Forecasting the Stock Index of SBI Using Wave-Nets

Thesis submitted in partial fulfilment of the requirement for

the award of the degree of

Masters of Science

in

Mathematics and Computing

Submitted by

Shivi Bansal

Roll No. – 300803018

Under

the guidance of

Dr. R. K. Sharma

JULY 2010

School of Mathematics and Computer Applications

Thapar University

Patiala – 147004(PUNJAB)

INDIA

CONTENTS

CERTIFICATE

I hereby certify that the work which is being presented in the thesis entitled "Forecasting the Stock Index of SBI Using Wave-Nets" in partial fulfilment of the requirements for the award of degree of Master of Mathematics and Computing, School of Mathematics and Computer Applications, Thapar University, Patiala is an authentic record of my own work carried out under the supervision of Dr. R. K. Sharma. The matter presented in this thesis has not been submitted for the award of any other degree of this or any other university.

(Shivi Bansal)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

(Dr. R. K. Sharma)

Dean of Academic Affairs

Thapar University

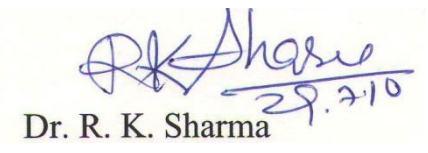
Patiala

Countersigned by:

(Professor and Head)

SMCA, Thapar University,

Patiala



(Dean of Academic Affairs)

Thapar University

Patiala

ACKNOWLEDGEMENT

A journey is easier when travelled together. Interdependence is certainly more valuable than Independence. The completion of this thesis has involved a lot of people to whom I would like to express my sincere thanks and gratitude for their help.

No amount of words can adequately express the debt, I owe to Dr. R. K. Sharma, Dean of Academic Affairs, Thapar University, Patiala, for his kind support, motivation and inspiration that triggered me for the thesis work. I owe him lots of gratitude for having me shown this way of research. I am fortunate that I got an opportunity to work under his supervision.

I express my regards and gratitude to Dr. S. S. Bhatia, Head of Department School of Mathematics and Computer Applications, Thapar University, Patiala for providing keen interest, unfailing support, inspiration and necessary research facilities in the school.

I would like to thank my beloved parents, my in-laws and my husband for their unconditional support and deep interest in me, without whom my project would have been a mere dream rather than a reality.

I would also thank all the academic and administrative staff of School of Mathematics and Computer Applications, Thapar University, Patiala.

Finally, I am thankful to all my friends who also contributed a lot in accomplishing this peace work.

(Shivi Bansal)

Abstract

This thesis presents a new network to forecast the daily stock value of SBI using the technique of the adaptive wavelet-neural-network. This has been implemented by writing programs in MATLAB. The learning algorithm of this network is optimal on rate of convergence and allows tuning the synaptic weights, dilations and translations parameters of wavelet activation functions. Different types of wavelets can be used as the activation function. The simulation of developed wavelet-neural network architecture and its online learning algorithm justifies the effectiveness of the approach. The algorithm which has been implemented in MATLAB is used by varying different parameters. Thus by using different parameters we have generated a network which is very effective in forecasting of the stock value of the SBI.

List of figures

Chapter – 1

Introduction

At present time the wave-net based techniques are becoming increasingly popular soft computing techniques and are being successfully applied for the processing of information containing complex nonlinear regularities and distortions of all kinds. These systems combine the dilation and translation properties of the wavelet systems with the online learning and universal approximation capabilities of artificial neural networks. This means that they can be used in forecasting and emulation of the stochastic and chaotic signals and sequences with complex nonlinear trends and non-stationary parameters.

The wavelet transform has been an increasingly popular technique which provides a compact local signal representation in both time and frequency domain. At the turn of the artificial neural network and wavelets theories the wavelet neural networks have evolved for the analysis of non stationary processes with considerably nonlinear trends.

The main point in defining effectiveness of such systems is the choice of learning algorithm, which is usually based on the gradient procedures of the accepted criterion minimization. Combination of the gradient optimization with the error back propagation essentially reduces the rate of learning hybrid systems and leads to necessity of using rather large training samples. In the case when the data processing has to be carried out in real time, forecasted or emulated sequence is non stationary and distorted, conventional gradient descent learning algorithms appeared to be ineffective.

This work takes the Adaptive Wavelet Neural Network (AWNN) as a tool for the forecasting of stock value of the SBI and focuses on using different parameters during the training of AWNN. The results of simulation show that the given network is a good choice for the forecasting of the stock index of SBI.

Chapter 2

Artificial Neural Network

Neural networks are the simplified models of the biological nervous system and have been motivated by the kind of computing performed by a human being. In general, it is a highly interconnected network of a large number of processing elements called neurons in an architecture inspired by the brain. An artificial neural network can be greatly parallel and therefore is said to have parallel distributed processing. These properties make these models to be used in a variety of applications. We give a formal definition below.

Definition: A neural network is a mathematical model or computational model that tries to model the structure and functional aspects of biological neural network. It consists of interconnected group of artificial neurons and process information using a connectionist approach to computation. The objective of the neural network model is to transform the inputs into meaningful outputs.

2.1 The Biological Model

Artificial neural networks emerged after the introduction of simplified neurons by McCulloch and Pitts in 1943 (McCulloch and Pitts, 1943). These neurons were presented as models of biological neurons and as conceptual components for circuits that could perform computational tasks. The basic model of the neuron is motivated by the functionality of a biological neuron. "Neurons are the basic signalling units of the nervous system" and "each neuron is a discrete cell whose several processes arise from its cell body". The neuron has four main regions in its structure. The cell body, or soma, has two offshoots from it, the dendrites, and the axon, which end in presynaptic terminals. This model is given in Fig. 2.1.

The cell body is the heart of the cell, containing the nucleus and maintaining protein synthesis. A neuron may have many dendrites, which branch out in a treelike structure, and receive signals from other neurons. A neuron usually has one axon only which grows out from a part of the cell body called the axon hillock. The axon conducts electric signals generated at the axon hillock down its length. These electric signals are called action potentials. The other end of the axon may split into several branches than end in a presynaptic terminal. Action potentials are the electric signals that neurons use to convey information to

the brain. All these signals are identical. Therefore, the brain determines what type of information is being received based on the path that the signal took. The brain analyses the patterns of signals being sent and from that information it can interpret the type of

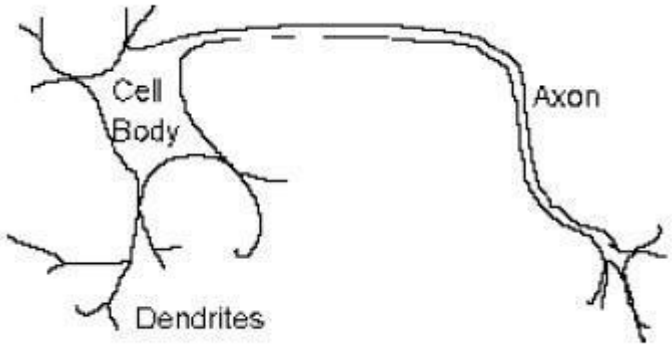


Figure 2.1: Biological neuron network

information being received. Synapse is the area of contact between two neurons. The neurons do not actually physically touch. They are separated by the synaptic cleft, and electric signals are sent through chemical interactions. The neuron sending the signal is called the presynaptic cell and the neuron receiving the signal is called the postsynaptic cell. The signals are generated by the membrane potential, which is based on the differences in concentration of sodium and potassium ions inside and outside the cell membrane.

Neurons can be classified by their number of processes (or appendages), or by their function [9]. If they are classified by the number of processes, they fall into three categories, namely, unipolar, bipolar and multipolar neurons. Unipolar neurons have a single process (dendrites and axon are located on the same stem), and are most common in invertebrates. In bipolar neurons, the dendrite and axon are the neuron's two separate processes. Bipolar neurons have a subclass called pseudo-bipolar neurons, which are used to send sensory information to the spinal cord. Finally, multipolar neurons are most common in mammals. Examples of these neurons are spinal motor neurons, pyramidal cells and Purkinje cells.

Based on their functioning, neurons can again be classified into three groups. The first group is sensory, or afferent neurons that provides information for perception and motor coordination. The second group provides information to muscles and glands and is therefore called motor neurons. The last group, interneuronal, contains all other neurons and has two subclasses. The first sub class called relay or projection interneurons has long axons and

connect different parts of the brain. The other sub class called local interneurons are only used in local circuits only.

2.2 The Mathematical Model

The behaviour of a neuron can be captured by a simple model as shown below. Every component of the model bears a direct analogy to the actual constituents of a biological neuron and hence is termed as artificial neuron.

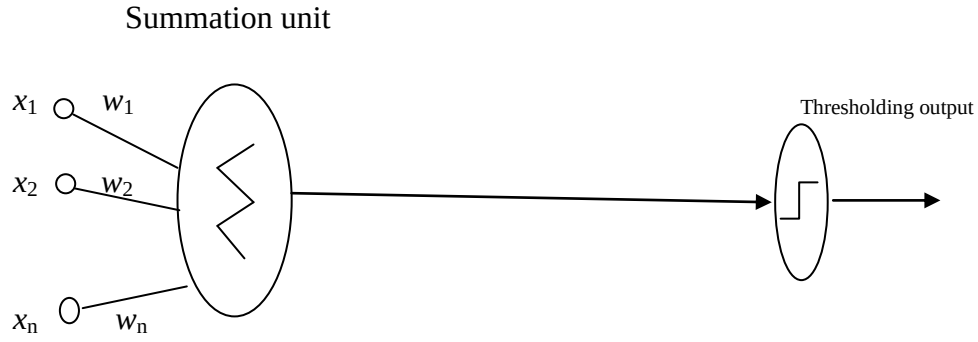


Figure 2.2: Mathematical neuron network

As illustrated in Fig. 2.2, x_1, x_2, x_3, …, x_n are n inputs to the network model and w_1, w_2, …, w_n are the weights attached to the input links.

One can note that a biological neuron receives all inputs through the dendrites, sum them and produces an output if the sum is greater than a threshold value. The input signals are passed on to the cell body through the synapse which may accelerate or retard an arriving signal. It is this acceleration or retardation of the input signals that is modelled by the weights. An effective synapse which transmits a stronger signal will have smaller weights. Thus, weights here are multiplicative factors of the inputs to account for the strength of the synapse. Total input received by the soma of artificial neuron is given by,

$$I = w_1 x_1 + w_2 x_2 + \ldots + w_n x_n$$

$$= \sum_{i=1}^{n} w_i x_i$$

2.3 Activation function

As mentioned previously, the activation function acts as a squashing function, such that the output of a neuron in a neural network is between certain values (usually 0 and 1, or -1 and 1). In general, there are three types of activation functions. Let us denote this by $\Phi(.)$.

First type of threshold functions take a value of 0 if the summed input is less than a certain threshold value (v), and the value 1 if the summed input is greater than or equal to the threshold value. This function can be defined as below.

$$\Phi(v) = 1 \quad if\ v \geq 0$$
$$0 \quad if\ v < 0$$

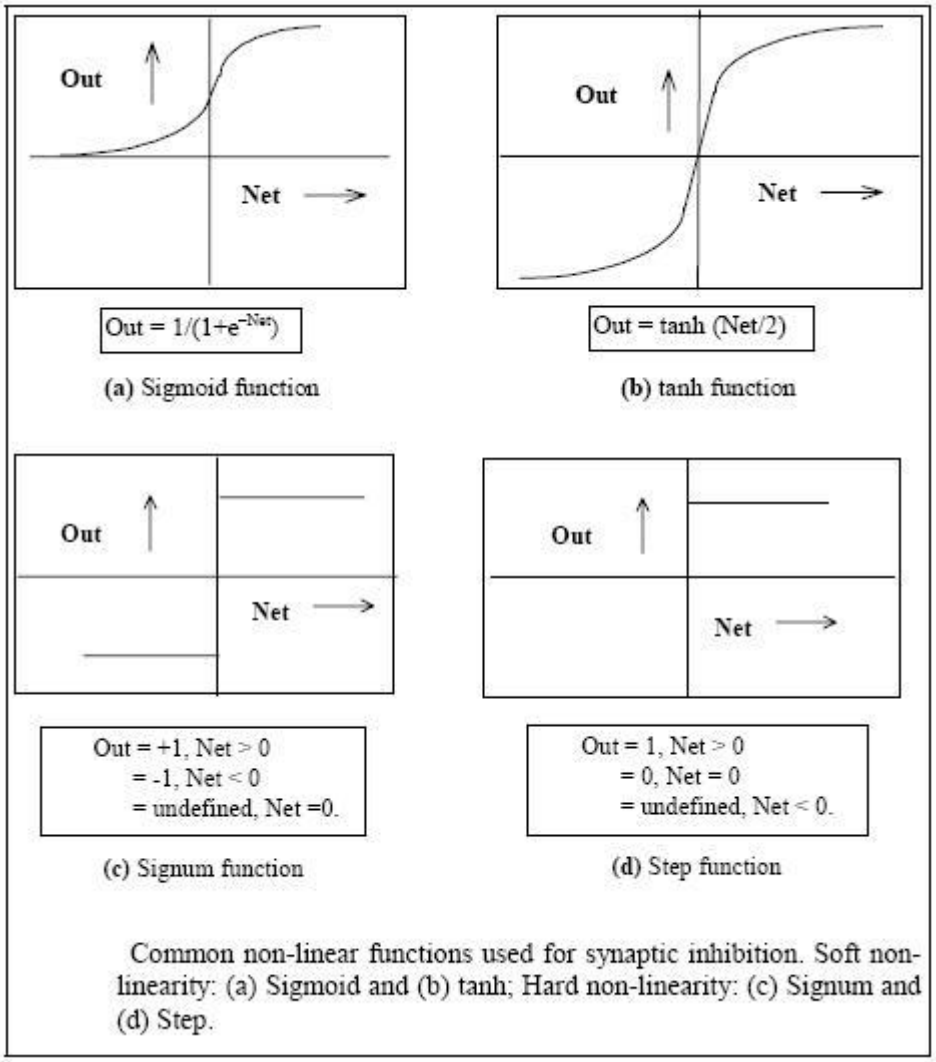


Figure 2.3: Different activation functions

Second type of activation functions is linear functions. This function again can take on the values of 0 or 1, but can also take on values between that depending on the amplification factor in a certain region of linear operation. This function can be defined as below.

$$\Phi(v) = 1 \quad if\ v \geq \tfrac{1}{2}$$
$$v \quad if\ -\tfrac{1}{2} > v > \tfrac{1}{2}$$
$$0 \quad if\ v \leq -\tfrac{1}{2}$$

Third kind of functions is called sigmoid functions. This function can range between 0 and 1, but it is also sometimes useful to use the -1 to 1 range, as shown in Fig. 2.3. An example of the sigmoid function is the hyperbolic tangent function. This function is given by,

$$\Phi(v) = tanh(v/2) = \frac{1-exp(-v)}{1+exp(-v)}$$

The artificial neural networks as described here are having the fundamental principle of Parallel and Distributed Processing behind them. The architecture of each neural network is based on very similar building blocks which perform the processing [11].

2.4 Learning Methods

Learning methods in neural networks can broadly be classified into three basic types as given below.

1) Supervised Learning
2) Unsupervised Learning
3) Reinforced Learning

2.4.1 Supervised Learning

In this type of learning, every input pattern that is used to train the network is associated with an output pattern, which is the target or the desired pattern. This is the kind of situation where a teacher is assumed to be present during the learning process, when a comparison is made between the networks computed output and the correct expected output, to determine the error. The error can then be used to change network parameters, which results in an improvement in performance.

2.4.2 Unsupervised Learning

In this type of learning, the target output is not present to the network. It is as if there is no teacher to present the desired patterns and hence the system learns on its own by discovering and adapting to structural features in the input patterns.

2.4.3 Reinforced Learning

In this type of learning method, a teacher though available, does not present the expected answer but only indicate if the computed output is correct or incorrect. The information

provided helps the network in its learning process. A reward is given for a correct answer computed and a penalty for a wrong answer. Reinforced learning method is not one of popular forms of learning.

2.5 ANN Architecture

The ANN Architecture has broadly been classified into the following categories.

1) Single layer feed forward network
2) Multi layer feed forward network
3) Recurrent network

There are other well-known neural networks. These are systems back propagation network, perceptron, ADALINE associative memory, Boltzmann machine, adaptive resonance theory, self organizing feature map and Hopfield network.

Generally, a neural network structure can be represented using a directed graph. The vertices of graph may represent neurons and the edges, the synaptic links. The edges are labelled by the weights attached to the synaptic links, as shown in Fig. 2.4.

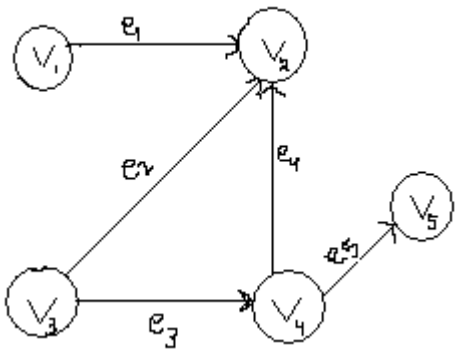


Figure 2.4: Representation of ANN using directed graph

Vertices are $V = \{v_1, v_2, v_3, v_4, v_5\}$

Edges are $E = \{e_1, e_2, e_3, e_4, e_5\}$

2.5.1 Single layer feed forward network

This type of network comprises of two layers, namely, the input layer and the output layer as shown in Fig. 2.5. The input layer neurons receive the input signals and the output layer neurons receive the output signal. The synaptic links carrying the weights connect every input

neuron to the output neuron but not vice-versa. Such a network is said to be feed forward in type or acyclic in nature. Despite the two layers, the network is termed single layer since it is the output layer, alone which perform computations. The input layer merely transmits the signals to the output layer. Hence, the name single layer feed forward network.

