to obtain interpretable IF-THEN rules. [14] [15]

The two major parameters that evaluate a fuzzy model are:

- Interpret-ability - determined by the Fuzzy Inference System(FIS) proposed by Ebrahim Mamdani in 1975. A FIS system consists of 4 modules as shown in the figure below.

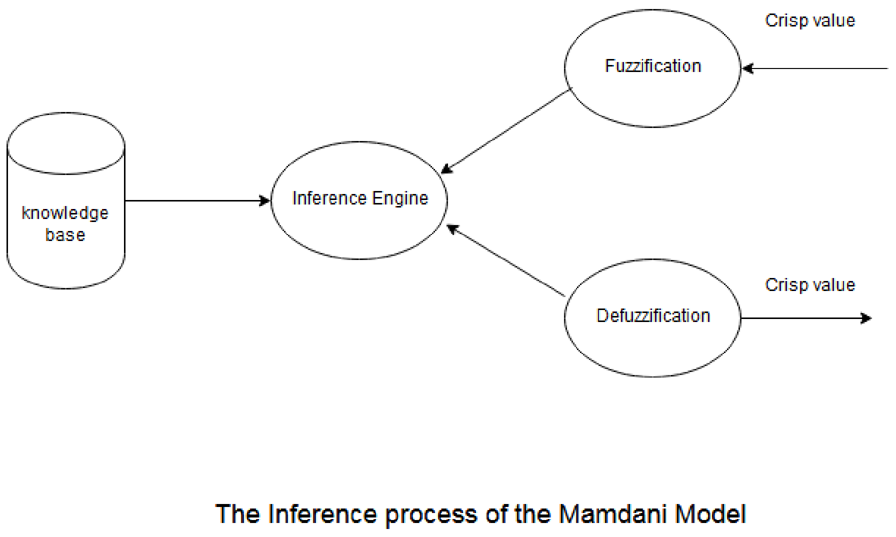


Figure 2.3: The Inference process of the Mamdani Model

First, the crisp inputs are fed into the *Fuzzification Module*. The *Knowledge Box* stores all the IF-THEN rules provided by the experts. The *Inference Engine* simulates the human thinking procedure by making Fuzzy Inference on the inputs and the IF-THEN rules. The *Defuzzification Module* transforms the fuzzy set obtained into crisp values.

- Accuracy - determined by the Sugeno Fuzzy model proposed by Takagi, Sugeno and Kang. The working of the model has been described below. [16]

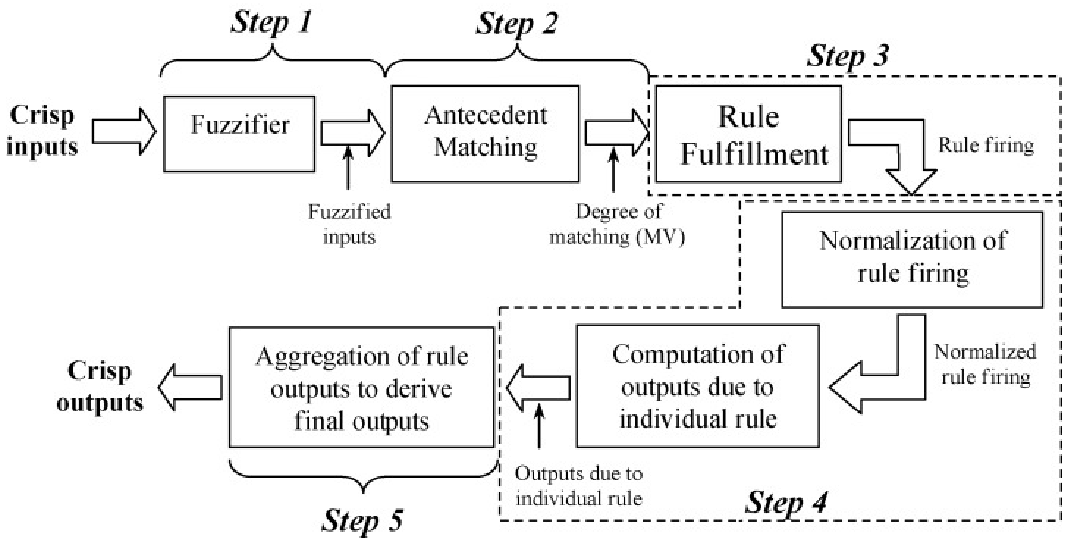


Figure 2.4: The Inference process of TSK fuzzy model

This model has various advantages over the Mamdani model. It can deal with complex and high-dimensional problems. It uses less number of rules as compared to Mamdani model. This model replaces the *"THEN"* part of Mamdani model with a function of input variables.

Other neuro-fuzzy models that can be used are Tsukamoto Model and Standard Additive Model(also called Kosko Model).

2.3.4 Deep Neural Networks

A deep neural network (DNN) is an artificial neural network with multiple hidden layers between input and the output layer. It has the capability to model complex non-linear relationships. [17] [18] The architecture of a basic deep neural network is shown as follows.

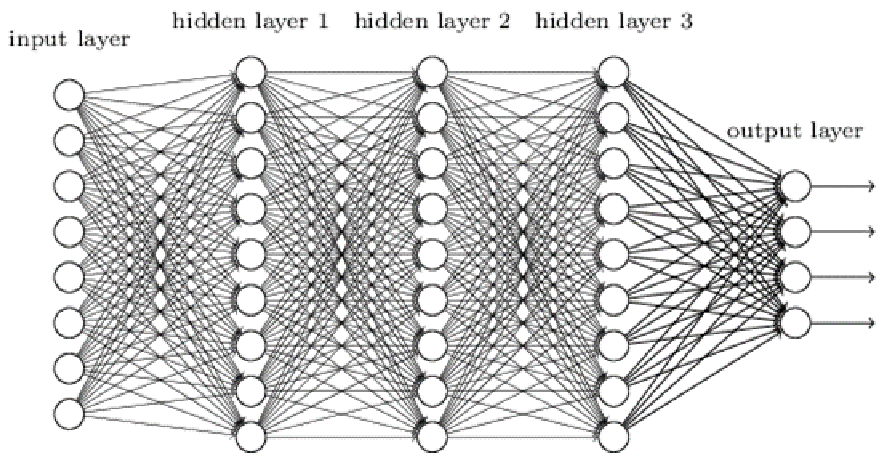


Figure 2.5: The architecture of a deep neural network

2.3.5 SVM for regression

Support Vector Machine(SVM) is a parameter-less technique developed by Vladimir Vapnik and his colleagues in 1992. [1] [19] The working of a basic SVM model is as shown below.

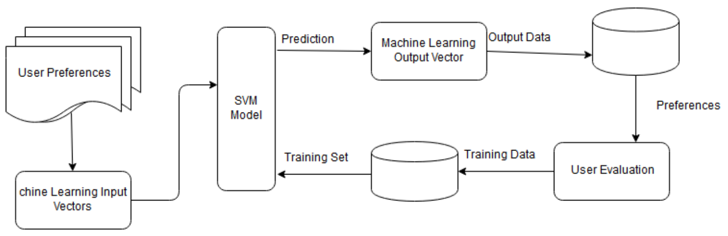


Figure 2.6: The basic working of a SVM model

2.4 Design of Training Algorithms for Neural Networks

As discussed in chapter 1, the better the model inherits the features of the time series, the better future predictions it can make. For this reason, various training algorithms are incorporated into the neural network. There are various training algorithms, few of which are listed below.

- Back Propagation (BP) - This is a training algorithm used in Artificial Neural Network to calculate a gradient that is needed for the updation of weights used in the network.
- Least Mean Square (LMS) - It is a form of supervised learning in which the model works on the inputs supplied by the user to calculate the output.
- Recursive Least Squares(RLS) - This is a popular training algorithm that recursively updates the coefficients that minimize a weighted linear least square cost functions corresponding to the input signals.
- Optimization Techniques -There are many optimization techniques which are used to train the weights (representing the connection strength) associated with the inputs. One of the major categories of Optimizations techniques is Swarm Computing. Various swarm computing techniques like Particle Swarm Optimization(PSO) [20], Shuffled Frog Leaping(SFL), Ant Colony Optimization(ACO), Ant Bee Colony(ABC), Fire Fly(FF) [21] have been implemented in neural networks and have been found to produce remarkable results. The idea behind the swarm computing techniques is that the Swarm Intelligence is utilized to improve the qualities of the harmony. Other optimizations that have been frequently used are Jaya, Harmony Search(HS) and Teaching-Learning Based Optimization(TLBO). From literature survey, it has been found that hybrid optimization techniques have been found to outperform the basic optimization techniques as they extract the best features of each optimization technique for the advantage. [22]

2.5 Role of Technical Indicators

As described in section 1.4, according to the third school of theory, the technical analysts undergo a technical analysis of the past and current stock data. Observing the trends, they have suggested few technical indicators which when included in the neural network model are believed to give a better forecast. [6] Few of them are as listed below:

Table 2.1: Technical Indicators and their calculation formulae

Technical Indicators	Formula
Simple Moving Average (SMA)	$\frac{1}{N}\sum_{i=1}^{N} x_i$ N = No. of Days. x_i = today's price
Exponential Moving Average (EMA)	$(P \times A) + (\text{Previous EMA} \times (1 - A))$; A=2/(N+1) P – Current Price, A- Smoothing factor, N-Time Period
Accumulation/ Distribution Oscillator (ADO)	$\frac{(\text{C.P - L.P}) - (\text{H.P - C.P})}{(\text{H.P - L.P}) \times (\text{Period's Volume})}$ C.P – Closing Price, H.P – Highest price, L.P – Lowest price
Stochastic Indicator (STOC)	$\%K = \frac{(\text{Today's Close - Lowest Low in K period})}{(\text{Highest High in K period - Lowest Low in K period})} \times 100$ $\%D$ = SMA of $\%K$ for the Period.
On Balance Volume (OBV)	If Today's Close > Yesterday's Close OBV = Yesterday's OBV + Today's Volume If Today's Close < Yesterday's Close OBV = Yesterday's OBV – Today's Volume
WILLIAM's %R	$\%R = \frac{(\text{Highest High in n period - Today's Close})}{(\text{Highest High in n period - Lowest Low in n period})} \times 100$
Relative Strength Index (RSI)	$\text{RSI} = 100 - \frac{100}{1 + (\text{U/D})}$
Price Rate Of Change (PROC)	$\frac{(\text{Today's Close - Close X-period ago})}{(\text{Close X-period ago})} \times 100$
Closing Price Acceleration (CPAcc.)	$\frac{(\text{Close Price - Close Price N-period ago})}{(\text{Close Price N-period ago})} \times 100$
High Price Acceleration (HPAcc.)	$\frac{(\text{High Price - High Price N-period ago})}{(\text{High Price N-period ago})} \times 100$

2.6 Measures used for evaluation of the prediction models

Due to the importance of time series forecasting in various real-life situations, the models for the analysis of the time series data should be carefully designed. To evaluate if our selected network model performs as per our expectations, performance indicators are used. Few important performance metrics are explained as below. Each and every indicator is a function of actual and predicted values of the time series.

2.6.1 The Mean Squared Error (MSE)

Mean Squared Error(MSE) is given by the formula:

$$MSE = \frac{1}{n}\sum_{i=1}^{n} e_i^2 \tag{2.3}$$

Its features are as follows:

- It determines the average squared deviation of predicted values.
- The magnitude of the positive and negative errors do not offset each other, thus, giving us the overall idea of the error that occurred during prediction.
- Unlike most of the metrics above, it penalizes extreme errors that occurred during forecasting.
- It does not provide an idea about the direction of the error.
- Although MSE is a good performance measure but is not as intuitive and predictable as the other measures discussed above.

2.6.2 The Sum of Squared Error (SSE)

Sum of Squared Error(SSE) is given by the formula:

$$SSE = \sum_{i=1}^{n} e_i^2 \tag{2.4}$$

Its features are as follows:

- It determines the total squared deviation of predicted values from the actual values.
- Other features are same as that of MSE.

2.6.3 The Signed Mean Squared Error (SMSE)

Signed Mean Squared Error(SMSE) is given by the formula:

$$SMSE = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{e_i}{\mid e_i \mid}\right)e_i^2 \tag{2.5}$$

Its features are as follows:

- It is same as MSE except for the fact that the original sign is kept for each individual squared error.
- It penalizes extreme errors that occur during prediction.
- It shows the direction of the overall error.

2.6.4 The Root Mean Squared Error (RMSE)

Root Mean Squared Error(RMSE) is given by the formula:

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}e_i^2} \tag{2.6}$$

Its features are as follows:

- It is the square root value of the MSE.
- Its other features are same as that of MSE.

2.6.5 The Normalized Mean Squared Error (NMSE)

Normalized Mean Squared Error(NMSE) is given by the formula:

$$NMSE = \frac{MSE}{\sigma^2} = \frac{1}{\sigma^2 n}\sum_{i=1}^{n}e_i^2 \tag{2.7}$$

Its features are as follows:

- It is the normalized value of MSE as the MSE is divided by the test variance.
- It is a balanced error measure and effectively calculates the forecast accuracy.
- The smaller the value of NMSE, the better is the forecast.

2.6.6 The Mean Forecast Error (MFE)

This metrics is defined as :

$$MFE = \frac{1}{n}\sum_{i=1}^{n} e_i \tag{2.8}$$

Its features are as follows:

- It determines the average deviation of the predicted value from the actual value.
- A zero MFE indicates that the predictions are on proper target.
- For a good predicted value, MFE should lie as close to zero as possible.
- A *Forecast Bias* is found out which indicates the direction of the error.
- It is dependent on the measurement scale selected.

2.6.7 The Mean Absolute Error (MAE)

The Mean Absolute Error(MAE) is defined as :

$$MAE = \frac{1}{n}\sum_{i=1}^{n} \mid e_i \mid \tag{2.9}$$

Its features are as follows:

- It determines the average absolute deviation of the predicted value from the actual value.
- It is also called *Mean Absolute Deviation(MAD).*
- It gives us the magnitude of the total error during the prediction process.
- In MAE, the magnitude of the positive and negative errors do not cancel out each other.
- Like MFE, extreme errors are not panelized by MAE.

