FAULT IDENTIFICATION USING NEURAL NETWORKS

A DISSERTATION

Submitted in partial fulfillment of the requirements for the award of the degree

of

MASTER OF TECHNOLOGY

in

ELECTRICAL ENGINEERING

(With Specialization in System Engineering and Operations Research)

By

PRAVEEN RANGISETTI

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
ROORKEE - 247 667 (INDIA)

JUNE, 2007

I.D. No.-M.T. 401/2007-34/GNP

CANDIDATE'S DECLARATION

I hereby declare that the work that is being presented in this dissertation report entitled " **FAULT IDENTIFICATION USING NEURAL NETWORKS**" submitted in partial fulfillment of the requirements for the award of the degree of **Master Of Technology** with specialization in **System Engineering and Operations Research,** to the **Department Of Electrical Engineering, Indian Institute Of Technology, Roorkee**, is an authentic record of my own work carried out, under the guidance of **Dr. G. N. Pillai,** Assistant Professor, Department of Electrical Engineering.

The matter embodied in this dissertation report has not been submitted by me for the Award of any other degree or diploma.

Date: 29-06-2007

Place: Roorkee

R-Praveen

(PRAVEEN RANGISETTI)

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

[illegible] 16/6/07

(Dr. G. N. Pillai)
Assistant Professor,
Department of Electrical Engineering,
Indian Institute of Technology,
ROORKEE – 247 667,
INDIA.

ACKNOWLEDGEMENTS

I am very indebted to my institution, **Indian Institute of Technology, Roorkee** for providing me opportunity to pursue my Masters.

Dr. G.N.Pillai, Professor, Department of Electrical Engineering, IIT Roorkee, my guide is the first person behind the success of this dissertation report. I frankly agreed that this work evolved through his support.

I am left indebted to all my friends, for letting their lives run beside mine for a while. They had always been a cheerful company to me. My gratitude to them cannot be expressed in words.

I thank each and every one heart fully for helping me in completing my dissertation report.

Dedicated

To

My Parents

ABSTRACT

In this thesis the aim is to detect the high impedance fault occurring on radial electrical distribution systems using neural network based relaying scheme. A multilayer perceptron is used for distinguishing the linear and nonlinear High Impedance Faults by taking the Feature vector as input. R.M.S values of third and fifth harmonic components of feeder voltage and feeder current are used as the feature vector obtained by applying the Fast Fourier Transform on the Feeder voltage and Feeder current.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

For Generator, Transmission Line and Load Model:

Vs = Source voltage

Ls = Source inductance

(Rs) = Source resistance

x*R_line = Resistance of the line upto fault point

x*L_line = Inductance of the line upto fault point

x*C_line= Capacitance of the line upto fault point

Lf = Fault inductance

Rf = fault resistance

(l-x)*R_line = Resistance of the line beyond fault point and upto load.

(l-x)*L_line = Inductance of the line beyond fault point and upto load.

(l-x)*C_line= Capacitance of the line beyond fault point and upto load.

R_load = Load resistance

L_load = Inductance of load

Is = Source current

If = Fault current

Il = Load current

Vf = Voltage at fault point

Va or V1 = Voltage at line length X

Vb or Vl = Load Voltage

v3= Third harmonic component of feeder voltage

v5 = Fifth harmonic component of feeder voltage

i3 = Third harmonic component of feeder current

i5 = Fifth harmonic component of feeder current

a = v3 * i3/p3

b = v5 * i5/p5

c = i3/v3

d = i5/v5

For Perceptron Neuron Model:

T or Z = Target of the network

O or y = Output of the network

e = T-y = Error

$X_{P1,}X_{P2,}.........X_{PN0}$ = P th Pattern fed as input to all the N_0 neuron in input layer.

$T_{P1,}T_{P2,}.........T_{PNM}$ = Target of the network i.e. Expected output from all the Nm neuron in output layer.

$O_{P1,}O_{P2,}.........O_{PNM}$ = Actual output from all the Nm neuron in output layer.

Y_{ji} = Output from the i th neuron in layer j for p th pattern

W_{jik} = Connection weight from k th neuron in layer j-1 to i th neuron in layer j

δ_{ji} = Error value associated with the i th neuron in layer j.

α or β or η = Learning rate

W^{new} or W^{1} or X_{k+1} = New weight obtained after addition of change in weight

E_P = Root Mean Square (RMS) of the errors in the output layer

Net= output of neuron before application of transfer function

δy = Product of error and weight propagating backward to hidden layers from output layer

df(e)/de or g = Gradient

CHAPTER-I INTRODUCTION

1.1 Introduction to High Impedance Fault

High impedance fault can be described as the fault which doesn't draw sufficient fault current to be recognized and cleared by over current devices. High impedance fault is generally occurred when a current carrying wire or conductor makes a non solid or temporary contact with the ground or temporarily short circuited with another current carrying conductor through a high resistance material. E.g.: A cable lying and touching the ground and a tree branch touching the two power cables are the examples for high impedance fault [1]. It's estimated that one-third to one-half of downed conductor faults are high impedance faults. High impedance faults are safety hazard to humans, live stocks, and electric utility persons. If left undetected for hours, weeks and months they can be fatal to humans and animals at a typical current level of 50 milli ampere or above [4]. In some cases the high impedance fault could transform into a fire hazard if the fault media are a building or a dc trolley in a coal mine [5]. Typical electrical faults on distribution network will be detected by conventional over current or ground relays which operate on the solid state comparator or electromechanical principle. But unfortunately these relays are not being able to detect the high impedance fault as the fault current generated due to the high impedance fault is far below the set point level of the over current relay. This necessitates the development of alternative methods for the detection of high impedance faults [6]. Up to two decades before the detection of high impedance fault was being an impossible task but due to advancements in artificial intelligence technology especially Artificial Neural Networks and due to the discovery of fast and efficient Fourier transform using the feature vector as input made the detection of high impedance fault possible [7].

High impedance faults are characterized by their low fault current levels and waveform distortion due to nonlinearity of ground return path due to arcing, soil fusing and time variations in the ground resistance [8]. This nonlinearity produces particular patterns of frequency spectral distribution in phase voltages and currents which are indicative of high impedance fault. These harmonic spectral variations are reflected back at the substation in varying magnitudes and phase shifts depending on type of fault and its location [9]. High impedance faults in electrical distribution networks also exhibit ripple type frequencies and noise patterns as well as abnormal sub, super and inter harmonics due to ground resistance variations, soil resistance nonlinearity and arcing [8-12]. Most of the present detection schemes are based on conventional over current protection, ground relays based on ratio of neutral current to fundamental positive sequence currents , harmonic identification using one or more characterizing harmonic signals and ripple detection based on high frequency power spectra in the frequency range (2 - 10 kHz)[6]. New ideas start emerging to utilize artificial neural networks as a robust detection scheme. ANN based detection provides nonlinear mapping from input diagnostic vector to the output outcome of fault classification.

1.2 Literature Review:

B.M. Aucoin , B. Don Russell [1] from Texas instruments discussed loop holes in Warrington's ground fault detector which used increase in the magnitude of fundamental harmonic. He revealed that this approach can't work practically as the normal switching events like feeder switching, capacitor bank switching also produce large increase in lower order harmonics and these will falsely identify the load rich in harmonics as high impedance fault, and they have proposed that high impedance faults exhibit marked increase in the high frequency current components over normal system conditions which persist for the entire duration of the arc, and for his experimentation on high impedance faults he considered the high frequency current components of frequency greater than 2 KHz , rather than taking low frequency components which vary widely under different loads. This report became very valuable for the later research on High impedance fault detection.

Later Emanuel, A.E., Cyganski, D., Orr, J.A., Shiller, S.,Gulachcnski, E.M., [2] from their experimentation work proposed that if we consider the harmonics of very high frequency it's becoming difficult to guess the fault as their magnitude is very low , and suggested that it's better to consider the Second and Third harmonic currents as their magnitude is also considerable for detection of the fault and as well as the requirement of much variation in magnitude through out the duration of arc for the detection of high impedance fault will also be fulfilled if second and third harmonics are considered.

Aucoin., B. Michael, and Jones., Robert H.,[3] considered the implementation issues of High impedance fault detection procedures. In this paper they discussed background research on this topic. They revealed that at the beginning research is concentrated on developing a sensitive detector to detect and clear faults; most cost effective approach is a substation based detector operating on electrical parameters using existing transducers and interrupters. One detector is to be incorporated in each feeder, this causes tripping of entire feeder during fault. Later a microcomputer based technique was developed but none of these were being able to distinguish high impedance fault from other faults. They gave some examples for fault and hazardous condition. An electrical fault is a condition for which an energized conductor contacts another conductor or a grounded object. An electrical hazard is a condition in which electrocution or fire may readily occur as the conductor is near ground or an object or with in reach of public. A hazardous condition can be one in which no fault is present. An intact primary conductor fallen off insulators hanging one meter above the ground is not a fault but hazardous, since it will not convey any electrical information upon which action can be taken. Conversely a circuit may be faulted but may not be a hazard. e.g.: A tree limb in contact with an intact overhead primary and a conductor on ground in a remote location may not be a threat though it's a fault. One will expect to have high impedance fault detectors on every feeder to have at most safety, but it's not practically possible. So they have classified few areas for locating the HIFD on each feeder, they are urban and suburban feeders with high population density; feeders with small conductor which is more likely to break; feeders with prior history of downed conductor trouble; areas with heavy tree growth, very dry areas or feeders with numerous , lengthy single phase laterals. They suggested the operating time of HIFD as 15 to 60 sec. they also told that it's necessary to bring awareness in the people as most of them can't distinguish among power, telephone and cable TV lines. Later many researchers started using different harmonic components from fundamental to fifth harmonic.

A.M. Sharaf , L.A. Snider , K. Debnath [4] obtained positive, negative and zero sequence components of third harmonic component voltage and current values. They have trained the neural network using these components and succeeded in well detection of the high impedance fault compared to his predecessors.

L. A. Snider, Yuen Yee Shan [5-7] proposed that many previous methods developed based on DSP can't work well in the detection of high impedance fault as the data available is very less, and he suggested that ANN is the best one to use because it can distinguish the fault well if we train it using the data available due to its pattern recognition capability.

Adel M. Sharaf ,Guosheng Wang [9-10] in their work have considered four types of fault conditions i.e. Linear fault, Nonlinear fault, Bolted fault , No fault and simulated these faults at different lengths of line , different source and fault impedances. They observed some common patterns in each fault and proposed some patterns to distinguish the fault occurred based on those patterns of waveforms obtained.

L.A. Snider [5-7] in all his papers modeled the high impedance fault by a diode model. Sharaf A.M. used no modeling for the fault. He just modeled the transmission line using nominal pi model. All most all the researchers using ANN have used Back propagation algorithm for it's simplicity.

A.M. Sharaf, L.A. Snider, K. Debnath [12] used negative and zero sequence components of second, third, fifth harmonic components for training the neural network.

T.M. Lai, L. A. Snider,E. Lo, D.Sutanto [13-14] developed a High impedance fault detection technique based on Discrete Wavelet transform. But this can't determine the properties of output coefficients.
M. M. Eissa, G.MA. Sowilam, A.M. Sharaf [15] used third and fifth harmonic components of feeder voltage and current obtained by applying FFT for training the neural network. They have also specified some patterns observed in their simulation of fault at different locations of line. This method is followed in this thesis for obtaining the feature vector which will be used for training the neural network.
Howard Demuth, Mark Beale [16] discussed different transfer functions, back propagation algorithms.

1.3 Organization of Thesis:

Chapter 1 discusses the introduction and literature review of the entire thesis. Chapter 2 discusses modeling and simulation of High Impedance Fault and the way of distinguishing the linear and nonlinear faults by observing few patterns is revealed. Chapter 3 discusses the weight changing methodology by Back propagation algorithm and this is represented with diagrams. The transfer functions are represented with diagrams. Chapter 4 gives all the results of thesis. The conclusions are given in chapter 5. The data obtained by simulation and the parameters used for simulation are given in Appendix.

CHAPTER-2 HIGH IMPEDANCE FAULT SIMULATION

Introduction:

High impedance fault can be described as the fault which doesn't draw sufficient fault current to be recognized and cleared by over current devices. If left undetected for hours, weeks and months they can be fatal to humans and animals at a typical current level of 50 milli ampere or above. The description of this fault is described below with help of a radial distribution model of the power system.

2.1 Single Line Diagram Representation of HIF System

The single line diagram of sample system used for the study of high impedance fault detection is shown in figure 2.1 below.

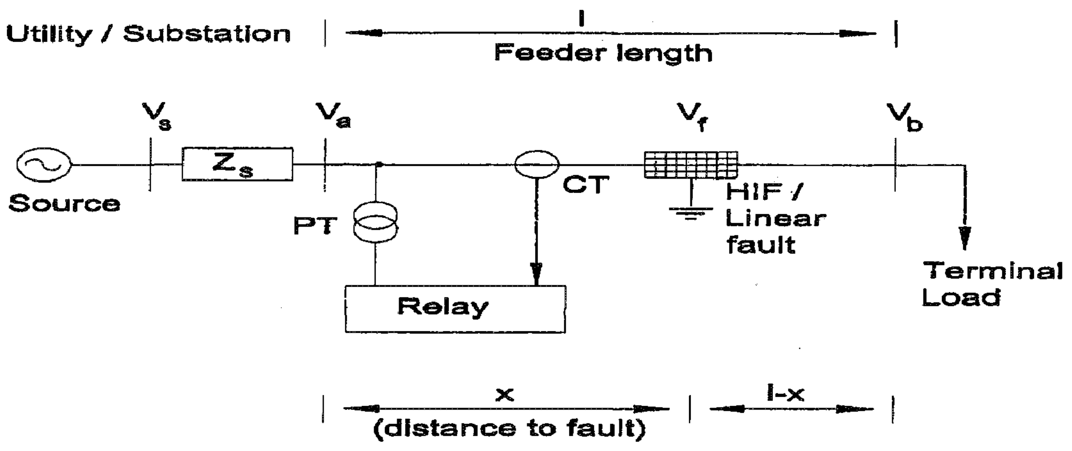


Figure 2.1: Single line diagram of system subjected to high impedance fault [15]

Figure 2.1 Shows a generator connected to load by a transmission line in which an ordinary over current relay is placed.

