2.1 Introduction

This chapter maps out the development of the PSO based Functional Link Interval Type-2 Fuzzy Neural System (FLIT2FNS) model used to forecast the stock market indices. In the process, it discusses the architecture of Functional Link Artificial Neural Network (FLANN), FLANN & Type-1 Fuzzy Logic System (Type-1FLS) and the differences between Type-1 FLS and Interval Type-2 Fuzzy Logic System (IT2FLS). The discussion aims at highlighting the relative advantages and disadvantages of all these models. It also shows the superiority of the hybrid model FLIT2FNS.

Secondly, in this chapter, the architecture of Local Linear Wavelet Neural Network (LLWNN) model has been discussed as this model has been taken for a comparative study with FLIT2FNS. Thirdly, Backpropagation and Particle Swarm Optimization (PSO) learning algorithms, used to optimize parameters of all the models are described. To test the model performance, three well known stock market indices namely: Standard’s & Poor’s 500 (S&P 500), Bombay Stock Exchange (BSE), and Dow Jones Industrial Average (DJIA) are taken as experimental data. The Mean Absolute Percentage Error (MAPE) and Root Mean Square Error (RMSE) are used to find out the performance of the models.
2.2 The models used for Stock price forecasting

2.2.1 Functional Link Artificial Neural Network (FLANN)

Researchers have used ANN tools such as, radial basis function (RBF), recurrent neural network (RNN) and Multilayered Perceptron (MLP) (discussed in chapter one) for financial time series forecasting [6, 24, 35, 46, 47]. Of them, MLP is mostly used for its inherent capabilities to approximate any non-linear function to a high degree of accuracy. However, it has two major limitations: (1) Convergence speed is slow; and (2) Computational complexity is higher. To overcome these two limitations, a different kind of ANN i.e. FLANN is used which is a single neuron and single layer architecture [26, 54]. Pao, the pioneer of FLANN, has shown that this network may be conveniently used for functional approximation and pattern classification with faster convergence rate and lesser computational load. In FLANN, each input of the network undergoes functional expansion through a set of basis functions. The functional link generates a set of linearly independent functions. The inputs expanded by a set of linearly independent functions in the functional expansion block cause an increase in input vector dimensionality. Fig. 2.1 describes the structure of FLANN with the BP learning algorithm.

![Fig. 2.1 Structure of FLANN with BP](image-url)
In this model, the functional expansion block comprises a set of trigonometric functions.

\[ \hat{y}_j = \tanh(w_{110} + w_1 x_1 + w_2 x_2 + \ldots + w_n x_n + w_{net} \text{sma}) \]  

where \( X=\{x_1, x_2, \ldots, x_n\} \) is the input vector and \( W=\{w_1, w_2, \ldots, w_n\} \) is the weight vector associated with the \( J^{th} \) output of FLANN. \( w_{110} \) is the bias and \( \text{sma} \) (simple moving average) is an important technical indicator among the five indicators used to develop the model. One independent weight is attached for each indicator. \( \hat{y} \) denotes the output of FLANN structure.

### 2.2.2 Functional Link Artificial Neural Network and Type-1FLS

Fuzzy Logic System (discussed in chapter one) is famous for its precision and dealing with uncertainty. On the other hand, neural network is known for learning, adaptation, fault tolerance, parallelism, and generalization [5, 48, 50, 51]. In this model, both FLANN and FLS are combined to get the advantages from both the tools [13, 29, 47, 56, 90, 91]. The model uses a functional link neural network to the consequent part of the fuzzy rules. The consequent part of the model is a non-linear combination of input variables. Each fuzzy rule corresponds to a sub-FLANN, comprising a functional link.
The structure of the FLANN and Type-1FLS model is given below (Fig. 2.2).

This model uses a fuzzy IF-THEN rule in the following form:

Rule j:

IF \( x_1 \) is \( A_{1j} \) and \( x_2 \) is \( B_{1j} \) ………. and \( x_7 \) is \( G_{1j} \)

THEN \( \hat{y}_j = \sum_{k=1}^{M} w_{kj} \phi_k \) \hspace{1cm} (2.2)

where \( X = [x_1, x_2, \ldots, x_7] \) is the input vector and \( \hat{y}_j \) is the local output variables. \([A_{1j}, B_{1j}, \ldots, G_{1j}]\) are the linguistic terms of the precondition part with Gaussian membership function. \( w_{kj} \) is the link weight of the local output. \( \phi_k \) is the basis trigonometric function of input variables. \( M \) is the number of basis function and rule \( j \) is the \( j^{th} \) fuzzy rule. There are five layers in FLANN and Type-1FLS model and the functions of nodes in each layer are described below.
Layer 1

No computation is required in layer 1. In this layer, each node only transmits input values to the next layer.

\[ O^{(1)}_i = x_i \]  

(2.3)

Layer 2

Each fuzzy set \( A_{ij} \) is described by a Gaussian membership function. Symbolically, it is represented by the symbol \( \mathcal{A} \) in Fig. 2.

\[ O^{(2)}_y = \frac{(\exp(-((x_y - c_{ij}))^2) / \sigma_{ij}^2) \in A_y}{\sigma_{ij}^2} \]  

(2.4)

where \( c_{ij} \) is the center and \( \sigma_{ij} \) is the width.

Layer 3

Layer 3 receives one dimensional membership degrees of the associated rule from the nodes of layer 2. Now, the output function of each inference node is

\[ O^{(3)}_j = \mu_j = \prod O^{(2)}_y \]  

(2.5)

where \( \prod O_y \) of a rule node represents the firing strength of its corresponding rule.