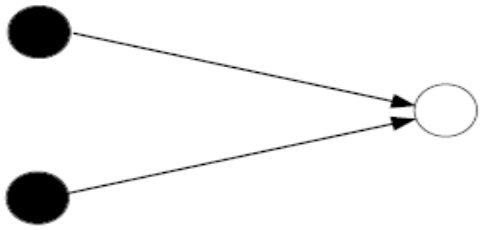


Figure 2.5: Single layer feed forward network

2.5.2 Multi layer feed forward network

Architecture of this class besides possessing an input and an output layer also has one or more intermediary layers called hidden layers, as shown in Fig. 2.6. The computational units of the hidden layer are known as the hidden neurons or hidden units. The hidden layer aids in performing useful intermediary computation before directing the input to the output layer. The input layer neurons are linked to the hidden layer neurons and the weights on these links are referred to as input hidden layer weights. Again, the hidden layer neurons are linked to the output layer neuron and the corresponding weights are referred to as hidden output layer neuron.

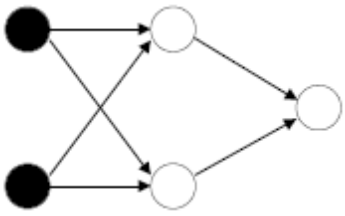


Figure 2.6: Multi layer feed forward network

2.5.3 Recurrent network

In this type of network, there is at least one feedback loop, as shown in Fig. 2.7. Here, we can have one layer with feedback connections and there could also be neurons with self- feedback link.

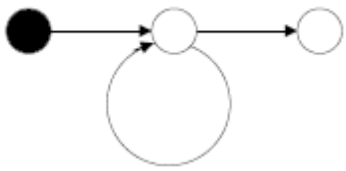


Figure 2.7: Recurrent network

2.6 ADALINE Network

The Adaptive Linear Neural Element Network was framed by Bernard Widrow and his student Marcian (Ted) Hoff (Widrow and Hoff 1960). There is only one output neuron and the output values are bipolar (-1 to 1). However input x_i could be binary, bipolar or real valued. If the weighted sum of the inputs is greater than or equal to 0 then the output is 1 otherwise it is -1. The supervised learning algorithm adopted by this network is similar to the perceptron learning algorithm, the learning algorithm is also known as Least Mean Square or Delta rule. The rule is given by

$$W_i^{new} = W_i^{old} + \alpha\ (t\text{-}y)x_i$$

where, α is the learning coefficient, t is the target output, y is the computed output and x_i is the input. This network has a wide range of applications because it is used virtually in all high speed modems and telephone switching system to cancel the echo in long distance communication circuit.

2.7 MADALINE Network

A MADALINE (Many ADALINE) network is created by combining a number of ADALINE. This was framed by Windrow and Hoff, 1960; and by Windrow and Lehr, 1990. The use of multiple ADALINE helps counter the problem of non-linear separability. The learning rule adopted by MADALINE network is termed as ‘MADALINE’ Adaptation Rule

(MR) based on ‘minimal disturbance principle’, and is a form of supervised learning. In this method the object is to adjust the weights such that the error is minimum for the current training pattern, with as little damage to the learning acquired through previous training patterns.

Chapter 3

Fundamentals of Wavelets

The term 'wavelet' as it implies means a little wave. A wavelet is a wave-like oscillation with amplitude that starts out at zero, increases, and then decreases back to zero. We can describe wavelets as the brief oscillations. The wavelets have the specific properties that make them useful for the signal processing. Technically, a wavelet is a mathematical function used to divide a given function or continuous-time signal into different scale components. In wavelets, the scale we use to look at the data plays an important role. Wavelets process data at different scales in order to study the signal properly. Wavelet transforms have advantages over traditional Fourier transforms for representing functions that have discontinuities and sharp peaks, and also for deconstructing and reconstructing finite, non-periodic and non-stationary signals. Wavelet transforms are classified into Discrete Wavelet Transforms (DWTs) and Continuous Wavelet Transforms (CWTs).

3.1 Wavelet Theory

In this thesis we will mainly deal with Continuous Wavelet Transforms (CWTs). Let *g(t)* be any square integrable function. Let us denote the CWT of *g(t)* with respect to a wavelet *ψ(t)* by *W(a, b).* It is defined as

$$W(a, b) = \int_{-\infty}^{\infty} g(t) \frac{1}{\sqrt{|a|}} \overline{\psi\left(\frac{t-b}{a}\right)}\, dt \qquad \ldots(3.1)$$

where *a* and *b* are real. Thus, the wavelet transform is a function of two variables. Both the functions *g*(*t*) and *ψ(t)* belongs to $L^2(R)$, (the set of square integrable function). We can write the equation (3.1) in a compact form by defining $\psi_{a,b}(t)$ as

$$\psi_{a,b}(t)=\frac{1}{\sqrt{|a|}}\, \psi\left(\frac{t-b}{a}\right) \qquad \ldots(3.2)$$

Substituting the value of $\psi_{a,b}(t)$ we can write the equation (3.1) in the form

$$W(a, b) = \int_{-\infty}^{\infty} g(t)\overline{\psi_{a,b}(t)}\, dt \qquad \ldots(3.3)$$

$\frac{1}{\sqrt{|a|}}$ is the normalizing factors which ensures that the energy stays the same for all *a* and *b*. The function value $\psi_{a,b}(t)$ is a shift of $\psi_{a,0}(t)$ by an amount *b* along the time axis. Thus, the variable *b* represents time shift or translation. Also we observe that *a* determines the amount of time scaling or dilation. Fig. 3.1 explains the wavelets with different value of dilation. In Fig. 3.1 we have shown a wavelet with $a = 1/2$, $a = 1$, $a = 3/2$, $a = 2$ respectively. We can generate many daughter wavelets from the main wavelet by changing the value of dilation and translation.

Figure 3.1: Wavelets with different dilation values

3.2 Types of different mother wavelets

3.2.1 Morlet wavelet

The Morlet's basic wavelet function is a multiplication of the Fourier basis with Gaussian window

$$h(t)= exp(iw_o t)exp(-0.5t^2)$$

Its real part is a Cos-Gaussian and the imaginary part is a Sin-Gaussian function. The Cos-Gaussian is a real even function.

3.2.2 SLOG wavelet

The ***S***uperposed ***LOG***istic functions (SLOG) results from finite term sums of weighted and delayed logistic functions. A logistic function is a type of monotonically increasing, smooth, asymptotic sigmoid (S-shaped) function which usually represents the threshold function at the neuron output of the neural network model. A logistic function characterizes an asymmetric unipolar representation having the form, as given below.

$$f(t) = \frac{1}{1+e^{-t}}$$

3.2.3 POLYWOG wavelets

Numerous useful mother wavelets arise from ***POLY***nomials ***W***ind***O***wed with ***G***aussians (POLYWOG) types of functions. For example, all derivatives of a Gaussian function are POLYWOGs and so are admissible mother wavelets.

POLYWOG1: $h_{POLYWOG1} = k_1 t\, exp(\frac{-t^2}{2})$, $k_1 = \sqrt{e}$

POLYWOG2: $h_{POLYWOG2} = k_2(t^3-3t)\, exp(\frac{-t^2}{2})$, $k_2 = 0.7246$

3.2.4 Mexican hat wavelets

The Mexican hat wavelets, is the negative normalized second derivative of a Gaussian function, i.e., up to scale and normalization, the second Hermite function. It is a special case

of the family of continuous wavelets known as Hermitian wavelets, usually referred to as the Mexican hat.

$$\psi(t) = \frac{2}{\sqrt{3\sigma}\pi^{\frac{1}{4}}}\left(1 - \frac{t^2}{\sigma^2}\right)e^{\frac{-t^2}{2\sigma^2}}$$

Chapter 4

Wave-Nets: Combining Wavelets and Neural Networks

The combination of wavelet theory and neural networks has led to the development of wavelet networks. Wavelet networks are feed-forward neural networks using wavelets as activation functions. Wavelet networks have been used in classification and identification problems with some success. The strength of wavelet networks lies in their capabilities of catching essential features from the signals. In wavelet networks, both the position and the dilation of the wavelets are optimized besides the weights. Wave-net is another term to describe wavelet networks. In wave-nets, the position and dilation of the wavelets are fixed and the weights are optimized by the network. We can use wave-nets in the forecasting and betterment of stochastic and chaotic signals.

To use wave-nets in the forecasting, Bodyanskiy (2008) proposed an architecture of adaptive wavelet-neural-network and its learning algorithm for solving non stationary processes forecasting and betterment of stochastic and chaotic signals [1]. We briefly describe this architecture here in this chapter.

At present time the wave-nets have been an increasingly popular technique of soft computing. Combination of the gradient optimization with the error back propagation essentially reduces the rate of learning in hybrid systems and leads to necessity of using rather large training samples. When the data processing has to be carried out in real time, forecasted or emulated sequence is non stationary and distorted. As such, conventional gradient descent learning algorithms appear to be ineffective in such cases. The main idea of defining such systems is the choice of learning algorithm, which is usually based on the gradient procedures of the accepted minimization criterion.

4.1 Architecture of the adaptive wavelet-neural network

Now we explain the five layers architecture, shown in Fig. 5.1, which was given by Bodyanskiy [1]. The input layer of the architecture is formed by the time-delay function t^{-1}. This function can be defined as below.

$$t^{-1}(y(k)) = y(k-1) \quad \ldots(4.1)$$

Under the input of current signal value $y(k)$ the prehistory vector $Y(k) = (y(k-1), y(k-2), \ldots, y(k-n))^T$ is formed as an output of this layer.

The first hidden layer has hn wavelets (h wavelets for each input), with $2hn$ tuning parameters of dilation (centre) c_{ji} and translation (width) σ_{ji}. We can define the general function of wavelets as $\varphi(y(k-i)) = \varphi(y(k-i), c_{ji}, \sigma_{ji}) = \varphi_{ji}(k)$, where $j=1, 2, \ldots, h$, $i=0, 1, 2, \ldots, n$.

Various kinds of analytical wavelets can be used as the activation functions in adaptive wavelet-neural network, for example, Morlet wavelets, Mexican hat wavelets, POLYWOG wavelets and Rasp wavelets.

The second hidden layer performs the operation as defined below.

$$w_j(k) = \prod_{i=1}^{n} \varphi_{ji}\big(y(k-i)\big), \quad j = 1, 2, \ldots, h \quad \ldots(4.2)$$

Then the normalization is performed in the third hidden layer using,

$$\bar{w}_j(k) = \frac{w_j(k)}{\sum_{j=1}^{h} w_j(k)} = \frac{\prod_{j=1}^{n} \varphi_{ji}(y(k-i))}{\sum_{j=1}^{h} \prod_{i=1}^{n} \varphi_{ji}(y_i(k-i))} \quad \ldots(4.3)$$

The condition that $\sum_{j=1}^{h} \bar{w}_j(k) = 1$ should also be fulfilled.

In fourth hidden layer, we use a function $g_j(y(k))$ which is in the linear form, and is given by,

$$g_j(Y(k)) = q_{j0} + \sum_{i=1}^{n} q_{ji} y(k-i) \quad \ldots(4.4)$$

This function can also be written in the following form

$$\bar{w}_j(k)\big(q_{j0} + \sum_{i=1}^{n} q_{ji} y(k-i)\big) = \bar{w}_j(k) q_j^T \bar{Y}(k), \quad \ldots(4.5)$$

where $\bar{Y}(k) = (1, Y^T(k))^T$, $q_j = (q_{j0}, q_{j1}, \ldots, q_{jn})^T$. In this layer we have $h(n+1)$ parameters q_{ji}, $j=1, 2, \ldots, h$, $i = 0, 1, 2, \ldots, n$ which are to be determined. The final output signal ($\hat{y}(k)$) of network is computed as

$$\hat{y}(k) = \sum_{j=1}^{h} \bar{w}_j(k) g_j(Y(k)) =$$

$$\sum_{j=1}^{h} \frac{w_j(k)}{\sum_{j=1}^{h} w_j(k)} g_j\big(Y(k)\big) = \sum_{j=1}^{h} \frac{\prod_{i=1}^{n} \varphi_{ji}\big(y_i(k-i), c_{ji}, \sigma_{ji}\big)}{\sum_{j=1}^{h} \prod_{i=1}^{n} \varphi_{ji}\big(y_i(k-i), c_{ji}, \sigma_{ji}\big)} g_j\big(Y(k)\big) \quad \ldots(4.6)$$

To write the final output signal in the compact form, we introduce the variables vectors *g(Y(k))* and q which are defined as *g(X(k))* = ($\bar{w}_1(k)$, $\bar{w}_1(k)y(k-1)$, ..., $\bar{w}_1(k)y(k-n)$, $\bar{w}_2(k)$, $\bar{w}_2(k)y(k-1), \ldots, \bar{w}_2(k)$, $\bar{w}_2(k)y(k-n), \ldots, \bar{w}_h(k)y(k-n))^T$ and q = (q_{10}, q_{11}, …, q_{1n}, q_{20}, …, q_{2n}, q_{h0}, q_{h1}, ..., q_{hn} $)^T$. The dimension of each variable vector is *h(n +1).*

$$\hat{y}(k) = q^T g\big(Y(k)\big) \qquad \ldots(4.7)$$

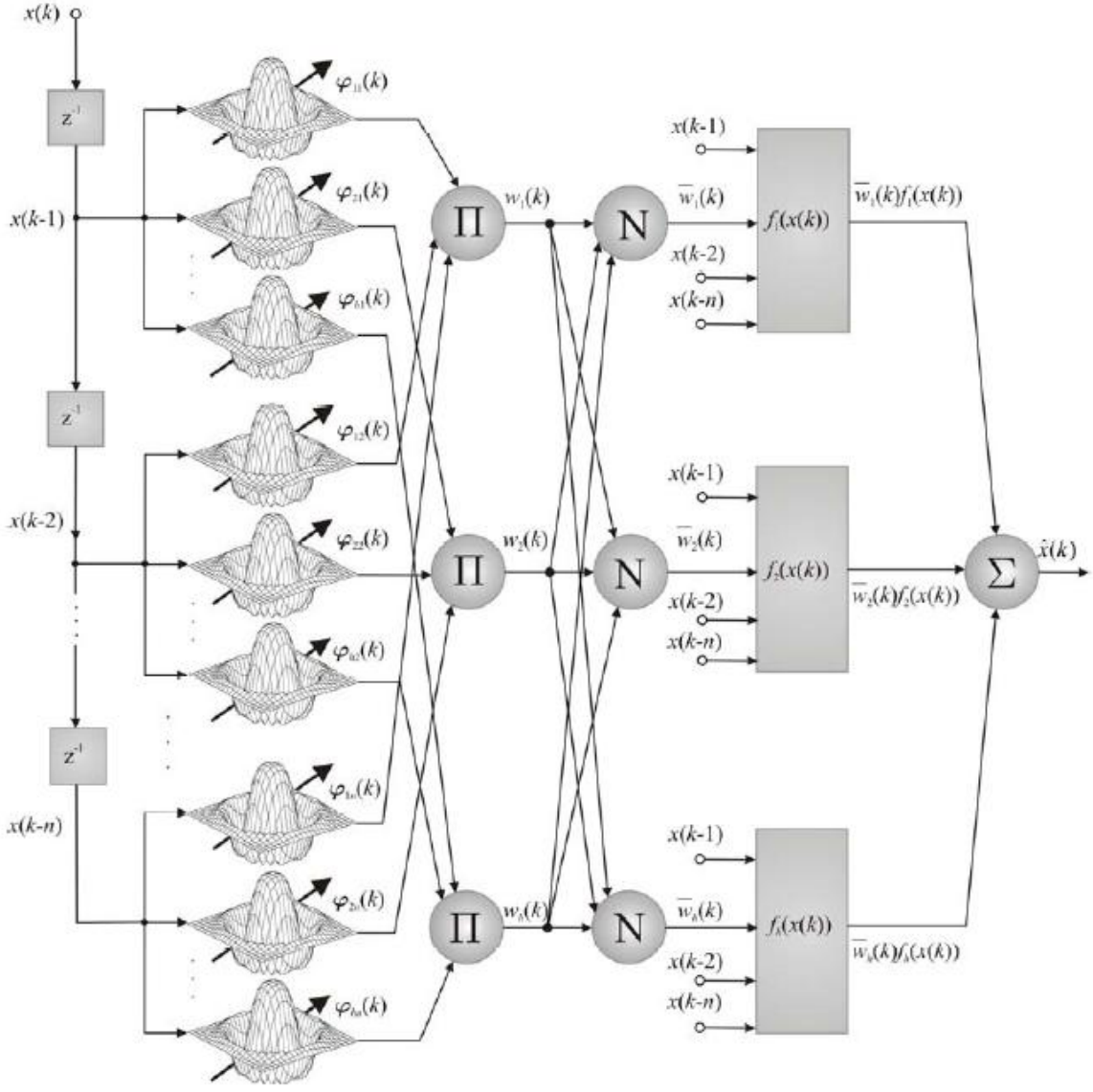


Figure 4.1: Five layers of Adaptive wavelet-neural network

The tuneable parameters of this network are located only in the first and fourth hidden layers. These are *2hn* wavelets parameters c_{ji} and σ_{ji} , and *h(n+1)* parameters of the linear models q_{ji}. These must be determined during the learning process.