Where Vs = Source voltage.

Vf = Voltage at fault point.

Vb = Load voltage.

Va= Voltage generated after excluding the drop in source.

l = Feeder length

x= Distance of fault location from the source point.

2.2 Per Phase Equivalent Circuit of HIF Model

The per-phase equivalent circuit of the above system is modeled as shown below in figure 2.2. From figure 2.2 it's obvious that the transmission line is modeled using nominal pi model which will be used generally for medium length transmission lines.

Where

Vs = Source voltage

Ls = Source inductance

(Rs) = Source resistance

x*R_line = Resistance of the line upto fault point

x*L_line = Inductance of the line upto fault point

x*C_line= Capacitance of the line upto fault point

Lf = Fault inductance

Rf = fault resistance

(l-x)*R_line = Resistance of the line beyond fault point and upto load.

(l-x)*L_line = Inductance of the line beyond fault point and upto load.

(l-x)*C_line= Capacitance of the line beyond fault point and upto load.

R_load = Load resistance

L_load = Inductance of load

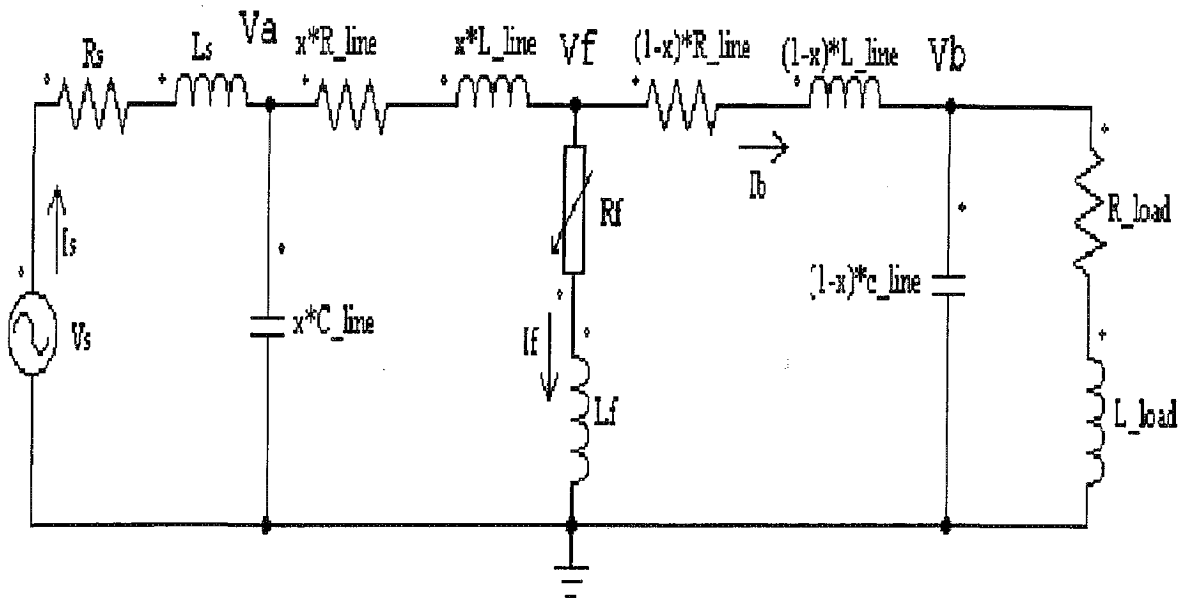


Figure 2.2: Per phase equivalent circuit of high impedance fault model [15]

In general the high impedance fault can be linear or nonlinear. In case of linear fault the fault resistance is constant and is not a function of any parameter.

In case of non linear fault the fault resistance is a function of current and is given by Rf = Rf0 (1+α (If/If0) ^ β) where α and β are constants.

2.3. Simulation Diagram of HIF Model

The simulation diagram is developed for fault from figure 2.2 and it is shown in figure 2.3 given below.

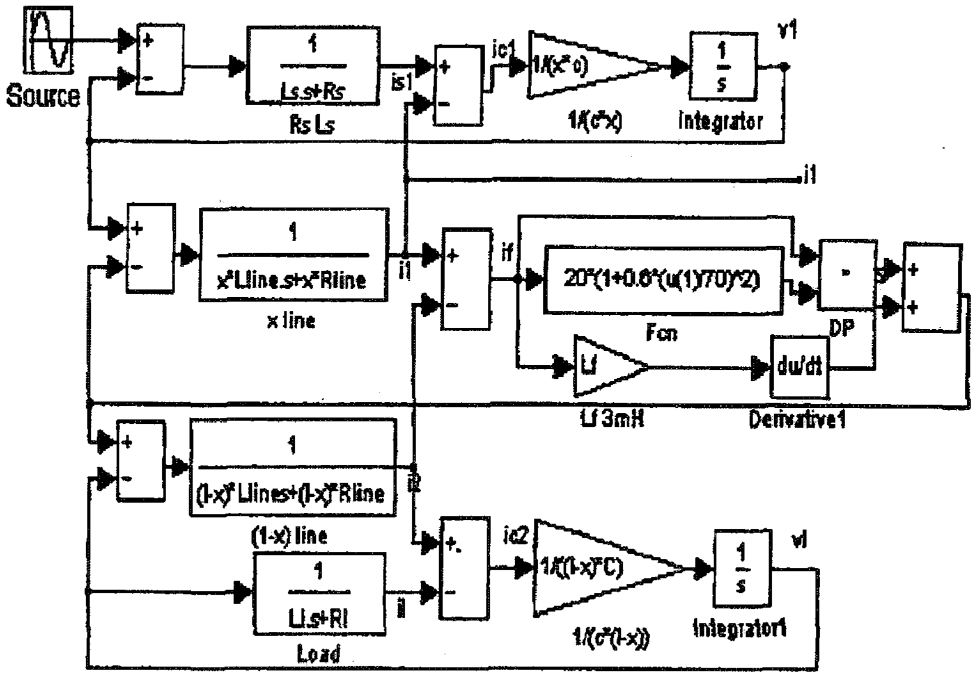


Figure2. 3: Simulation diagram used to simulate the non linear fault

The diagram shown above is obtained by modeling the per-phase equivalent circuit shown in figure 2.2 by nodal analysis. The equations used in the modeling of above diagram are shown below.

Source current (Is) = $\dfrac{Vs - Va}{Rs + sLs}$

Fault current (If) = Ia – Ib

where Ia = $\dfrac{\text{Va- Vf}}{\text{line impedance upto fault point}} = \dfrac{\text{Va- Vf}}{(\text{x*L\_line})s+\text{x*R\_line}}$

and Ib = $\dfrac{\text{Vf- Vl}}{\text{line impedance upto the end of line from fault point}}$

$= \dfrac{\text{Vf- Vl}}{((\text{l-x})\text{*L\_line})\text{s} + (\text{l-x})\text{*R\_line}}$

Load current (Il) = $\dfrac{\text{Vl}}{\text{load impedance}} = \dfrac{\text{Vl}}{\text{R\_load} + (\text{L\_load})\text{s}}$

Voltage at fault point (Vf) = (If) * (Rf + Lf*$\dfrac{\text{dIf}}{\text{dt}}$)

Current flowing through capacitor x* C_line = Source current (Is) – Current flowing through line before fault point (Ia)

Current flowing through capacitor (l-x)* C_line = Current flowing through line beyond fault point (Ib) – Load current (Il)

Va = {Current flowing through capacitor x* C_line} * {Impedance of capacitor x* C_line } = (Is- Ia)*$\dfrac{1}{\text{s(x*C\_line)}}$

Vb= {Current flowing through capacitor (1-x)* C_line} * {Impedance of capacitor (1-x)* C_line } = (Ib- Il)*$\dfrac{1}{\text{s((1-x)*C\_line)}}$

[Note: The suffixes a (or) 1 and b (or) l indicate the same]

2.4. High Impedance Fault Pattern Characteristics

Here we have obtained voltage and current at the feeder. By pattern recognition method it's concluded that the plot between v1 and i1 is skewed banana in case of nonlinear fault as shown in figure 2.4 below.

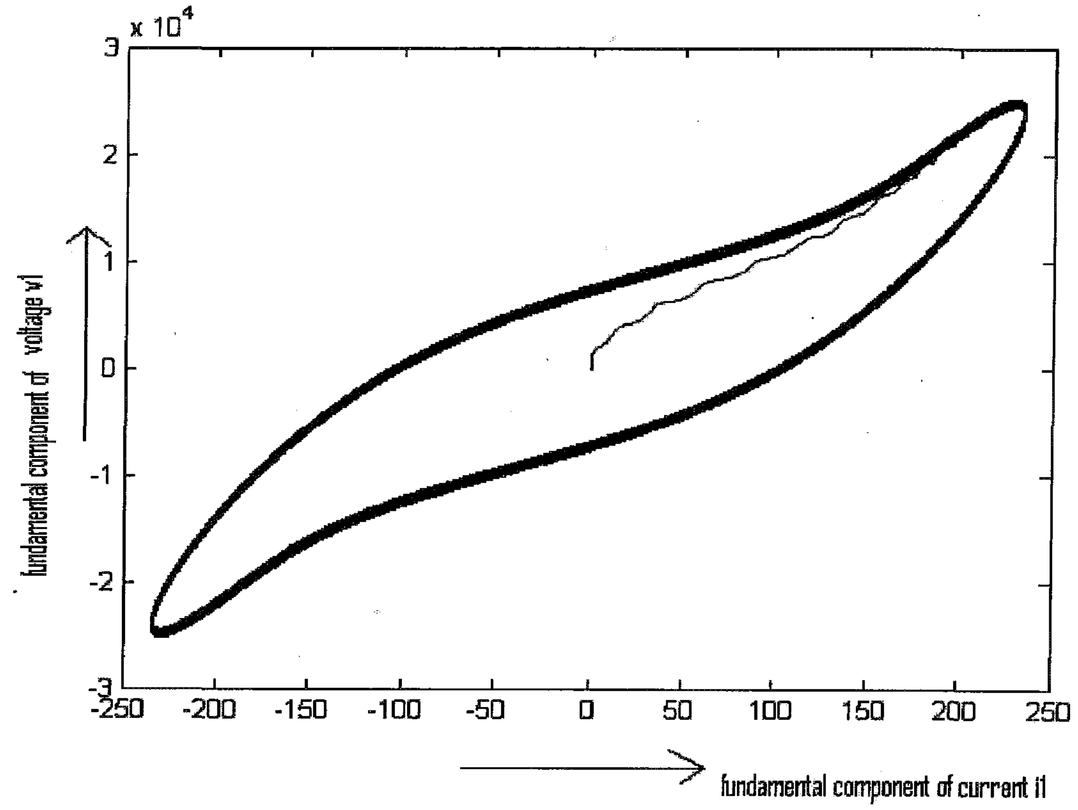


Figure 2.4 Plot of v1 vs i1 in case of nonlinear fault at x=0.05

3rd and 5th harmonic components of these fundamental values can be found out by applying the one cycle Fast Fourier transform. This is represented in the figure 2.5 given below. These harmonic components exhibit certain pattern recognition characters using which the Linear and Non linear High impedance faults can be distinguished. It's observed that the plot of a vs b is collapsed lines, plot of p3 vs p5 is collapsed and retraced lines; plot of c vs d is triangle shaped lines in case of non- linear fault.

Where $a = v3 * i3/p3$

$b = v5 * i5/p5$

$c = i3/v3$

$d = i5/v5$

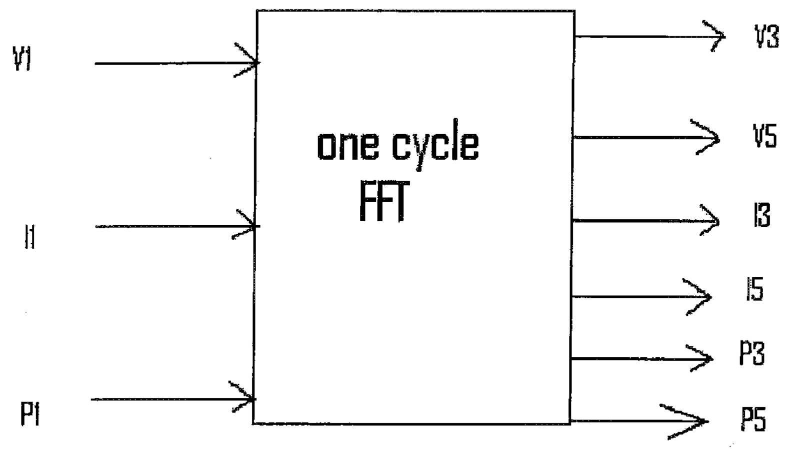


Figure 2.5: plot representing harmonic extraction using one cycle FFT

The plot of b vs a for nonlinear fault is shown in figure 2.6 given below.