2.6.8 The Mean Absolute Percentage Error (MAPE)

Mean Absolute Percentage Error (MAPE) is given by the formula:

$$MAPE = \frac{1}{n}\sum_{i=1}^{n} | \frac{e_i}{y_i} | \times 100 \tag{2.10}$$

Its features are as follows:

- It determines the percentage of average absolute error occurred.
- It is independent of the measurement scale but varies with data transformation.
- Unlike MFE, it doesn't show the direction of error.
- Like MAE, extreme errors are not panelized by MAPE.
- In MAPE, the magnitude of the positive and negative errors do not offset each other.

2.6.9 The Mean Percentage Error (MPE)

Mean Absolute Error (MAE) is given by the formula:

$$MPE = \frac{1}{n}\sum_{i=1}^{n} \left(\frac{e_i}{y_i}\right) \times 100 \tag{2.11}$$

Its features are as follows:

- It determines the percentage of average error occurred.
- It is similar to MAPE in most regards except it shows the direction of error.
- The magnitude of the positive and negative errors offset each other.
- Like MFE, even though the value of MPE becomes close to zero, we can not conclude that our selected model performed well. However, it is desirable to keep its value very small.

2.7 Summary

In this chapter, we understood the basics of financial time series forecasting, how to acquire datasets, normalize them within a range of [0,1] and feed them as inputs to the artificial neural networks. Various methods of forecasting have been explained like through Statistical Models, Neural Networks, Hybrid Neuro-fuzzy System and SVM model. We also came across different training algorithms which can be incorporated into Neural Networks, Neuro-fuzzy Systems and Deep Neural Networks. From literature survey, we also concluded that optimization techniques like Swarm Computing techniques, Jaya, HS, TLBO and hybrid optimization techniques can be inculcated in Neural Networks to get better accuracy. According to technical analysts, models with technical indicators included will perform with a better accuracy than models with no technical indicator. They are basically the numerical form of expression of the trends in the market. Lastly, we discussed various performance measures that can be used to check the efficiency of our designed model. They parametrically represent how efficiently our developed model fits our selected time series data.

Chapter 3

A Jaya Algorithm trained Neural Network for Stock Market Prediction

3.1 Introduction

In section 2.3, we discussed various methods of forecasting. In this chapter, we shall deal with a simple ANN structure i.e Functional Link Artificial Neural Network(FLANN) model. From the literature survey, we have found that FLANN has been trained with Back Propagation (BP), Particle Swarm Optimization (PSO) [23] and Differential Evolution (DE) [7] and it has been observed that the DE and PSO outperform the BP algorithm. In this chapter, we shall discuss how Jaya based FLANN performs in predicting stock market indices in comparison to BP. The two types of FLANN models used are - Chebyshev-FLANN(CFLANN) and Trigonometric-FLANN(TFLANN).

First, we shall train the model with BP and understand the functioning of the model. We shall make predictions for 1-day and 7-days. Further, we shall test the same model with Jaya optimization technique. The stock indices BSE500, DJIA and NASDAQ with few technical indicators are fed as inputs for our experimental time series data. The Mean Square Error(MSE) and Mean Absolute Percentage Error(MAPE) are calculated to evaluate the FLANN models. The MSE and MAPE in case of Jaya are found to be very less in comparison to the BP method.

3.2 Training of FLANN with BP

From literature survey, we found out that FLANN has less computational complexity as compared to the Multi-layer Perceptron (MLP). MLP has hidden layers which makes it more complex and reduces its convergence rate. So, to reduce the computational burden, we selected the FLANN model. It is a single layer ANN which has the ability to form complex decision regions by creating non-linear decision boundaries.

In the proposed FLANN model, each input is fed into a *functional expansion unit* that produces an enhanced representation of the original pattern. In our proposed model, there are six inputs from the time series data and the seventh input is the mean of rest six inputs. The function used in this unit may be any orthogonal function like Trigonometric, Chebyshev, Legendre or Laguerre. It has been observed that the Trigonometric and Chebyshev outperform the other two models [24]. Each input is assigned with a randomly selected weight.

$$Y_1 = [w_1 * cos(\pi x_1) + w_2 * sin(\pi x_1) + ... + w_{13} * cos(\pi x_7) + w_{14} * sin(\pi x_7)] \quad (3.1)$$

$$Y_2 = [w_{15}x_1 + w_{16} * x_2 + ... + w_{20} * x_7 + w_{21} * x_7] \quad (3.2)$$

Finally, the output $''Y''$ is represented as the sum of Y_1 an Y_2. The output is passed through an activation function, which in this case is a tan-hyperbolic function.

$$Y = tanh(Y_1 + Y_2) \quad (3.3)$$

The final output is then compared with our Desired Output(D) to produce an error.

$$Error(e) = Y - D \quad (3.4)$$

Then, the initially selected random weights $[w_1, w_2, w_3, w_4, ...w_{20}, w_{21}]$ are updated using the BP learning algorithm.

$$W_j(new) = W_j(old) + \mu * e(i) * \frac{\partial(Y)}{\partial(W_j)} \tag{3.5}$$

where μ represents the learning rate, e(i) represents the error during i^{th} iteration and $\frac{\partial(Y)}{\partial(W_j)}$ represents the change in weights. [7]

The proposed TFLANN and CFLANN are as follows:

3.2.1 TFLANN

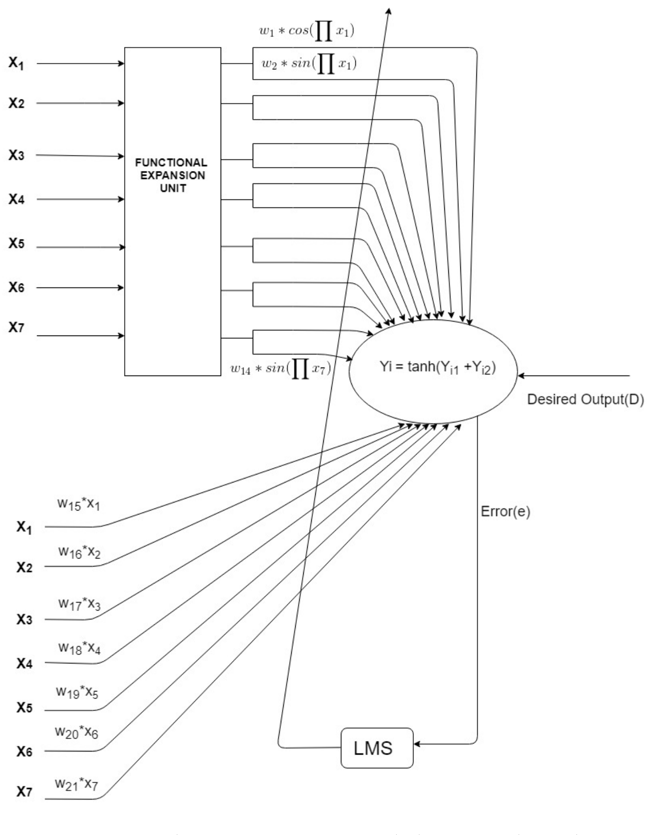


Figure 3.1: The TFLANN model trained with BP

The expansion of inputs is done by the Fourier Expansion because it provides a compact representation of the function in the mean square sense [6]. Each input is expanded to have two components as follows:

$$t_1 = cos(\pi x) \tag{3.6}$$

$$t_2 = sin(\pi x) \tag{3.7}$$

3.2.2 CFLANN

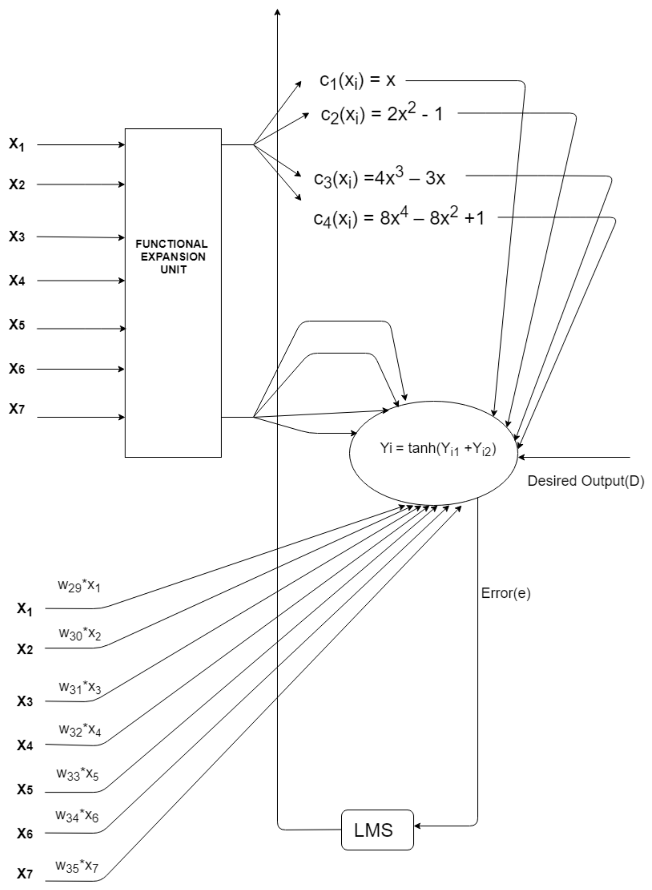


Figure 3.2: The CFLANN model trained with BP

The expansion of inputs is done by the general formula which is given as:

$$c_{r+1} = 2 * x * c_r(x) - c_{r-1}(x) \tag{3.8}$$

The four components of each input can be obtained from the general formula and they are as follows:

$$c_1(x_i) = x \tag{3.9}$$

$$c_2(x_i) = 2x^2 - 1 \tag{3.10}$$

$$c_3(x_i) = 4x^3 3x \tag{3.11}$$

$$c_4(x_i) = 8x^4 8x^2 + 1 \tag{3.12}$$

3.3 Basics of Jaya Algorithm

According to R. Venkata Rao [21], Jaya algorithm is a powerful optimization algorithm that solves both constrained and unconstrained problems [25]. The main idea behind the algorithm is the solution for a given problem should move towards the best solution and avoid the worst solution. Unlike many other algorithms, the Jaya algorithm only requires few common parameters. It does not require any algorithm-specific control parameters. [21]

The population-based heuristic algorithms are mainly of two type:

- Evolutionary Algorithm(EA): Some of the popular evolutionary algorithms are Genetic Algorithm(GA), Evolution Strategy(ES), Differential Evolution(DE), etc.
- Swarm Intelligence(SI) based algorithm: Some of the well-known SI based algorithms are Particle Swarm Optimization(PSO) [20], Shuffled Frog Leaping(SFL), Ant Colony Optimization(ACO), Ant Bee Colony(ABC), Fire Fly(FF) [21], etc.

Besides there are other algorithms which work on different principles of the nature like Whale Optimization Algorithm(WOA), Harmony Search(HS), etc. All the EA and SI-based algorithms have their own algorithm-specific controlling parameters like number of generations, population size, and harmony memory considering rule(hmcr) and pitch adjusting rate(par) in case of HS algorithm. An improper tuning of these algorithm-specific parameters may result in increased computational effort or may give poor results. For this reason, R. Venkata Rao had suggested Teaching Learning Based Optimization(TLBO) that has only two common controlling parameters: population size and the number of generations. Thus, this optimization technique has gained much popularity among the researchers. [21]

However, the TLBO algorithms possess two phases: the *Teaching Phase* and the *Learning Phase*. Thus, the Jaya optimization technique was suggested by R. Venkata Rao that has only one phase and is much easier to apply.

3.4 Jaya based FLANN Model

Let *f(x)* is the objective function to be minimized, which in our case is to minimize the Mean Square Error(MSE). At any iteration i, let there be 'm' design variables(i.e j=1,2,3,...,m) and 'n' candidate solution(i.e population size = k =1,2,3,...,n). Let the best candidate best obtains the value of objective function f(x) (i.e. $f(x)_{best}$) and the worst candidate worst obtains the value of objective function f(x) (i.e. $f(x)_{worst}$) in the entire solution set. [26] If $X_{j,k,i}$ is the value of the j^{th} variable for the k^{th} candidate during i^{th} iteration, $X_{j,k,i}$ is modified by the formula:

$$X'_{j,k,i} = X_{j,k,i} + r_{1,j,i}(X_{j,best,i} - \mid X_{j,k,i} \mid) - r_{2,j,i}(X_{j,worst,i} - \mid X_{j,k,i} \mid) \tag{3.13}$$

where $X_{j,best,i}$ is the value of the j^{th} variable for the best candidate and $X_{j,worst,i}$ is the value of the j^{th} variable j for the worst candidate. $X'_{j,k,i}$ is the modified value of the $X_{j,k,i}$ and $r_{1,j,i}$ and $r_{2,j,i}$ are two random values for the j^{th} variable during i^{th} iteration and it has to be within the range [0,1].

In the equation, the $r_{1,j,i}(X_{j,best,i} - \mid X_{j,k,i} \mid)$ is the tendency of the solution to move closer to the best solution and the $-r_{2,j,i}(X_{j,worst,i} \mid X_{j,k,i} \mid)$ is the tendency of the solution to avoid the worst solution.

The value of $X'_{j,k,i}$ is accepted only if it gives a better value of the objective function than that of $X_{j,k,i}$. All the accepted function values are maintained to the next iteration.

3.4.1 Hand Calculation of Jaya Algorithm

Let our objective function to be minimized be :

$$minf(x) = \sum_{i=1}^{n} x_i^2 \tag{3.14}$$

Let us assume the initial population size as 5(i,e candidate solutions). Let there be 2 design variables x_1 and x_2. The termination criterion for the example is assumed to be 2 iterations.