4.2 The learning of adaptive wavelet-neural network

There are three methods that can be used to tune the components of the parameter *q*. The first method called as exponentially weighted recurrent least square method can be given by the following relation.

$$q(k+1) = q(k) + \frac{Q(k)\left(y(k) - q^T(k)g(Y(k))\right)}{\alpha + g^T(Y(k))Q(k)g(Y(k))}\, g(Y(k)) \qquad \ldots(4.8)$$

$$Q(k+1) = \frac{1}{\alpha}\left(Q(k) - \frac{Q(k)g(Y(k+1))g^T(Y(k+1))Q(k)}{\alpha + g^T(Y(k+1))Q(k)g(Y(k+1))}\right)$$

where $y(k) - q^T(k)\, g(y(k)) = y(k) - \hat{y}(k) = e(k)$ is the forecasting error, $0 < \alpha \leq 1$ is the out-dated information forgetting factor.

The second method is one-step gradient Kaczmarz algorithm [6] [7]. This method is characterized by the following relationship.

$$\mathrm{q}(k+1) = q(k) + \frac{y(k) - q^T(k)g(Y(k))}{g^T(Y(k))g(Y(k))}\, g(Y(k)) \qquad \ldots(4.9)$$

Third method is given by Goodwin-Ramadge-Caines [8] and this described with the following relationship.

$$\mathrm{q}(k+1) = q(k) + r^{-1}(k)\left(y(k) - q^T g(Y(k))\right) g(Y(k))$$

$$r(k+1) = r(k) + ||g(Y(k+1))||^2 \qquad \ldots(4.10)$$

Here the exponentially weighted recurrent least squares method as given in equation (4.8), can be unstable under small values of parameter α and the convergence of the algorithm shown in equation (4.9) under the intensive disturbance ξ is disrupted, and stochastic approximation procedures, and the algorithm shown in equation (4.10) can be used only in the stationary conditions.

For the tuning of first hidden layer parameters in Adaptive-wavelet-neural-network, back propagation learning algorithm based on the chain rule of differentiation and gradient descend optimization of local criterion is used. The error has been defined as,

$$\mathrm{E(k)} = \frac{1}{2}\mathrm{e}^2(\mathrm{k}) = \frac{1}{2}(\mathrm{y(k)} - \hat{\mathrm{y}}(\mathrm{k}))^2 \qquad \ldots(4.11)$$

In the general case, learning procedure in this layer has the following form.

$$c_{ji}(k+1) = c_{ji}(k) - \eta_c(k)\frac{\partial E(k)}{\partial c_{ji}(k)}$$

$$\sigma_{ji}(k+1) = \sigma_{ji}(k) - \eta_\sigma(k)\frac{\partial E(k)}{\partial \sigma_{ji}(k)} \quad \ldots(4.12)$$

and its properties are completely determined by the learning rate parameter $\eta_c(k)$, $\eta_\sigma(k)$, selected according to the empirical reasons.

Convergence rate can be increased using other algorithms than gradient procedures, such as Hartley [30] or Marquardt algorithms which can be written in general form as,

$$\Phi(k+1) = \Phi(k) + \lambda(J(k)J^T(k) + \eta I)^{-1}J(k)e(k) \quad \ldots(4.13)$$

where $\Phi(k) = (c_{11}(k), \sigma_{11}^{-1}(k), c_{21}(k), \ldots\ldots, c_{hn}(k), \sigma_{hn}^{-1}(k))^T$ is the ($2hn \times 1$) tuneable parameter vector, $J(k)$ is the ($2hn \times 1$) gradient vector of output signal $\hat{y}(k)$ on the tuneable parameters, I is the ($2hn \times 2hn$) identity matrix, η is a scalar regularizing parameter, and λ is the positive scalar gain.

To compute elements of gradient vector

$$J(k) = \left(\frac{\partial \hat{y}(k)}{\partial c_{11}}, \frac{\partial \hat{y}(k)}{\partial \sigma_{11}^{-1}}, \ldots, \frac{\partial \hat{y}(k)}{\partial c_{ji}}, \frac{\partial \hat{y}(k)}{\partial \sigma_{ji}^{-1}}, \ldots, \frac{\partial \hat{y}(k)}{\partial c_{hn}}, \frac{\partial \hat{y}(k)}{\partial \sigma_{hn}^{-1}}\right)^T \quad \ldots(4.14)$$

the chain rule can be used as given by,

$$\frac{\partial \hat{y}(k)}{\partial c_{ji}} = \frac{\partial \hat{y}(k)}{\partial \bar{w}_j} \cdot \frac{\partial \bar{w}_j}{\partial w_j} \cdot \frac{\partial w_j}{\partial \varphi_{ji}} \cdot \frac{\partial \varphi_{ji}}{\partial c_{ji}} = g_j\left(Y(k)\right)\bar{w}_j(k)\left(1 - \bar{w}_j(k)\right)\frac{1}{\varphi_{ji}\left(y_i(k), c_{ji}, \sigma_{ji}^{-1}\right)} \cdot \frac{\partial \varphi_{ji}}{\partial c_{ji}} \quad \ldots(4.15)$$

$$\frac{\partial \hat{y}(k)}{\partial \sigma_{ji}^{-1}} = \frac{\partial \hat{y}(k)}{\partial \bar{w}_j} \cdot \frac{\partial \bar{w}_j}{\partial w_j} \cdot \frac{\partial w_j}{\partial \varphi_{ji}} \cdot \frac{\partial \varphi_{ji}}{\partial \sigma_{ji}^{-1}} = g_j\left(Y(k)\right)\bar{w}_j(k)\left(1 - \bar{w}_j(k)\right)\frac{1}{\varphi_{ji}\left(y_i(k), c_{ji}, \sigma_{ji}^{-1}\right)} \cdot \frac{\partial \varphi_{ji}}{\partial \sigma_{ji}^{-1}}$$

where $\partial\varphi_{ji}/\partial c_{ji}$ and $\partial\varphi_{ji}/\partial \sigma_{ji}^{-1}$ are partial derivatives of wavelet activation function. To reduce the computational complexity of the learning algorithm we can use the matrix inversion lemma in the form

$$(JJ^T + \eta I)^{-1} = \eta^{-1}I - \frac{\eta^{-1}IJJ^T\eta^{-1}I}{1 + J^T\eta^{-1}IJ}, \quad \ldots(4.16)$$

using which it is easy to obtain the relation

$$\lambda(JJ^T + \eta I)^{-1}J = \lambda\frac{J}{\eta + ||J||^2} \quad \ldots(4.17)$$

Substituting this relation in the equation (4.13), we obtain first hidden layer parameters learning algorithm in the form

$$\Phi(k+1) = \Phi(k) + \lambda \frac{J(k)e(k)}{\eta + ||J(k)||^2} \quad \ldots(4.18)$$

It is easy to see, that the equation (4.18) is the nonlinear additive-multiplicative modification of Kaczmarz algorithm, and under λ=1, η=0 coincides with it structurally. To provide the filtering properties to the learning algorithm given in equation (4.18) let us introduce additional tuning procedure of the regularizing parameter η in the form

$$\Phi(k+1) = \Phi(k) + \lambda \frac{J(k)e(k)}{\eta(k)} \quad \ldots(4.19)$$

$$\eta(k+1) = \alpha\eta(k) + ||J(k+1)||^2 .$$

If α = 0, then this procedure coincides with equation (4.18) and has the highest rate of convergence, and if α = 1, then this procedure obtains properties of stochastic approximation.

Here we see that the algorithm (4.19) is stable at any value of forgetting factor α, what favourably differs it from the exponentially weighted recurrent least squares method. As a result this procedure can be used in the form

$$\mathrm{q}(k+1) = q(k) + \lambda_q \eta_q^{-1}(k) \left(y(k) - q^T(k) g\left(Y(k)\right)\right) g\left(Y(k)\right)$$

$$\eta_q(k+1) = \alpha\eta_q(k) + ||g(y(k))||^2 \quad \ldots(4.20)$$

for the fourth layer parameters tuning. The close relation of the algorithms (4.8) and (4.20) is shown as

$$\eta^{-1}(\mathrm{k}) = \mathrm{TrP(k)}, \quad \ldots(4.21)$$

However algorithm (4.20) is much simpler in the computing implementation and easily reconstructs its properties from the most following to the most filtering ones.

4.3 Simulation of Adaptive Wavelet-Neural-Network

4.3.1 Training and Testing Strategy

To demonstrate the effectiveness of the Adaptive Wavelet-Neural-Network and its learning algorithm (4.19), (4.20), Adaptive Wavelet-Neural-Network was trained to emulate the time

series forecasting. In the online mode of learning, Adaptive Wavelet-Neural-Network was trained with algorithm (4.19), (4.20) using signal $u(k) = sin(2k/250)$, where k = 1.1, 1.2, …, 30.0 as our input. The first 9 values were kept for training and the next input signal was kept for testing. Now this procedure was repeated for all inputs. Thus we have total 261 inputs for training and 29 signals for testing the network. The values *y(t-3), y(t-2) and y(t-1)* were used to emulate y*(t)*, i.e., we have used 3 inputs to predict the 4^{th} input. The number of wavelets used in each layer was 60. The parameters of the learning algorithm were α = 0.6, λ_q = 2, λ = 1. Initial values were $\eta(0)$ = 1 and $\eta_q(0)$ = 10000. The mother wavelet used is Mexican hat wavelet.

The MATLAB program developed for the simulation purpose is given in Fig. 4.2

```
j=1;
for i=1.1:0.1:30
   inp(j)=sin(2*i/250);
   j=j+1;
end
l = 1;
u = 1;
a = 0.6;
r = 2;
v = 1000;
n = input('Enter the value of n:');
h = input('Enter the value of no. of wavelets:');
[q c b dc db] = first(inp, n, h);
[w sum]= second(q,n,h);
nor = third(w,sum,h);
[f  p]=fourth(inp,n,h);
y=output(nor,f,h);
for i=1:n
   x(i)=inp(i);
end
x(n+1)=y;
e=inp(n+1)-y;
k=1;
for i=1:h
   f1(k)= nor(i);
   k=k+1;
   for j=1:n
      f1(k)= nor(i)*inp(j);
      k=k+1;
```

```
    end
end
k=1;
for i=1:h
    for j=1:n+1
        p1(k)=p(i,j);
        k=k+1;
    end
end
p1=p1';
%training
k=1;
for i=1:h*n
  Q(k)=c(i);
  k=k+1;
  Q(k)=1/(b(i));
  k=k+1;
end
Q=Q';
for i=1:n
    for j=1:h
        ddc(j,i)=f(j)*nor(j)*(1-nor(j))*dc(j,i)/q(j,i);
        ddb(j,i)=-f(j)*nor(j)*(1-nor(j))*db(j,i)/(q(j,i)*b(j,i)*b(j,i));
    end
end
con=0;
for i=1:n
    for j=1:h
        jj=con+2*j-1;
        J(jj)=ddc(j,i);
        J(jj+1)=ddb(j,i);
     end
 con=con+(2*h);
end
for d=1:2
 for m=1:289-n
   if mod(n+m+1,10)~= 0
     Q = Q + (l*e)*J'/u;
      p1 = p1 + (r *( inp(n+1) - (p1'*f1')))* f1'/ v;
    v= (a*v) + ((f*f')^2);
    k=1;
    for i=1:h*n
```

```
        c(i)= Q(k);
        k=k+1;
        b(i)=1/(Q(k));
        k=k+1;
     end
     k=1;
     for i=1:h
       for j=1:n+1
         p(i,j)=p1(k);
         k=k+1;
       end
     end
     [q dc db]= tfirst(inp, n, h,c,b,m);
     [w sum]= second(q,n,h);
     nor=third(w,sum,h);   f = tfourth(inp,m,n,h,p);
     y=output(nor,f,h);
     x(n+1+m)=y;
     e=inp(n+m+1)-y;
     k=1;
     for i=1:h
      f1(k)= nor(i);
      k=k+1;
      for j=1:n
        f1(k)= nor(i)*inp(j+m);
        k=k+1;
      end
     end
     k=1;
     for i=1:h
        for j=1:n+1
p1(k)=p(i,j);
           k=k+1;
        end
     end
     k=1;
     for i=1:h*n
       Q(k)=c(i);
       k=k+1;
       Q(k)=1/(b(i));
       k=k+1;
     end
     for i=1:n
```

```
      for j=1:h
        ddc(j,i)=f(j)*nor(j)*(1-nor(j))*dc(j,i)/q(j,i);
        ddb(j,i)=-f(j)*nor(j)*(1-nor(j))*db(j,i)/(q(j,i)*b(j,i)*b(j,i));
      end
    end
    k=1;
    for i=1:n
      for j=1:h
        J(k)=ddc(j,i);
        k=k+1;
        J(k)=ddb(j,i);
        k=k+1;
      end
    end
    u = (u*a) + (J*J');
else
  q=ttfirst(inp,n,h,c,b,m);
  [w sum]= ttsecond(q,n,h);
  nor=ttthird(w,sum,h);
  f = ttfourth(inp,n,h,p,m);
  y=ttoutput(nor,f,h);
  i=(n+m+1)/10;
  rr(i)=y;
end
end
d=d+1;
end
j=1;
k=1;
for i=1:290
   if mod(i,10)~=0
    xx(j)=x(i);
    j=j+1;
end
end
j=1;
for i=1:290
   if mod(i,10)~=0
    trinp(j)=inp(i);
    j=j+1;
else
    testinginp(k)=inp(i);
```

```
   k=k+1;
end
end
plot(trinp,xx)
plot(testinginp,rr)
```

Figure 4.2: MATLAB program for simulation

The results from this program are given in Fig. 4.3. The Fig. 4.3(a) contains the training graph of simulation exercise and the Fig. 4.3(b) consists of the results of testing of network.

4.3.2: Results of Simulation

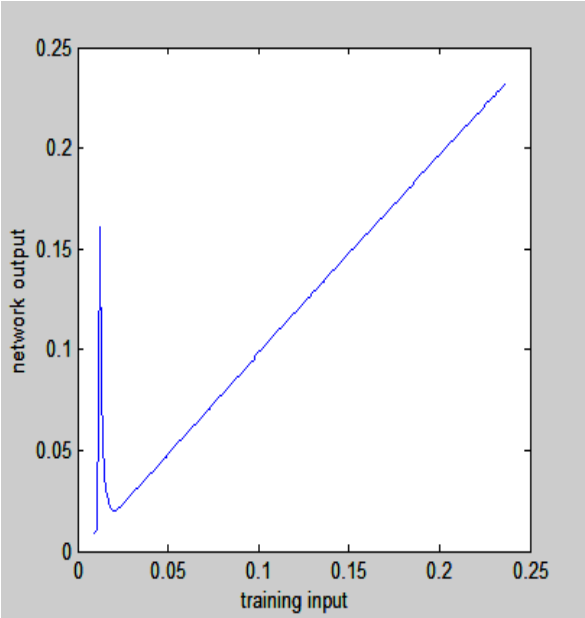


Figure 4.3(a) Training graph of AWNN network

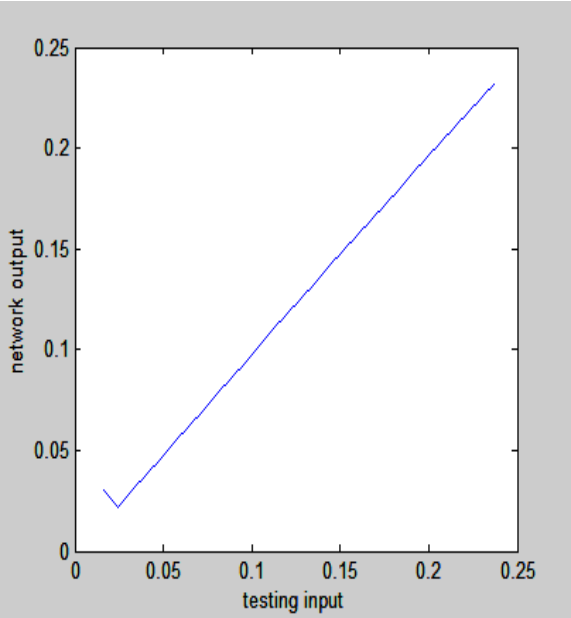


Figure 4.3(b) Testing graph of AWNN network

It can be clearly seen from the results shown in Fig 4.3 that the Adaptive Wavelet-Neural-Network with the learning algorithm given in the equation (4.19) and equation (4.20) ensures a good quality of emulation and high learning rate. Thus this network can be further used in different areas such as stock market; weather forecasting etc. to obtain the forecasting results.

Chapter 5

Stock Forecasting using Wave-Nets

5.1 Introduction

Innovation and easing of life through innovation has been the cherished objective of human beings. The prevailing notion in society is that wealth brings comfort and luxury, so it is not surprising that there has been a good amount of work on the problem of how to predict the stock markets. Various technical, fundamental, and statistical indicators have been proposed and used with varying results. However, no single technique has been successful enough to consistently predict the market behaviour. With the development of wave-nets, researchers and investors are hoping that the market mysteries can be unravelled with the help of this technique.

Wave-nets can be used to predict stock market prices because they are able to learn nonlinear relationships between inputs and outputs. Several researchers claim that stock market and other complex systems exhibit chaos. Chaos is a nonlinear deterministic process that is random and can not be expressed easily. With the wave-nets ability to learn nonlinear, chaotic systems, it may be possible to outperform traditional analysis and other computer-based methods by adopting wave-net architecture.