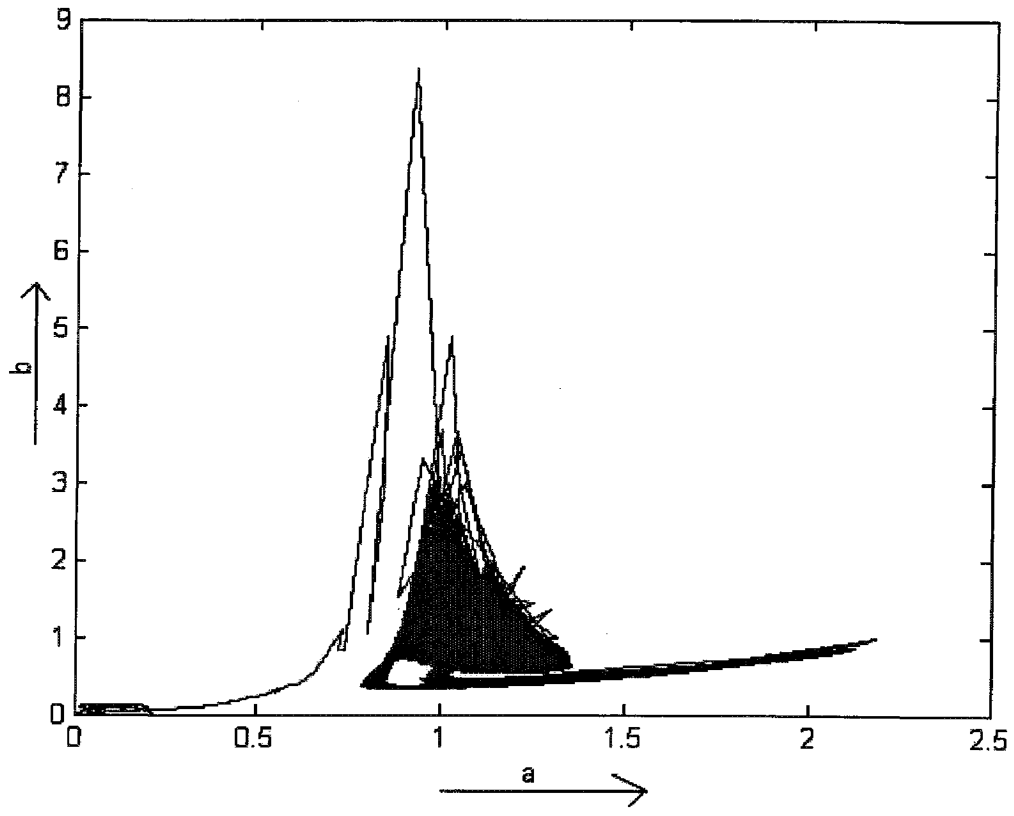


Figure 2.6: plot of b vs a in case of non linear high impedance fault

The plot of d vs c for nonlinear fault is shown in figure 2.7 given below.

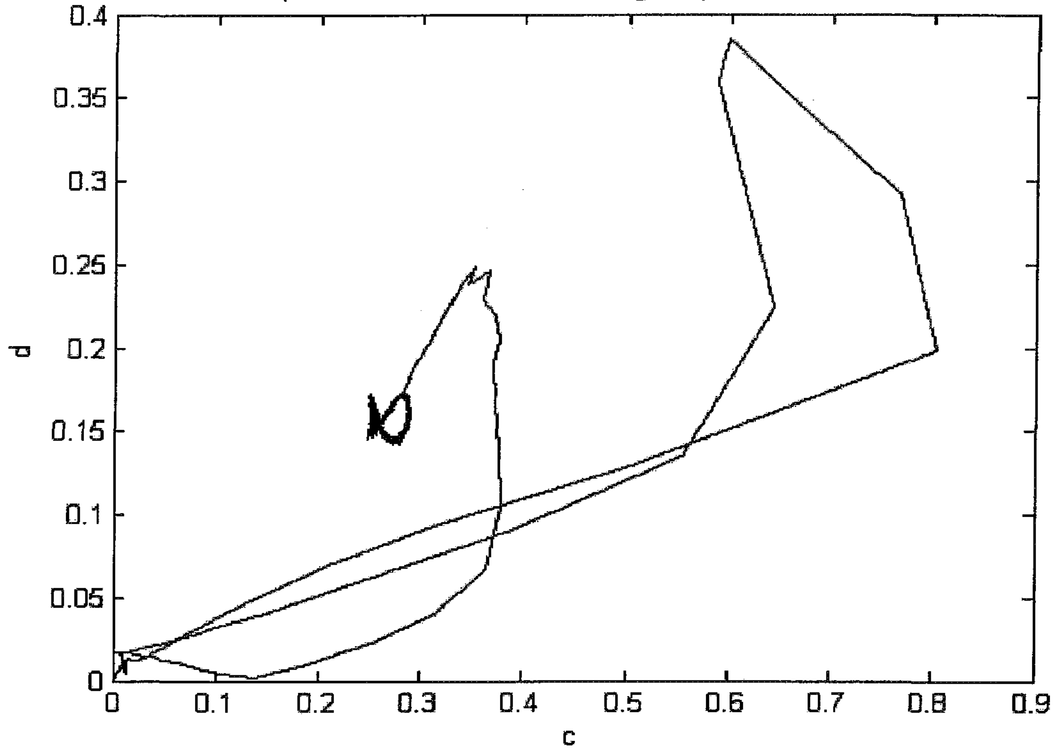


Figure 2.7: plot of d vs c in case of non linear high impedance fault

The plot of p5 vs p3 for nonlinear fault is shown in figure 2.8 given below.

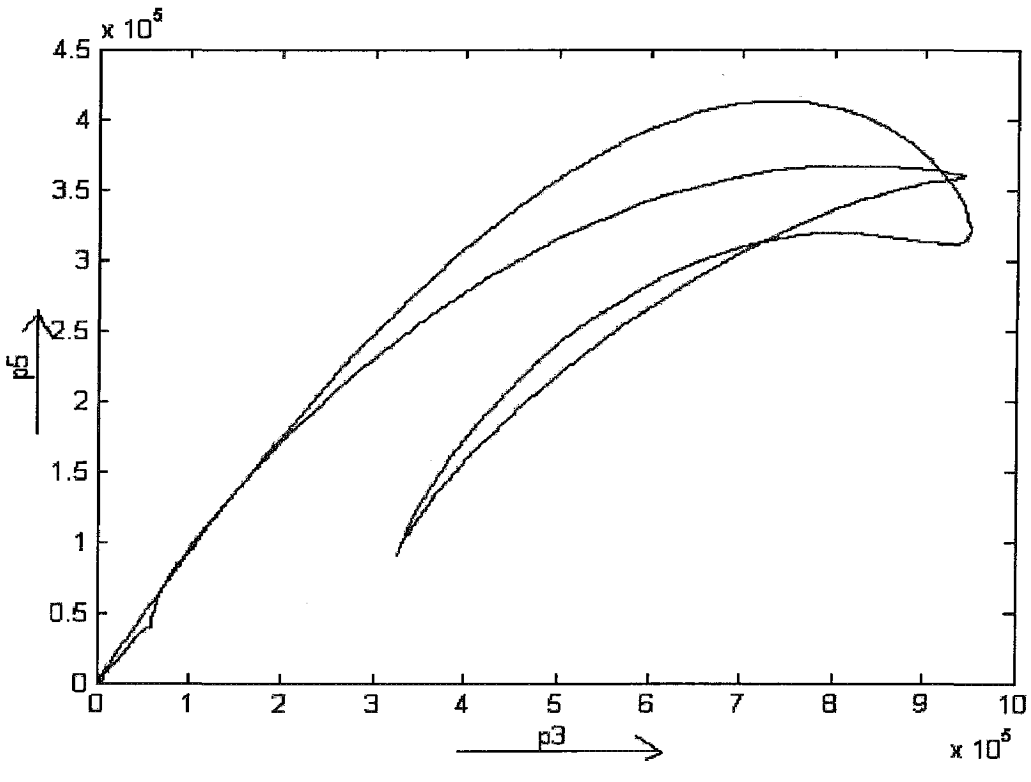


Figure 2.8: plot of p5 vs p3 in case of non linear high impedance fault

By pattern recognition method it's concluded that the plot between v1 and i1 is ellipse in case of linear fault as shown in figure 2.9 below.

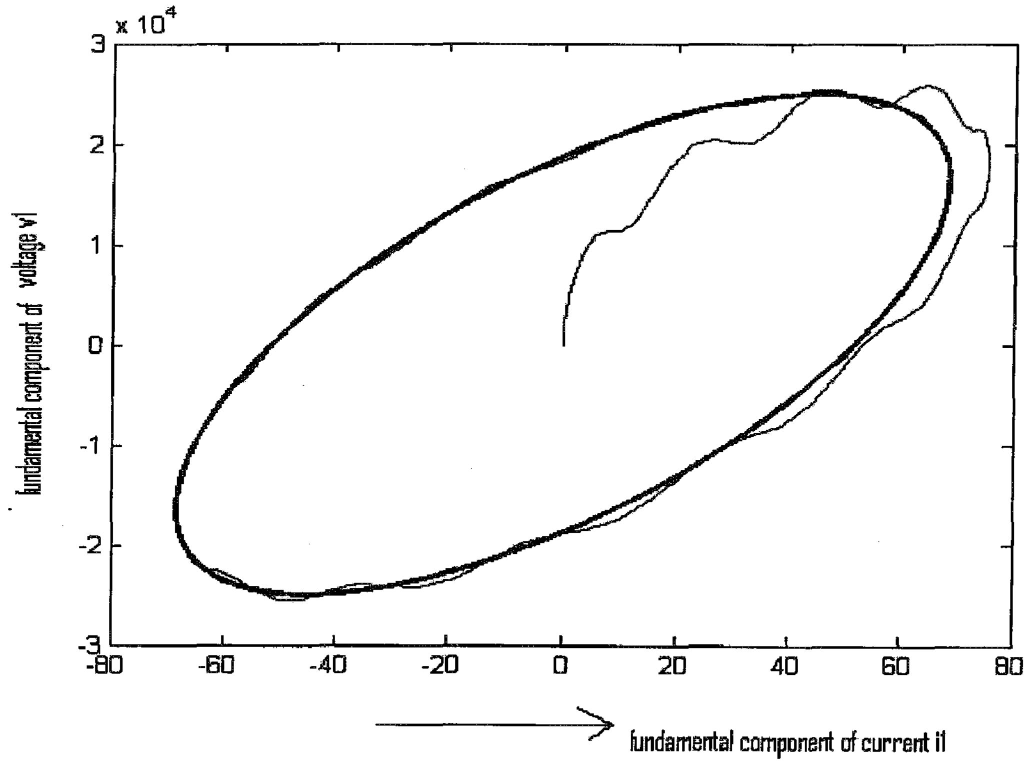


Figure 2.9 Plot of v1 vs i1 in case of linear fault at x=0.867

The characteristics exhibited by the 3rd and 5th harmonic components in case of Linear fault are given by the following patterns.

The plot of a (vs) b is bow shaped loop.

Plot of c (vs) d is closed loop.

Plot of p3 (vs) p5 is retraced lines.

Where $a = v3 * i3/p3$

$b = v5 * i5/p5$

$c = i3/v3$

$d = i5/v5$

The plot of b vs a for linear fault is shown in figure 2.10 given below.

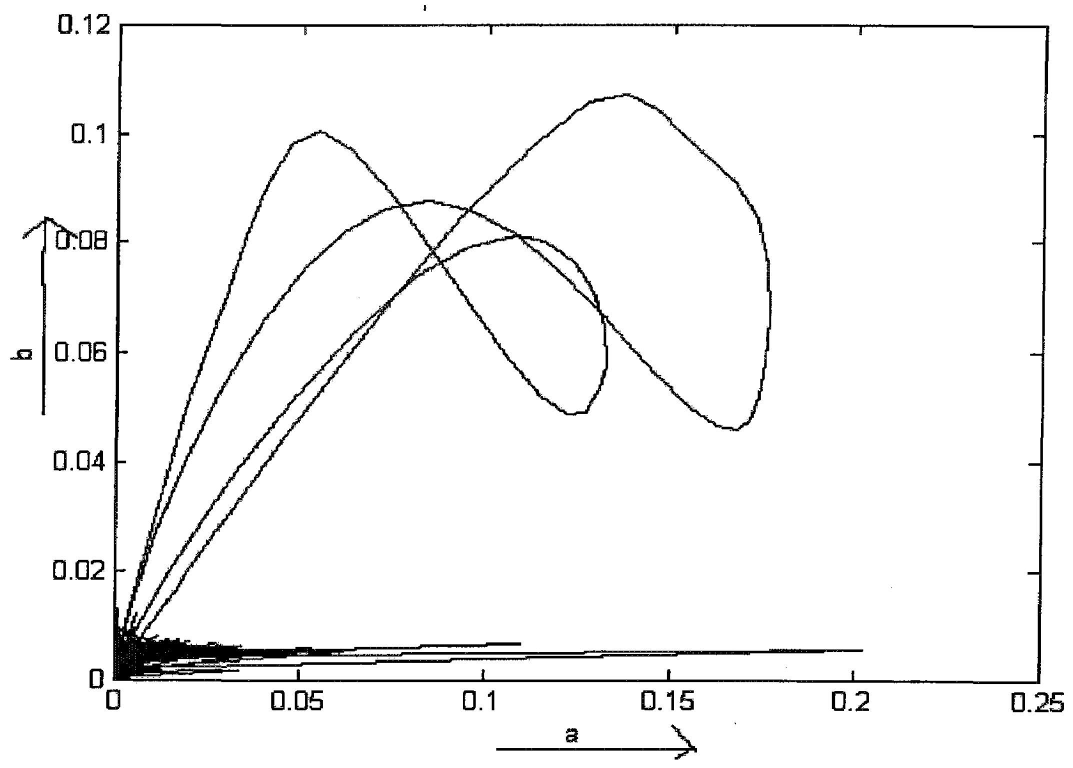


Figure 2.10 Plot of b vs a in case of linear fault at x=0.867

The plot of d vs c for linear fault is shown in figure 2.11 given below.