Table 3.1: Initial Population

Candidate	*x₁*	*x₂*	*f(x)*	*Status*
1	-3	15	234	
2	12	61	3865	
3	65	-6	4261	*worst*
4	-6	7	85	*best*
5	-10	-20	500	

From the above table, it can be observed that the best solution corresponds to the 4th candidate and the worst solution corresponds to the 3rd. Now assuming r_1 and r_2 values as 0.58 and 0.81 for x_1 and 0.92 and 0.49 for x_2, the new values for x_1 and x_2 are calculated. For example, for the first candidate, the formulae for calculation of x_1 and x_2 will be:

$$X'_{1,1,1} = X_{1,1,1} + r_{1,1,1}(X_{1,4,1} - \mid X_{1,1,1} \mid) - r_{2,1,1}(X_{1,3,1} - \mid X_{1,1,1} \mid) \tag{3.15}$$

$$X'_{2,1,1} = X_{2,1,1} + r_{1,2,1}(X_{2,1,1} - \mid X_{2,1,1} \mid) - r_{2,2,1}(X_{2,3,1} - \mid X_{2,1,1} \mid) \tag{3.16}$$

So, by the formula, the values of x_1 and x_2 will be:

$X'_{1,1,1} = -3 + 0.58(-6 - \mid -3 \mid) - 0.81(65 - \mid -3 \mid) = -58.44$

$X'_{2,1,1} = 15 + 0.92(7 - \mid 15 \mid) - 0.49(-6 - \mid 15 \mid) = 17.93$

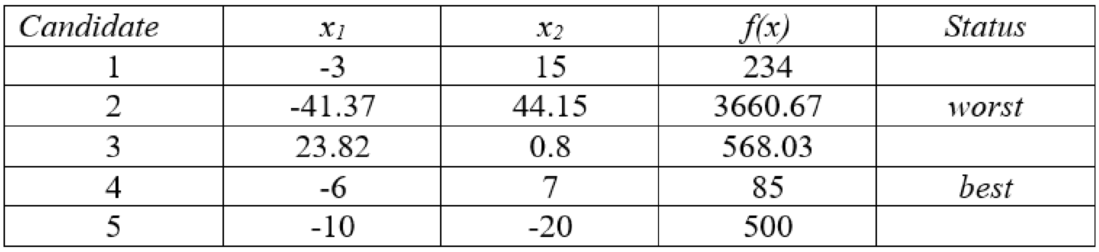
Similarly, the x_1 and x_2 for other candidates are calculated and maintained in another table as shown below.

Table 3.2: New values of the variables and the objective function during iteration 1

Candidate	*x₁*	*x₂*	*f(x)*
1	-58.44	17.93	3736.72
2	-41.37	44.15	3660.7
3	23.82	0.8	568.03
4	-60.75	13.37	3869.32
5	-63.83	-19.22	4443.67

Now, the above 2 tables are compared with respect to the value of the objective function and the best of them is selected. The corresponding values of the x_1 and x_2 are updated.

Table 3.3: Updated values of the variables and the objective function based on fitness comparison at the end of iteration 1



Candidate	x_1	x_2	*f(x)*	*Status*
1	-3	15	234	
2	-41.37	44.15	3660.67	*worst*
3	23.82	0.8	568.03	
4	-6	7	85	*best*
5	-10	-20	500	

Here, the first iteration ends and the updated values are used for the next iteration.

From the above table, it can be observed that the best solution corresponds to the 4^{th} candidate and the worst solution corresponds to the 2^{nd}. Now assuming r_1 and r_2 values as 0.58 and 0.81 for x_1 and 0.92 and 0.49 for x_2, the new values for x_1 and x_2 are calculated. For example, for the first candidate, the formulae for calculation of x_1 and x_2 will be:

$$X'_{1,1,2} = X_{1,1,2} + r_{1,1,2}(X_{1,4,2} - \mid X_{1,1,2} \mid) - r_{2,1,2}(X_{1,2,2} - \mid X_{1,1,2} \mid) \tag{3.17}$$

$$X'_{2,1,2} = X_{2,1,2} + r_{1,2,2}(X_{2,4,1} - \mid X_{2,1,2} \mid) - r_{2,2,2}(X_{2,2,2} - \mid X_{2,1,2} \mid) \tag{3.18}$$

So, by the formula, the values of x_1 and x_2 will be:

$X'_{1,1,2} = -3 + 0.58(-6 - \mid -3 \mid) - 0.81(-41.37 - \mid -3 \mid) = 38.16$

$X'_{2,1,2} = 15 + 0.92(7 - \mid 15 \mid) - 0.49(44.15 - \mid 15 \mid) = -6.64$

Similarly, the x_1 and x_2 for other candidates are calculated and maintained in another table as shown below.

Table 3.4: New values of the variables and the objective function during iteration 2

Candidate	x_1	x_2	$f(x)$
1	38.16	-6.64	1500.28
2	-1.83	10.11	105.56
3	59.32	-14.73	3735.84
4	25.41	-11.2	771.11
5	22.33	-43.8	2417.07

Now, the above 2 tables are compared with respect to the value of the objective function and the best of them is selected. The corresponding values of the x_1 and x_2 are updated.

Table 3.5: Updated values of the variables and the objective function based on fitness comparison at the end of iteration 2

Candidate	x_1	x_2	$f(x)$	*Status*
1	-3	15	234	
2	-1.83	10.11	105.56	
3	23.82	0.8	568.03	*worst*
4	-6	7	85	*best*
5	-10	-20	500	

Here, the second iteration ends.

3.4.2 JAYA Optimized FLANN Model

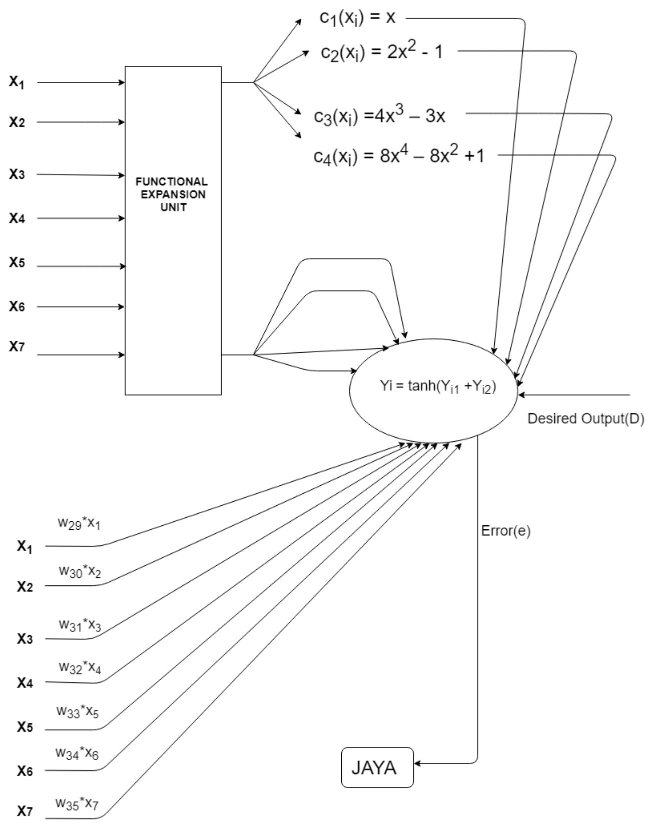


Figure 3.3: Jaya optimized TFLANN model

In the Jaya optimized FLANN model, the random weights are updated by the Jaya algorithm. In the above FLANN model, there are 7 inputs(i.e candidates) each corresponding to 3 components(or design variables).

In the first iteration, the value of the objective function is calculated, from which the best and the worst values are selected. Based on these values, the corresponding design variables are updated and thus, the value of the objective function is also found out. Now, the best out of the old and new values of the objective function is selected and their corresponding design variables are used for evaluation in the next iteration.

The basic working of Jaya can be demonstrated by the flowchart given below:

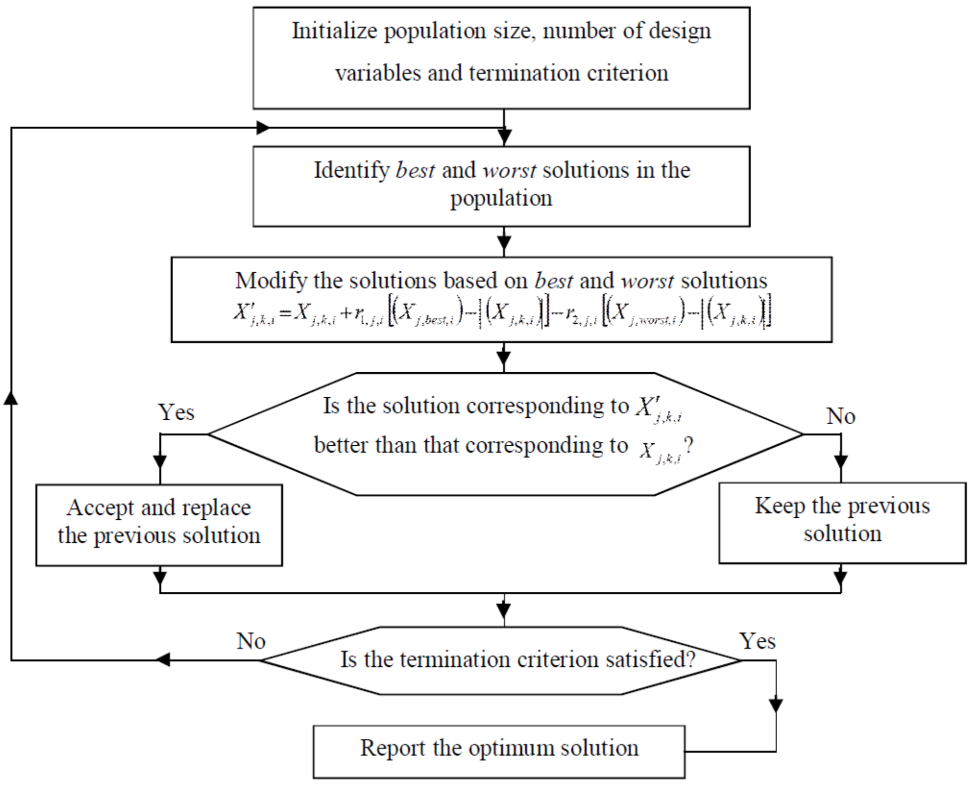

Initialize population size, number of design variables and termination criterion
Identify *best* and *worst* solutions in the population
Modify the solutions based on *best* and *worst* solutions
$X'_{j,k,i} = X_{j,k,i} + r_{1,j,i}[(X_{j,best,i}) - |(X_{j,k,i})|] - r_{2,j,i}[(X_{j,worst,i}) - |(X_{j,k,i})|]$
Is the solution corresponding to $X'_{j,k,i}$ better than that corresponding to $X_{j,k,i}$?
Yes
No
Accept and replace the previous solution
Keep the previous solution
Is the termination criterion satisfied?
No
Yes
Report the optimum solution

Figure 3.4: FLowchart 0f Jaya optimization technique

3.5 Simulation Study

3.5.1 Experimental Data for training and testing

The daily closing price of various stock market data i.e BSE500, DJIA and NASDAQ are considered here as the experimental data. They are obtained from *www.yahoo.finance.com* and *www.bseindia.com*. All the inputs are normalized within the range [0,1] using the following formula.

$$X_{norm} = \frac{X_{orig} - X_{min}}{X_{max} - X_{min}} \tag{3.19}$$

where X_{norm} represents normalized value, X_{orig} represents current closing price and X_{min} and X_{max} are the minimum and maximum prices of the stock data respectively. [7]

The datasets used in the process are as given below:

Table 3.6: Stock Datasets used

STOCK DATA SETS	TOTAL SAMPLE	DATA RANGE	TRAINING SAMPLE	TESTING SAMPLE
BSE500	19th Feb 2003 19th Feb 2018	3700	1300	250
DJIA	19th Feb 2003 19th Feb 2018	2500	1300	250
NASDAQ	19th Feb 2003 19th Feb 2018	3200	1300	250

The technical indicators used in the process are as given below:

Table 3.7: The technical indicators and their formulae

TECHNICAL INDICATORS	FORMULA
Simple Moving Average(SMA)	$\frac{1}{N}\sum_{i=1}^{N} x_i$ N = No. of Days. x_i = today's price
Price Rate of Change (PROC)	$\frac{\text{Today's close} - \text{Close X−periods ago}}{\text{Close X−periods ago}} *100$

3.5.2 Training and Testing of the forecast model

The performance parameter MSE is calculated by the formula:

$$MSE = \frac{1}{n}\sum_{i=1}^{n} e_i^2 \tag{3.20}$$

as the process in repeated for N number of times. The performance parameter MAPE is calculated by the formula:

$$MAPE = \frac{1}{n}\sum_{i=1}^{n} \mid \frac{e_i}{y_i} \mid \times 100 \tag{3.21}$$

3.6 Results Discussion

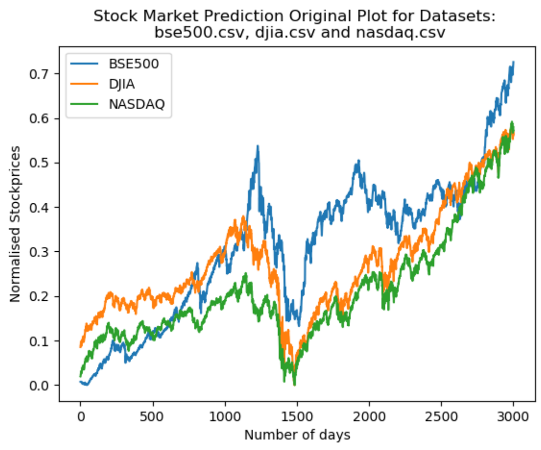
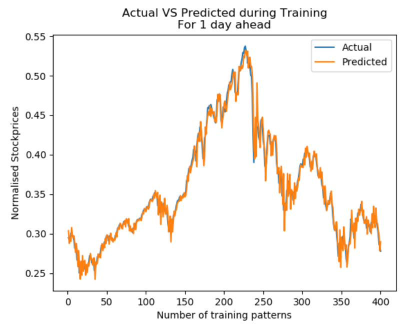