Layer 4

In layer 4, nodes are called consequent nodes. The outputs collected both from FLANN and layer 3 are considered as the input to layer 4.

\[ O^{(4)}_i = \delta_j = \mu_j \sum_{k=1}^{M} w_{kj}\phi_k \]  

(2.6)

where \( w_{kj} \) is the link weight of FLANN and \( \phi_k \) is the functional expansion of input variables.
Layer 5

The output node in layer 5 acts as a defuzzifier that combines all the actions recommended by layer 3 and layer 4.

\[ O^{(5)} = y = \frac{\mu_1(\delta_1) + \mu_2(\delta_2)}{\mu_1 + \mu_2} \]  

(2.7)

where \( \delta_1 \) and \( \delta_2 \) are the outputs of FLANN and \( \mu_1 \) and \( \mu_2 \) are the outputs of Layer 3.

After passing through four stages, such as, fuzzifier, rule base, fuzzy inference engine and defuzzifier, a crisp value is obtained from FLANN and Type-1FLS model. This crisp value is considered as the model output

2.2.3 Type-1 Fuzzy Logic System vs. Interval Type-2 Fuzzy Logic System

Fuzzy logic system is one of the most efficient tools to handle uncertainty in real-life decision making process. Unlike neural network, it is transparent and robust in nature [33]. Commonly, two types of FLS are seen in the literature: Type-1FLS and Type-2FLS [92, 93, 94]. As Type-1 fuzzy sets express the belongingness of a crisp value \( x' \) of a base variable \( x \) in a fuzzy set \( A \) by a crisp membership value \( \mu_x(x') \), they cannot capture the uncertainties due to imprecision in identifying membership functions. In other words, the membership function (MF) of a traditional FLS i.e. Type-1 fuzzy sets has no uncertainty associated with it. To overcome this limitation, Zadeh, the pioneer of fuzzy logic system, has introduced Type-2FLS that can be applied to minimize the effects of the uncertainty in the rule base. Type-2 FLS is used in the areas like control of mobile robots, fuzzy granulation, image processing, pattern
recognition and speech recognition. But in recent years, because of its computational complexity researchers preferred to use IT2FLS, where the computation is more similar with the Type-1FLS. In other words, Type-2FLS is computationally difficult but the process can be simplified to a great extent if Type-2 fuzzy sets are defined as Interval Type-2 fuzzy sets [34, 95, 96, 97]. Symbolically, the difference between Type-1 and both Type-2 and IT2FLS are marked by a tilde symbol. For example, if \( A \) denotes a Type-1 fuzzy set, than \( \tilde{A} \) denotes the Interval Type-2 fuzzy set or Type-2 fuzzy set.

A Type-2 fuzzy set, denoted \( \tilde{A} \), is characterized by a Type-2 MF \( \mu_{\tilde{A}}(x,u) \), where \( x \in X \) and \( u \in J_x \subseteq [0,1] \), that is

\[
\tilde{A} = \{(x,u), \mu_{\tilde{A}}(x,u)\} | \forall x \in X, \forall u \in J_x \subseteq [0,1] \}
\]

(2.8)
in which \( 0 \leq \mu_{\tilde{A}}(x,u) \leq 1 \).

In equation (2.8), \( \forall u \in J_x \subseteq [0,1] \) is the first restriction. It is consistent with Type-1 constraint that is \( 0 \leq \mu_{\tilde{A}}(x) \leq 1 \); it means when uncertainties disappear a Type-2 MF reduce to Type-1MF and the third dimension disappears. In this case, the variable \( u \) equals \( \mu_{\tilde{A}}(x) \) and \( 0 \leq \mu_{\tilde{A}}(x) \leq 1 \). The second restriction is \( 0 \leq \mu_{\tilde{A}}(x,u) \leq 1 \). It is consistent with the fact that the amplitude of a MF should lie between or be equal to 0 and 1. When all \( \mu_{\tilde{A}}(x,u) = 1 \), then \( \tilde{A} \) is an Interval Type-2 fuzzy set.

Like Type-2 fuzzy set, Interval Type-2 fuzzy set is also a three-dimensional set; but the value of third dimension in Interval Type-2 fuzzy set is always same. This means that no new information is contained in the third dimension. In IT2FLS, there
are two membership functions $\mu_{A(x)}$ and upper membership functions $\overline{\mu}_{A(x)}$. Each of two membership functions can be represented by a Type-1 fuzzy set membership function. The interval between these two membership values represents the Foot print Of Uncertainty (FOU). FOU is used to describe an Interval Type-2 fuzzy set. It is the union of all primary membership functions and consists of a bounded region shown in Fig. 2.3. It shows the uncertainty in shapes or other parameters of membership functions. When the uncertainty disappears, the Interval Type-2 fuzzy sets get reduced to Type-1 fuzzy sets. Most of the steps for Type-1 and Interval Type-2 fuzzy sets are similar except the type reducer one that marks the difference between these two fuzzy logic systems. This step is necessary because membership grades of Interval Type-2 fuzzy sets are fuzzy set in $(0, 1)$ and thus it has to be reduced to Type-1 fuzzy set before defuzzifier is able to reduce the output fuzzy set further into crisp value. This step is called type reduction because this takes us from IT2FLS output sets to a Type-1 set.
2.2.4 Structure of Functional Link Interval Type-2 Fuzzy Neural System (FLIT2FNS)

The proposed integrated model FLIT2FNS - a combination of FLANN and IT2FLS - has drawn on the advantages of both techniques discussed above. The architecture of the model is represented in Fig. 2.4.
In this model, the functional expansion block comprises a set of trigonometric functions.

\[ y_1 = \tanh \left( w_{110} + w_{11}x_1 + w_{12}x_2 + \ldots + w_{1n}x_n + sma \right) \]  

\[ y_2 = \tanh \left( w_{220} + w_{12}x_1 + w_{22}x_2 + \ldots + w_{2n}x_n + sma \right) \]

where \( X = [x_1, x_2, \ldots, x_n] \) is the input vector and \( W = [w_1, w_2, \ldots, w_n] \) is the weight vector of FLANN. The weight \( w_{110} \) and \( w_{220} \) are the bias and \( sma \) is an important technical indicator among the five indicators used to develop the model.