5.2 Data Collection

In the present dissertation work, we have taken the daily closing value of stock index of State Bank of India (SBI) from 1st January 2009 to 30th January 2010 as the time series for training and testing of the wave-nets. Thus, we have 261 values in the time series. It is worth mentioning here that this data is from the open source and has been taken from the site *www.moneycontrol.com.* The time series data used in this work is:

{1315.8, …, 2058} with the first element in this set indicating SBI stock index on January 1, 2009, second element on January 2, 2009, …, last element on January 30, 2010.

5.3 Training and Testing Strategy

We have used, as mentioned and illustrated in earlier chapters, the AWNN in order to forecast the stock index of SBI. The network was trained with the algorithm explained in Chapter 4 using the patterns representing the closing value of the stock index of SBI, as input.

The AWNN proposed in this work has 3 parameters, namely, the number of inputs, the number of wavelets in each layer and number of iterations. The number of iterations is the number of times we repeat the procedure of training. The example values taken for number of inputs are 3, 4 and 5; the example values taken for number of wavelets in second layer are 30, 40 and 50 and the numbers of iterations that have been tried in this simulation study are taken as 3, 4 and 5.

The time series data has been divided into training and testing patterns as per the following method. If we have taken 3 as the input value of the training patterns then first three training patterns are used to emulate the 4^{th} training pattern, then 2^{nd}, 3^{rd} and 4^{th} training patterns are used to emulate the 5^{th} training pattern. This procedure is repeated till we get the 9^{th} training pattern. Now, we keep the 10^{th} pattern for the testing. We again use the patterns for training till we get 19^{th} training pattern and 20^{th} pattern is again kept for testing. We repeated this procedure for all input patterns. For the betterment of the results we have used another parameter, i.e., the number of iterations, it is the number of times we are repeating this procedure. A similar method has been adopted when we consider the number of inputs as four and five.

The values of the parameters used in the algorithm are $\alpha = 0.6$, $\lambda_q = 2$ and $\lambda = 1$. Initial values are taken as $\eta(0) = 1$ and $\eta_q(0) = 10000$. The mother wavelets used here in this work is Mexican Hat wavelets.

The AWNN has been implemented in MATLAB for the forecasting of stock index of SBI and the program developed is given in Fig. 5.1.

inp=[1315.8 1330 1361.2 1324.2 1238.7 1215.9 1156.85 1177.2 1199.8 1146.75 1165.5 1147.4 1112.3 1081.95 1089.05 1041.75 1088.6 1110.45 1096.85 1152.2 1095.65 1091.3 1096.9 1093.9 1118.35 1147.55 1164.05 1158.85 1159.5 1194 1136 1100.35 1070.8 1059.55 1046.6 1028.25 1037.75 1024.15 1027.1 995.25 975.85 957.55 934.55 940.85 896.8 910.9 953.05 987.8 949.6 960.95 968.2 953.55 1023.45 1034.65 1050.05 1093.8 1125.35 1022 1066.55 1073.95 1145.35 1128.95 1123.9 1140.4 1217.9 1295.6 1261.7 1306.35 1295.95 1255.35 1232.6 1265.7 1307.8 1282.15 1240 1277.7 1366.2 1343.65 1323.7 1366.95 1325.15 1260.7 1295.45 1262.15 1267.85 1312.25 1577.95 1754.45 1780.2 1714.3 1731.7 1719.85 1687.95 1790.55 1828.9 1869.1 1879.8 1909.5 1873.2 1875.9 1817.9 1695.1 1763.3 1756.75 1696.25 1637 1642 1714.1 1663.4 1702.2 1724.35 1694.75 1708.35 1716.5 1704.3 1748.95 1765.1 1742.05 1779.8 1758.9 1810.65 1655 1636.35 1587 1601.95 1543.6 1546.1 1582.6 1633.9 1607.35 1674.65 1721.15 1708.4 1693.55 1725.9 1698.5 1709.55 1680.5 1656.45 1722.8 1814 1846.5 1856.6 1844.7 1797.2 1741.85 1711.9 1729.7 1702.4 1799.05 1797.4 1713.05 1733.8 1691.1 1753.6 1776.75 1789.6 1747.35 1743.5 1754.2 1781.55 1743.05 1733.4 1735.3 1747.15 1763.9 1815 1894.15 1894.65 1879.3 1918.8 1956.15 2009.25 2089.6

```
2106.7 2142.3 2172.55 2142.55 2158.65 2139.15 2090.85 2195.7 2209.9 2131.9 2158.85
2101.35 2115 2066 2172.55 2269.45 2330.45 2453.9 2462.25 2465.25 2385.3 2325.7
2353.85 2305.6 2202.95 2218.2 2195 2191 2103.5 2163.4 2138.8 2204.2 2318.55 2368.1
2379.35 2295.9 2298.05 2344.65 2351.25 2330.6 2280.45 2335.75 2320.7 2304.6 2322.85
2252.95 2242.5 2238.15 2292.15 2307.35 2353.95 2327.65 2325.35 2307.15 2294.3 2295.8
2265.75 2248.3 2180.75 2152.65 2165.45 2144.95 2146.85 2166.05 2209.45 2220.85
2225.25 2269.45 2291.6 2291.9 2305.05 2292.6 2287 2267.55 2203.25 2177.85 2157.05
2144 2156.55 2172.9 2159.75 2123.95 2090.2 2093.35 1987.15 2003.45 2058 ];
z=minmax(inp);
inp=premnmx(inp);
l = 1;
u = 1;
a = 0.6;
r = 2;
v = 1000;
n = input('Enter the value of n:');
h = input('Enter the value of no. of wavelets:');
g = input('Enter the no. of iteration:');
[q c b dc db] = first(inp, n, h);
[w sum]= second(q,n,h);
nor = third(w,sum,h);
[f p]=fourth(inp,n,h);
y=output(nor,f,h);
for i=1:n
   x(i)=inp(i);
end
x(n+1)=y;
e=inp(n+1)-y;
k=1;
for i=1:h
   f1(k)= nor(i);
   k=k+1;
   for j=1:n
      f1(k)= nor(i)*inp(j);
      k=k+1;
   end
end
k=1;
for i=1:h
   for j=1:n+1
      p1(k)=p(i,j);
      k=k+1;
```

```
    end
end
p1=p1';
%training
k=1;
for i=1:h*n
Q(k)=c(i);
k=k+1;
Q(k)=1/(b(i));
k=k+1;
end
Q=Q';
for i=1:n
    for j=1:h
        ddc(j,i)=f(j)*nor(j)*(1-nor(j))*dc(j,i)/q(j,i);
        ddb(j,i)=-f(j)*nor(j)*(1-nor(j))*db(j,i)/(q(j,i)*b(j,i)*b(j,i));
    end
end
con=0;
for i=1:n
    for j=1:h
        jj=con+2*j-1;
        J(jj)=ddc(j,i);
        J(jj+1)=ddb(j,i);
     end
con=con+(2*h);
end
for d=1:g
 for m=1:260-n
 if mod(n+m+1,10)~= 0
   %new values of c and b
   lll=(l*e)/u;
   lll= lll*J';
   Q = Q + lll;
   %new value of p1
   p1 = p1 + (r*( inp(n+1) - (p1'*f1')))* f1'/ v;
   v= (a*v) + ((f*f')^2);
   k=1;
   for i=1:h*n
        c(i)= Q(k);
        k=k+1;
        b(i)=1/(Q(k));
```

```
    k=k+1;
end
k=1;
for i=1:h
   for j=1:n+1
      p(i,j)=p1(k);
      k=k+1;
   end
end
[q dc db]= tfirst(inp, n, h,c,b,m);
[w sum]= second(q,n,h);
nor=third(w,sum,h);
f = tfourth(inp,m,n,h,p);
y=output(nor,f,h);
x(n+1+m)=y;
e=inp(n+m+1)-y;
k=1;
for i=1:h
 f1(k)= nor(i);
 k=k+1;
 for j=1:n
    f1(k)= nor(i)*inp(j+m);
    k=k+1;
 end
end
k=1;
for i=1:h
   for j=1:n+1
       p1(k)=p(i,j);
       k=k+1;
   end
end
k=1;
for i=1:h*n
  Q(k)=c(i);
  k=k+1;
  Q(k)=1/(b(i));
  k=k+1;
end
for i=1:n
  for j=1:h
    ddc(j,i)=f(j)*nor(j)*(1-nor(j))*dc(j,i)/q(j,i);
```

```
      ddb(j,i)=-f(j)*nor(j)*(1-nor(j))*db(j,i)/(q(j,i)*b(j,i)*b(j,i));
    end
  end
  k = 1;
  for i=1:n
    for j=1:h
      J(k)= ddc(j,i);
      k = k+1;
      J(k)= ddb(j,i);
      k = k+1;
    end
  end
  u = (u*a) + (J*J');
else
 q=ttfirst(inp,n,h,c,b,m);
 [w sum]= ttsecond(q,n,h);
 nor=ttthird(w,sum,h);
 f = ttfourth(inp,n,h,p,m);
 y=ttoutput(nor,f,h);
 i=(n+m+1)/10;
 rr(i)=y;
end
end
d=d+1;
end
j=1;
k=1;
for i=1:261
  if mod(i,10)~=0
   xx(j)=x(i);
   j=j+1;
  end
end
j=1;
for i=1:261
  if mod(i,10)~=0
   trinp(j)=inp(i);
   j=j+1;
 else
   testinginp(k)=inp(i);
   k=k+1;
 end
```

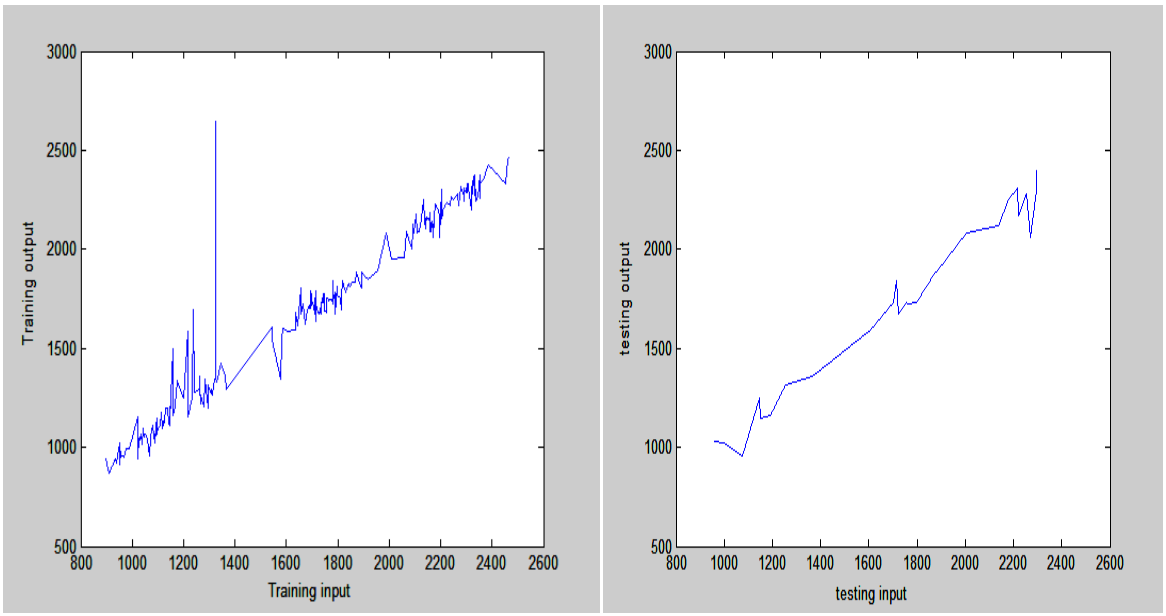
```
end
[actual x1]=opr(trinp,xx);
actual=postmnmx(actual,z(1),z(2));
x1=postmnmx(x1,z(1),z(2));
plot(actual,x1)
[testinginp1,rr1]=opr10(testinginp,rr);
testinginp1=postmnmx(testinginp1,z(1),z(2));
rr1=postmnmx(rr1,z(1),z(2));
```

Figure 5.1: MATLAB program for forecasting the data of SBI

Using the algorithm shown in Fig. 5.1, which was written in MATLAB and varying the different parameters, we have generated different networks.

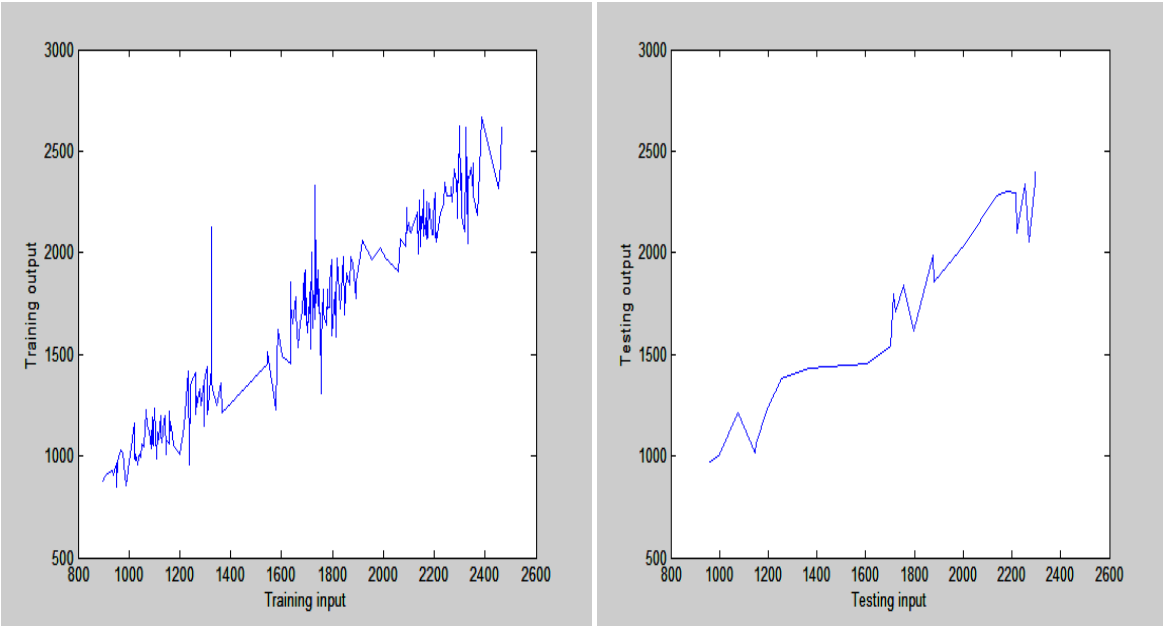
5.4 Forecasting the stock index of SBI

Using the training and testing strategy as explained above we have generated 27 graphs by varying different parameters. These graphs are given in Fig. 5.2, 5.3, …, 5.28. In the part (a) of each figure we have training graph and in the part (b) we have testing graph.



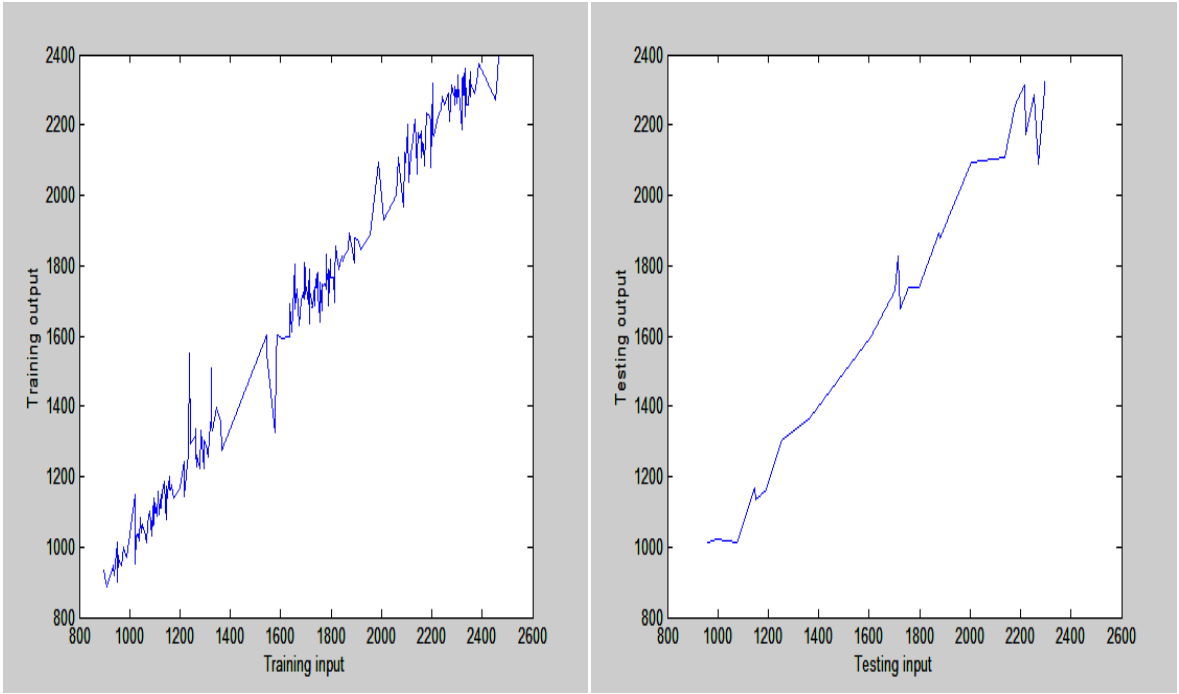
(a) Training graph **(b) Testing graph**

Figure 5.2: Training and testing graph for number of inputs 3, number of wavelets 30, and 3 iterations