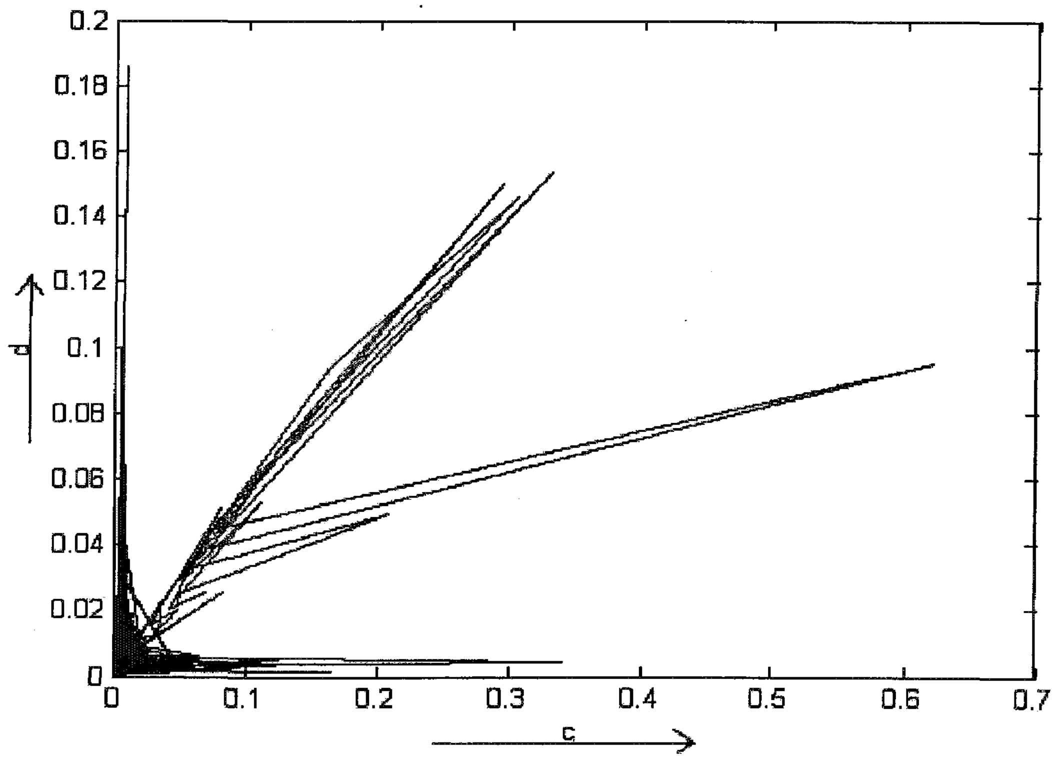


Figure 2.11 Plot of d vs c in case of linear fault at x=0.867

The plot of p5 vs p3 for linear fault is shown in figure 2.12 given below.

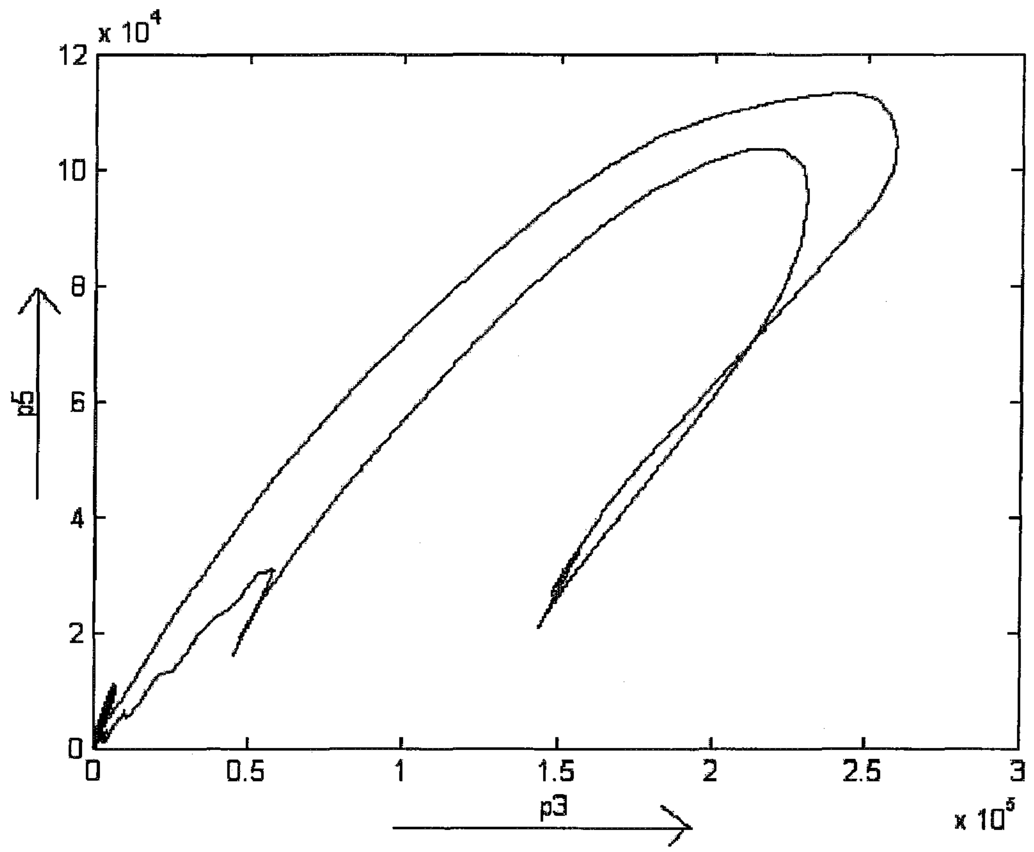


Figure 2.12 Plot of p5 vs p3 in case of linear fault at x=0.867

These patterns are generalized by observing many plots at different fault location and for different values of source and fault impedances at a particular fault location. For this purpose the fault is simulated by creating it at different locations through out the length of the line. [Note: location of fault can be changed by just changing the value of x in the simulation diagram shown in figure 2.3.] It's very cumbersome to observe all the plots every time and identify the fault. And the plots may become very close in most of the cases if there is no much nonlinearity. But if a neural network is trained with a set of fault data it easily recognizes the type of fault if occurred at later time based on its pattern observing capability.

CHAPTER-3 BACK PROPAGATION ALGORITHM

3.1. General Architecture of Neuron

The perceptron neural network was developed based on data processing characteristics of human brain. The advantage in using this technique is due to their ability to learn from examples and generalize the result for the inputs not seen in the training phase.

An elementary neuron with R input is shown in figure 3.1 below. Each input is denoted by p and weight is denoted by w. Each input p is weighted with some weight. A neuron can be seen as a combination of summer of weighted inputs and transfer function.

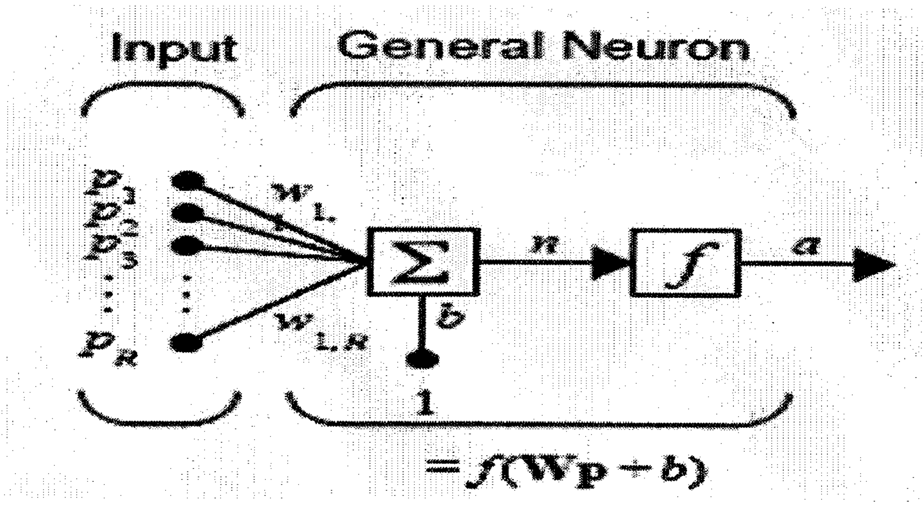


Figure 3.1: An elementary neuron with R input [16]

The sum of the weighted inputs with bias forms the net input to the transfer function f and is given by the following expression.

$$n=w1,1p1+w1,2p2+\ldots + w1, RpR + b$$

Any differentiable transfer function can be used by the neuron to generate the output.

3.2. Types of transfer functions:

There are many transfer functions in the literature, but in general only four types of transfer functions will be used. Their description is given below.

3.2.1. Hard limit transfer function: The hard-limit transfer function is shown in figure 3.2 below.

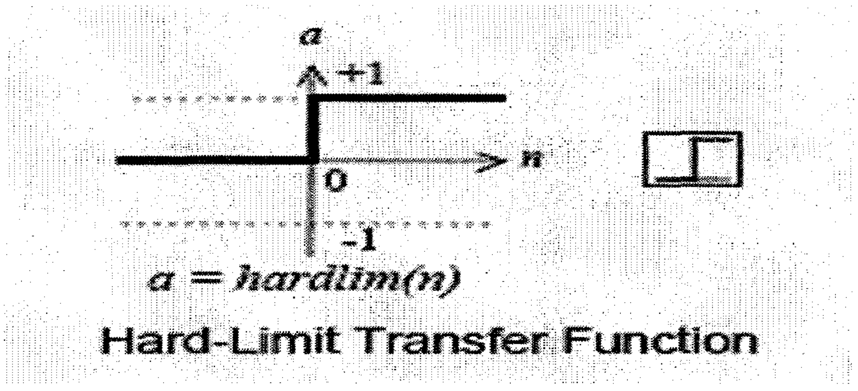


Figure 3.2: Representation of Hard Limit Transfer function [16]

This function limits the output of the neuron to 0 or 1 depending on the net input applied to transfer function.If the net input n is less than 0 output of the transfer function is 0 and if *n* is greater than or equal to 0 the output is 1.

3.2.2. Linear transfer function: The linear transfer function is shown in figure 3.3 below.

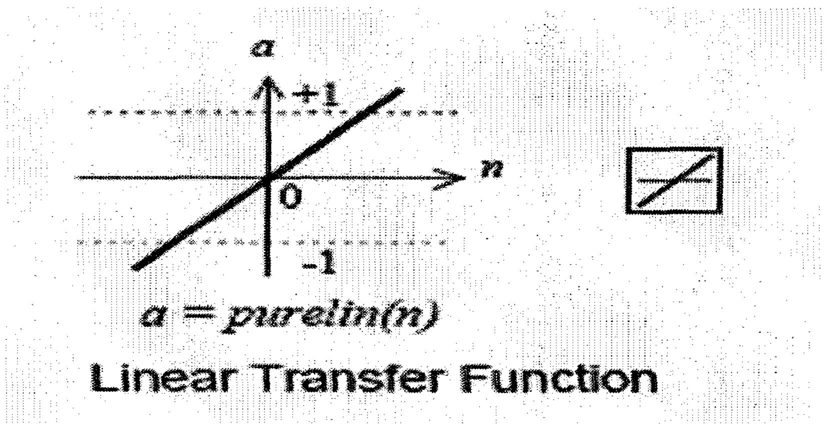


Figure 3.3: Representation of Linear Transfer function [16]

The linear transfer function will not change the output and it simply passes out the net output.

3.2.3. Log sigmoid transfer function: The log sigmoid transfer function is shown in figure 3.4 given below.

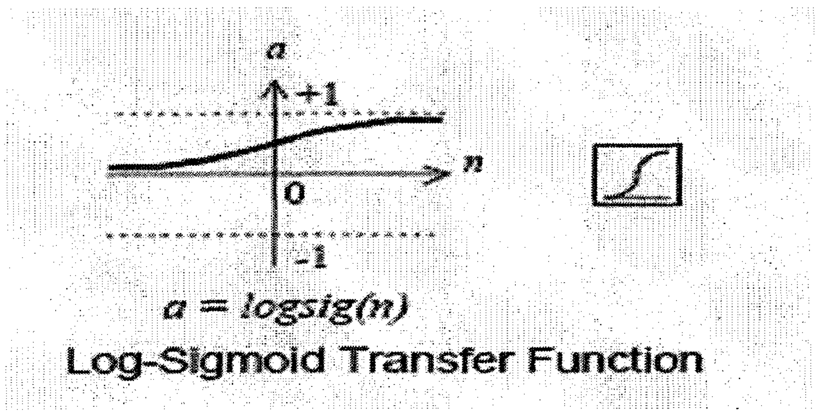


Figure 3.4: Representation of Log- Sigmoid Transfer function [16]

The log-sigmoid transfer function takes any value between $\pm\infty$ as the input and squashes the output into the range 0 to 1. This transfer function is commonly used in back propagation networks since it is differentiable.

3.2.4. Tan-sigmoid transfer function: The Tan-sigmoid transfer function takes any value between $\pm\infty$ as the input and the output will also be in the range $\pm\infty$. The Tan-sigmoid transfer function is shown in figure 3.5 below.

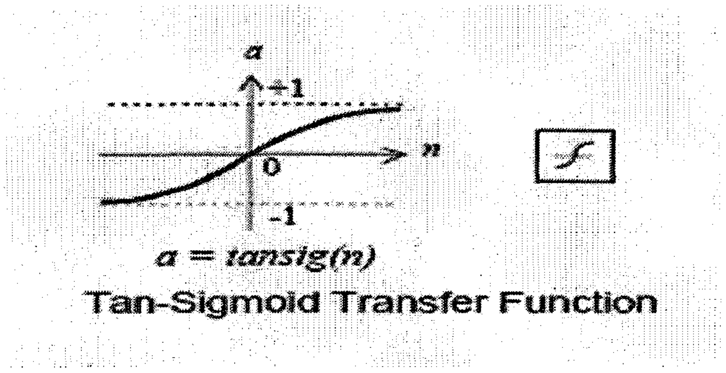


Figure 3.5: Representation of Tan- Sigmoid Transfer function [16]

Till now we have seen perceptron neuron of only one layer but we can adapt neural network of any layers. The sample multilayer perceptron neuron is shown in figure 3.6 given below. Here we have shown equal number of neuron in all the 3 layers. But practically this may not happen and each layer may have different number of neuron. Feed forward networks often have one or more hidden layers of sigmoid neurons followed by an output layer of linear neurons. Multiple layers of neurons with nonlinear transfer functions allow the network to learn non linear and linear relationships between input and output vectors. The linear output layer lets the network produce values outside the range -1 to +1.On the other hand, if you want to constrain the outputs of a network (such as between 0 and 1), then the output layer should use a sigmoid transfer function (such as logsig).