For each indicator, one independent weight is attached; \( y_1 \) and \( y_2 \) denote the local...
outputs of FLANN structure and the consequent part of $j^{th}$ fuzzy rule in both FLIT2FNS and FLANN and Type-1FLS models. The number of fuzzy rules should match with the number of local outputs of FLANN. The $tanh$ function is used to provide the required nonlinearity in the output. Further, in terms of real number, it provides a range from -1 to +1 and is used extensively in system identification and as an activation function in MLP networks.

The antecedent part of the FLIT2FNS model is developed by using the IT2FLS. The details of IT2FLS and the difference between Type-1 and Type-2 FLS have been discussed above. The structure of an IT2FLS used in the model is based on the Takagi–Sugano Kang (TSK) principle [3, 27, 98]. It is:

$j^{th}$ rule:

$$\text{If } x_1 \text{ is } \tilde{A}_j \text{ and } x_2 \text{ is } \tilde{B}_j \text{ and } \ldots \ldots \text{ and } x_7 \text{ is } \tilde{G}_j$$

Then $y_j = \sum_{j=1}^{r} \tanh(a_{0j} + a_{1j}x_1 + \ldots + a_{7j}x_7)$  \hspace{1cm} (2.11)

where $j = [1,2,\ldots,r]$, $r$ is the number of rules. There are two rules in FLIT2FNS model. $\tilde{A}_j$, $\tilde{B}_j$, $\ldots$, $\tilde{G}_j$ represent Interval Type-2 fuzzy sets of input state in the $j^{th}$ rule. $(x_1, x_2, \ldots, x_7)$ are the inputs and $(a_1, a_2, \ldots, a_7)$ are the coefficients of output function. These rules are helpful to design the uncertainties encountered in the antecedents. Like FLANN and Type-1FLS model, FLIT2FNS also consists of five layers. The operations of nodes in each layer are described below.
Layer 1

It is the input layer. Total 7 samples from the respective datasets and five technical indicators are considered as the input to the FLIT2FNS model. Among the five indicators, only one indicator i.e. \( \text{sma} \) is shown in the Fig.2.4. As no computation is needed in this layer, it only transmits the value to the next layer.

\[
O^{(1)}_i = x_i
\]  

(2.12)

Layer 2

Each fuzzy set is described by a Gaussian membership function. As IT2FLS is used, each member deals with two membership functions, one lower and another upper. Only one sample \( \tilde{A}_i \) is described as other samples from \( \tilde{B}_i \) to \( \tilde{G}_i \) are similar.

\[
O^{(2)}_i = \tilde{A}_i = \exp \left( -\frac{\left( O^{(1)}_i - m_i \right)^2}{\sigma_i^2} \right)
\]  

(2.13)

\[
O^{(2)}_i = \tilde{A}_i = \exp \left( -\frac{\left( O^{(1)}_i - m_i \right)^2}{\sigma_i^2} \right)
\]  

(2.14)

where \( m_i \) and \( \bar{m}_i \) are the centers of the lower and upper membership function respectively and \( \sigma_i \) and \( \bar{\sigma}_i \) are the widths of two membership functions.  

\( i = [1,2,\ldots,n] \), \( i \) is the number of iteration. Similarly, other outputs of this layer have been computed.
Layer 3

Nodes in layer 3 receive the membership degrees of associated rule from the nodes of layer 2. In the proposed model, two rules are generated each having two values (lower and upper). Thus, total output of this layer is 4. Both rules being similar, only one rule having two values is given below:

\[ O^{(3)}_y = \mu_i = A_i \ast \bar{B}_i \ast \cdots \cdots \ast G_i \]  \hspace{1cm} (2.15)

\[ O^{(3)}_\bar{y} = \bar{\mu}_i = A_i \ast \bar{B}_i \ast \cdots \cdots \ast \bar{G}_i \]  \hspace{1cm} (2.16)

here, \( i \) denotes \([1, ..., n]\) and \( n \) is the number of iterations and \( j \) represents \([1, ..., r]\) where \( r \) is the number of rules. In the above equation, \( \ast \) represents the t-norm i.e. a product t-norm. It is a binary algebraic operation on the interval \((0, 1)\).

Layer 4

Nodes in layer 4 are called consequent nodes. The outputs obtained from layer 3 and the two local outputs of FLANN are considered as the inputs to this layer. Two outputs of layer 4, \( y_1 \) and \( y_r \), are

\[ O^{(4)}_1 = y_1 = \frac{\mu_1 \cdot \bar{y}_1 + \mu_2 \cdot \bar{y}_2}{\mu_1 + \mu_2} \]  \hspace{1cm} (2.17)

\[ O^{(4)}_r = y_r = \frac{\bar{\mu}_1 \cdot y_1 + \bar{\mu}_2 \cdot y_2}{\bar{\mu}_1 + \bar{\mu}_2} \]  \hspace{1cm} (2.18)

where \([\mu_1, \bar{\mu}_1, \mu_2, \bar{\mu}_2]\) are the outputs of layer 3 and \([y_1, y_2]\) are outputs of the FLANN.
Layer 5

It is the defuzzification layer, and the output obtained from this layer is given by

$$O^{(s)}_i = y = \lambda y_i + (1 - \lambda) y_r$$  

(2.19)

where $\lambda$ lies between 0 and 1. When $\lambda = 0.5$, the equation (2.19) will become

$$y = \frac{y_i + y_r}{2}$$

Finally, a crisp value is obtained from FLIT2FNS model after passing through four stages: fuzzifier, rule base, fuzzy inference engine, and output processor. The output processor includes type reducer and defuzzifier. For an IT2FLS, the crisp value is the center of the type-reduced set. This crisp value is considered as the model output.