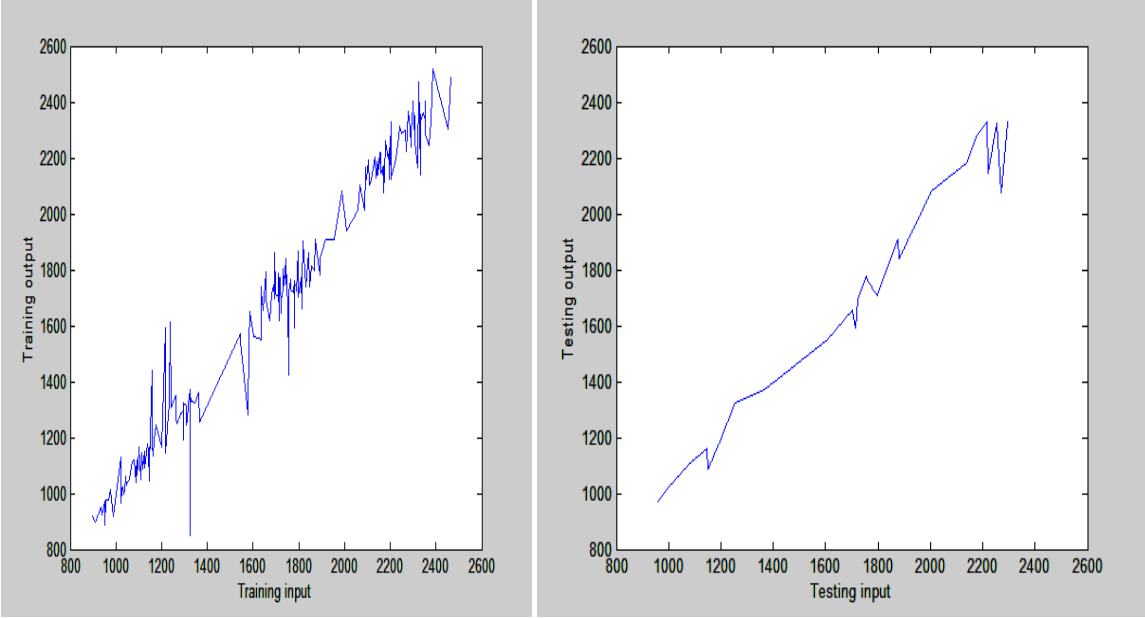
(a) Training graph **(b) Testing graph**

Figure 5.3: Training and testing graph for number of inputs 3, number of wavelets 30, and 4 iterations



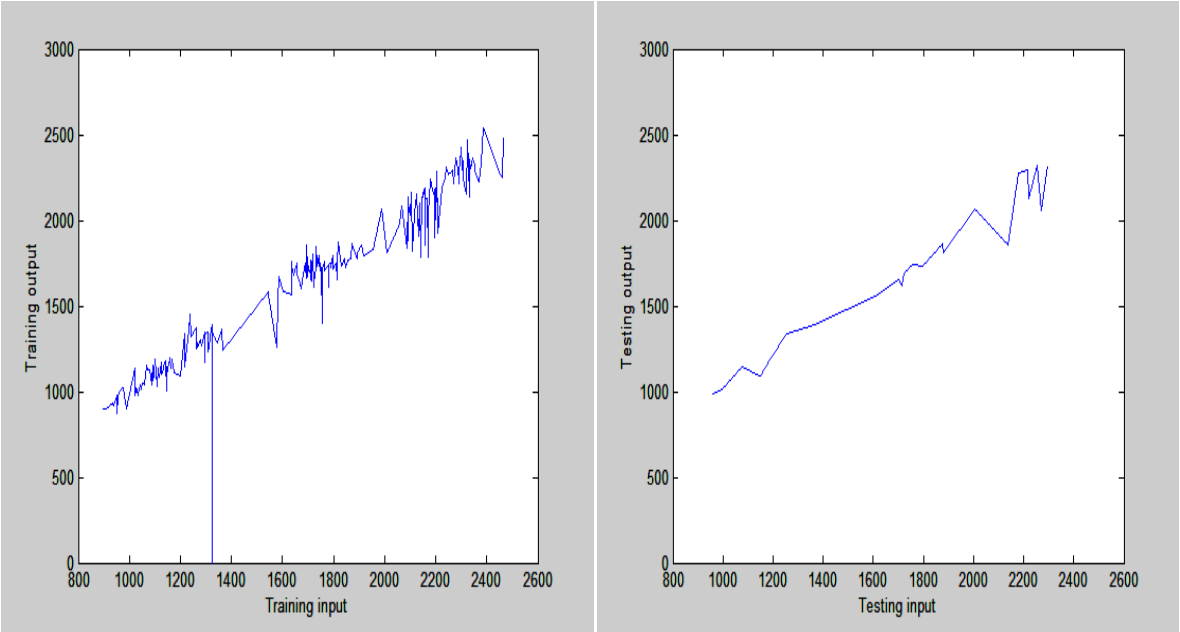
(a) Training graph **(b) Testing graph**

Figure 5.4: Training and testing graph for number of inputs 3, number of wavelets 30, and 5 iterations



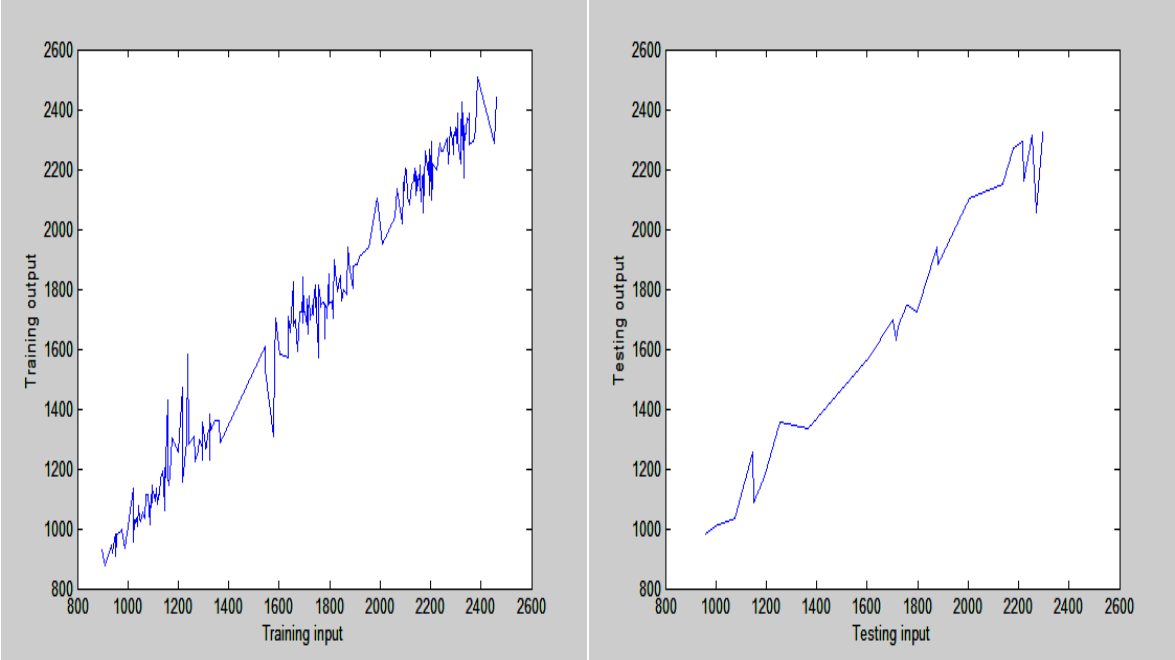
(a) Training graph **(b) Testing graph**

Figure 5.5: Training and testing graph for number of inputs 3, number of wavelets 40, and 3 iterations



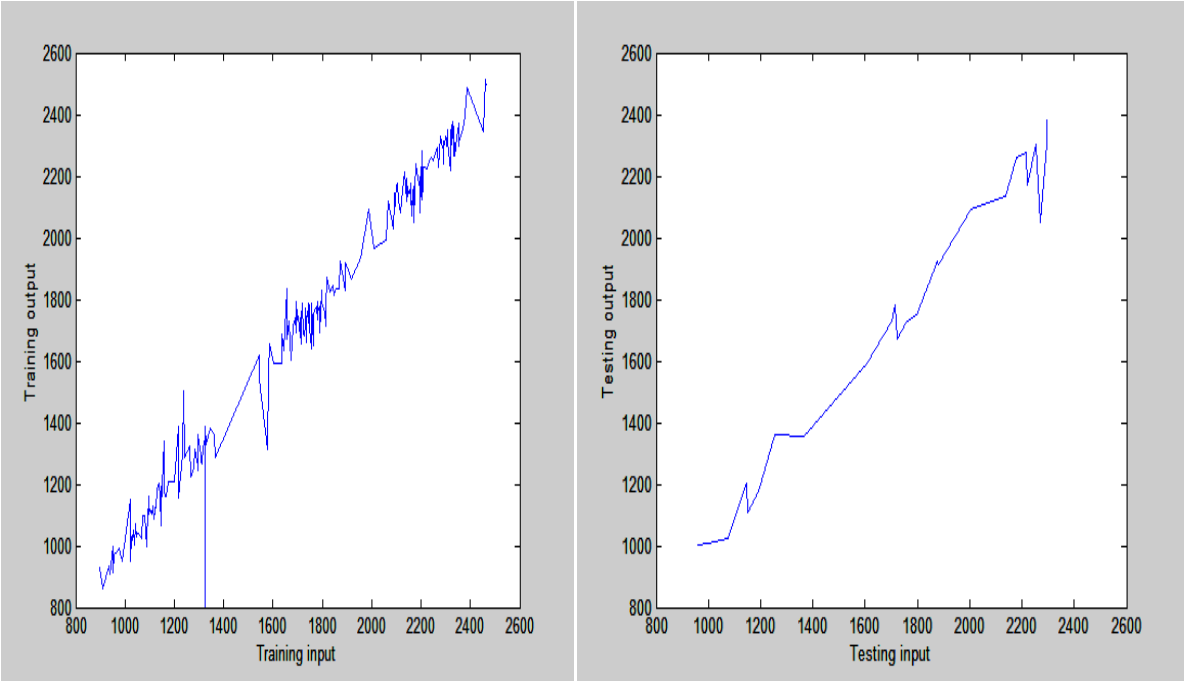
(a) Training graph **(b) Testing graph**

Figure 5.6: Training and testing graph for number of inputs 3, number of wavelets 40, and 4 iterations



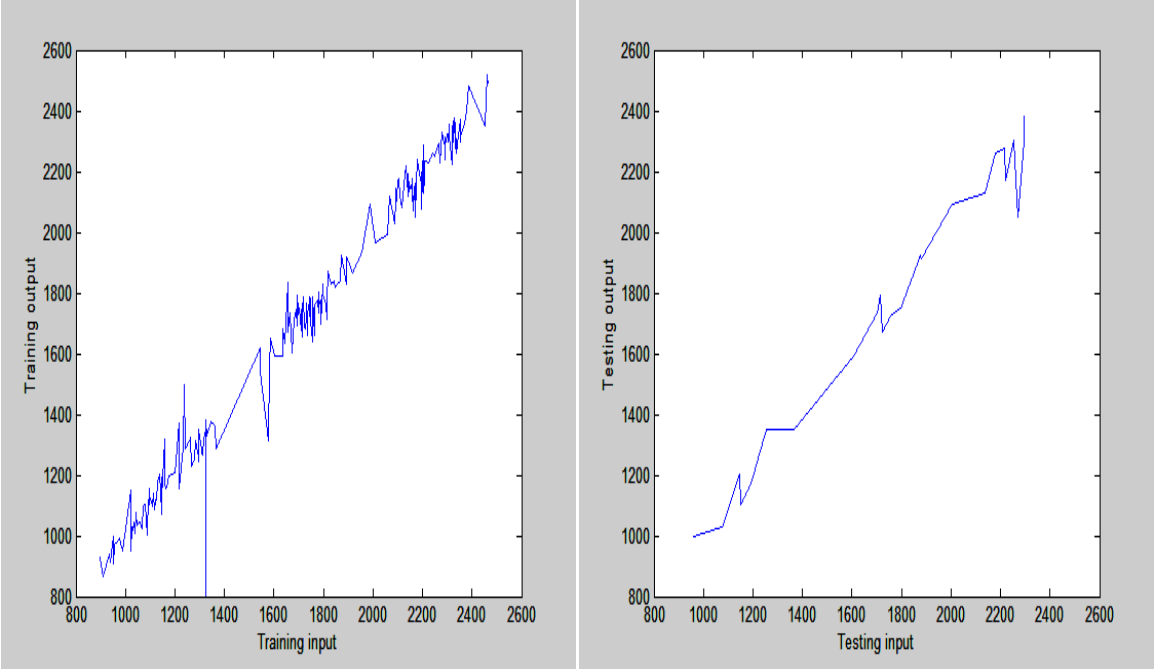
(a) Training graph **(b) Testing graph**

Figure 5.7: Training and testing graph for number of inputs 3, number of wavelets 40, and 5 iterations



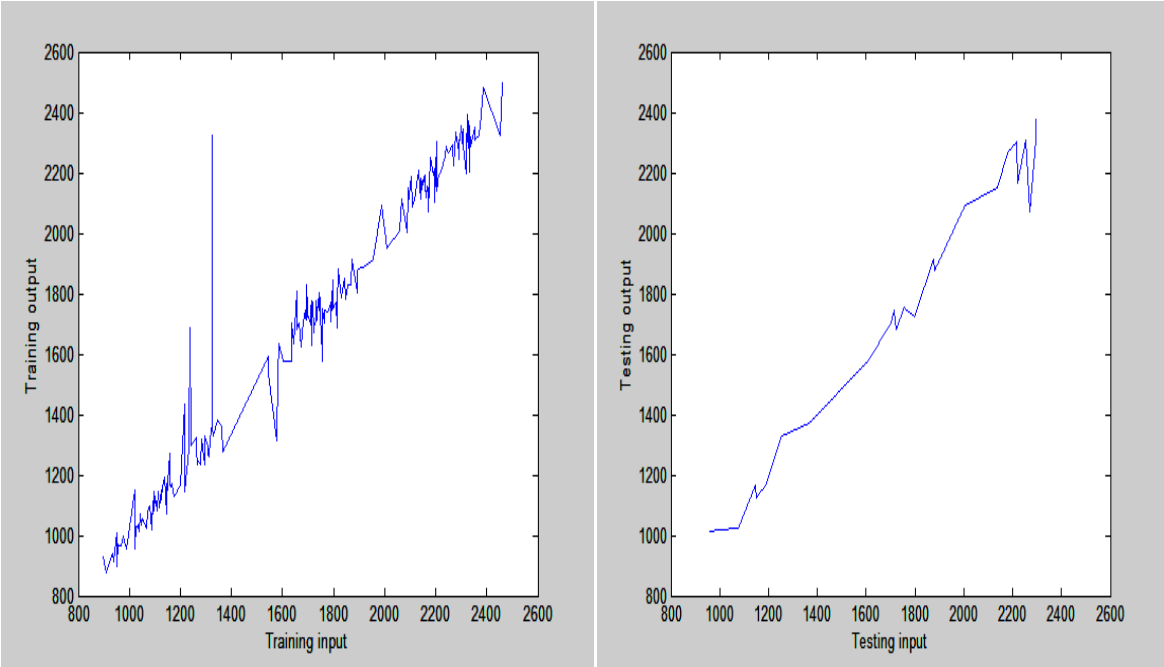
(a) Training graph **(b) Testing graph**

Figure 5.8: Training and testing graph for number of inputs 3, number of wavelets 50, and 3 iterations



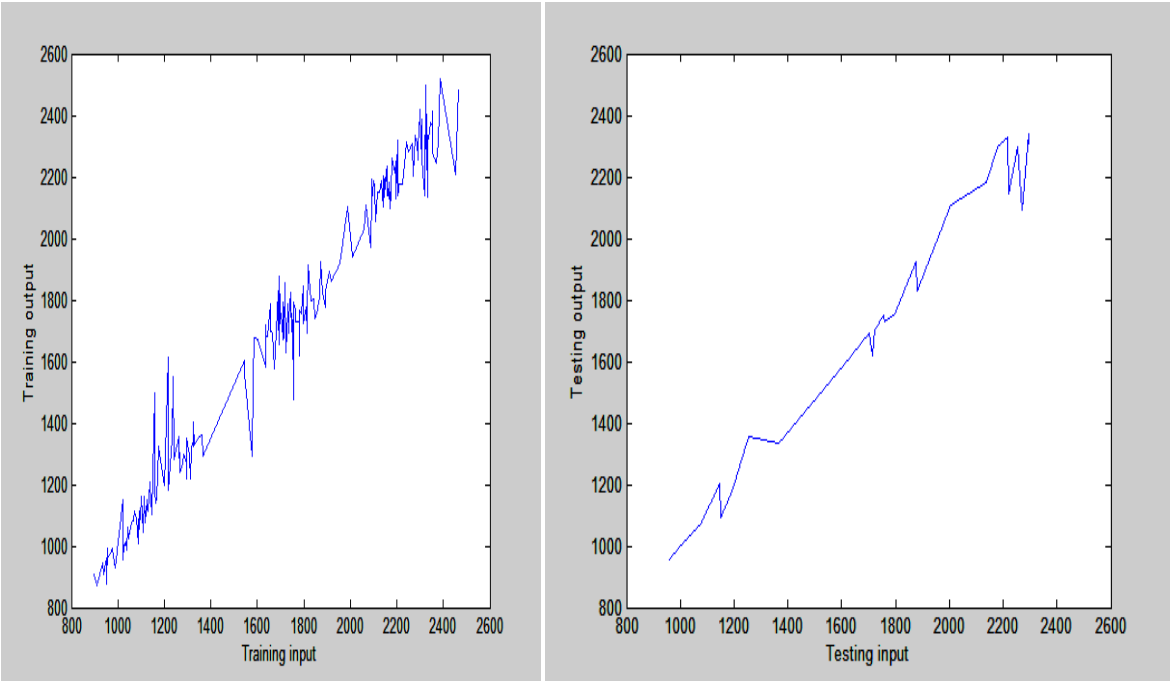
(a) Training graph **(b) Testing graph**