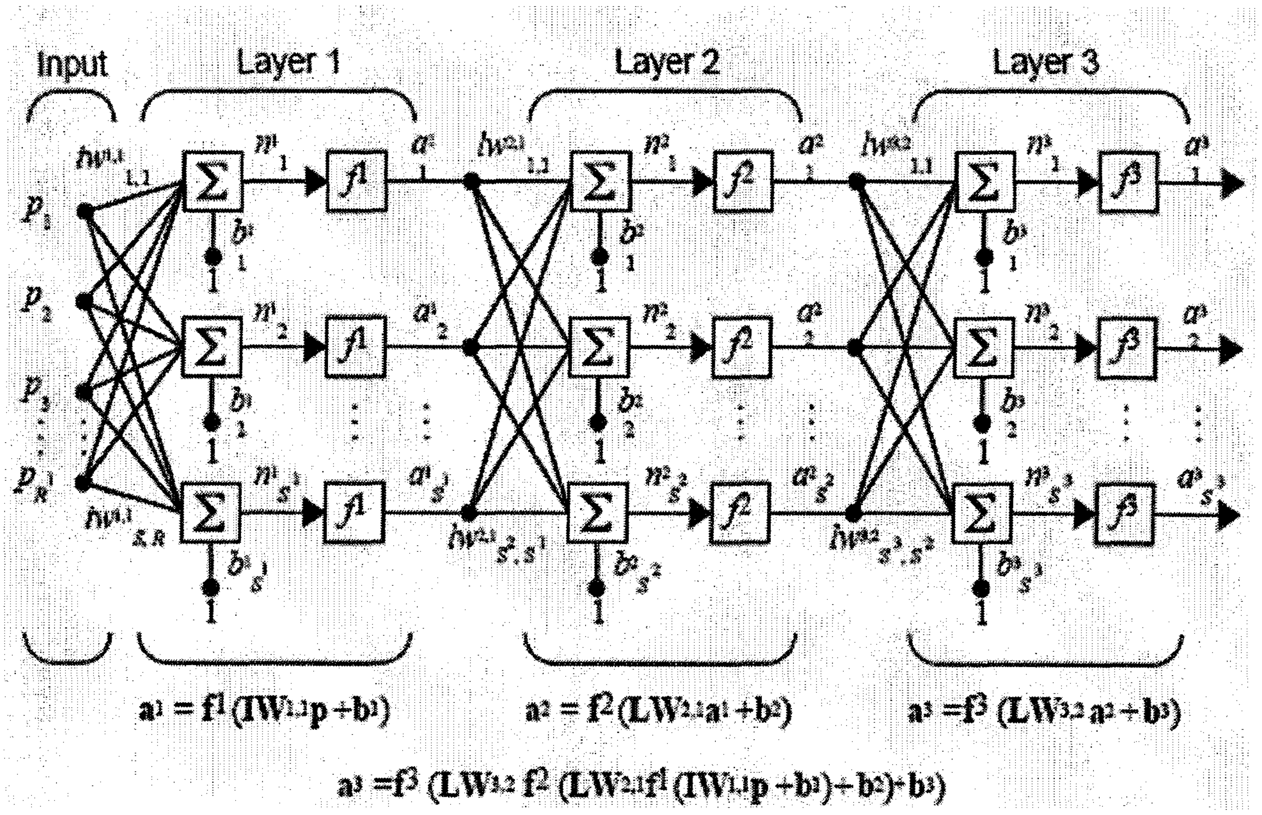

Input
Layer 1
Layer 2
Layer 3
$a^1 = f^1(IW^{1,1}p + b^1)$
$a^2 = f^2(LW^{2,1}a^1 + b^2)$
$a^3 = f^3(LW^{3,2}a^2 + b^3)$
$a^3 = f^3(LW^{3,2}f^2(LW^{2,1}f^1(IW^{1,1}p + b^1) + b^2) + b^3)$

Figure 3.6: Representation of Multi Layer Perceptron architecture [16]

Here we are training the network to give output in the range [0 1] so logsig transfer function is used. This training is done using the back propagation algorithms. **Backpropagation** is a supervised learning technique used for training artificial neural networks. It was first described by Paul Werbos in 1974, and further developed by David E. Rumelhart, Geoffrey E. Hinton and Ronald J. Williams in 1986. It is most useful for feed-forward networks (networks that have no feedback, or simply, that have no connections that

loop). The term is an abbreviation for "backwards propagation of errors". Backpropagation requires that the transfer function used by the artificial neurons (or "nodes") be differentiable.
since it will be used up in updating the weights. The derivative of logsig can be obtained by dlogsig. The back propagation algorithm trains a given feed-forward multilayer neural network for a given set of input patterns with known classifications. When each entry of the sample set is presented to the network, the network examines its output response to the sample input pattern. The output response is then compared to the known and desired output and the error value is calculated. Based on the error, the connection weights are adjusted. The back propagation algorithm is based on *Widrow-Hoff delta learning rule* in which the weight adjustment is done through *mean square error* of the output response to the sample input .The set of these sample patterns are repeatedly presented to the network until the error value is minimized. The algorithm is shown below.

3.3. Back Propagation Algorithm

```
1. Initialize the weights in the network (often randomly)
2. Repeat
   * For each example e in the training set do
    a. O = neural-net-output (network, e); forward pass
    b. T = teacher output for e
    c. Calculate error (T - O) at the output units
    d. Compute $\delta y$ for all weights from hidden layer to output layer backward pass
    e. Compute $\delta y$ for all weights from input layer to hidden layer; backward pass continued
    f. Update the weights in the network
        * end
 3. until all examples classified correctly or stopping criterion satisfied
 4. Return (network)
```

The figure 3.7 illustrates the back propagation multilayer network with M layers. N_j represents the number of neurons in layer j. Here the network is presented the P th pattern of training sample set with N_O dimensional input $X_{P1,}X_{P2,}.........X_{PN0}$ and N_M dimensional known output response $T_{P1,}T_{P2,}.........T_{PNM}$. The actual response to the input pattern by the network is represented as $O_{P1,}O_{P2,}.........O_{PNM}$. Let Y_{ji} be the output from the i th neuron in layer j for p th pattern; W_{jik} be the connection weight from k th neuron in layer j-1 to i th neuron in layer j ; and δ_{ji} be the error value associated with the i th neuron in layer j.

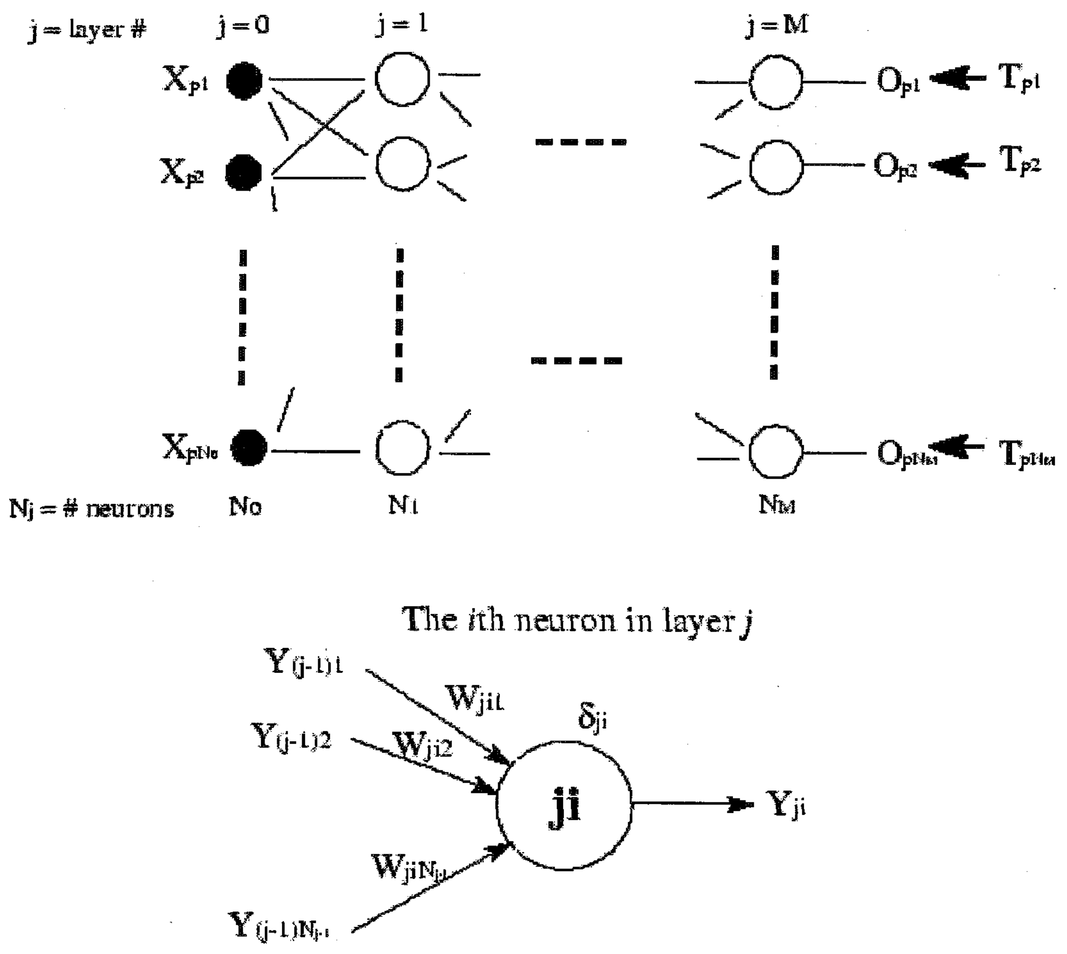


Figure 3.7: Back propagation neural network [17]

The algorithm description is given below.

1. Connection weights are initialized with small random values.
2. Present the p th sample input vector of pattern $X_P = X_{P1,} X_{P2,} ....X_{PN0}$ and the corresponding output target $T_P = T_{P1,} T_{P2,} ....T_{PNM}$ to the network.

3. Pass the input values to the first layer, layer 1. For every input node i in layer 0, perform:

$$Y_{oi} = X_{pi}$$

4. For every neuron i in every layer j = 1, 2,........M. from input to output layer, find the output from the neuron:

$$Y_{ji} = f\left(\sum_{k=1}^{N_{j-1}} Y_{(j-1)k} W_{jik}\right)$$

where $$f(x) = \frac{1}{1+\exp(-x)}$$

5. Obtain output values for every output node i in layer M , perform:

$$O_{Pi} = Y_{Mi}$$

6. Calculate error value δ_{ji} for every neuron i in every layer in backward order $j = M, M-1, ..., 2, 1$ from output to input layer, followed by weight adjustments.

For the output layer, the error value is: $\delta_{Mi} = Y_{Mi}(1 - Y_{Mi})(T_{Pi} - Y_{Mi})$ and for hidden layers:

$$\delta_{ji} = Y_{ji}(1 - Y_{ji})\sum_{k=1}^{N_{j+1}} \delta_{(j+1)k} W_{(j+1)ki})$$

The weight adjustment can be done for every connection from neuron k in layer i-1 to every neuron i in every layer i

$$W_{jik}^{new} = W_{jik} + \beta\delta_{ji}Y_{ji}$$

Where β represents weight adjustment factor normalized between 0 and 1. The derivation of the equations above will be discussed soon.

The above steps will be repeated for every training sample pattern p, and repeated for these sets until the root mean square (RMS) of output errors is minimized.

We now attempt to derive the error and weight adjustment equations shown above. Let's begin with the Root Mean Square (RMS) of the errors in the output layer defined as:

$$E_P = \frac{1}{2}\sum_{J=1}^{N_M} (T_{Pj} - O_{Pj})^2 \qquad \text{for the P th sample pattern.}$$

In *generalized delta rule* the error value δ_{ji} associated with the i th neuron in layer j is the rate of change in the RMS error E_P respect to the sum-of-product of the neuron:

$$\delta_{ji} = -\frac{\partial E_P}{\partial net_{ji}}$$

where net_{ji} represents the sum-of-product value.

With the chain rule, we can obtain the rate of change in the RMS error E_P in response to weight change:

$$\frac{\partial E_P}{\partial W_{jik}} = \frac{\partial E_P}{\partial net_{ji}} \frac{\partial net_{ji}}{\partial W_{jik}}$$

$$= -\delta_{ji} \frac{\partial}{\partial W_{jik}} [Y_{(j-1)0} W_{(j-1)i0} + \ldots\ldots + Y_{(j-1)k} W_{(j-1)ik} + \ldots\ldots]$$

$$= -\delta_{ji} \frac{\partial}{\partial W_{jik}} Y_{(j-1)k} W_{(j-1)ik}$$

$$= -\delta_{ji} Y_{(j-1)k}$$

We can say that the weight change is proportional to this value above
$\Delta W_{jik} = \beta \delta_{ji} Y_{(j-1)k}$ where β is a constant.

Thus, weight change can be performed as: $W_{jik}{}^{new} = W_{jik} + \Delta W_{jik}$

To find an error value associate with the neuron, again using the chain rule, we get:

$$\delta_{ji} = -\frac{\partial E_P}{\partial Y_{ji}} \frac{\partial Y_{ji}}{\partial net_{ji}}$$

For output layer j=M and $O_{Pi} = Y_{Mi}$.