2.2.5 Local Linear Wavelet Neural Network

The second model chosen for the comparison is the Local Linear Wavelet Neural Network (LLWNN) model [52, 53]. Wavelet Neural Network (WNN) is a combination of neural network and wavelet technique. The disadvantage of WNN is that when the dimension increases, the number of hidden layers increases to an undefined number. Similar is the problem with neurons in the hidden layers. As a result, calculation becomes more complicated. But, the advantage of LLWNN is that it needs less hidden layer units. Secondly, the number of neurons in hidden layer is equal to the number neurons present in input layer. Thirdly, connection weights between the hidden layer units and output units are replaced by a local linear model.
According to wavelet transformation theory, wavelets are defined as:

\[ \psi = \left\{ \psi_j = \left| a_i \right|^{-1/2} \psi \left( \frac{x-b_i}{a_i} \right) : a_i, b_i \in \mathbb{R}^n, i \in \mathbb{Z} \right\}, \]  

\[ x = (x_1, x_2, \ldots, x_n), \]

\[ a_i = (a_{i1}, a_{i2}, \ldots, a_{in}), \]

\[ b_i = (b_{i1}, b_{i2}, \ldots, b_{in}), \]

are a family of functions generated from one single function \( \psi(x) \) by the operation of dilation and translation. \( \Psi(x) \) is localized in both time space and the frequency space. It is called a mother wavelet and the parameters \( a_i \) and \( b_i \) are the scale and translation parameters, respectively.

The mother wavelet is

i) \[ \psi(x) = \frac{-x^2}{2} e^{-x^2/\sigma^2} \]  

\[ (2.21) \]

ii) \[ \psi(x) = e^{-\left( \frac{x-c}{\sigma} \right)^2} \]  

\[ (2.22) \]

where \( x = \sqrt{p_1^2 + p_2^2 + \ldots + p_n^2} \)  

\[ (2.23) \]

Instead of the straightforward weight \( w_i \) (piecewise constant model), a linear model

\[ v_i = w_{i0} + w_{i1}x_1 + \ldots + w_{in}x_n \]  

\[ (2.24) \]
The activities of the linear models $v_i$ (i=1,2,--------,M) are determined by the associated locally active wavelet functions $\psi_i(x)$ (i=1,2,--------,M), thus $v_i$ is only locally significant.

The standard output of WNN is

$$f(x) = \sum_{i=1}^{M} w_i \psi_i(x)$$

$$= \sum_{i=1}^{M} w_i |a_i| ^{1/2} \psi \left( \frac{x-b_i}{a_i} \right) \tag{2.25}$$

where $\psi_i$ is the wavelet activation of $i^{th}$ unit of hidden layer and $w_i$ is the weight connecting the $i^{th}$ unit of the hidden layer to the output layer unit.

In LLWNN model, the output in the output layer is

$$Y = \sum_{i=1}^{M} (w_{i0} + w_{i1} x_1 + \cdots + w_{in} x_n) \psi_i(x)$$

$$= \sum_{i=1}^{M} (w_{i0} + w_{i1} x_1 + \cdots + w_{in} x_n) |a_i|^{1/2} \psi \left( \frac{x-b_i}{a_i} \right) \tag{2.26}$$

The architecture of LLWNN with backpropagation learning is given in Fig. 2.5.
2.3 Learning Algorithms

This Section has briefly delineated the two learning algorithms i.e., BP and PSO used independently to optimize the parameters of all the models considered.

2.3.1 Backpropagation learning algorithm for FLIT2FNS Model

Backpropagation algorithm, a tested supervised learning algorithm, is used to minimize the objective function by adjusting the link weights used in the consequent part and the parameters used to generate Gaussian membership functions. The gradient of the cost function with respect to that particular weight parameter is calculated and the parameters are adjusted with the negative gradient. The objective function is given by

$$E(t) = \frac{1}{2}[(\hat{y}(t) - y(t))]^2$$  \hspace{1cm} (2.27)
where \( j(t) \) is the model output and \( y(t) \) the desired value. As this is a hybrid model and the outputs of FLANN are fed to the consequent part of the FLIT2FNS model, the work of the algorithm is to optimize the weights used in both antecedent and consequent part of the model. In the antecedent part, the parameters (center-\( m \) and width-\( \sigma \)) are used to generate the membership functions of IT2FLS. As all parameters are chosen by hit and trial method, they have to be optimized by the gradient descent algorithm. The parameters in the antecedent part (both center-\( m \) and width-\( \sigma \)) are updated by the formula given below.

\[
m(t + 1) = m(t) - s\eta \frac{\partial E}{\partial m} \tag{2.28}
\]

\[
\sigma(t + 1) = \sigma(t) - s\eta \frac{\partial E}{\partial \sigma} \tag{2.29}
\]

where \( s\eta \) is the learning coefficient used with the membership functions. Again in FLANN, three categories of weights (with the data, indicators and bias) are used. The link weights used with data, indicators as well as bias are also to be optimized by the gradient descent algorithm. As the two local outputs are obtained from the consequent part of the model i.e. FLANN, two sets of weights are required for each input including indicators and bias. Weight updating for all (data, indicators and bias) are described as follows.