Figure 5.9: Training and testing graph for number of inputs 3, number of wavelets 50, and 4 iterations



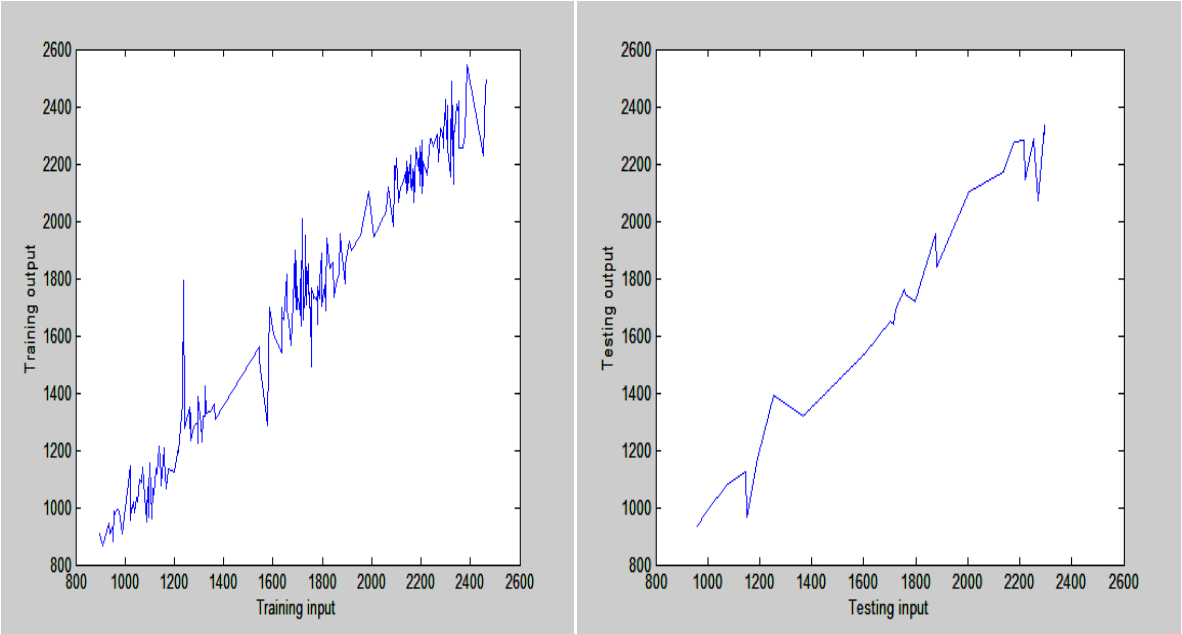
(a) Training graph **(b) Testing graph**

Figure 5.10: Training and testing graph for number of inputs 3, number of wavelets 50, and 5 iterations



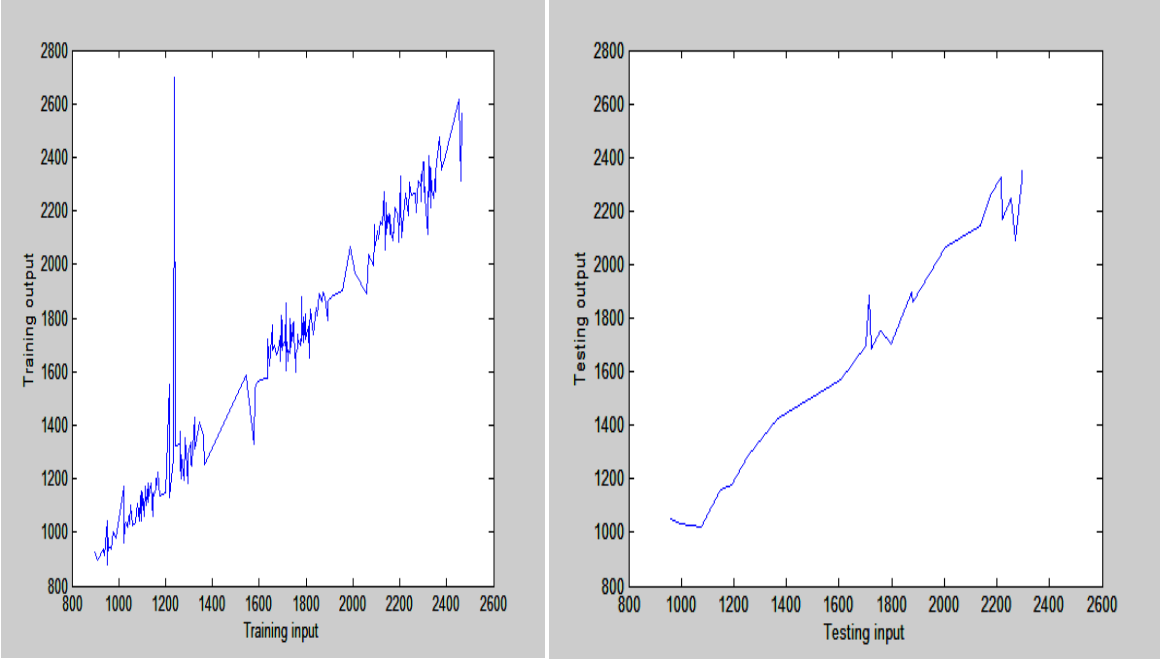
(a) Training graph **(b) Testing graph**

Figure 5.11: Training and testing graph for number of inputs 4, number of wavelets 30, and 3 iterations



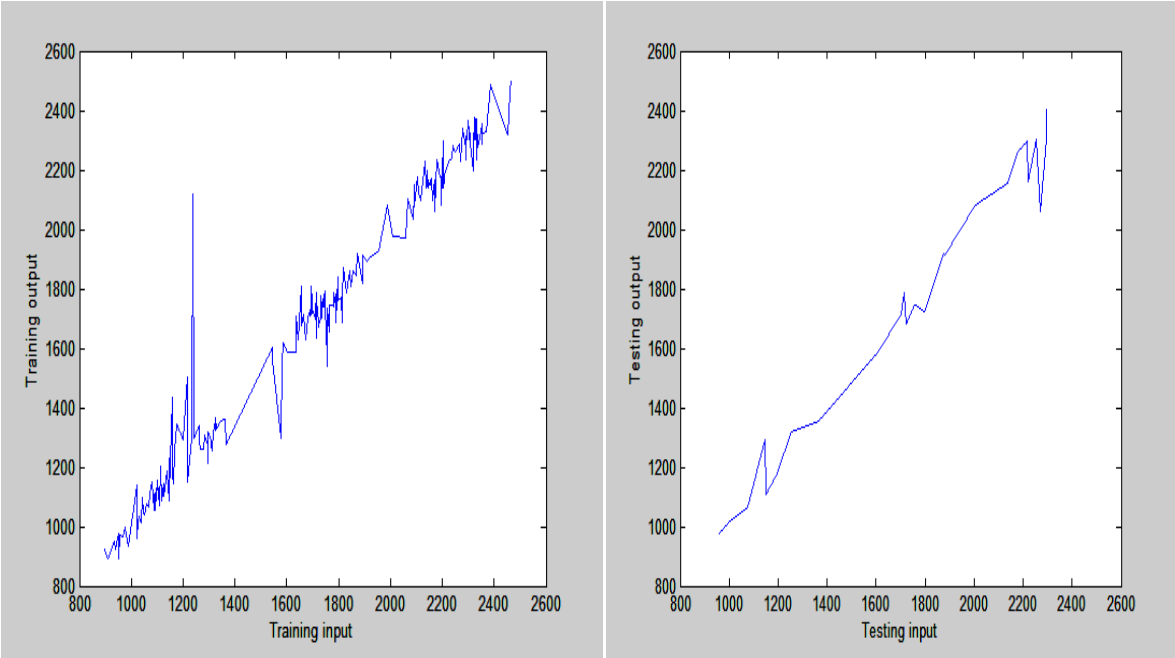
(a) Training graph **(b) Testing graph**

Figure 5.12: Training and testing graph for number of inputs 4, number of wavelets 30, and 4 iterations



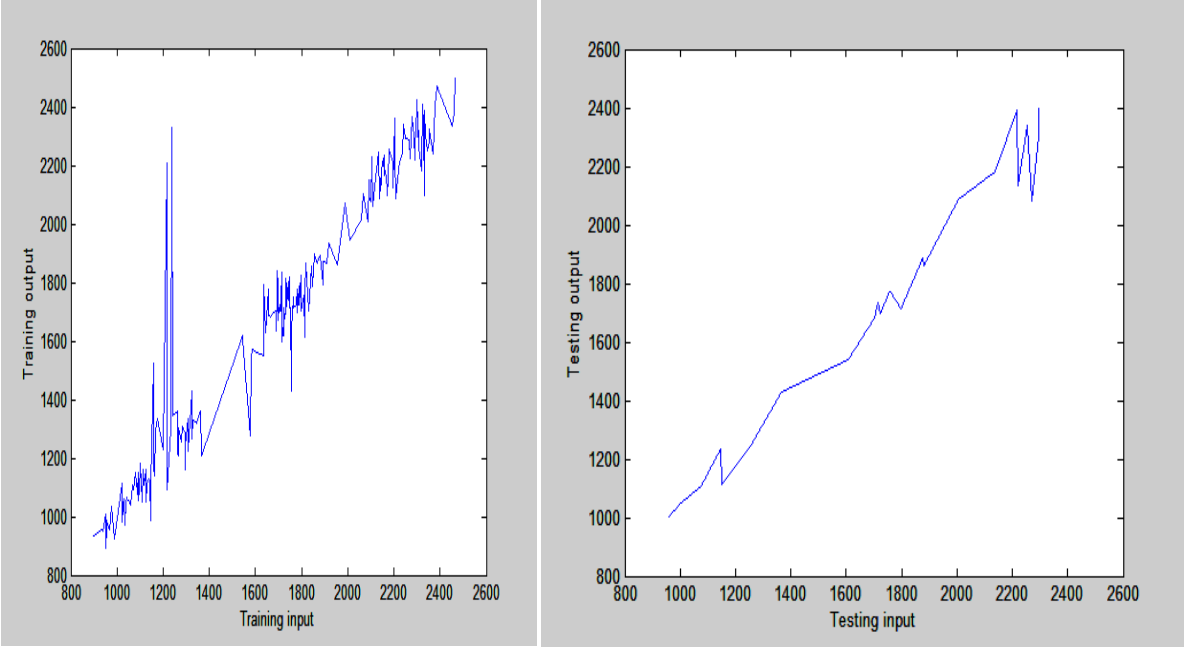
(a) Training graph **(b) Testing graph**

Figure 5.13: Training and testing graph for number of inputs 4, number of wavelets 30, and 5 iterations



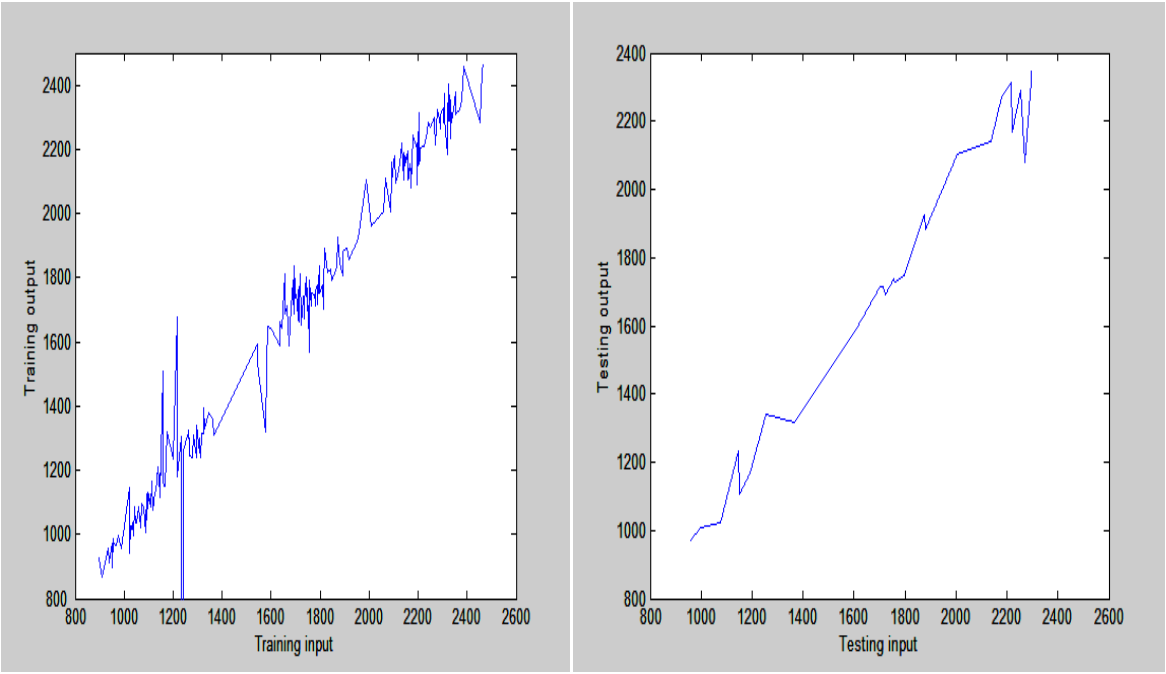
(a) Training graph **(b) Testing graph**

Figure 5.14: Training and testing graph for number of inputs 4, number of wavelets 40, and 3 iterations



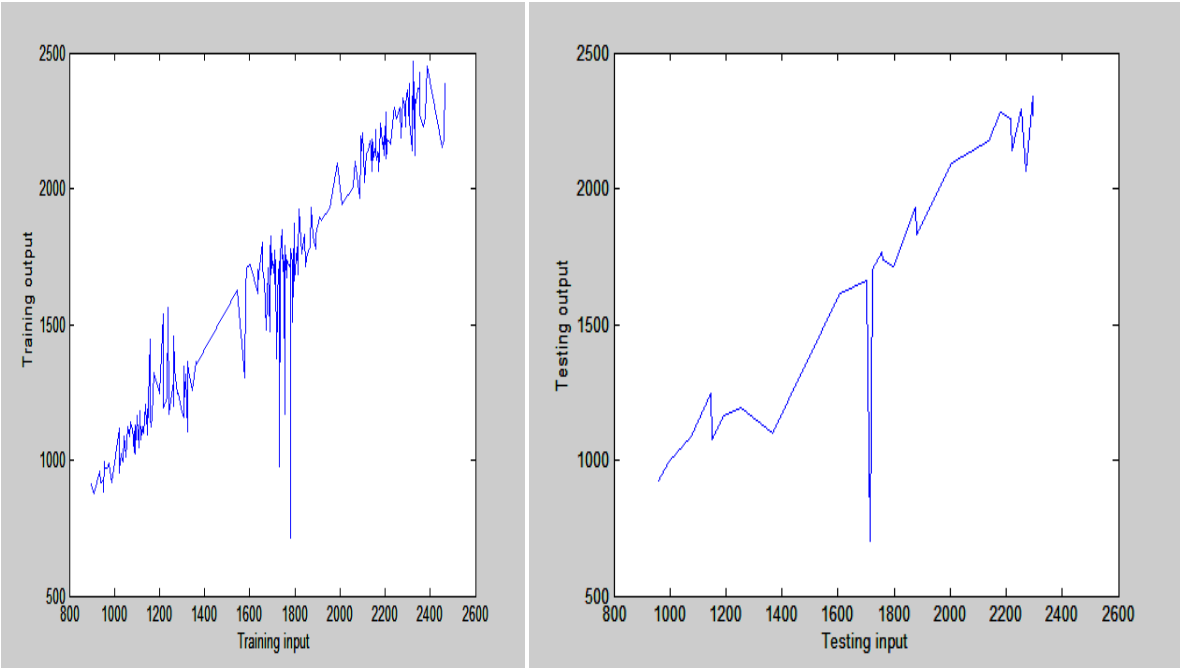
(a) Training graph **(b) Testing graph**

Figure 5.15: Training and testing graph for number of inputs 4, number of wavelets 40, and 4 iterations