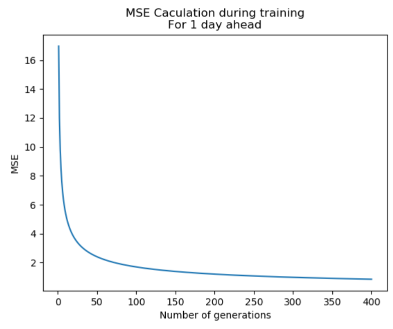
Figure 3.5: Stock Market Prediction Original Dataset bse500.csv, djia.csv, nasdaq.csv

Table 3.8: The Performance of BP based FLANN (During Training and Testing)

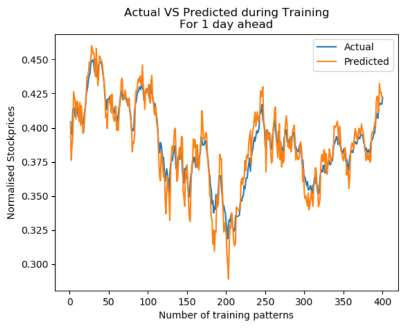
STOCK DATA	FLANN MODEL	PREDICTION	MSE (train)	MAPE in % (train)	MSE (test)	MAPE in % (test)
BSE500	TFLANN	1-day	0.7277	1.956	0.036	1.642
		1-week	0.8524	6.77	0.045	4.9
	CFLANN	1-day	0.2885	1.02	0.006	0.415
		1-week	0.2925	1.75	0.39	0.424
DJIA	TFLANN	1-day	0.579	5.06	0.11	1.71
		1-week	0.9824	4.27	0.486	2.7
	CFLANN	1-day	0.2433	2.35	0.0143	1.09
		1-week	0.2484	2.09	0.115	1.31
NASDAQ	TFLANN	1-day	0.6171	9.33	0.025	3.15
		1-week	0.8326	11.13	0.082	3.96
	CFLANN	1-day	0.2737	3.37	0.047	1.78
		1-week	0.2778	6.9	0.061	2.59



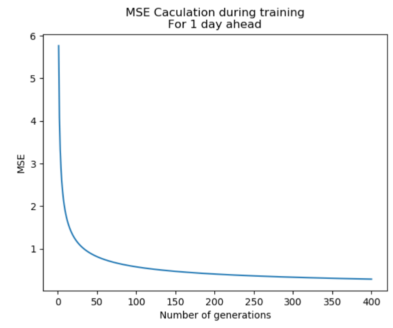
(a) Actual vs Predicted of BP trained TFLANN during Training



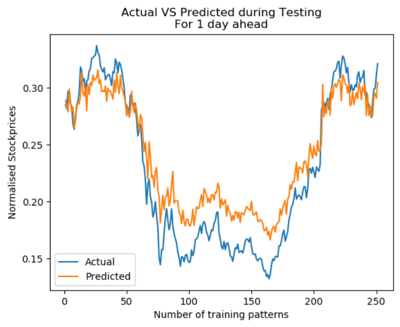
(b) MSE of BP trained TFLANN during Training



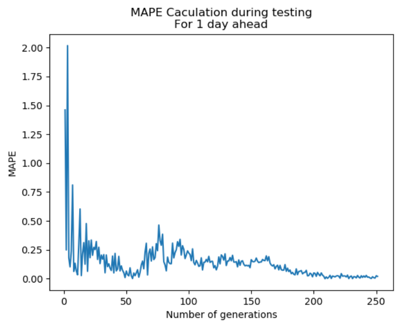
(c) Actual vs Predicted of BP trained CFLANN during Training



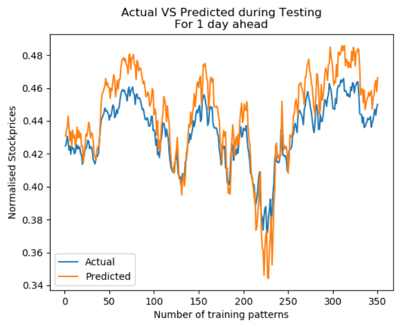
(d) MSE of BP trained CFLANN during Training



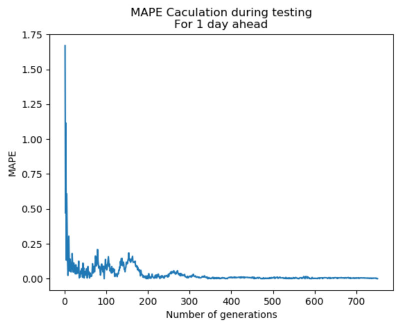
(e) Actual vs Predicted of BP trained TFLANN during Testing



(f) MSE of BP trained TFLANN during Testing

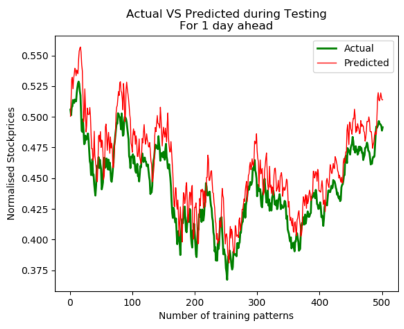


(g) Actual vs Predicted of BP trained CFLANN during Testing

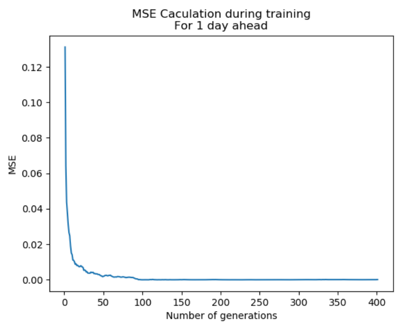


(h) MSE of BP trained CFLANN during Testing

Figure 3.6: BP trained TFLANN and CFLANN during training and testing



(a) Actual vs Predicted of JAYA trained CFLANN during Training



(b) MSE of JAYA trained CFLANN during Testing

Figure 3.7: JAYA trained CFLANN during training and testing

The MSE of JAYA trained CFLANN model after training is $1.25664568e^{-05}$ and after testing is $2.85819102e^{-09}$.

3.7 Analysis and Conclusion

Accurate and precise stock market prediction is always a difficult task. We also found out that the CFLANN gives us better results than TFLANN as far as MSE and MAPE are concerned. So, the CFLANN model was trained with Jaya and it was found that it outperforms BP in terms of MSE and MAPE. Thus, we shall compare Jaya with various other optimization techniques by implementing them on the CFLANN model in the next chapter.

Chapter 4

DE, PSO, WOA, Jaya, HS trained Neural Network for Stock Market Prediction

4.1 Introduction

In the previous chapter, we already discussed how Jaya optimization was applied to CFLANN model. In this chapter, we shall compare the performance of Jaya with four other optimizations techniques like Differential Evolution(DE), Particle Swarm Optimization(PSO), Whale Optimization Algorithm(WOA) and Harmony Search(HS). The datasets used were BSE500, DJIA and NASDAQ. We calculate the MSE during training and plot the Actual vs Predicted graph for testing. In the end, we shall make a conclusion about each of the optimization techniques.

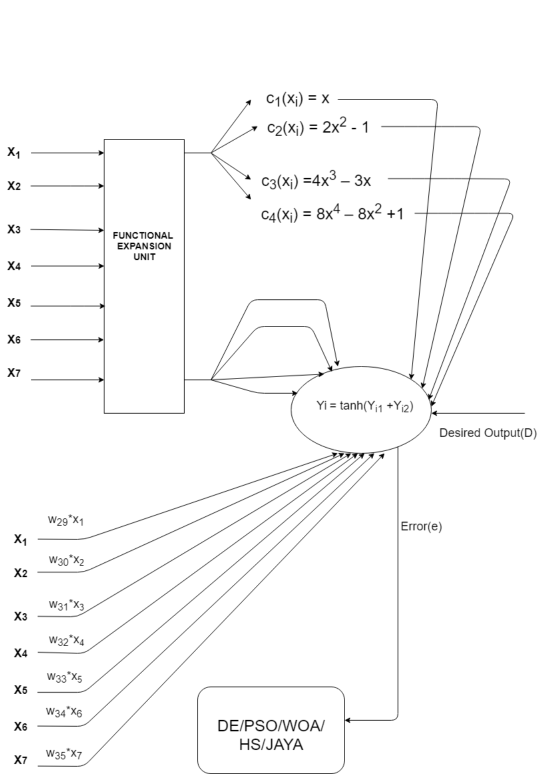


Figure 4.1: Optimized CFLANN Model

We shall discuss each algorithm in brief and study their respective results in further sections.

4.2 DE

DE was first introduced in 1995 by Kenneth Price. It is a population-based stochastic search algorithm. This algorithm has been successfully applied in text document clustering, image pixel clustering, etc. The main advantages of DE are it finds the true global minimum regardless of the initial parameters. It also possesses a fast convergence rate and requires very few control parameters. It is very similar to Genetic Algorithm and uses similar operators: crossover, mutation and selection.The only thing in which DE differs from GA is that DE relies on mutation whereas GA relies on the crossover. In this algorithm, the components of population vectors are used to build the trial vectors. Thus, the recombination operator shuffles successful information from the population to search for a better solution space. [7] [27].

The working of DE has been explained in the following flowchart.

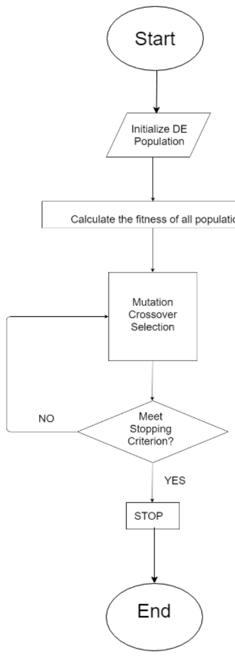

Start
Initialize DE Population
Calculate the fitness of all population
Mutation Crossover Selection
Meet Stopping Criterion?
NO
YES
STOP
End

Figure 4.2: Flowchart of Differential Evolution Method

4.3 PSO

PSO is an optimization technique that was proposed in 1995 by Eberhart and Kennedy. This algorithm is inspired by fish and birds to display a collective behaviour. Each participating member in PSO has a position as well as velocity vector represented as $X_i = [x_1, x_2...., x_i]$ and $V_i = [v_1, v_2...., v_i]$ respectively [28]. For every member, there is a personal best represented as pbest = $P_b = [pb_i, pb_i...., pb_i]$ and global best gbest. For each member, the velocity vector is calculated for the next iteration by taking into consideration position vector, velocity vector, a personal best vector as well as a global best vector. From the updated velocity vector and previous position vector, the position vector for the next iteration is calculated. [29] [28]

The working of PSO can be described in brief by the flowchart given below.

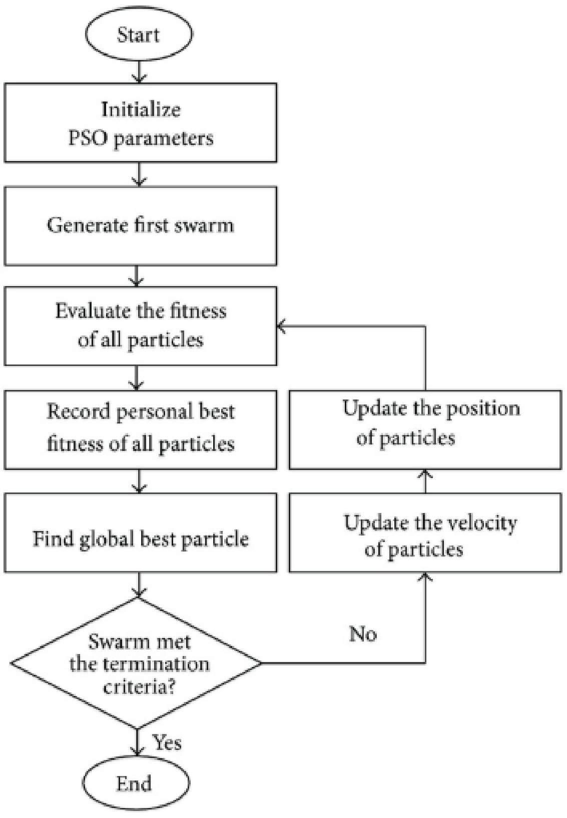

Start
Initialize PSO parameters
Generate first swarm
Evaluate the fitness of all particles
Record personal best fitness of all particles
Find global best particle
Swarm met the termination criteria?
No
Yes
Update the velocity of particles
Update the position of particles
End

Figure 4.3: Flowchart of Particle Swarm Optimization

4.4 WOA

From literature survey, we understood that the meta-heuristic optimization algorithms are becoming much popularity because of 4 main reasons as follows:

- They are simple concepts and easy to implement.
- They can easily bypass local optima.
- They have a wide range of applications.
- They work pretty well without the information about gradient.

WOA is a meta-heuristic optimization algorithm that was first proposed by S. Mirjalili in the year 2016. Whales are very fancy creatures. According to Hof and Van Der Gucht, whales have cells in their brain that resemble the human spindle cells [30]. These cells make them capable of making a judgement and be emotional and social like humans. In fact, these cells are in double the quantity in whales as compared to an adult brain which makes the whales extra smart.

Whales exhibit a different kind of social behaviour. Some whales live alone and some like *Killer Whales* live their life for their entire life period. One of the biggest whale species is a humpback whale whose size is almost like a school bus. They live in groups and possess a unique hunting method. This method is the root of this proposed algorithm. [30]

The hunting technique of the humpback whales is popularly known as bubble-net feeding method. They develop a "9" like pattern with the bubbles as shown in the following diagram:

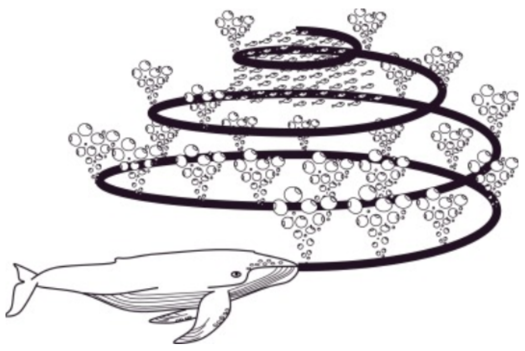


Figure 4.4: The Bubble-net feeding technique of humpback whales

The working of WOA can be described in brief by the flowchart given below.