\[
w(t + 1) = w(t) + \Delta w(t) = w(t) + \left( -w\eta \frac{\partial E(t)}{\partial w(t)} \right) \tag{2.30}
\]

where \( w\eta \) is the learning rate used in FLANN and \( \frac{\partial E}{\partial w} \) for \( w11, w12 \) attached with the data are described in (2.31) and (2.32), respectively.
\[
\frac{\partial E}{\partial w_{11}} = e(i) \times (\frac{1}{2} \times A_{1(i)} \times B_{1(i)} \ldots \times G_{1(i)} \times \tanh(w_{110} + w_{11} \times x_{1} \ldots + w_{81} \times \text{sma}) + A_{2(i)} \times B_{2(i)} \ldots \times G_{2(i)}) + \frac{1}{2} \times (\frac{A_{1(i)} - B_{1(i)}}{\overline{A_{1(i)}} \times \overline{B_{1(i)}}} \ldots \times \overline{G_{1(i)}} \times \tanh(w_{220} + w_{12} \times x_{1} \ldots + w_{82} \times \text{sma}) / A_{1(i)} \times B_{1(i)} \ldots \times G_{1(i)} + A_{2(i)} \times B_{2(i)} \ldots \times G_{2(i)})
\]

\[
\frac{\partial E}{\partial w_{12}} = e(i) \times (\frac{1}{2} \times A_{1(i)} \times B_{1(i)} \ldots \times G_{1(i)} \times \tanh(w_{220} + w_{12} \times x_{1} \ldots + w_{82} \times \text{sma}) / A_{1(i)} \times B_{1(i)} \ldots \times G_{1(i)} + A_{2(i)} \times B_{2(i)} \ldots \times G_{2(i)}) + \frac{1}{2} \times (\frac{A_{1(i)} - B_{1(i)}}{\overline{A_{1(i)}} \times \overline{B_{1(i)}}} \ldots \times \overline{G_{1(i)}} \times \tanh(w_{220} + w_{12} \times x_{1} \ldots + w_{82} \times \text{sma}) / A_{1(i)} \times B_{1(i)} \ldots \times G_{1(i)} + A_{2(i)} \times B_{2(i)} \ldots \times G_{2(i)})
\]

\[
(2.31)
\]

Similarly, all other parameters used to develop the model are optimized using BP.

### 2.3.2 Backpropagation learning algorithm for LLWNN Model

The objective function for LLWNN model is
\[ E = \frac{1}{2} \left[ y(t) - w_{i0} \psi_i(x) - w_{i1} p_{i1} \psi_i(x) - \ldots - w_{i(n-1)} p_{i(n-1)} \psi_i(x) - w_{i(n)} p_{i(n)} \psi_i(x) \right]^2 \]  

(2.33)

where \( y(t) \) the desired value.

A weight update from iteration \( w(t) \) to \( w(t+1) \) is

\[
w(t + 1) = w(t) + \Delta w(t) = w(t) + \left( - w \eta \frac{\partial E(t)}{\partial w(t)} \right)
\]

(2.34)

where \( \frac{\partial E}{\partial w} \) for all weights are described in (2.35) to (2.38).

\[
\frac{\partial E}{\partial w_{10}} = w_{10} + w \eta * e^* \left( \frac{1}{2} \right) * \left( x_1^2 + x_2^2 + \ldots + x_n^2 \right) * \exp \left( - \left( (x_1 - c_1)^2 + (x_2 - c_1)^2 + \ldots + (x_n - c_1)^2 \right) \right) * x_2
\]

(2.35)

\[
\frac{\partial E}{\partial w_{12}} = w_{12} + w \eta * e^* \left( \frac{1}{2} \right) * \left( x_1^2 + x_2^2 + \ldots + x_n^2 \right) * \exp \left( - \left( (x_1 - c_1)^2 + (x_2 - c_2)^2 + \ldots + (x_n - c_1)^2 \right) \right) * x_2
\]

(2.36)

\[
\frac{\partial E}{\partial w_{20}} = w_{20} + w \eta * e^* \left( \frac{1}{2} \right) * \left( x_1^2 + x_2^2 + \ldots + x_n^2 \right) * \exp \left( - \left( (x_1 - c_2)^2 + (x_2 - c_2)^2 + \ldots + (x_n - c_2)^2 \right) \right) * x_1
\]

(2.37)

\[
\frac{\partial E}{\partial w_{21}} = w_{21} + w \eta * e^* \left( \frac{1}{2} \right) * \left( x_1^2 + x_2^2 + \ldots + x_n^2 \right) * \exp \left( - \left( (x_1 - c_2)^2 + (x_2 - c_2)^2 + \ldots + (x_n - c_2)^2 \right) \right) * x_1
\]

(2.38)

Similarly, other weights are also updated. \( w \eta \) is the learning rate used in LLWNN. Choosing an appropriate learning rate is a crucial task because if it is too low the network learns very slowly. Similarly, if it is too high, it may lead to no learning at all. Sometimes, it depends on the datasets also.
Since BP converges locally, if these initial weights are located on a local grade, the BP algorithm will likely to be trapped in local solution that may or may not be the global solution. To overcome this limitation, PSO has been used to optimize the model parameters.

### 2.3.2 Particle Swarm Optimization Learning Algorithm

Particle swarm optimization (PSO) is a population based, self adaptive search optimization technique first introduced by Kennedy and Eberhart in 1995 [45]. Similar to other population based optimization techniques like GA (Genetic Algorithm) [35, 38, 49, 55] the PSO starts with the random initialization of a population of particles in the search space. The PSO algorithm works on the social behavior of particles in the swarm and finds the global best solution by simply adjusting the trajectory of each individual towards its best location and towards the best particle of the entire population at each generation [46, 57]. The PSO method is becoming very popular due to its simplicity in implementation and its ability to quickly converge to an optimal solution. For a particle moving in a multidimensional search space let $S_i$ denotes the position of the $i^{th}$ particle and $v_i$ denotes the velocity at the sampling instant $k$, and the dimension $j$ denotes the number of parameters to be optimized which if too high will result in allowing the particles to fly past good solutions. On the other hand, if $v_{\text{max}}$ is too small, particles end up in local solutions only.

The modified velocity and position of each particle at the sampling instant ($k+1$) can be calculated as
\[ v_i(k+1) = \omega_i v_i(k) + c_1 \text{rand}( ) + (pbest_i - s_i(k)) + c_2 \text{rand}( )(gbest_i - s_i(k)) \]  

(2.39)

\[ S_i(k+1) = S_i(k) + v_i(k+1) \]  

(2.40)

when \( c_1 \) and \( c_2 \) are constants known as acceleration co-efficient, \( \text{rand}( ) \) is a uniformly distributed random number in the range \((0, 1)\), and \( \omega_i \) is the inertia weight for the \( i^{th} \) particle.