Thus,

$$\delta_{Mi} = -\frac{\partial E_P}{\partial o_{Pi}} \frac{\partial Y_{Mi}}{\partial net_{Mi}}$$

$$= -\frac{\partial}{\partial o_{Pi}} [\frac{1}{2} \sum_{J=1}^{N_M} (T_{Pj} - O_{Pj})^2] \frac{\partial Y_{Mi}}{\partial net_{Mi}}$$

$$= -\frac{\partial}{\partial o_{Pi}}[\frac{1}{2}[(T_{P1} - O_{P1})^2 + \ldots\ldots + (T_{Pi} - O_{Pi})^2 + \ldots]\frac{\partial f(net_{Mi})}{\partial net_{Mi}}$$

$$= -\frac{\partial}{\partial o_{Pi}}[\frac{1}{2}[(T_{Pi} - O_{Pi})^2] f'(net_{Mi})$$

Using equation

$\delta_{Mi} = (T_{Pi} - O_{Pi})[f(net_{Mi})[1 - f(net_{Mi})]]$

$$= (T_{Pi} - O_{Pi})(O_{Pi})(1 - O_{Pi})$$

This should correspond with equation For error values associated with the hidden layer neurons, we cannot use target values. For this reason, the part $\frac{\partial E_P}{\partial Y_{ji}}$ in equation needs to be found using a different approach. We use the chain rule applied to the sum-of-product values of neurons in the front layer j+1.

$$\frac{\partial E_P}{\partial Y_{ji}} = \frac{\partial E_P}{\partial net(_{j+1})1}\frac{\partial net(_{j+1})1}{\partial Y_{ji}} + \frac{\partial E_P}{\partial net(_{j+1})2}\frac{\partial net(_{j+1})2}{\partial Y_{ji}} + \ldots$$

$$= \sum_{a=1}^{N_{j+1}}[\frac{\partial E_P}{\partial net(_{j+1})a}\frac{\partial net(_{j+1})a}{\partial Y_{ji}}]$$

$$= \sum_{a=1}^{N_{j+1}}[-\delta_{(j+1)a}\frac{\partial}{\partial Y_{ji}}(Y_{j0}W_{(j+1)a0} + \ldots\ldots + Y_{ji}W_{(j+1)ai} + \ldots)]$$

$$= \sum_{a=1}^{N_{j+1}}[-\delta_{(j+1)a}\frac{\partial}{\partial Y_{ji}}[Y_{ji}W_{(j+1)ai}]$$

$$= \sum_{a=1}^{N_{j+1}} [-\delta_{(j+1)a} W_{(j+1)ai}]$$

Finally, combining with $\frac{\partial Y_{ji}}{\partial net_{ji}}$ we get:

$$\delta_{ji} = -\sum_{a=1}^{N_{j+1}} [-\delta_{(j+1)a} W_{(j+1)ai}] \frac{\partial Y_{ji}}{\partial net_{ji}} = Y_{ji}(1-Y_{ji}) \sum_{a=1}^{N_{j+1}} [\delta_{(j+1)a} W_{(j+1)ai}]$$

By substituting this in $W_{jik}^{new} = W_{jik} + \beta \delta_{ji} Y_{ji}$ new weight is obtained.

3.4. Description of Back Propagation Algorithm using 3 layer neural network

The diagrammatic representation of the calculation of weights for multilayer neural network by Back propagation algorithm is shown below.To illustrate this process consider the three layer neural network with two inputs and one output as shown in figure 3.8 below.

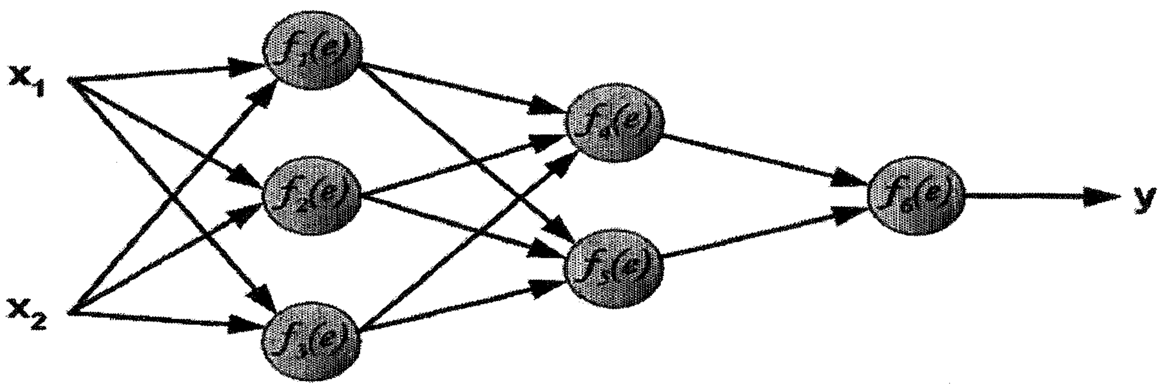


Figure 3.8: Three Layer Neural Network [17]

Each neuron is composed of two units. First unit adds products of weights coefficients and input signals. The second unit realise nonlinear function, called neuron activation function. Signal *e* is adder output signal, and *y* = *f(e)* is output signal of nonlinear element. Signal *y* is also output signal of neuron.

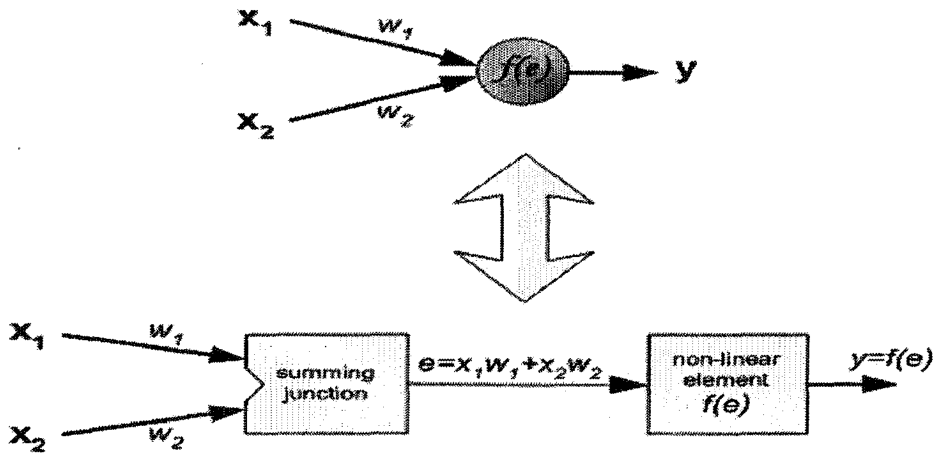
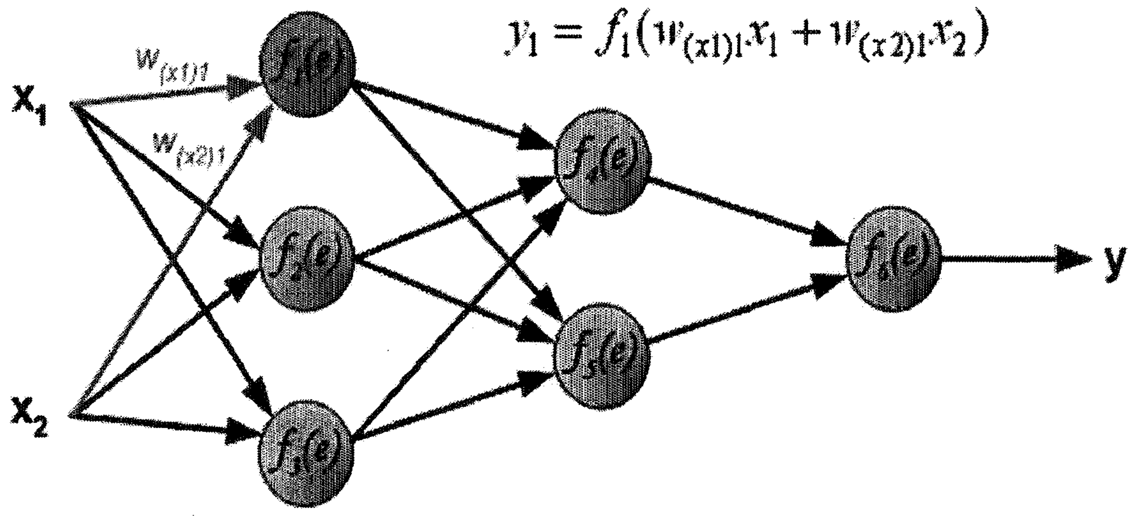
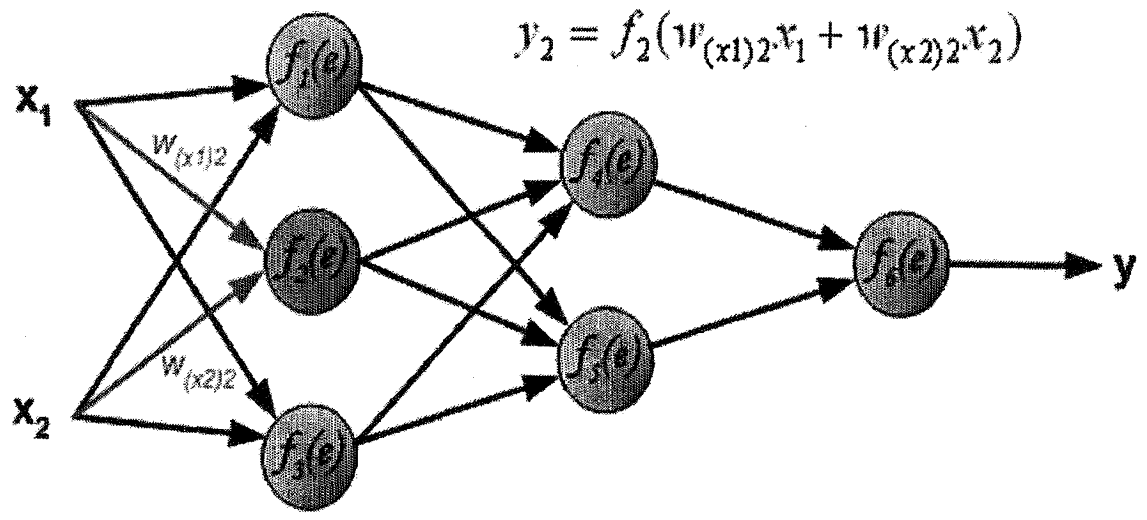


Figure 3.9: Description of Neuron as a Composition of 2 units [17]

To teach the neural network a set of data will be used for training it. The training data set consists of input signals (x_1 and x_2) assigned with corresponding target (desired output) z. The network training is an iterative process. In each iteration weights coefficients of nodes are modified using new data from training data set. pictures below illustrate how signal is propagating through the network, Symbols $w_{(xm)n}$ represent weights of connections between network input x_m and neuron n in input layer. Symbols y_n represents output signal of neuron n.





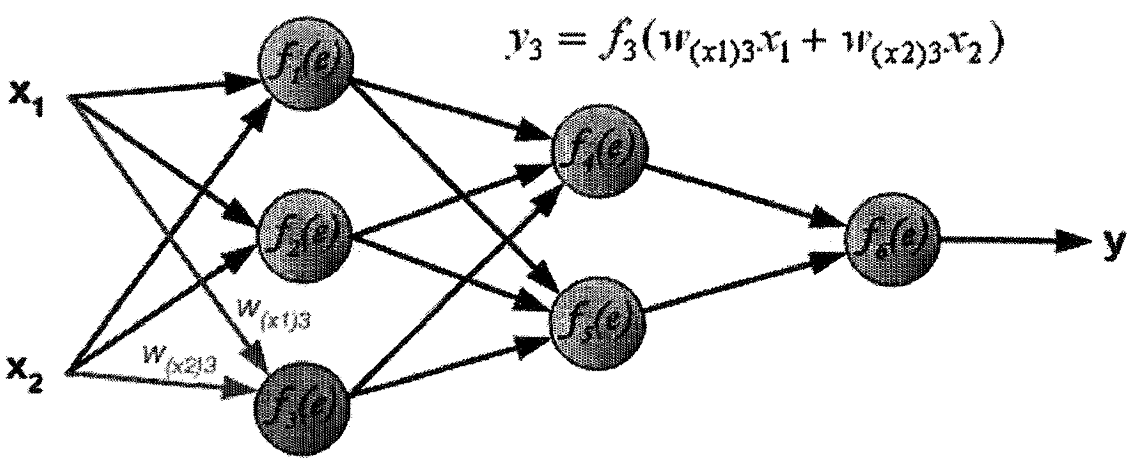
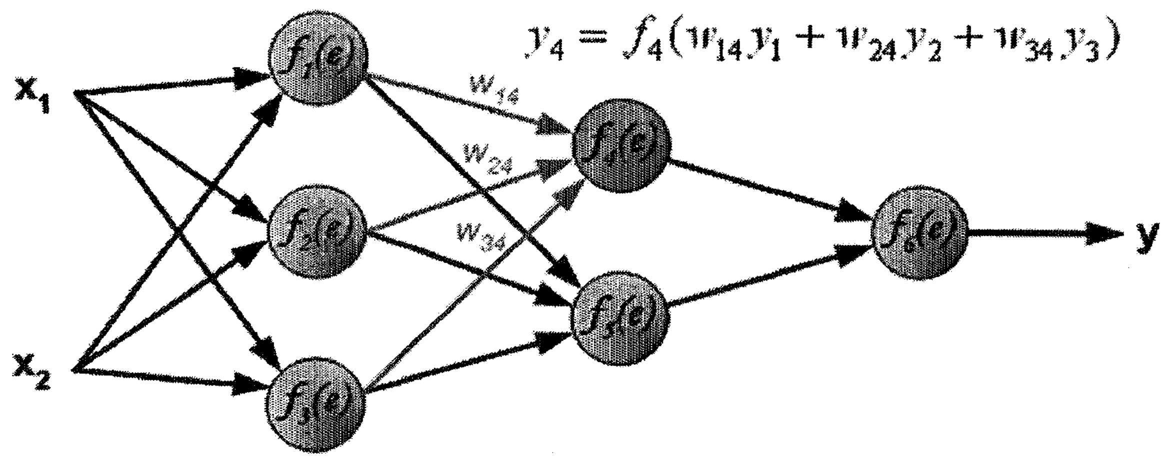


Figure 3.10: Outputs of 3 Neuron in first hidden layer (y_1, y_2, y_3) [17]

Outputs of all the 3 neurons in first layer are shown in figure 3.10 shown above.Propagation of signals through the hidden layer are shown below. Symbols w_{mn} represent weights of connections between output of neuron *m* and input of neuron *n* in the next layer. Outputs of the 2 neurons in second layer are shown in figure 3.11



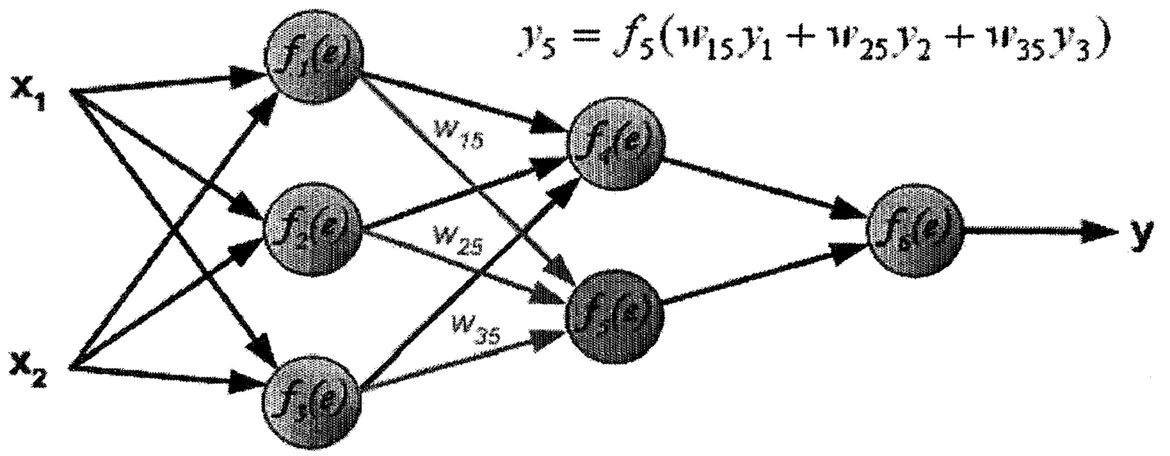


Figure 3.11: Outputs of 3 Neuron in second hidden layer (y_4, y_5) [17]

Propagation of signals through the output layer is shown in figure 3.10.