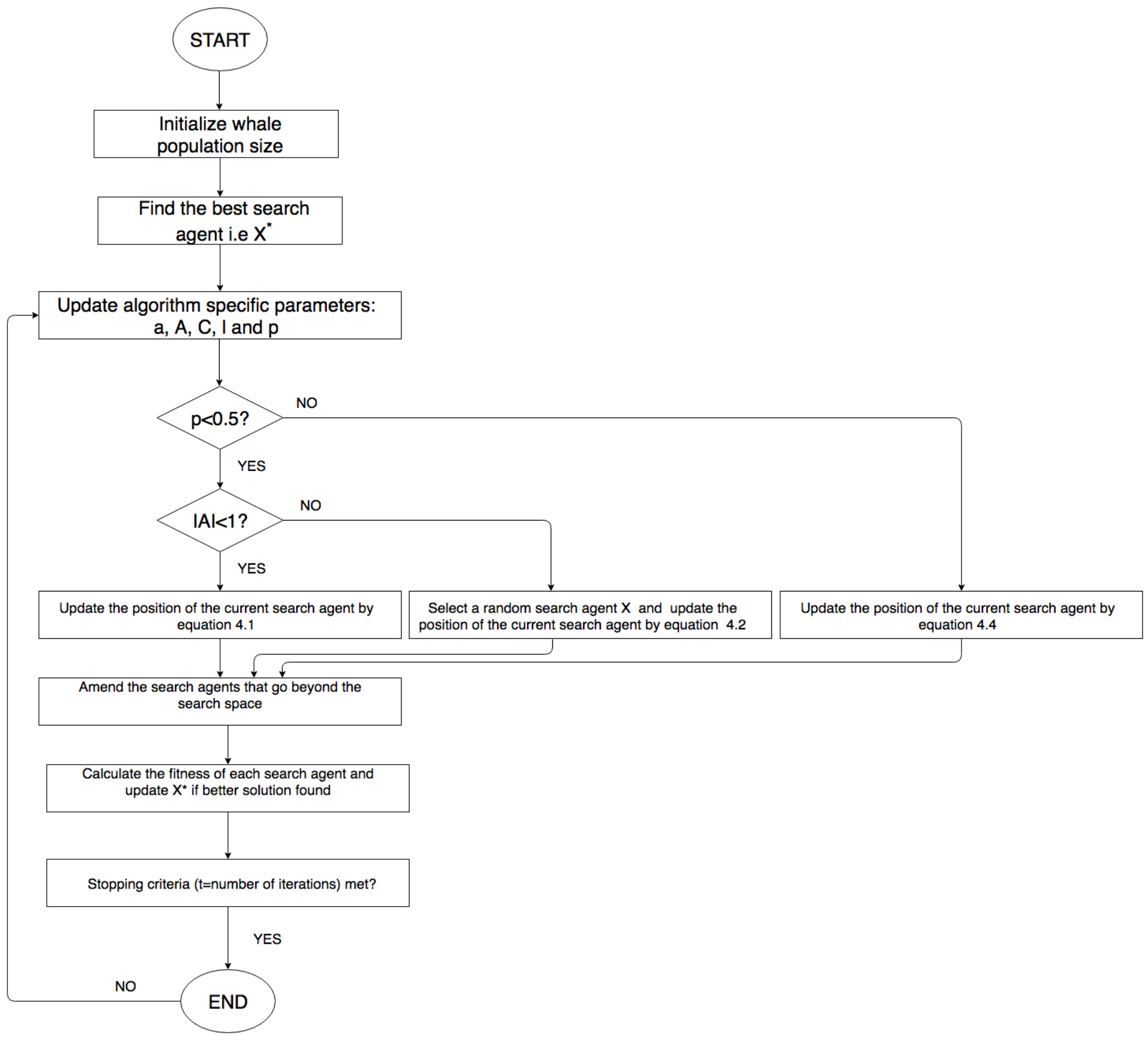

START
Initialize whale population size
Find the best search agent i.e X*
Update algorithm specific parameters: a, A, C, l and p
p<0.5?
NO
YES
IAI<1?
NO
YES
Update the position of the current search agent by equation 4.1
Select a random search agent X and update the position of the current search agent by equation 4.2
Update the position of the current search agent by equation 4.4
Amend the search agents that go beyond the search space
Calculate the fitness of each search agent and update X* if better solution found
Stopping criteria (t=number of iterations) met?
YES
NO
END

Figure 4.5: Flowchart of Whale Optimization Algorithm

$$\vec{D} = | \vec{C}\,\overrightarrow{X^*}(t) - \vec{X}(t) | \tag{4.1}$$

$$\vec{X}(t+1) = \overrightarrow{X_{rand}}\,\vec{A}\,\vec{D} \tag{4.2}$$

where

$$\vec{D} = | \vec{C}\,\overrightarrow{X_{rand}} - \vec{X} | \tag{4.3}$$

$$\vec{X}(t+1) = \overrightarrow{D'}\,e^{bt} cos(2\pi l) + \overrightarrow{X^*}(t) \tag{4.4}$$

Other parameters like:

- a is linearly decreased from 2 to 0 over the course of its iterations.
- r is a random vector in [0,1]
- A = $2\vec{a}\ \vec{r}\ -\vec{a}$
- C = $2\vec{r}$
- l is a random number in [-1,1]
- p is a random number in [0,1]
- $\overrightarrow{D'} = \overrightarrow{X^*}(t) - \vec{X}(t)$

4.5 JAYA

Kindly refer to chapter 3 to understand about this optimization technique in details.

4.6 HS

HS is a meta-heuristic population-based algorithm whose main goal is to find the perfect state of harmony in a musical process. It was proposed in the year 2001 by Geem. The objective function determines the fitness of the decision variables like the pitch of the musical instrument determines the aesthetic quality of the music. During an improvisation process, all the music players try to bring the pitches within a particular range. If all the pitches combine to produce a soothing harmony, each music player stores the pitches in his memory to modify the process the next time. Something similar happens in an optimization procedure as well. Initially, random decision variables are chosen. If the objective function gives a good result, the random values are stored in memory to enhance the solution in the next iteration. [31]

The working of HS algorithm is explained stepwise as follows:

- The initial population for the HS algorithm is randomly chosen where x_j = x_{ij} where i= 1, 2, 3...., HMS and j=1, 2, 3,...n. The HM Matrix now looks as follows:

$$HM = \begin{bmatrix} x_{11} & x_{12} & x_{13} & \cdot\ \cdot\ \cdot & x_{1n} \\ x_{21} & x_{22} & x_{23} & \cdot\ \cdot\ \cdot & x_{2n} \\ \cdot & \cdot & \cdot & \cdot\ \cdot\ \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot\ \cdot\ \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot\ \cdot\ \cdot & \cdot \\ x_{HMS1} & x_{HMS2} & x_{HMS3} & \cdot\ \cdot\ \cdot & x_{HMSn} \end{bmatrix} \tag{4.5}$$

- By the Harmony Memory Consideration(HMC) rule, a new random number r_1 is generated and is compared to Harmony Memory Consideration Rate(hmcr).
  If $r_1 < hmcr$, then the x_{ij}^{new} is generated randomly by the formula:

$$x_{ij}^{new} = x_{ij}, x_{ij} \in \{x_{1j}, x_{2j}, x_{3j}..., x_{HMSj}\} \tag{4.6}$$

  If $r_1 \geq hmcr$, then by Random Initialization Rule, the first decision variable is chosen randomly from the current HM values by the rule:

$$x_{ij}^{new} = l_{ij} + (u_{ij} - l_{ij}).rand(0,1) \tag{4.7}$$

  where l, u are the lower and upper bound of the problem respectively.

- Next, the pitch of the decision variables selected by HMC rule is examined to check if pitch adjustment is required or not. Thus, a new random number r_2 is generated within the range [0,1]. If $r_2 <$ Pitch Adjusting Rate(par), then the decision variable undergoes pitch adjustment by the formula:

$$x_{ij}^{new} = x_{ij} \pm rand(0,1).BW \tag{4.8}$$

  where BW represents Band Width.

- Now, if the value of the objective function corresponding to x^{new} is less than that corresponding to x^{worst}, we update the harmony memory as $x^{worst} = x^{new}$

Further, the working of HMS is demonstrated in brief by the flowchart below:

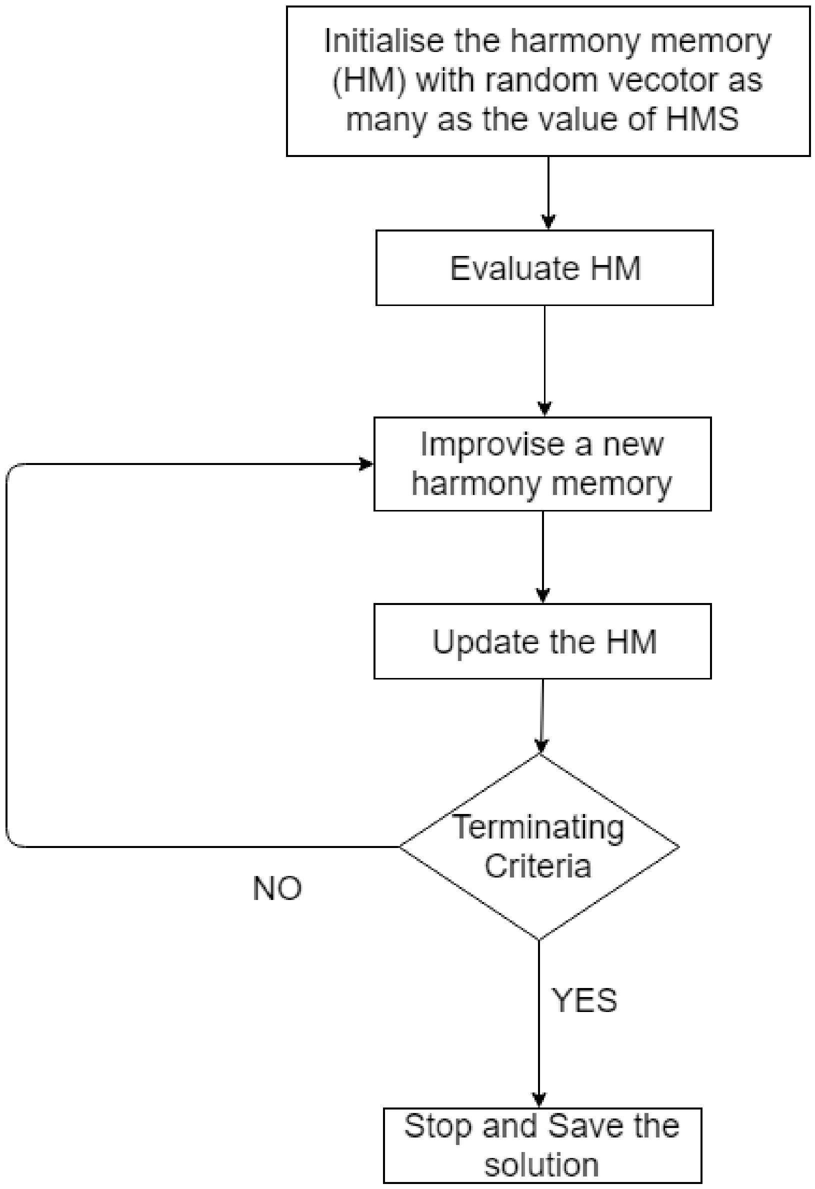
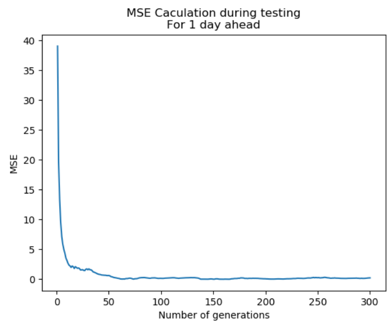
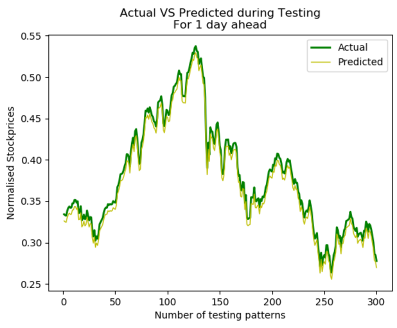

Initialise the harmony memory (HM) with random vecotor as many as the value of HMS
Evaluate HM
Improvise a new harmony memory
Update the HM
Terminating Criteria
NO
YES
Stop and Save the solution

Figure 4.6: Flowchart of HS Algorithm

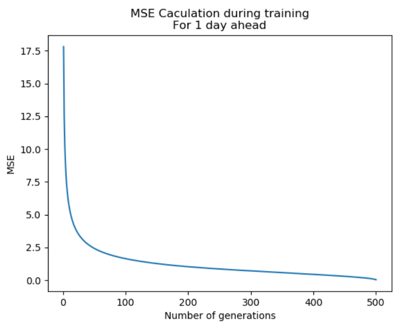
4.7 Results Discussion



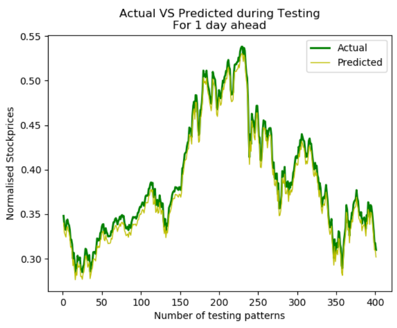
(a) MSE of DE optimized CFLANN during Training