A suitable selection of inertia weight \( \omega \) and constants \( c_1 \) and \( c_2 \) is crucial in providing a balance between the global and local search in the flying space. The particle velocity at any instant is limited to \( v_{\text{max}} \).

As with GA, PSO has premature convergence and thus results in an inaccurate solution. To circumvent this problem, an alternative approach is used to dynamically vary the inertia weight based on the variance of the population fitness. This results in better local and global searching ability of the particles which improves the convergence velocity and better accuracy for stock market indices prediction.

Since the PSO algorithm depends on the fitness function to guide the search, the choice is made as

\[ F_i = \frac{1}{NS} \sum_{k=1}^{NS} (\hat{y}(t) - y_d(t))^2 \]  

(2.41)

\( NS \) = number of samples over which the RMSE is compared

\( F_i \) = fitness for the \( i^{th} \) particle.

The inertia weight \( \omega_i \) is updated by finding the variance of the population fitness as
\[ \sigma = \sqrt{\frac{M}{\sum_{i=1}^{M} \left( \frac{F_i - F_{\text{avg}}}{F} \right)^2}} \]  

(2.42)

\[ F_{\text{avg}} = \text{average fitness of the population} \]

\[ F_i = \text{fitness of the } i^{\text{th}} \text{ particle in the population, and} \]

\[ M = \text{total number of particles} \]

\[ F = \max \| F_i - F_{\text{avg}} \|, i = 1, 2, \ldots, M \]

\[ = 1, \quad \text{if } \max \| F_i - F_{\text{avg}} \| > 1 \]

\[ = \frac{1}{\max \| F_i - F_{\text{avg}} \|} \quad \text{if } \max \| F_i - F_{\text{avg}} \| < 1 \]  

(2.43)

If \( \sigma \) is found to be large, the population will be a random searching mode, while for small \( \sigma \) the solution tends towards a premature convergence and will give the local best position of the particles. To circumvent this problem the inertia weight is changed in the following way:

\[ \omega(k) = \lambda \omega(k-1) + (1 - \lambda) \sigma \]  

(2.44)

The forgetting factor \( \lambda \) is chosen as 0.9 for faster convergence.

Another alternative will be

\[ \omega(k) = \lambda \omega(k-1) + \text{rand}(0, 1)/2 \]  

(2.45)

with \( 0 \leq \lambda \leq 0.5 \)

where \( \text{rand}(0, 1) \) is a random number between (0, 1). Besides, the influence of the past velocity of a particle on the current velocity is chosen to be random and the inertia weight is adapted randomly depending on the variance of the fitness value of a
population. This result is an optimal coordination of local and global searching abilities of the particles. The flow chart of PSO is as under:

Fig. 2.6 Flow chart of PSO
2.4 Stock Market Datasets

2.4.1 Standard’s & Poor’s 500 (S&P 500)

The S&P 500 is an index of the prices of 500 large publicly held companies. It has been publishing stock market indices since 1957. It introduced its first stock indices in 1923. Before 1957, its primary daily stock market index was the "S&P 90", a value-weighted index based on 90 stocks. Standard & Poor's also published a weekly index of 423 companies. The S&P 500 index in its present form began on March 4, 1957.

2.4.2 Bombay Stock Exchange (BSE)

The BSE is the oldest stock exchange in Asia and on August 2007 as many as 4700 companies were listed in the Exchange which is the largest stock exchange in the world [99]. The equity market capitalization of the companies listed on the BSE was US$1.63 trillion as of December 2010, making it the 4th largest stock exchange in Asia and the 8th largest in the world. The BSE has the largest number of listed companies in the world [99].

2.4.3 Dow Jones Industrial Average (DJIA)

The DJIA known as “the barometer of stock market” began publishing the composite list of stocks of major companies in the year 1984. In 1997, it appeared in Wall Street Journal. In the beginning, it averaged the stocks of 12 companies with the idea of giving a picture of the trend in the stock market which can help in the forecasting.
2.5 Data Series

The following four time series were used as raw input to forecast the stock market indices of the above stock markets.

- Closing price (price of the last fulfilled trade during the day);
- Highest price paid during the day;
- Lowest price paid during the day;
- Volume (total amount of traded stocks during the day).

2.6 Input Selection

The experimental data obtained from three significant stock markets i.e. S&P 500, BSE and DJIA comprise daily closing prices. The data are available in the web site www.forecasts.com. All the models are used to predict the S&P 500, BSE and DJIA stock indices one day, one week and one month in advance. For all models, inputs are scaled between 0 and +1 using the standard formula described below.

\[
\tilde{X}^i = \frac{X^i - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}}
\]

where \(X^i\) is the current day closing price, \(\tilde{X}^i\) is the scaled price and \(X_{\text{min}}\) and \(X_{\text{max}}\) are the minimum and maximum of the dataset respectively. Scaling is not mandatory; but it will help to improve the performance. The normalized value of all the datasets is shown in Fig. 2.7. All the datasets are divided into two parts, one for training and another for testing. The total number of samples and the samples used for training and testing are given in Table 2.1. Five important technical indicators are taken into
consideration along with the daily closing prices of the respective stocks. The technical indicators that have been used and the formula are given in Table 2.2. The Mean Absolute Percentage Error (MAPE) and Root Mean Square Error (RMSE) are used to measure the performance of the trained models. The MAPE and RMSE are defined as

\[
\text{MAPE} = \frac{1}{N} \sum_{i=1}^{N} \left| \frac{y_i - \hat{y}_i}{y_i} \right| \times 100
\]

(2.47)

\[
\text{RMSE} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2}
\]

(2.48)

where \(y_i\) is desired value and \(\hat{y}_i\) is the predicted value. \(N\) is the total number of test data.