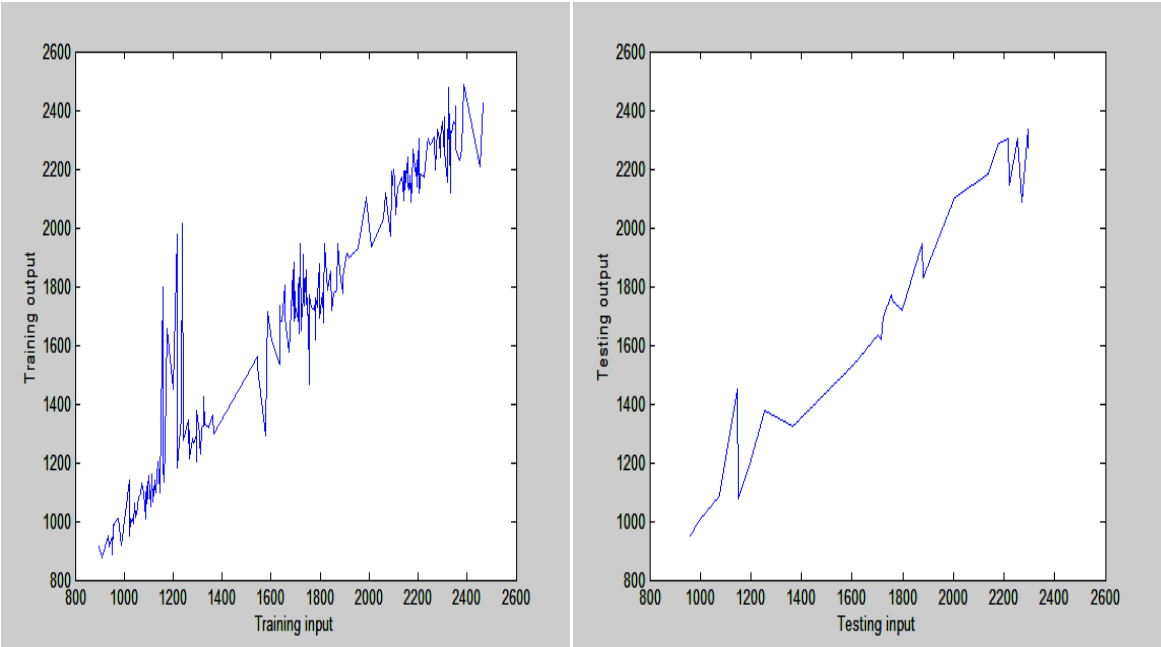
(a) Training graph **(b) Testing graph**

Figure 5.16: Training and testing graph for number of inputs 4, number of wavelets 40, and 5 iterations



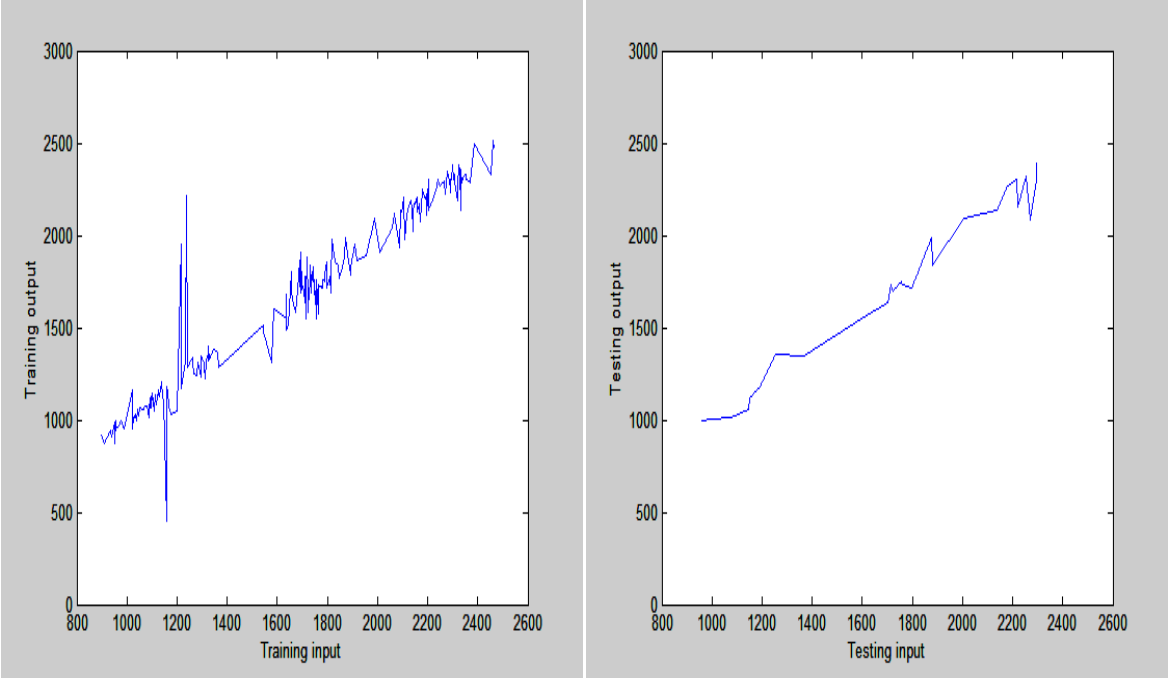
(a) Training graph **(b) Testing graph**

Figure 5.17: Training and testing graph for number of inputs 4, number of wavelets 50, and 3 iterations



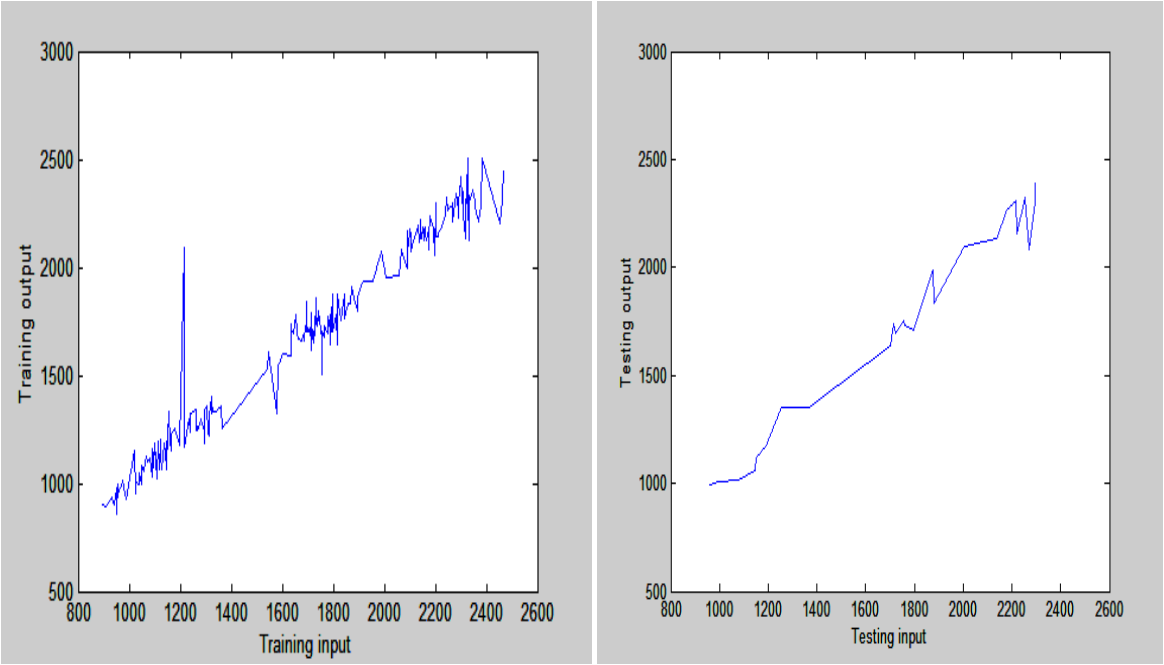
(a) Training graph **(b) Testing graph**

Figure 5.18: Training and testing graph for number of inputs 4, number of wavelets 50, and 4 iterations



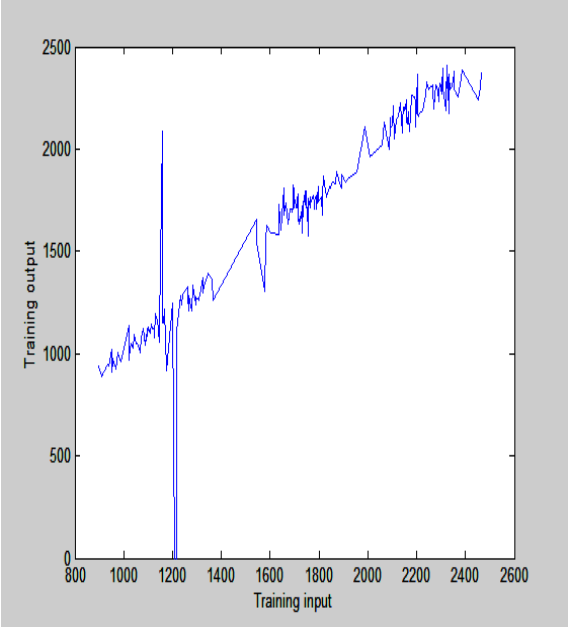
(a) Training graph **(b) Testing graph**

Figure 5.19: Training and testing graph for number of inputs 4, number of wavelets 50, and 5 iterations

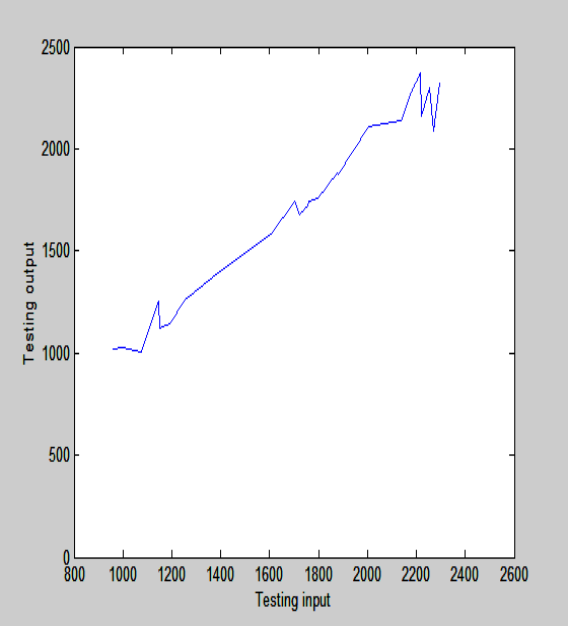


(a) Training graph **(b) Testing graph**

Figure 5.20: Training and testing graph for number of inputs 5, number of wavelets 30, and 3 iterations

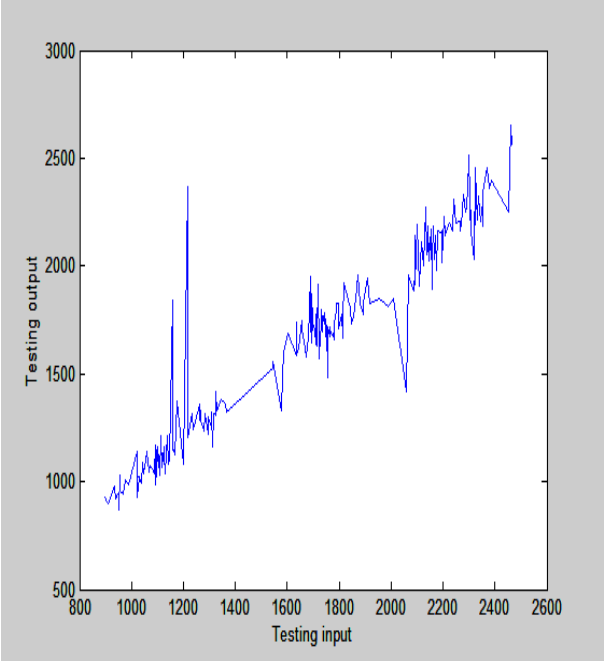


(a) Training graph

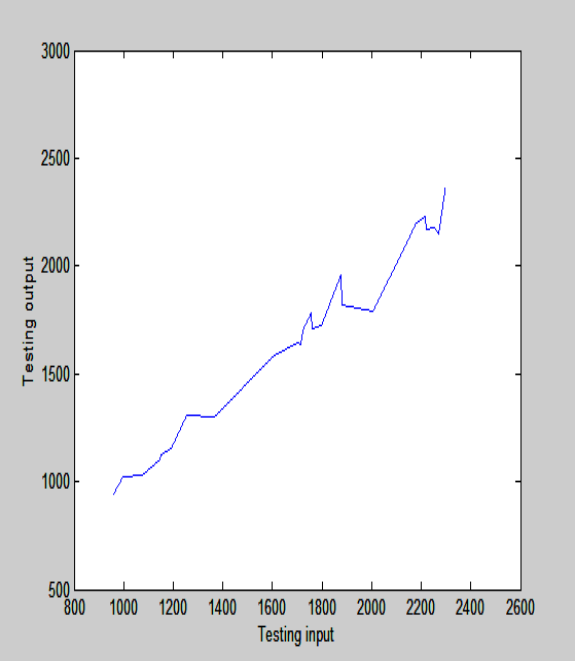


(b) Testing graph

Figure 5.21: Training and testing graph for number of inputs 5, number of wavelets 30, and 4 iterations

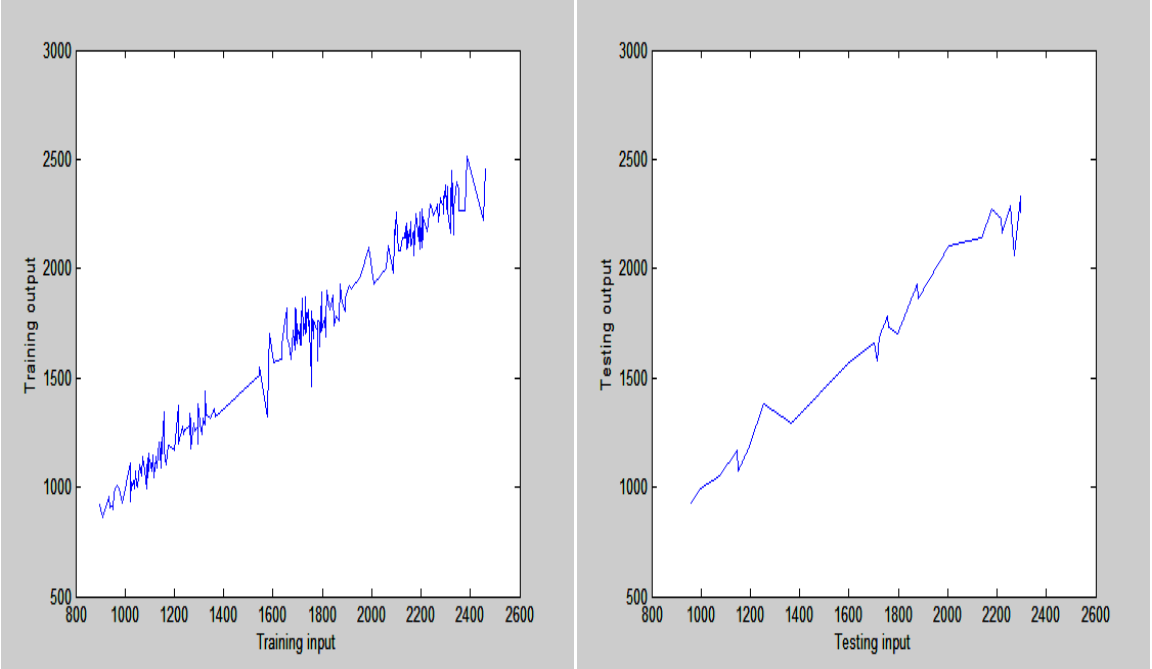


(a) Training graph



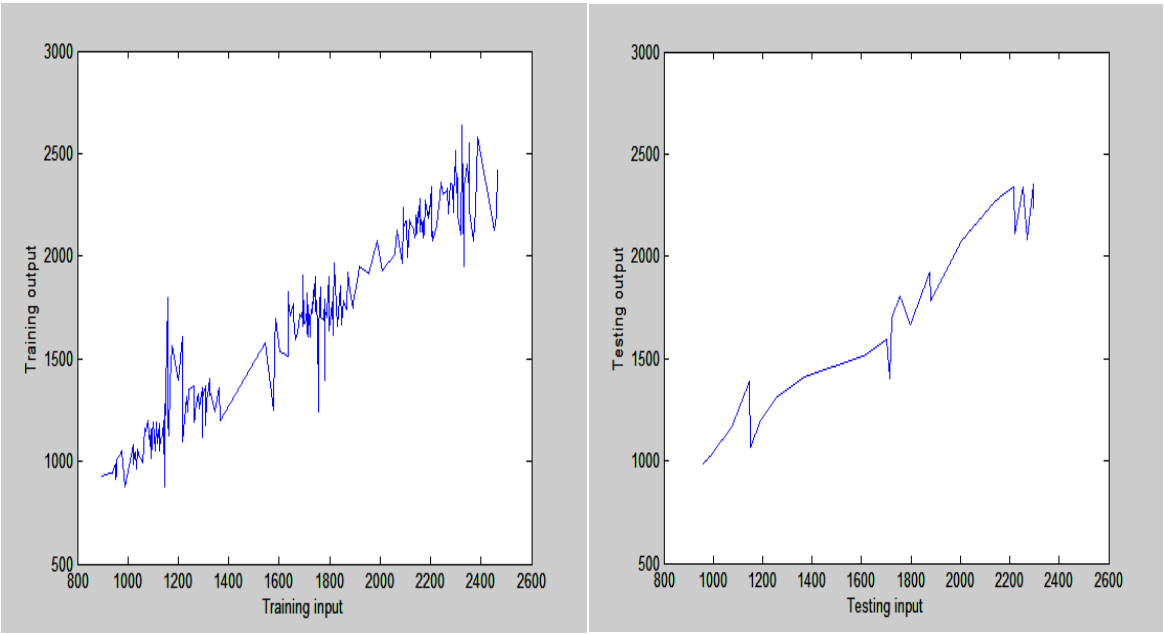
(b) Testing graph

Figure 5.22: Training and testing graph for number of inputs 5, number of wavelets 30, and 5 iterations