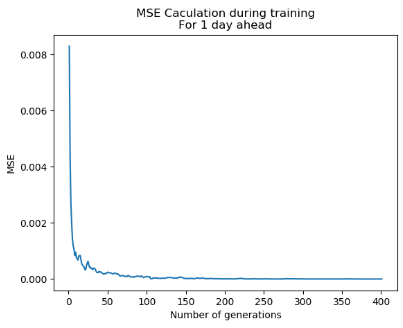
(b) Actual vs Predicted of DE optimized CFLANN during Testing



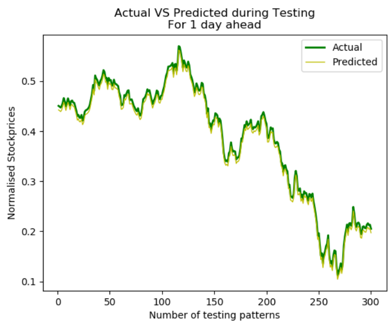
(c) MSE of PSO optimized CFLANN during Training



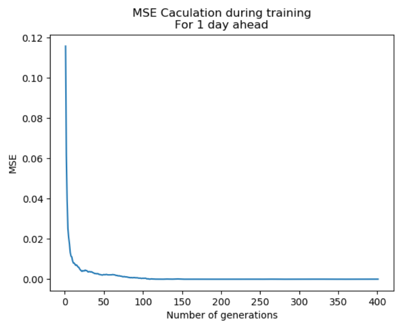
(d) Actual vs Predicted of PSO optimized CFLANN during Testing



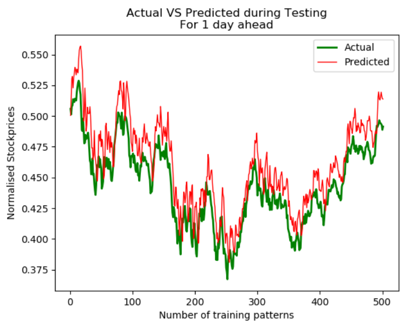
(e) MSE of WOA optimized CFLANN during Training



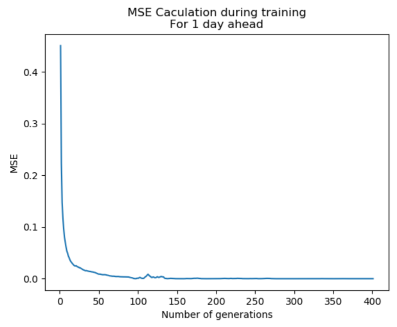
(f) Actual vs Predicted of WOA optimized CFLANN during Testing



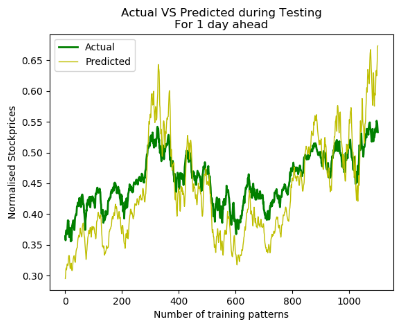
(g) MSE of JAYA optimized CFLANN during Training



(h) Actual vs Predicted of JAYA optimized CFLANN during Testing



(i) MSE of HS optimized CFLANN during Training



(j) Actual vs Predicted of HS trained CFLANN during Testing

Figure 4.7: DE, PSO, WOA, JAYA, HS optimized CFLANN during training and testing

From the MSE calculated, we found out that the best algorithm among DE, PSO, WOA, JAYA and HS is WOA in terms of Mean Square Error(MSE). Thus, keeping that aside we wanted to test if any hybrid algorithms could be formed that would perform better than basic algorithms [32]. The next best optimization techniques as observed from the experiment were JAYA and HS. Although DE and PSO also performed well in terms of MSE, JAYA and HS outperformed DE and PSO.

4.8 Analysis and Conclusion

We performed five powerful optimization techniques and compared between them in terms of the performance measure MSE. Out of the five models(DE, PSO, WOA, Jaya and HS), WOA performed the best. The next best optimization techniques were found to be Jaya and HS. Thus, a hybrid Jaya-HS model has been proposed in the next chapter. This hybrid model integrates the best features of the second superior optimization techniques Jaya and HS.

Chapter 5

Hybrid Jaya-HS trained Neural Network for Stock Market Prediction

5.1 Introduction

In chapter 3, we discussed the working of Jaya optimization technique [21]. In the previous chapter, we discussed the working of Harmony Search(HS) algorithm. We also compared the results of the famous Jaya optimization technique with other algorithms such as DE, PSO, WOA and HS. We found that WOA gave the best results and is superior to rest of the optimization techniques. As it is already performing very nicely, we shall keep that aside and try integrating models next superior to WOA, i.e Jaya and HS. From literature survey, we found that a hybrid Harmony Search-Teaching Learning Based Optimization (HSTLBO) performs remarkably on high dimensional optimization problems [33]. Thus, in this chapter, a hybrid model has been proposed that integrates the advantages of both Jaya and HS optimization techniques. We shall test this hybrid optimization technique on the same CFLANN model and study the results.

5.2 Proposed Model

The Jaya-HS optimized CFLANN model looks as follows:

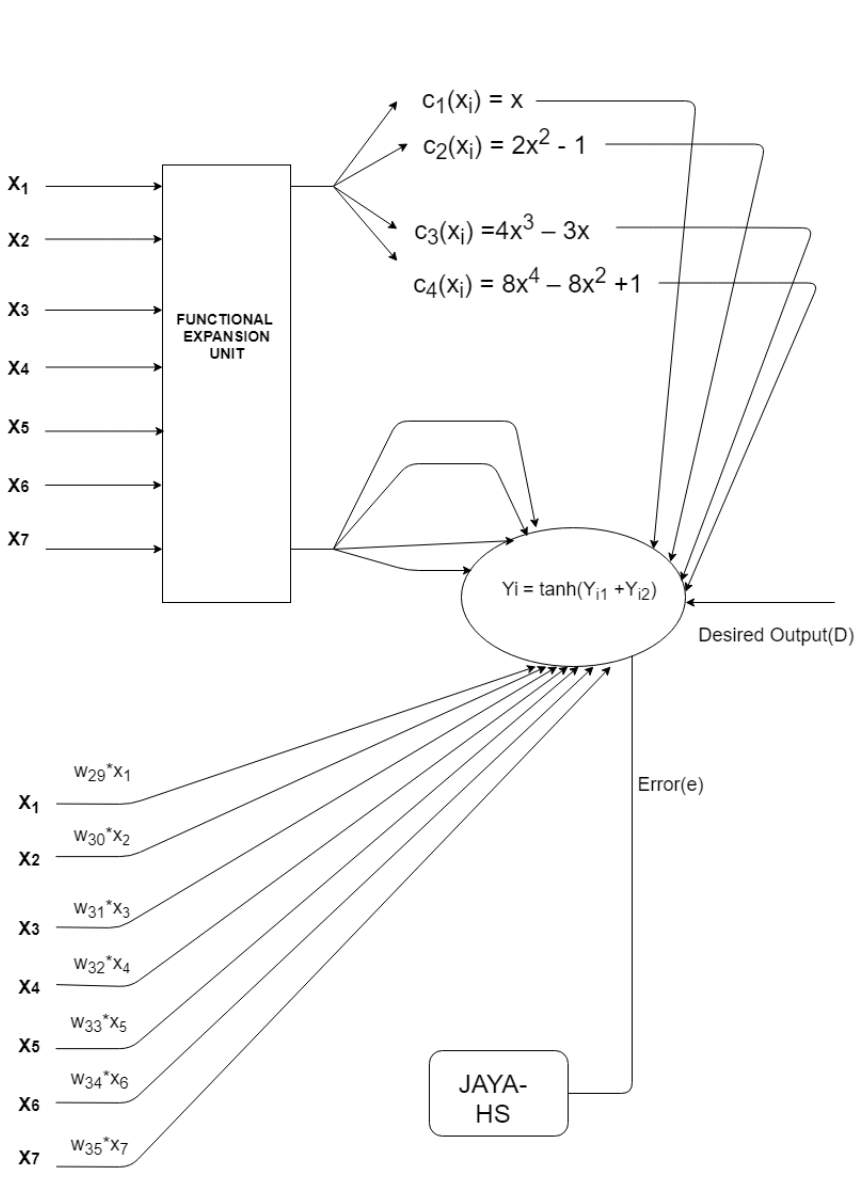


Figure 5.1: Hybrid Jaya-HS Optimized CFLANN Model

As we can see in this model, each input is fed into a *functional expansion unit* that produces an enhanced representation of the original pattern. In our proposed model, there are six inputs from the time series data and the seventh input is the mean of rest six inputs. The function used in this functional expansion unit is Chebyshev polynomial function. Thus, each input is expanded into 4 components. These 4 components along with inputs themselves are fed into the activation function along with randomly selected weights.

We calculate the output and generate its corresponding error. Further, the weights are updated using the Jaya-HS optimization technique.

The working of Hybrid Jaya-HS Optimization Technique has been explained below in steps.

- The initial population for the HS algorithm is randomly chosen where x_j = x_{ij} where i= 1, 2, 3...., HMS and j=1,2...n. The HM Matrix now looks as follows:

$$HM = \begin{bmatrix} x_{11} & x_{12} & x_{13} & . \; . \; . & x_{1n} \\ x_{21} & x_{22} & x_{23} & . \; . \; . & x_{2n} \\ . & . & . & . \; . \; . & . \\ . & . & . & . \; . \; . & . \\ . & . & . & . \; . \; . & . \\ x_{HMS1} & x_{HMS2} & x_{HMS3} & . \; . \; . & x_{HMSn} \end{bmatrix} \tag{5.1}$$

  We generate the $error^2$ corresponding to each row in HM matrix(i.e HMS number of $error^2$ values for i=1,2,3... HMS).

- By the Harmony Memory Consideration(HMC) rule, a new random number r_1 is generated and is compared to Harmony Memory Consideration Rate(hmcr).
  If $r_1 < hmcr$, then the x_{ij}^{new} is generated randomly by the formula:

$$x_{ij}^{new} = x_{ij}, x_{ij} \in \{x_{1j}, x_{2j}, x_{3j}..., x_{HMSj}\} \tag{5.2}$$

  If $r_1 \geq hmcr$, then by Random Initialization Rule, the first decision variable is chosen randomly from the current HM values by the rule:

$$x_{ij}^{new} = l_{ij} + (u_{ij} - l_{ij}).rand(0, 1) \tag{5.3}$$

  where l, u are the lower and upper bound of the problem respectively.

- Next, the pitch of the decision variables selected by HMC rule is examined to check if pitch adjustment is required or not. [34] Thus, a new random number r_2 is generated within the range [0,1]. If r_2 < Pitch Adjusting

Rate(PAR), then the decision variable undergoes pitch adjustment by the formula:

$$x_{ij}^{new} = x_{ij} \pm rand(0,1).BW \tag{5.4}$$

where BW represents Band Width. Now, we generate the $error_{new}^2$ corresponding to each row in HM matrix(i.e HMS number of $error^2$ values for i=1,2,3... HMS).

- By using Jaya algorithm, we will compare the $error^2$ with $error_{new}^2$ for each row i= 1, 2, 3..., HMS. If for the i^{th}, $error_{new}^2 < error^2$, the harmony memory is updated by the equation:

$$x_{worst}^i = x_{new}^i \tag{5.5}$$

and

$$error^2 = error_{new}^2 \tag{5.6}$$

otherwise, the previous values are maintained.

This updated HM matrix and $error_2$ array are further used for the next iteration to find the optimal solution and this continues until the stopping criterion is satisfied(i.e for the number of specified iterations).

The working of Jaya-HS has been explained in the flowchart below:

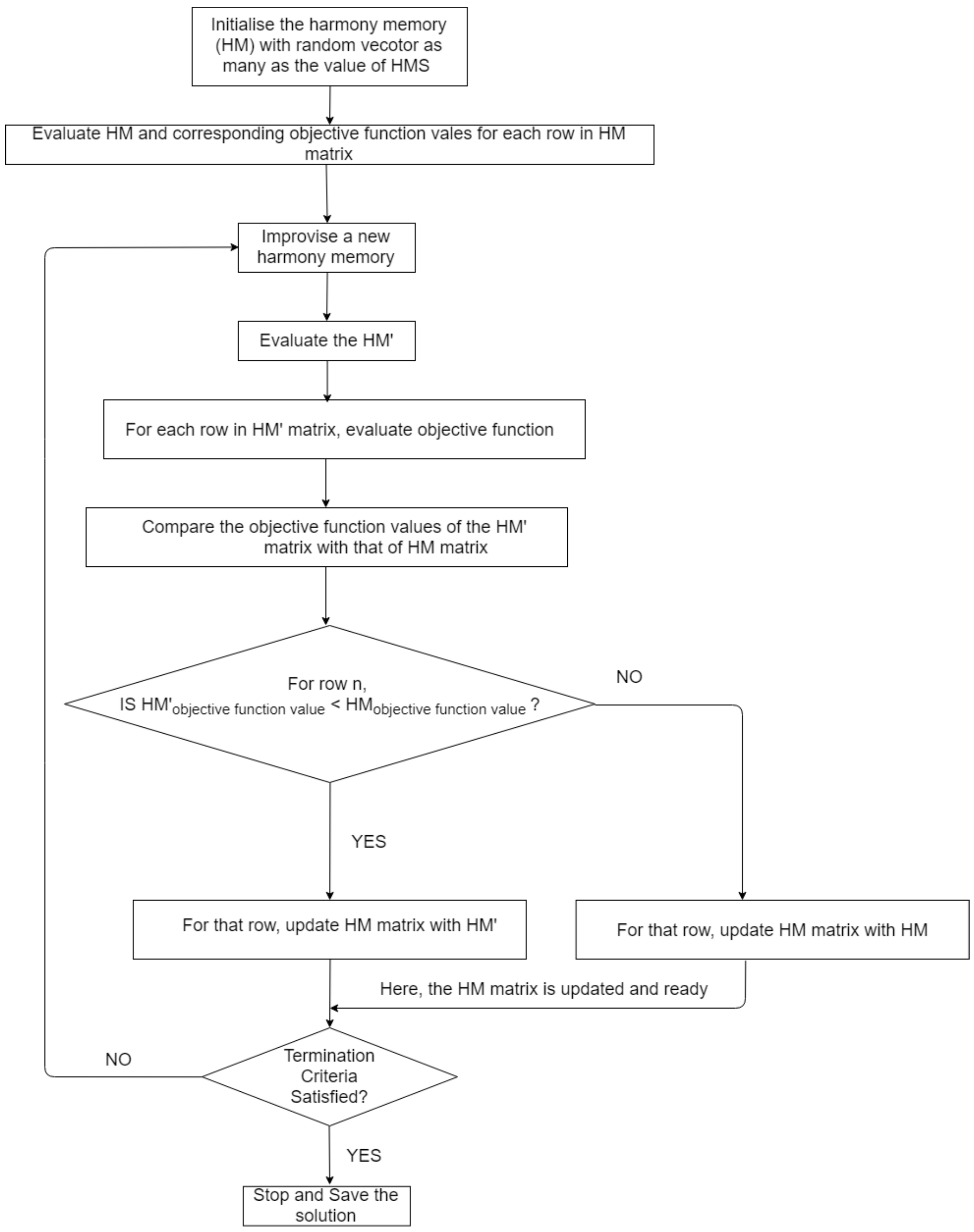
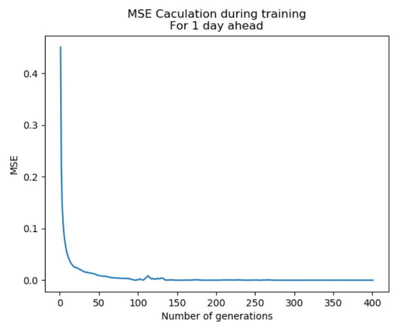