Table 2.1 Total No. of Samples used for all models

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Number of Samples</th>
<th>Data range</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Total</td>
<td>Training</td>
</tr>
<tr>
<td>S &amp; P 500</td>
<td>3228</td>
<td>2000</td>
</tr>
<tr>
<td>BSE</td>
<td>4000</td>
<td>2000</td>
</tr>
<tr>
<td>DJIA</td>
<td>2301</td>
<td>1000</td>
</tr>
</tbody>
</table>
Fig. 2.7 Normalized Stock Markets indices

Table 2.2 Technical Indicators with Formula

<table>
<thead>
<tr>
<th>Indicators</th>
<th>Formulae</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Moving Average</td>
<td>$\frac{1}{N} \sum_{i=1}^{N} x_i$, N=no. of Days, $x_i=$today’s price;</td>
</tr>
<tr>
<td>2. Exponential Moving Average</td>
<td>$(P \times A) + (\text{Previous EMA} \times (1-A))$; A=$\frac{2}{N+1}$;</td>
</tr>
<tr>
<td></td>
<td>P=Current Price; A=Smoothing factor; N=Time period;</td>
</tr>
<tr>
<td>3. High Price Acceleration (HP Acc)</td>
<td>(High Price-High Price N-Periods ago)/(High Price N-Periods ago)x100</td>
</tr>
<tr>
<td>4. Closing Price Acceleration (CP Acc)</td>
<td>(Close Price-Close Price N-Periods ago)/(Close Price N-Periods ago)x100</td>
</tr>
<tr>
<td>5. Accumulation/Distribution Oscillator (ADO)</td>
<td>$\frac{(\text{C.P}-\text{L.P})-(\text{H.P}-\text{C.P})}{(\text{H.P}-\text{L.P}) \times \text{Period’s volume}}$;</td>
</tr>
<tr>
<td></td>
<td>C.P-Closing price; H.P-Highest price; L.P-lowest Price;</td>
</tr>
</tbody>
</table>
2.7 Stopping Criteria

There are mainly two paradigms: (a) late stopping; and (b) early stopping to determine the number of iterations for training the network.

**Late stopping** means that the network is trained until a minimum error on the training set is reached. This process is also called pruning. By doing so, eventually, good generalization ability is reached. The main problem of late stopping is that the network gets overfitted. To overcome it, different techniques are used to exterminate nodes in the network.

**Early stopping** is a way of avoiding overfitting. During learning, the progression is monitored and training is terminated as soon as signs of overfitting appear. A clear advantage with early stopping is that the time of training is relatively short. On the downside, it is hard to know when to stop. The early stopping method is used to train all the three models used to forecast stock market indices to overcome the problem of overfitting.

2.8 Experimental results and Model Outputs

The experiment has taken into account three models, three datasets and three time horizons. The empirical results have been presented in terms of target vs. predicted values and error convergence speed. A comparative analysis across models and time horizons has also been presented.

Figs. 2.8 to 2.10 show the comparison of target vs. predicted during testing of S&P 500 for one day, one week and one month in advance respectively using all three models with BP and PSO learning algorithms separately. The number of weights to be
optimized by PSO is 74 for FLIT2FNS model, whereas 39 for FLANN and Type-1FLS and 56 for LLWNN. The parameters used for all three models (FLIT2FNS, FLANN and Type-1FLS, and LLWNN) with PSO as a learning algorithm are given in Table 2.3. For evaluating the fitness function 450 observations are taken (NS=450) in equation (2.41). The optimal values for the weights are obtained by finding the global best position of the particles which represent the weights of the neural models.

Fig. 2.11 demonstrates the error convergence speed of S&P 500 dataset using FLIT2FNS with PSO as learning algorithm for all three time horizons considered in this study. Table 2.4 shows the MAPE during testing for FLIT2FNS model with BP and PSO as learning algorithms separately with all three datasets. Similarly Figs. 2.12, 2.13 and 2.14 illustrate the comparison of target vs. predicted during testing of BSE for one day, one week and one month in advance, respectively with BP and PSO separately. Fig. 2.15 depicts the error convergence speed of BSE for FLIT2FNS model with PSO for all three time horizons. Fig. 2.16 shows the error convergence speed of DJIA for FLIT2FNS model for the same three time horizons. Fig. 2.17 shows the error convergence speed of all three models with PSO learning algorithm using S&P500 stock indices for one day in advance. Also Fig. 2.18 presents a bar graph, showing the comparative picture of all three models with BP and PSO technique separately with all time horizons for one dataset i.e. S&P 500. In Tables 2.5 and 2.6, MAPEs for all the datasets obtained from other two models (FLANN and Type-1FLS, and LLWNN) are given for all the three time horizons.
Fig. 2.8 One Day ahead Prediction (S&P 500)

Fig. 2.9 One Week ahead Prediction (S&P500)
Fig. 2.10 One Month ahead Prediction (S&P500)

Table 2.3 Parameters used for all three models (PSO)

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population size P</td>
<td>30</td>
</tr>
<tr>
<td>Maximum No. of generation G</td>
<td>200</td>
</tr>
<tr>
<td>Dimension or No. of variables D (FLIT2FNS)</td>
<td>74</td>
</tr>
<tr>
<td>Dimension or No. of variables D (FLANN&amp;Type-1FLS)</td>
<td>39</td>
</tr>
<tr>
<td>Dimension or No. of variables D (LLWNN)</td>
<td>56</td>
</tr>
<tr>
<td>No. of consecutive generation for which no improvement is observed (FLIT2FNS Model)</td>
<td>15</td>
</tr>
<tr>
<td>Acceleration co-efficient $C_1, C_2$</td>
<td>2.1,2.1</td>
</tr>
<tr>
<td>Lower bound</td>
<td>0.0001</td>
</tr>
<tr>
<td>Upper bound</td>
<td>1</td>
</tr>
</tbody>
</table>
Fig. 2.11 Convergence speed of S&P500 (FLIT2FNS with PSO)