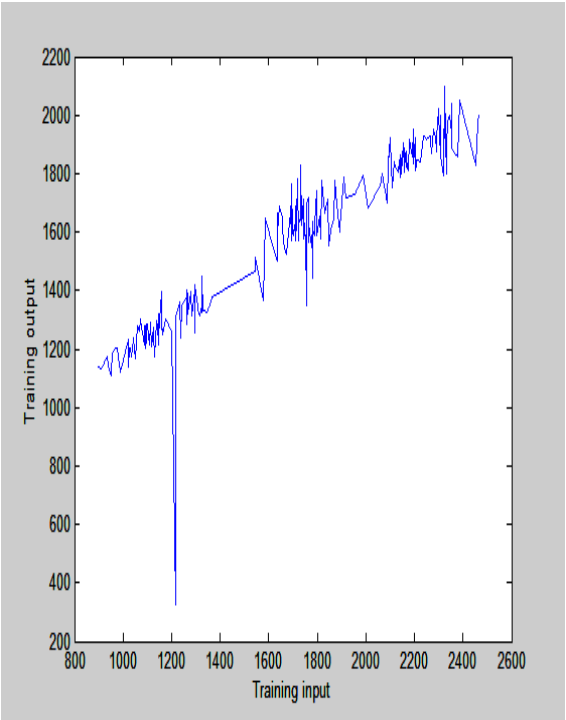
(a) Training graph **(b) Testing graph**

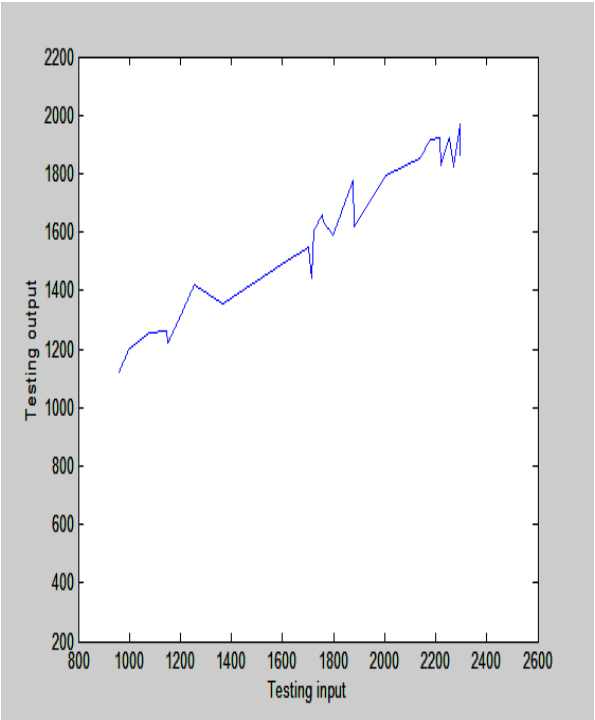
Figure 5.23: Training and testing graph for number of inputs 5, number of wavelets 40, and 3 iterations



(a) Training graph **(b) Testing graph**

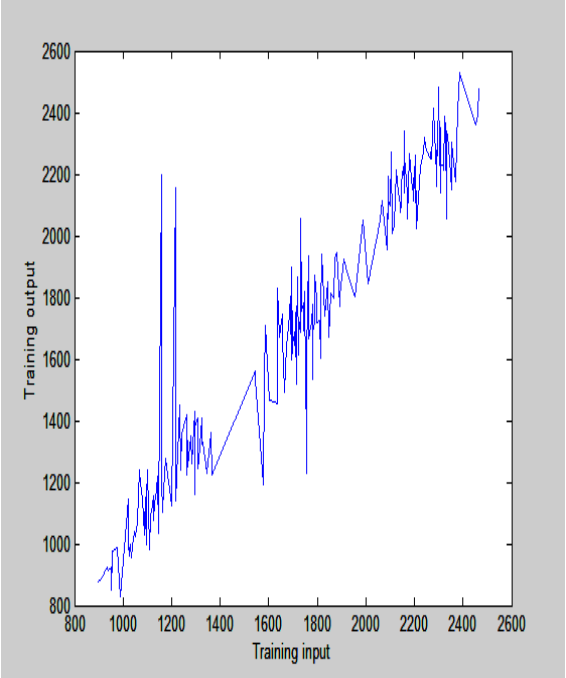
Figure 5.24: Training and testing graph for number of inputs 5, number of wavelets 40, and 4 iterations

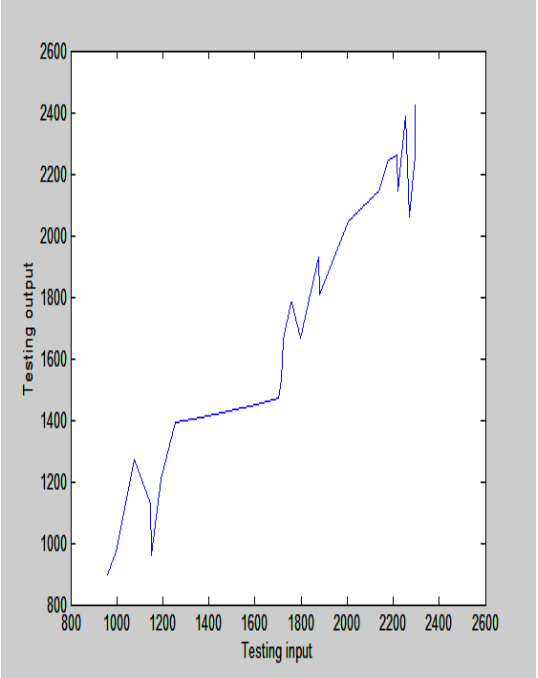




(a) Training graph **(b) Testing graph**

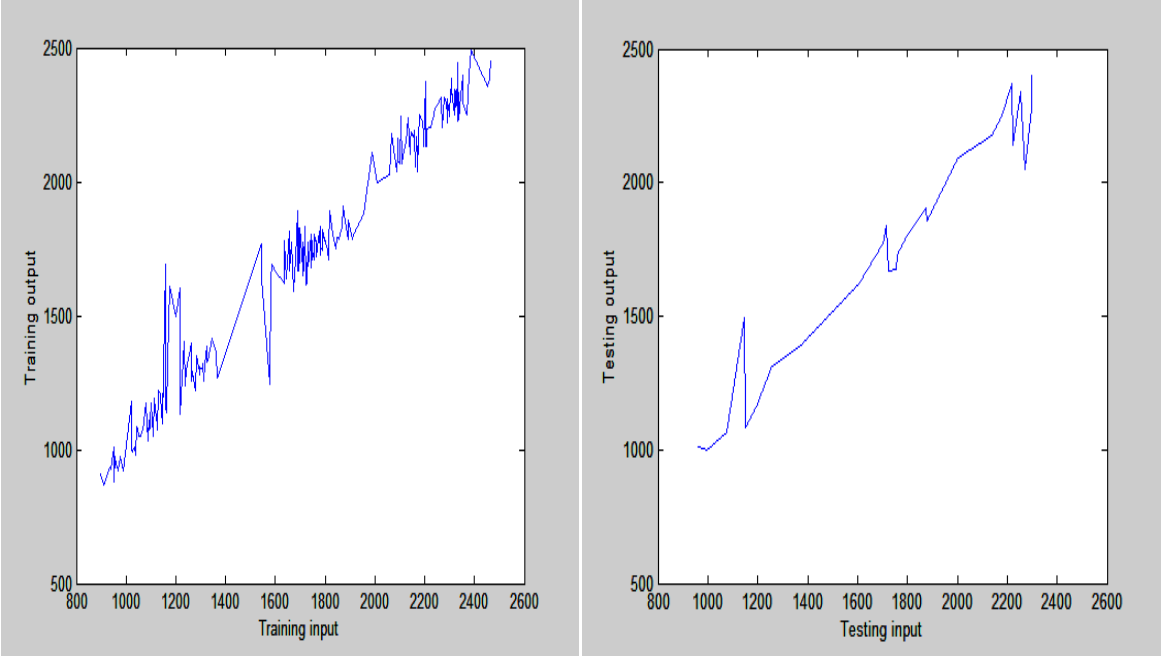
Figure 5.25: Training and testing graph for number of inputs 5, number of wavelets 40, and 5 iterations





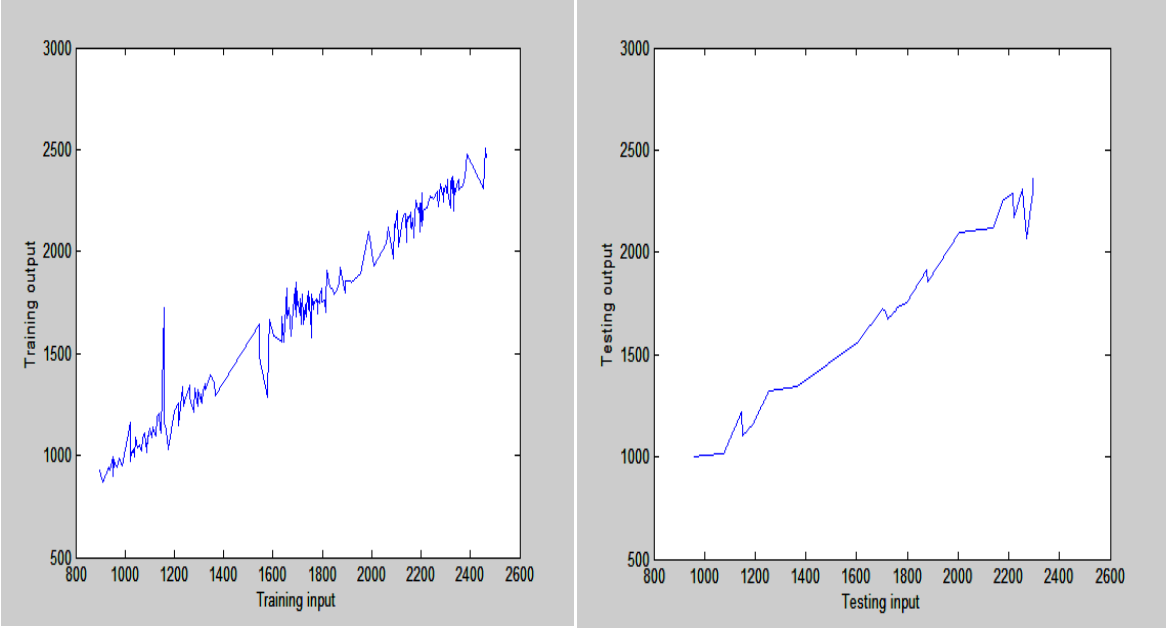
(a) Training graph **(b) Testing graph**

Figure 5.26: Training and testing graph for number of inputs 5, number of wavelets 50, and 3 iterations



(a) Training graph **(b) Testing graph**

Figure 5.27: Training and testing graph for number of inputs 5, number of wavelets 50, and 4 iteration



(a) Training graph **(b) Testing graph**

Figure 5.28: Training and testing graph for number of inputs 5, number of wavelets 50, and 5 iterations

The results of stimulation of the network clearly show that the algorithm given in MATLAB is very good network for forecasting of stock index of SBI. After comparing all the results of different parameters we observe that when we take 5 inputs and 30 wavelets in each layer and 3 iterations then we get a network which gives us better results.

Chapter 6

Conclusion and Future Scope

Nowadays, the area of applications of artificial neural network and wavelets is getting very vast. We have seen that these are very important tools in the forecasting and betterment of distorted and chaotic signals. When we combine the artificial neural networks and wavelets, into wave-nets, it becomes even a more powerful tool for the forecasting and betterment of distorted and chaotic signals. In artificial neural network we have weights in the hidden layer, which are trained to get the results, but in wave-nets we use daughter wavelets of the mother wavelet as weights in the hidden layer. As wavelet is the effective tool in betterment of the non-stationary signals, the wave-nets become a more powerful tool in the forecasting of the data. The area of applications of wave-net is not limited with the forecasting and betterment of signal but we can also see its application in other areas.

We have used the AWNN and its learning algorithm to emulate the time-series forecasting for the stock index of SBI. In the proposed mode of learning, we have trained the AWNN and the simulation results justify that the algorithm is computationally effective.

In order to implement AWNN, we have developed a program in MATLAB. There are three parameters of the network, namely, the number of inputs, the number of wavelets in second layer and the number of iterations. The number of iterations is the number of times the procedure of training is repeated. We have taken 27 different networks for the forecasting of the stock index of SBI. The results of simulation of these networks are given in the Fig. 5.2 to Fig. 5.28. In these figures, the output from AWNN has been plotted as a function of actual time series data. All the graphs here follow a similar trend, although not perfect, indicating the effectiveness of the learning process. After comparing the results of different parameter combinations, we have observed that when we take 5 inputs and 30 wavelets in second layer and perform 3 iterations, then we get a network which gives us best results.

Here are few observations which we would like to highlight for the future work in this direction.

In our work, we have taken 261 values of the time-series data of stock index of SBI for training and testing of AWNN. The data which we have used is from 1st January 2009 to 31st January 2010. One can take a larger time series data, i.e., one can used the data of last 5 years or last 10 years for training and testing of the AWNN.

As mentioned earlier, we have taken 3 parameters, namely, the number of inputs, the number of wavelets in second layer, and the number of iterations for the training of AWNN. We have tried 27 combinations to study the network. One can try some other combinations which can give more effective results for the purpose of SBI stock index forecasting.

Also, we have used Mexican Hat wavelets in the second later of AWNN, as the mother wavelet to produce the daughter wavelets. We can experiment to use some other mother wavelet whose daughter wavelets can be used in the second layer. We have mentioned some other mother wavelets which can be used in the AWNN network, namely, SLOG wavelets, Morlet wavelets and POLYWOG wavelets in Chapter 3.

In the proposed work, we have only used the time-series data of stock index of SBI for the training and testing of the AWNN network, to forecast the value of stock index. The results show the effectiveness of the AWNN technique and we can apply this technique in forecasting of stock index of other banks and also other industries.

In the proposed learning, we have used a fixed time-series of the data for the learning and testing of AWNN. One can implement a network which will have the on-line mode of learning. The network may take the current value of stock index and then forecast the next value of stock index in real-time.

References

[1] Bodyanskiy Y., Pliss I. and Vynokurova O., *Adaptive Wavelet- Neuro Network in the Forecasting and Emulation task*, International Journal of Information Theories & Applications, 15, 2008, 47-54.

[2] Jang J., *ANFIS Adaptive Network – Based fuzzy inference systems*, IEEE Transactions on Systems, Man and Cybernetics, 23, 1993, 665-685.

[3] Hartley H., *The modified Gauss-Newton method for the fitting of nonlinear regression functions of least-squares*, Techno metrics, 3, 1961, 269-280.

[4] Marquardt D., *An algorithm for least squares estimation of non-linear parameters*, SIAM Journal of Applied Mathematics, 11, 1963, 431-441.

[5] Ljung L., *System identification: Theory for the User*, PTR Prentice Hall, Upper Saddle River, N.J., 1999.

[6] Zhang J., Walter G.G. and Miao Y., *Wavelet neural networks for function learning*, IEEE Transactions Signal Processing, 43(6), 1995, 1485-1496.

[7] Kaczmarz S., *Approximate solution of systems of linear equation*, International Journal of Control, 53, 1993, 1269-1271.

[8] Goodwin Y.C., Ramadge P.J. and Canines P.E., *A globally convergent adaptive predictor*, Automatic, 17, 1981, 135-140.

[9] Laurane F., *Fundamentals of Neural Networks, Architectures, Algorithms and Applications*, Prentice Hall, Englewood Cliffs, 1994.

[10] Rajasekaran S. and Vijyalakshmi Pai G.A., *Neural networks, Fuzzy Logic and Genetic Algorithms: Synthesis and applications*, PHI Learning Private Limited, New Delhi, 2003.

[11] Lippman R.P., *Introduction to computing with Neural Nets*, IEEE Computer, 21(3), 1987, 105-17.

[12] Minsky M., and Papert S., *Perceptrons*, MIT Press, Cambridge, MA, 1969.

[13] Tom Mitchell M., *Machine learning*, WCB-McGraw- Hill, 1997, ISBN 0-07-042807-7.

[14] Chui C.K., *An Introduction to Wavelets*, New York: Academic. 1992, 264.

[15] Daubechies I., *Ten Lectures on Wavelets*, Philadelphia, PENNSYLVANIA, SIAM, 1992.

[16] Zhang Q.H., and Benveniste A., *Wavelet networks*, IEEE Transactions on Neural Networks, 3(6), 1992, 889–898.

[17] Ikonomopoulos A. and Endou A., *Wavelet decomposition and radial basis function networks for system monitoring*, IEEE Transactions on Nuclear Sciences, 45(5), 1998, 2293-2301.

[18] Jiao L., Pan J. and Fang Y., *Multiwavelet neural network and its approximation properties*, IEEE Transactions on Neural Networks, 12(5), 2001, 1060-1066.

[19] Poggio T. and Girosi F., *Networks for approximation and learning*, Proceedings of IEEE, 78(9), 1990, 1481-1497.

[20] Zhang Q., *Using wavelet network in non-parameters estimation*, IEEE Transactions on Neural Networks, 8(2), 1997, 227-236.

[21] Suresh babu N. and Farrell J.A., *Wavelet-Based System Identification for Non linear Control*, IEEE Transactions on Automatic Control, 44(2), 1999, 412-417.

[22] Meyer Y., *Wavelets: Algorithms and Applications*, Philadelphia, P.A, SIAM, 1993.

[23] Billings S.A., Wei H.L, *A new class of wavelet networks for nonlinear system identification*, IEEE Transactions on Neural Networks, 16(4), 2005, 862-874.

[24] Oussar Y., Dreyfus G., Initialization by selection for wavelet network training, Neurocomputing, 34, 2000, 131-143.

[25] Narendra K.S., Parthasarathy K., *Identification and control of dynamic systems using neural networks*, IEEE Transactions on Neural Network, 1990, 1(1), 4-26.