Initialise the harmony memory (HM) with random vecotor as many as the value of HMS
Evaluate HM and corresponding objective function vales for each row in HM matrix
Improvise a new harmony memory
Evaluate the HM'
For each row in HM' matrix, evaluate objective function
Compare the objective function values of the HM' matrix with that of HM matrix
For row n, IS HM'objective function value < HMobjective function value ?
NO
YES
For that row, update HM matrix with HM'
For that row, update HM matrix with HM
Here, the HM matrix is updated and ready
Termination Criteria Satisfied?
NO
YES
Stop and Save the solution

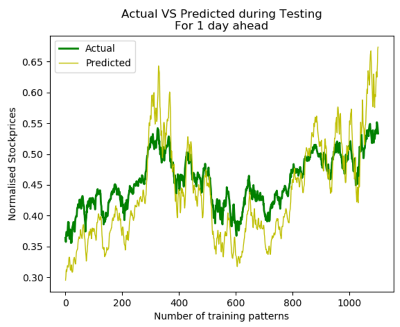
Figure 5.2: Flowchart of Hybrid Jaya-HS Optimization Technique

5.3 Results Discussion

The MSE during training and the Actual vs Predicted graph during testing of HS, Jaya and Hybrid Jaya-HS optimized CFLANN are depicted as follows:

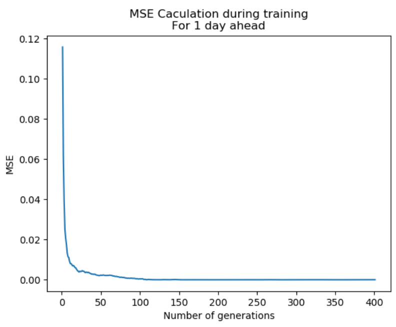


(a) MSE of HS optimized CFLANN during Training

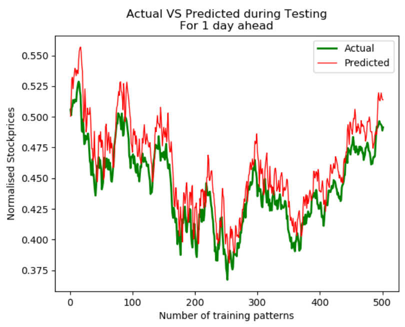


(b) Actual vs Predicted of HS optimized CFLANN during Testing

Figure 5.3: HS optimized CFLANN during training and testing

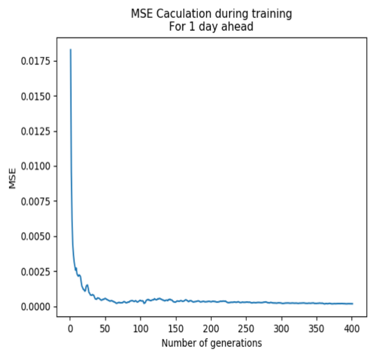


(a) MSE of Jaya optimized CFLANN during Training

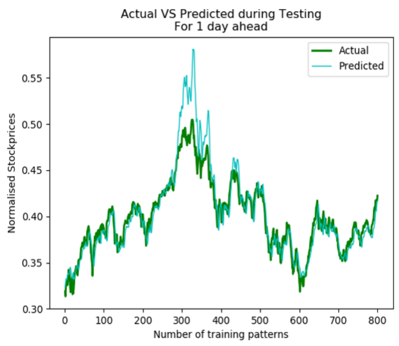


(b) Actual vs Predicted of Jaya optimized CFLANN during Testing

Figure 5.4: Jaya optimized CFLANN during training and testing



(a) MSE of Hybrid Jaya-HS optimized CFLANN during Training



(b) Actual vs Predicted of Hybrid Jaya-HS optimized CFLANN during Testing

Figure 5.5: Hybrid Jaya-HS optimized CFLANN during training and testing

At the end of the training process of optimized CFLANN, the MSE for HS, Jaya and Hybrid Jaya-HS were found as follows:

- HS - $11.12208e^{-05}$
- Jaya - $8.49541587e^{-05}$
- Hybrid Jaya-HS - $3.76993704e^{-05}$

5.4 Analysis and Conclusion

In this chapter, we extracted the advantageous features of HS and Jaya and performed hybrid HS-Jaya optimization on CFLANN model. We found out that the hybrid optimization model outperformed the basic HS and JAYA optimized models taking MSE into consideration. In the next chapter, in order to test the efficiency of the proposed hybrid optimization technique, we shall test it on a more complex neural network model.

Chapter 6

LRNFIS based Hybrid Jaya-HS optimized Deep Net

6.1 Introduction

In the previous chapter, we found out that the hybrid Jaya-HS trained FLANN model outperformed jaya and HS individually trained FLANN models. In this chapter, we shall check the efficiency of the hybrid Jaya-HS on a more complex model i.e a Locally Recurrent Neuro-fuzzy Information System (LRNFIS). [22]

In this model, there exists a feedback loop in the firing strength layer (i.e the layer that measures the degree to which the rule matches the inputs). This helps the model in memorizing the temporal nature of the fuzzy rule base. A similar loop is also included in the output layer which provides a dynamic nature to the time series data. In most neuro-fuzzy information systems, the fuzzy rule base is of TSK-type(i.e resembles Takagi Sugeno Kang model). However, the TSK-type feed forward recurrent fuzzy neural information system doesn't take the full advantage of the fuzzy rule base in information approximation [35]. Thus, we use Chebyshev polynomial function to complement that part. The Chebyshev Polynomial function model expands an input to the input space in a non-linear fashion, which will capture the chaotic variations in the time series data. From literature survey [22], we found that usually fuzzy information systems parameters are trained by gradient descent because of which they suffer from slow convergence and high computational complexity. Thus, evolutionary algorithms like PSO, HS and Firefly techniques are incorporated. To increase the accuracy further, we shall implement an improved hybrid Jaya-HS optimization technique on LRNFIS and check the accuracy of the optimization technique.

6.2 Architecture of LRNFIS Model

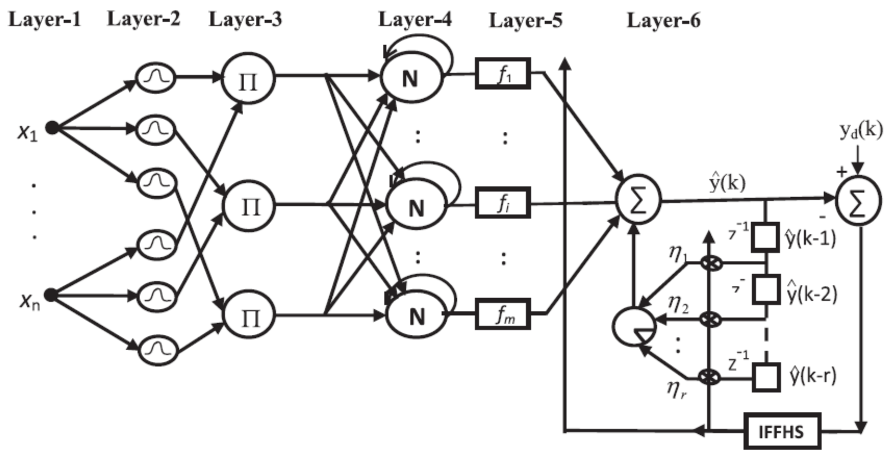


Figure 6.1: The Locally Recurrent Neuro-fuzzy Information System

The above architecture is a six-layer structure. As we can see, the main advantage of this model is that the firing strength of each fuzzy rule is fed back to itself. Thus, a locally recurrent fuzzy system is created that captures the dynamic behaviour of the highly fluctuating and volatile time series data. In this model, each layer performs a mathematical function which has been described below:

- *Layer 1 (The input layer)*: In this layer, inputs of the past time series data undergo a Chebyshev expansion and these crisp values are fed.

This layer involves no computation as there are no weights associated with the inputs. These inputs are further passed to the next layer for fuzzification.

The higher order Chebyshev polynomials can be given generalized recursive formula:

$$c_{r+1} = 2 * x * c_r(x) - c_{r-1}(x_i) \tag{6.1}$$

Thus we can obtain the Chevbyshev components as follows:

$\phi_0 = 1, \phi_1 = x_1, \phi_2 = 2x_1^2 - 1, \phi_3 = 4x_1^3 - 3x_1$

$\phi_{M-2} = x_n, \phi_{M-1} = 2x_n^2 - 1, \phi_M = 4x_N^3 - 3x_N$

– *Layer 2 (The fuzzification layer)*: Each node in this layer is associated with a Gaussian function that performs the fuzzification of the crisp inputs. For the i^{th} fuzzy set $A_{i,j}$, the input data x_j is fuzzified by the following formula.

$$\mu_{i,j}(x_j) = u_{j,2} = exp(\frac{-(x_j - c_{i,j})^2}{2\sigma_{i,j}^2}) \tag{6.2}$$

where $c_{i,j}$ = *center* of the i^{th} membership function for the j^{th} input and it is given by the formula:

$$c_{i,j} = \sum_{p=1}^{P} \frac{x_{p,j}}{P} \tag{6.3}$$

and $\sigma_{i,1}$ = *width* of the i^{th} membership function for the j^{th} input and it is given by the formula:

$$\sigma_{i,j} = \sum_{p=1}^{P} \frac{(x_{p,j} - \mu_{i,j})^2}{P} \tag{6.4}$$

for i=1,2,3,...,m , j=1,2,3...,n.

– *Layer 3 (The spatial firing layer)*: Each node of this layer functions represents a TSK-type fuzzy rule. This is used to evaluate the output by a T-norm (product or minimum) of the previous fuzzified inputs. The total number of nodes in this layer equals the total number of fuzzy rules used. The aggregate spatial firing strength is obtained as:

$$u_{j,3} = \prod_{i=1}^{N} exp\left\{ \frac{-(x_j - c_{i,j})^2}{2\sigma_{i,j}^2} \right\} \tag{6.5}$$

– *Layer 4 (The normalization and locally recurrent layer)*: In this layer, normalized temporal firing strength is found out by the formula:

$$u_{j,4} = u_{j,3} \Big/ \sum_{i=1}^{m} u_{i,3} \tag{6.6}$$

where m represents the total number of fuzzy rules used in the layer 2. In this layer, all the nodes are locally recurrent rule nodes. They provide a dynamic nature to the network structure and its outputs are temporal firing strength $O_{j,4}$ which is calculated as shown below.

$$O_{j,4}(k) = \lambda_j \times u_{j,4} + (1 - \lambda_j) \times O_{j,4}(k-1) \tag{6.7}$$

where $0 \leq \lambda_j \leq 1$ represents the internal recurrent feedback parameters. These feedback parameters control the contribution ratio of current spatial and delayed temporal strength to the final outputs to the network outputs.

– *Layer 5 (The activation layer)*: The non-linear functional output in this layer is obtained as:

$$f_j = tanh(\omega_{j0}\phi_0 + \omega_{j1}\phi_1 + ...\omega_{jM}\phi_M) \tag{6.8}$$

where M=3n and $\omega_{j0}, \omega_{j1}....\omega_{jM}$ are random weights.

– *Layer 6 (The output layer)*: In this layer, the recurrent weighted one step delayed past outputs are added to current outputs to increase the ability of the network. This is done to capture the dynamic behaviour of the time series data. Thus, the final output $\hat{y}(k)$ is the sum of the rule consequences and one step delayed past estimated outputs.

$$\hat{y}(k) = \sum_{j=1}^{M} O_{j,4}(k) f_j + \sum_{i=1}^{r} \eta_i y(\hat{k} - i) \tag{6.9}$$

where η_i represents the weights corresponding to the i^{th} delayed output for i=1,2,3,...r.

Thus, we can say the performance of the LRNFIS model is superior to FLANN in terms that it includes fine tuning of the internal and external feedback weights, and the antecedent and consequent weights.