Table 2.4 FLIT2FNS Model (D, W, M for Day, Week, Month respectively)

<table>
<thead>
<tr>
<th>Datasets</th>
<th>MAPE (%)</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PSO</td>
<td>BP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Time horizon</td>
<td>Time horizon</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1D</td>
<td>1W</td>
<td>1M</td>
<td>1D</td>
<td>1W</td>
<td>1M</td>
</tr>
<tr>
<td>S&amp;P500</td>
<td>0.31</td>
<td>0.41</td>
<td>0.62</td>
<td>1.0</td>
<td>1.1</td>
<td>2.0</td>
</tr>
<tr>
<td>BSE</td>
<td>0.23</td>
<td>0.43</td>
<td>0.61</td>
<td>1.3</td>
<td>1.39</td>
<td>2.1</td>
</tr>
<tr>
<td>DJIA</td>
<td>0.44</td>
<td>0.5</td>
<td>0.65</td>
<td>1.9</td>
<td>1.95</td>
<td>2.3</td>
</tr>
</tbody>
</table>
Fig. 2.12 One Day ahead Prediction (BSE)

Fig. 2.13 One Week ahead Prediction (BSE)
Fig. 2.14 One Month ahead Prediction (BSE)

Fig. 2.15 Convergence speed of BSE (FLIT2FNS with PSO)
Fig. 2.16 Convergence speed of DJIA (FLIT2FNS with PSO)

Fig. 2.17 Convergence speed of Three Models (S&P500)
Fig. 2.18 Comparison of Forecasting Errors for FLIT2FNS, FLANN&Type-1FLS and LLWNN Models (S&P500), 1D, 1W and 1M represent 1 Day, 1 Week and 1 Month respectively

Table 2.5 FLANN&Type-1FLS Model

<table>
<thead>
<tr>
<th>Datasets</th>
<th>MAPE (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PSO</td>
</tr>
<tr>
<td></td>
<td>Time horizon</td>
</tr>
<tr>
<td></td>
<td>1D</td>
</tr>
<tr>
<td>S&amp;P500</td>
<td>1.0</td>
</tr>
<tr>
<td>BSE</td>
<td>1.0</td>
</tr>
<tr>
<td>DJIA</td>
<td>1.4</td>
</tr>
</tbody>
</table>
Table 2.6 LLWNN Model

<table>
<thead>
<tr>
<th>Datasets</th>
<th>MAPE (%)</th>
<th>MAPE (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PSO</td>
<td>BP</td>
</tr>
<tr>
<td></td>
<td>1D</td>
<td>1W</td>
</tr>
<tr>
<td>S&amp;P500</td>
<td>1.3</td>
<td>1.35</td>
</tr>
<tr>
<td>BSE</td>
<td>1.3</td>
<td>1.4</td>
</tr>
<tr>
<td>DJIA</td>
<td>1.5</td>
<td>1.53</td>
</tr>
</tbody>
</table>

2.9 Results Analysis

Fig. 2.8 clearly shows that the actual and the predicted output of FLIT2FNS model integrated with PSO almost merge into same line, whereas, gaps are visible in case of other two models. This indicates the superiority of FLIT2FNS model vis-à-vis other two models.

In case of one day ahead prediction, when FLIT2FNS model is combined with BP during testing, the average MAPE for all the datasets stands at 1.4%, but with PSO it comes down to 0.32% with all other factors remaining unchanged. Similarly, for one week prediction, the average MAPE turns out to be 1.5% and 0.45%, and for one month prediction 2.1% and 0.62%, respectively, when FLIT2FNS is combined with BP and PSO.

From Tables 5 and 6, it can be seen that MAPE value is higher for other two models vis-à-vis the proposed FLIT2FNS model. The average RMSE, the second error measure taken in this study, during testing obtained from FLIT2FNS model integrated with PSO for all datasets is 0.000619 in case of one day ahead prediction,
whereas it is 0.002443 for one month ahead prediction. Similarly it is 0.005438 and 0.014538 for FLANN and Type-1FLS model for one day and one month ahead respectively and 0.015863 and 0.019893 for LLWNN model for the same time horizon.

To sum up the results it can be said:

- When the period is shorter, the prediction is more accurate and it holds well irrespective of the models used;
- Unlike BP, the stochastic PSO learning algorithm does not get trapped in local minima. Hence the degree of prediction accuracy is more than that of BP;
- When the time horizon is shorter, convergence speed is faster irrespective of datasets and models;
- The convergence speed for any dataset is fastest in case of FLIT2FNS model, followed by FLANN and Type-1FLS, and LLWNN model respectively;
- FLIT2FNS proves to be the best in terms of both prediction accuracy and error convergence speed amongst the three models and is more capable to handle more uncertainties.

2.10 Summary

An integrated model comprising functional link artificial neural network and IT2FLS has been used to predict three stock market indices, S&P 500, BSE, and DJIA for one day, one week, and one month in advance. From the experiment, it is clearly established that IT2FLS helps to deal with the uncertainties present in the stock market in an efficient manner. Other two models, FLANN and Type-1FLS, and LLWNN are discussed and implemented for a comparison purpose. Backpropagation learning algorithm is used to train the models. Further, to get more accuracy in
prediction and to avoid local minima problem PSO is incorporated with all the models. The simulation results show that FLIT2FNS model performs better than that of FLANN and Type-1FLS followed by LLWNN model irrespective of the learning algorithms it is combined with or irrespective of the periodicity of the prediction. The average MAPE obtained from FLIT2FNS-PSO model is 0.32% for one day ahead prediction for all three datasets whereas, it is 1.1% and 1.4% for FLANN and Type-1FLS-PSO, and LLWNN-PSO models respectively.