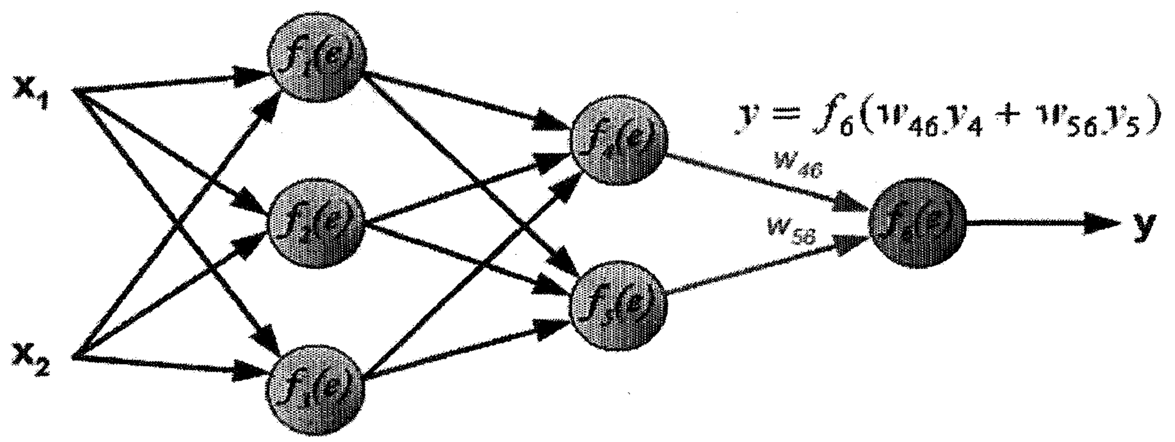


Figure 3.12: Output of Neuron in output layer (y_6) [17]

In the next algorithm step the output signal of the network y is compared with the desired output value (the target), which is found in training data set. The difference is called error signal δ of output layer neuron and the way of finding it is shown in figure 3.11.

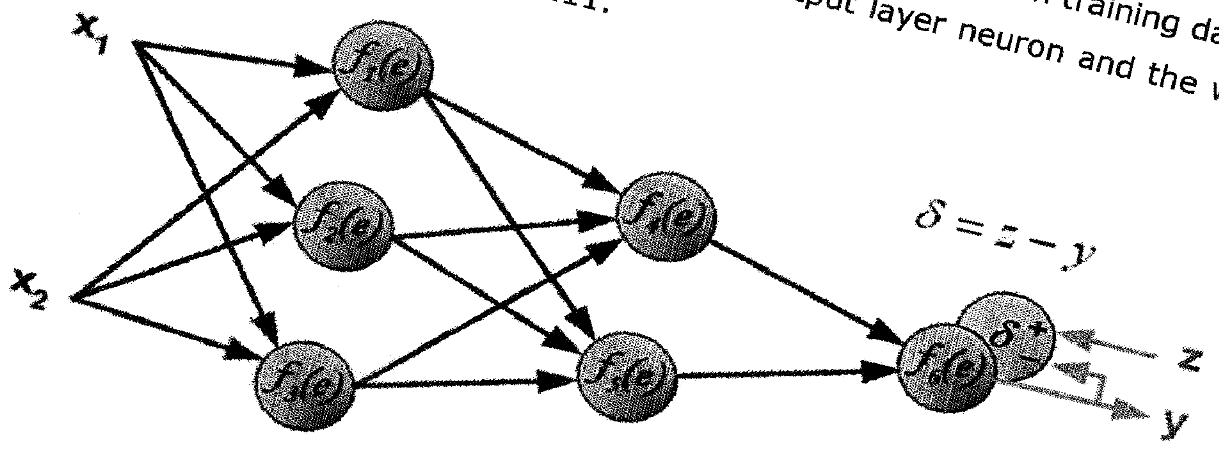
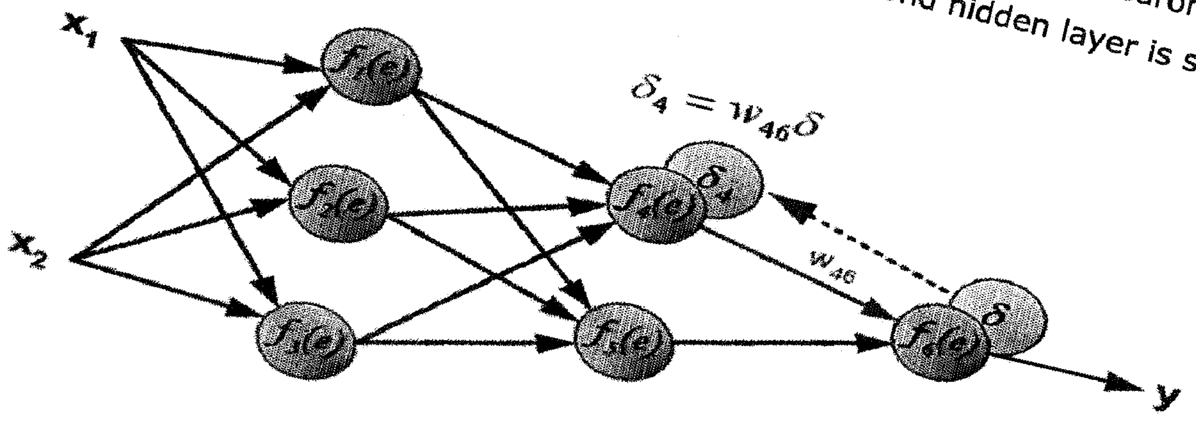


Figure 3.13: Finding out the Deviation in Output of Neural Network from target value

It is impossible to compute error signal for internal neurons directly, because output values of these neurons are unknown. For many years the effective method for training multiplayer networks has been unknown. Only in the middle eighties the backpropagation algorithm has been worked out. The idea is to propagate error signal δ (computed in single teaching step) back to all neurons, which output signals were input for discussed neuron. The propagation of error signal from output layer to second hidden layer is shown in figure 3.14.



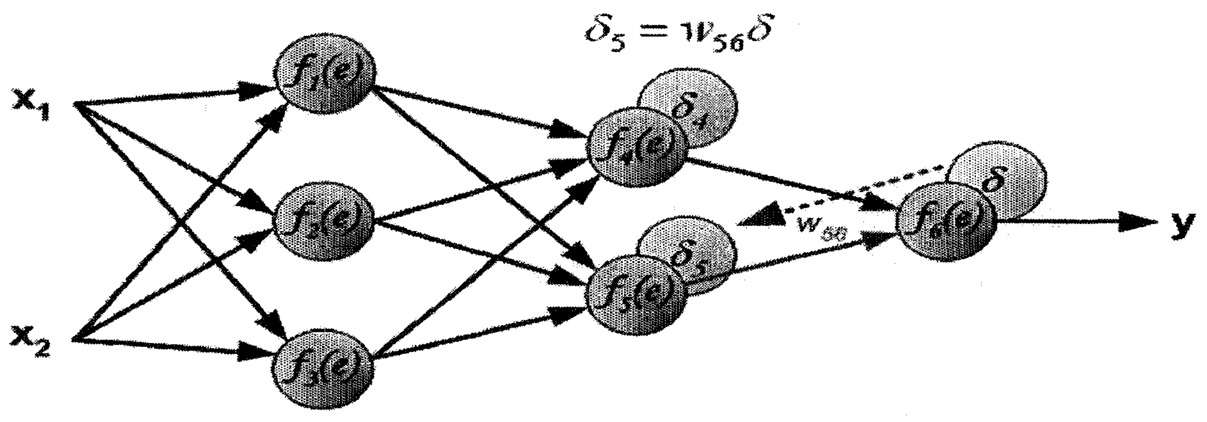
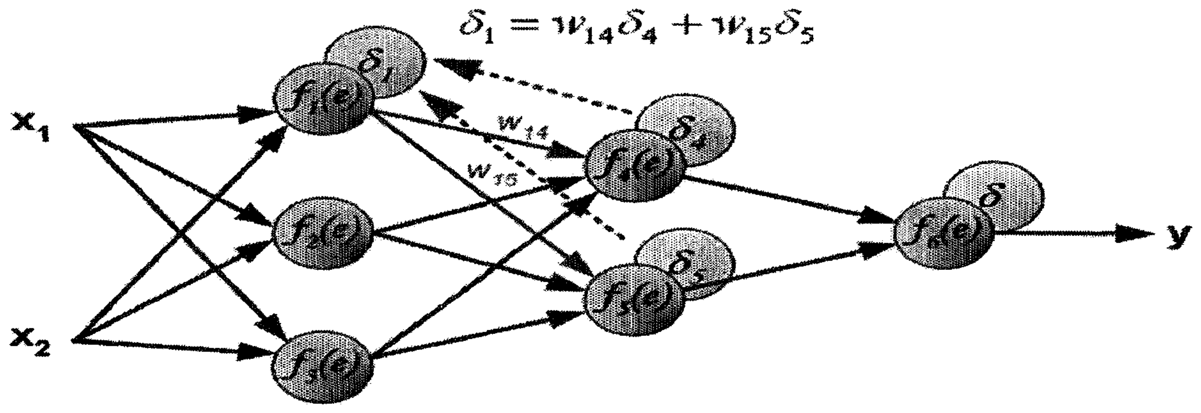
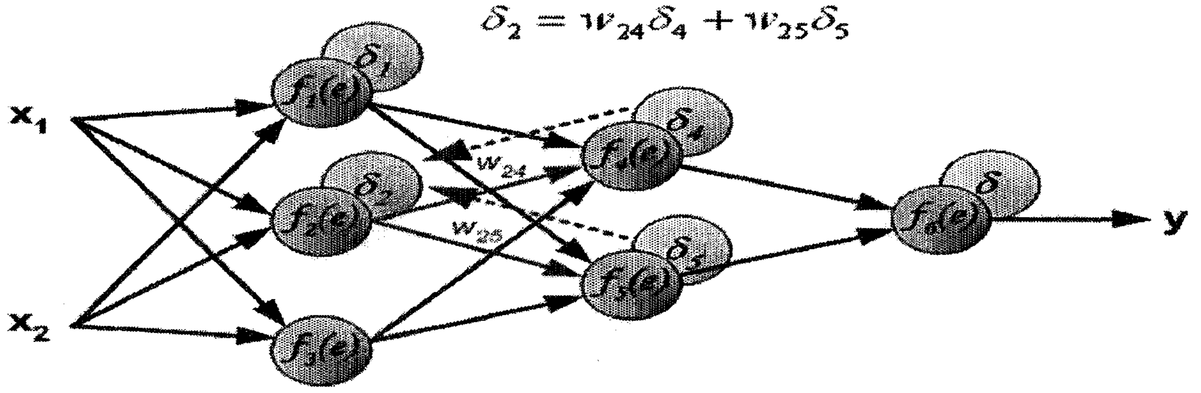
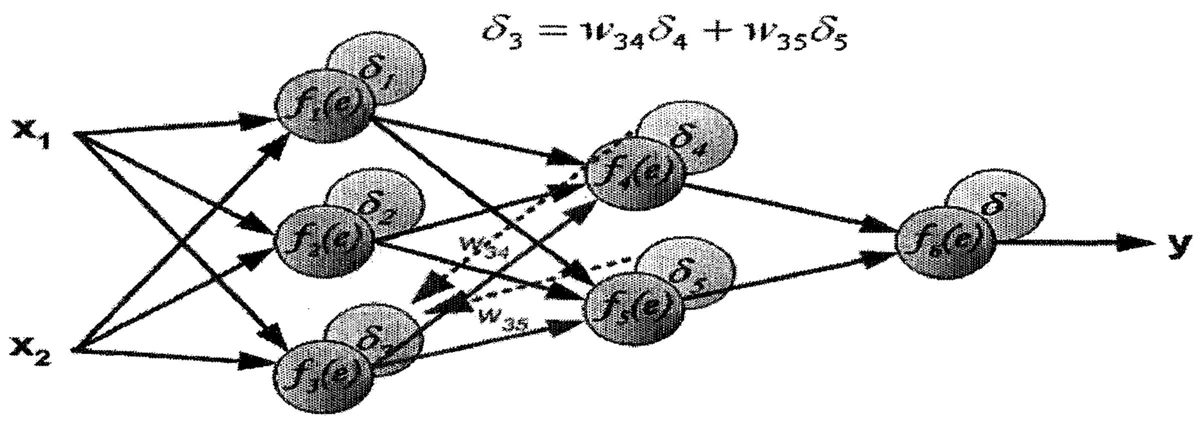


Figure 3.14: propagation of error signal from output layer to second hidden layer

The propagation of error signal from second hidden layer to first hidden layer is shown in figure 3.15.