6.3 Training the LRNFIS model with hybrid Jaya-HS algorithm

The resultant error of the model is represented as:

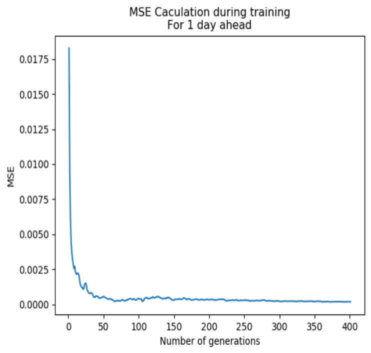
$$E(k) = \frac{1}{2}\Bigg(y_d(k) - \hat{y}(k) \Bigg) \tag{6.10}$$

Now, the weights of the internal and external feedback loops are updated with the main goal of minimizing the mean square error(MSE) of the prediction.

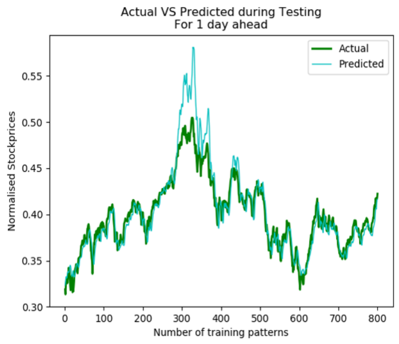
The weights of the layer 5 and layer 6 are trained by the hybrid Jaya-HS algorithm for around 10 generations. Kindly refer the previous chapter for the detailed working procedure of the hybrid Jaya-HS optimization algorithm. The updated weights are used in the next iteration and we train the network for our assumed number of iterations(i.e the stopping criterion).

6.4 Results Discussion

The MSE during training and the Actual vs Predicted graph during testing of the JAYA-HS optimised CFLANN and LRNFIS Deep Net are as follows.

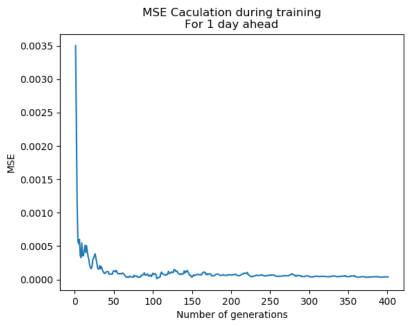


(a) MSE of Hybrid Jaya-HS optimized CFLANN during Training

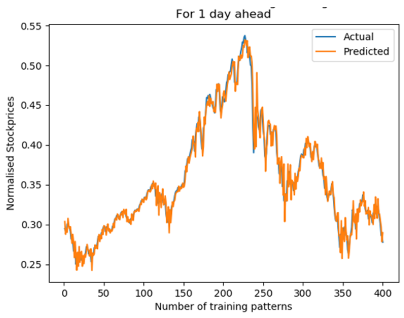


(b) Actual vs Predicted of Hybrid Jaya-HS optimized CFLANN during Testing

Figure 6.2: Hybrid Jaya-HS optimized CFLANN during training and testing



(a) MSE of Hybrid Jaya-HS optimized LRNFIS Deep Net during Training



(b) Actual vs Predicted of Hybrid Jaya-HS optimized LRNFIS Deep Net during Testing

Figure 6.3: Hybrid Jaya-HS optimized LRNFIN Deep Net during training and testing

At the end of the training process of optimized LRNFIS Deep Net, the MSE of hybrid Jaya-HS was found out and compared with MSE of hybrid Jaya-HS optimized CFLANN model. :

- Hybrid Jaya-HS on CFLANN Model- $3.76993704e^{-05}$
- Hybrid Jaya-HS on LRNFIS Deep Net- $1.25664568e^{-05}$

6.5 Analysis and Conclusion

In this chapter, we verified the working of hybrid Jaya-HS optimization technique on a more complex model i.e LRNFIS Deep Net and found very good results. The MSE after training LRNFIS model by our proposed optimization technique was found to be $1.25664568e^{-05}$ which is very less compared to that of the simple CFLANN model (in which the MSE after training was found to be $3.76993704e^{-05}$) i.e it got reduced by 66%. Thus, we verified that our proposed optimization technique successfully performs on stock data sets i.e BSE500, DJIA and NASDAQ on both simple FLANN model and LRNFIS Deep Net.

Chapter 7

Conclusion and Future Scope

Concluding chapter gives a summary of the thesis and states the future scope of the research. The study has explored and rigorously evaluated three popular stock datasets like BSE500, DJIA and NASDAQ.

Different types of neural network models and optimization techniques have been tested and investigated to develop suitable optimization technique for prediction of stock datasets.

Initially, different types of simple FLANN models optimized with BP were compared and was found that CFLANN outperformed TLANN in terms of MSE and MAPE. We also performed Jaya optimization of CFLANN model and found that it performs far better than the BP optimized CFLANN.

Further, we performed few more optimization techniques on CFLANN to compare the Jaya optimization technique with other already proven powerful optimization techniques. We concluded that WOA performed the best of all, Thus, keeping that aside we wanted to test if any hybrid algorithms could be formed that would perform tremendously better than basic algorithms. The second best algorithms were found to Jaya and HS. Thus, an integrated optimization technique was proposed that included the advantages of both HS and Jaya models. This technique is called Hybrid Jaya-HS. This technique was first tested on simple CFLANN and was found that it gave better results than basic Jaya and HS trained CFLANN model.

Further, to test the accuracy of the proposed optimization technique, we tested the algorithm with a more complex model called LRNFIS Deep Net. We found that it performed far better than the hybrid Jaya-HS optimised CFLANN model.

Finally, the conclusions of models and optimization techniques from the thesis are:

– Chebyshev-FLANN outperformed Trigonometric and Laguerre-FLANN.

– WOA was found to be superior to DE, PSO, Jaya and HS optimization techniques.

– A hybrid Jaya-HS optimization technique was suggested and was found to be superior to basic Jaya and HS optimization algorithms.

– The hybrid Jaya-HS was tested on both simple CFLANN and complex LRNFIS deep net. It was found that on LRNFIS model, the MSE was 66% less as compared to the that on simple CFLANN model.

The future development of the study might be following:

– Implementing the hybrid Jaya-HS optimization technique with Multilayer Perceptrons.

– Implementing the hybrid Jaya-HS optimization technique with Long Short-Term Memory(LSTM) Recurrent Neural Networks.

– Implementing the hybrid Jaya-HS optimization technique with Radial Basis Functional Neural Networks.

– Implementing the hybrid Jaya-HS optimization technique with other Deep Neural Network models.

Bibliography

[1] R. Adhikari and R. Agrawal, "An introductory study on time series modeling and forecasting," *arXiv preprint arXiv:1302.6613*, 2013.

[2] L. T. Bui, T. T. H. Dinh *et al.*, "A novel evolutionary multi-objective ensemble learning approach for forecasting currency exchange rates," *Data & Knowledge Engineering*, 2017.

[3] R. Majhi, G. Panda, and G. Sahoo, "Development and performance evaluation of flann based model for forecasting of stock markets," *Expert systems with applications*, vol. 36, no. 3, pp. 6800–6808, 2009.

[4] S. Das, A. Patra, S. Mishra, and M. R. Senapati, "A self-adaptive fuzzy-based optimised functional link artificial neural network model for financial time series prediction," *International Journal of Business Forecasting and Marketing Intelligence*, vol. 2, no. 1, pp. 55–77, 2015.

[5] J. Graf, "Stock market prediction with neural networks," in *Operations Research91*. Springer, 1992, pp. 496–499.

[6] C. Mohapatra and S. Ron, "Flann based model to predict stock price movements of stock indices," Ph.D. dissertation, 2007.

[7] P. Mohapatra, A. Raj, and T. K. Patra, "Indian stock market prediction using differential evolutionary neural network model," *International Journal of Electronics Communication and Computer Technology (IJECCT) Volume*, vol. 2, 2012.

[8] P. Chujai, N. Kerdprasop, and K. Kerdprasop, "Time series analysis of household electric consumption with arima and arma models," in

Proceedings of the International MultiConference of Engineers and Computer Scientists, vol. 1, 2013, pp. 295–300.

[9] P. Mohapatra, M. Anirudh, and T. K. Patra, “Forex forecasting: A comparative study of llwnn and neurofuzzy hybrid model,” *International Journal of Computer Applications*, vol. 66, no. 18, 2013.

[10] D. M. AL-Najjar, “Modelling and estimation of volatility using arch/garch models in jordans stock market,” *Asian Journal of Finance & Accounting*, vol. 8, no. 1, pp. 152–167, 2016.

[11] J. Koima, P. Mwita, and D. Nassiuma, “Volatility estimation of stock prices using garch method,” 2015.

[12] A. Birgul Egeli, “Stock market prediction using artificial neural networks,” *Decision Support Systems*, vol. 22, pp. 171–185, 2003.

[13] X. Xing, “Stock price forecasting using rbf neural network and hybrid particle swarm optimization.”

[14] J.-S. R. Jang, C.-T. Sun, and E. Mizutani, “Neuro-fuzzy and soft computing; a computational approach to learning and machine intelligence,” 1997.

[15] S. M. Heakel and M. E. Khedr, “Realization of strong ports and shipping services in developing countries using multi-criteria selection algorithm,” in *Logistics Systems and Intelligent Management, 2010 International Conference on*, vol. 3. IEEE, 2010, pp. 1638–1642.

[16] J. A. Roubos, S. Mollov, R. Babuška, and H. B. Verbruggen, “Fuzzy model-based predictive control using takagi–sugeno models,” *International Journal of Approximate Reasoning*, vol. 22, no. 1-2, pp. 3–30, 1999.

[17] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010, pp. 249–256.

[18] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Advances in neural information processing systems*, 2014, pp. 3320–3328.

[19] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[20] P. J. Angeline, "Evolutionary optimization versus particle swarm optimization: Philosophy and performance differences," in *International Conference on Evolutionary Programming*. Springer, 1998, pp. 601–610.

[21] R. Rao, "Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems," *International Journal of Industrial Engineering Computations*, vol. 7, no. 1, pp. 19–34, 2016.

[22] A. Parida, R. Bisoi, and P. Dash, "Chebyshev polynomial functions based locally recurrent neuro-fuzzy information system for prediction of financial and energy market data," *The Journal of Finance and Data Science*, vol. 2, no. 3, pp. 202–223, 2016.

[23] T. R. Benala, K. Chinnababu, R. Mall, and S. Dehuri, "A particle swarm optimized functional link artificial neural network (pso-flann) in software cost estimation," in *Proceedings of the International Conference on Frontiers of Intelligent Computing: Theory and Applications (FICTA)*. Springer, 2013, pp. 59–66.

[24] D. K. Bebarta, B. Biswal, and P. K. Dash, "Polynomial based functional link artificial recurrent neural network adaptive system for predicting indian stocks," *International Journal of Computational Intelligence Systems*, vol. 8, no. 6, pp. 1004–1016, 2015.

[25] G. Jothi, H. H. Inbarani, A. T. Azar, and K. R. Devi, "Rough set theory with jaya optimization for acute lymphoblastic leukemia classification," *Neural Computing and Applications*, pp. 1–20, 2018.

[26] R. Rao, K. More, J. Taler, and P. Ocłoń, “Dimensional optimization of a micro-channel heat sink using jaya algorithm,” *Applied Thermal Engineering*, vol. 103, pp. 572–582, 2016.

[27] J. Q. Puma and D. G. Colome, “Parameters identification of excitation system models using genetic algorithms,” *IET generation, transmission & distribution*, vol. 2, no. 3, pp. 456–467, 2008.

[28] P. Mohapatra, S. Das, T. K. Patra, and M. Anirudh, “Stock volatility forecasting using swarm optimized hybrid network,” *International Journal of Emerging Trends and Technology in Computer Science (IJETTCS)*, vol. 2, no. 3, pp. 78–85, 2013.

[29] X. Zhang, Y. Chen, and J. Y. Yang, “Stock index forecasting using pso based selective neural network ensemble.” in *IC-AI*, 2007, pp. 260–264.

[30] S. Mirjalili and A. Lewis, “The whale optimization algorithm,” *Advances in Engineering Software*, vol. 95, pp. 51–67, 2016.

[31] Z. W. Geem, “State-of-the-art in the structure of harmony search algorithm,” in *Recent advances in harmony search algorithm.* Springer, 2010, pp. 1–10.

[32] B. Al-hnaity and M. Abbod, “Predicting financial time series data using hybrid model,” in *Intelligent Systems and Applications.* Springer, 2016, pp. 19–41.

[33] S. Tuo, L. Yong, Y. Li, Y. Lin, Q. Lu *et al.*, “Hstlbo: A hybrid algorithm based on harmony search and teaching-learning-based optimization for complex high-dimensional optimization problems,” *PloS one*, vol. 12, no. 4, p. e0175114, 2017.

[34] M. A. Al-Betar, I. A. Doush, A. T. Khader, and M. A. Awadallah, “Novel selection schemes for harmony search,” *Applied Mathematics and Computation*, vol. 218, no. 10, pp. 6095–6117, 2012.

[35] N. Maknickienė, A. V. Rutkauskas, and A. Maknickas, “Investigation of financial market prediction by recurrent neural network,” *Innovative Technologies for Science, Business and Education*, vol. 2, no. 11, pp. 3–8, 2011.

Publications

Submitted papers

1. P. Mohapatra, R. Mishra, and T. K. Patra, "A Jaya Algorithm trained FLANN model for Stock Market Prediction," submitted to *International Journal of Latest Trends in Engineering and Technology*, e-ISSN: 2278-62IX.
2. P. Mohapatra, R. Mishra, and T. K. Patra, "Financial Stock Market Prediction using Whale Optimization ALgorithm Trained FLANN Models," submitted to *Intelligent Decission Technologies of Scientific Research*
3. P. Mohapatra, R. Mishra, and T. K. Patra, "An Integrated Jaya-HS trained FLANN Model for Stock Market Prediction," submitted to *International Journal of Knowledge-based and intelligent engineering systems*
4. R. Mishra and P. Mohapatra, "A Locally Recurrent Neuro-fuzzy information System based hybrid Jaya-HS optimized Deep Net for Stock Market Prediction," submitted to *Intelligent Decission Technologies of Scientific Research*