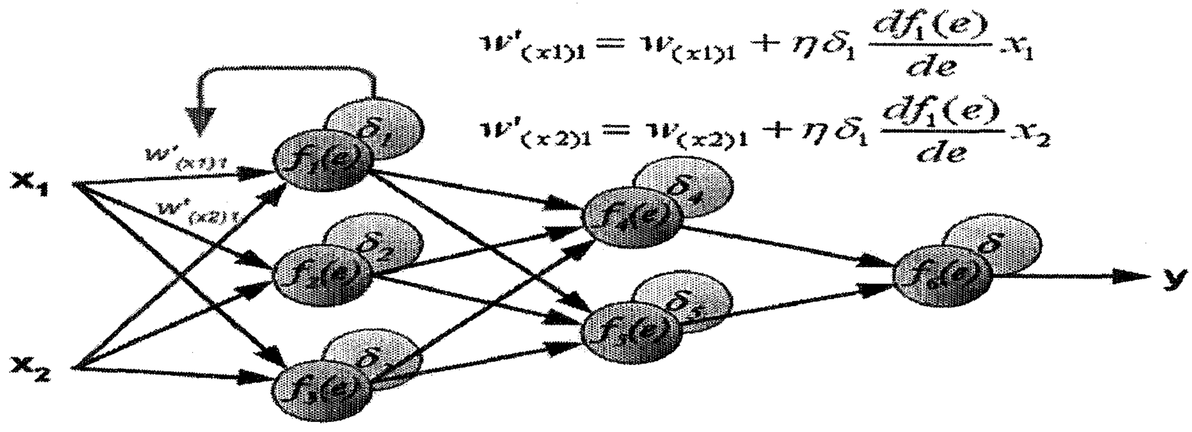




$$\delta_3 = w_{34}\delta_4 + w_{35}\delta_5$$

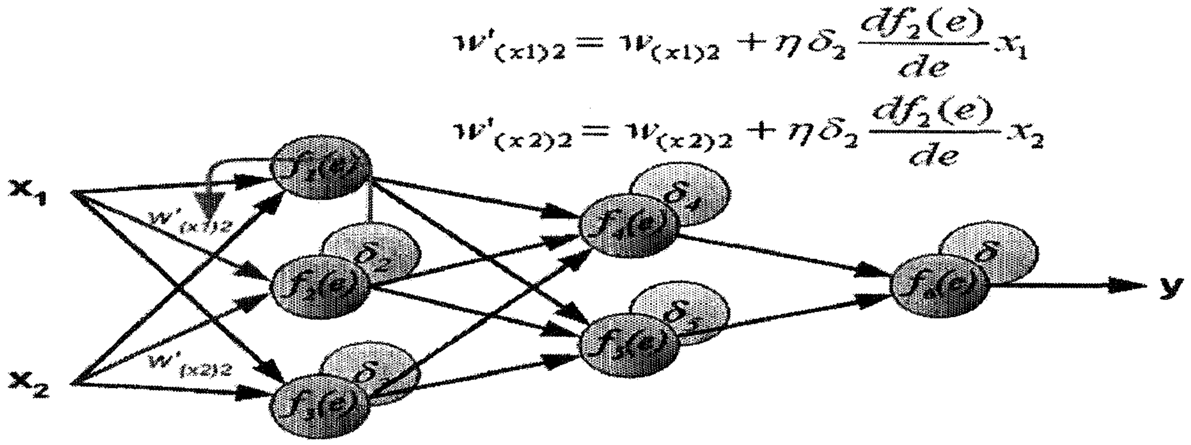
Figure 3.15: propagation of error signal from second hidden layer to first hidden layer

The modification in weights coefficients between first hidden layer and input will be as shown below in figure 3.16.



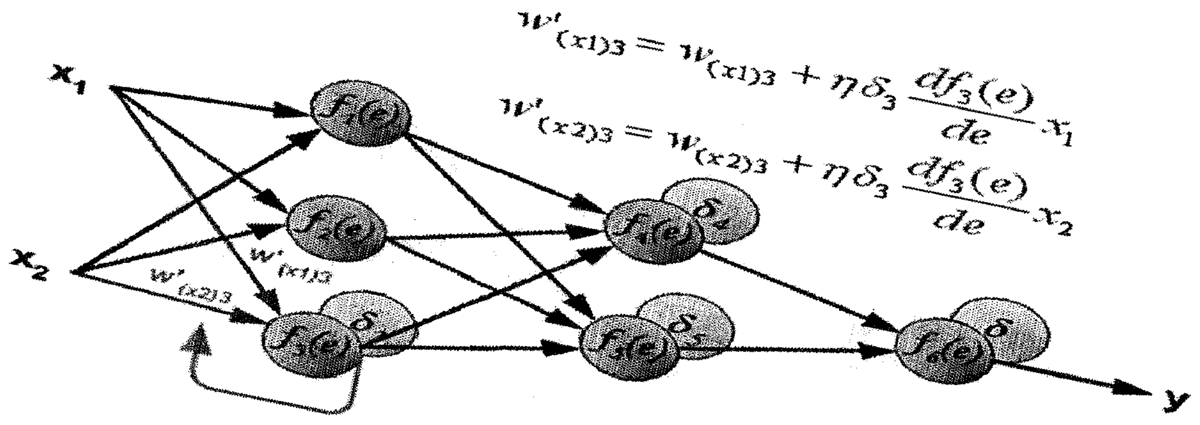
$$w'_{(x1)1} = w_{(x1)1} + \eta\delta_1 \frac{df_1(e)}{de} x_1$$

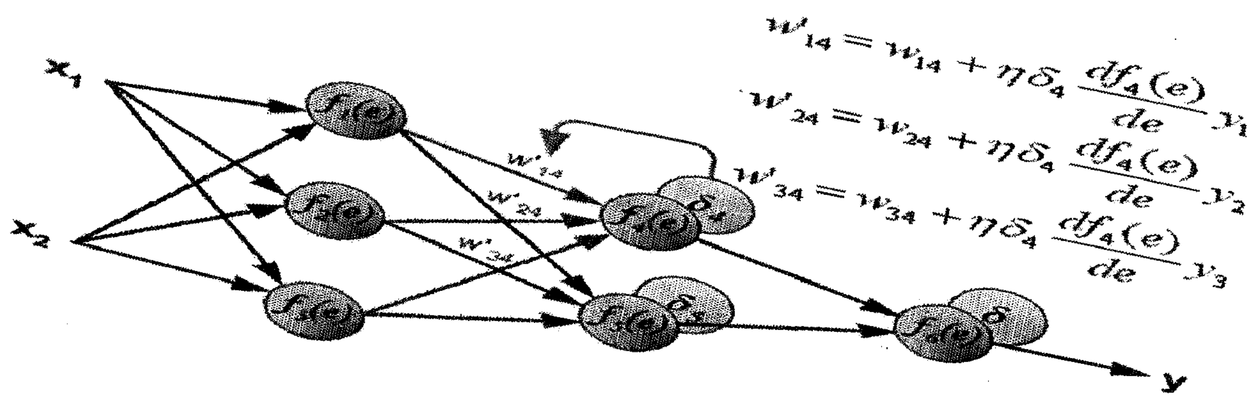
$$w'_{(x2)1} = w_{(x2)1} + \eta\delta_1 \frac{df_1(e)}{de} x_2$$

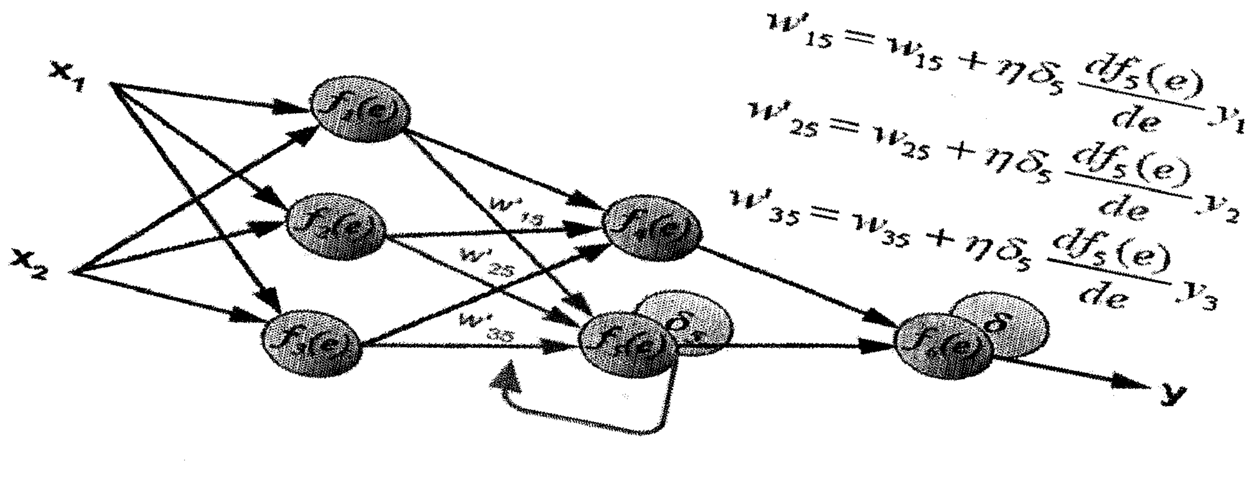


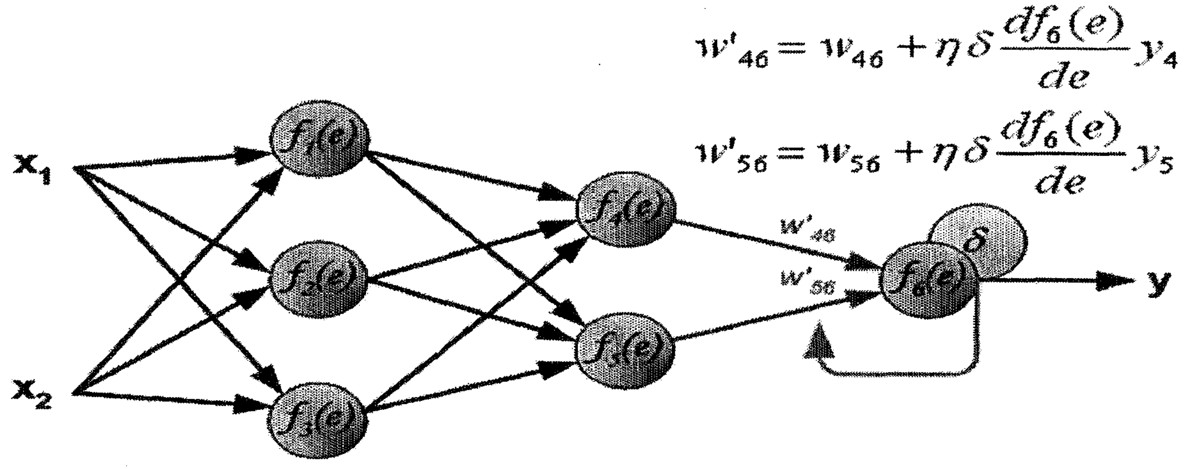
$$w'_{(x1)2} = w_{(x1)2} + \eta\delta_2 \frac{df_2(e)}{de} x_1$$

$$w'_{(x2)2} = w_{(x2)2} + \eta\delta_2 \frac{df_2(e)}{de} x_2$$


$w'_{(x1)3} = w_{(x1)3} + \eta\delta_3 \frac{df_3(e)}{de} x_1$
$w'_{(x2)3} = w_{(x2)3} + \eta\delta_3 \frac{df_3(e)}{de} x_2$
x_1
x_2
$f_1(e)$
$f_2(e)$
$f_3(e)$
$f_4(e)$
$f_5(e)$
$f_6(e)$
δ_3
δ_4
δ_5
δ
$w'_{(x1)3}$
$w'_{(x2)3}$
y


$w'_{14} = w_{14} + \eta\delta_4 \frac{df_4(e)}{de} y_1$
$w'_{24} = w_{24} + \eta\delta_4 \frac{df_4(e)}{de} y_2$
$w'_{34} = w_{34} + \eta\delta_4 \frac{df_4(e)}{de} y_3$
x_1
x_2
$f_1(e)$
$f_2(e)$
$f_3(e)$
$f_4(e)$
$f_5(e)$
$f_6(e)$
δ_4
δ_5
δ
w'_{14}
w'_{24}
w'_{34}
y


$w'_{15} = w_{15} + \eta\delta_5 \frac{df_5(e)}{de} y_1$
$w'_{25} = w_{25} + \eta\delta_5 \frac{df_5(e)}{de} y_2$
$w'_{35} = w_{35} + \eta\delta_5 \frac{df_5(e)}{de} y_3$
x_1
x_2
$f_1(e)$
$f_2(e)$
$f_3(e)$
$f_4(e)$
$f_5(e)$
$f_6(e)$
δ_5
δ
w'_{15}
w'_{25}
w'_{35}
y



$w'_{46} = w_{46} + \eta\delta\frac{df_6(e)}{de}y_4$
$w'_{56} = w_{56} + \eta\delta\frac{df_6(e)}{de}y_5$
x_1
x_2
$f_1(e)$
$f_2(e)$
$f_3(e)$
$f_4(e)$
$f_5(e)$
$f_6(e)$
δ
w'_{46}
w'_{56}
y

Figure 3.16 The modification in weights coefficients between first hidden layer and input

Coefficient η is the learning rate. There are 2 techniques to select this parameter.The first method is to start teaching process with large value of the parameter. While weights coefficients are being established the parameter is being decreased gradually. The second method starts teaching with small parameter value. During the teaching process the parameter is being increased when the teaching is advanced and then decreased again in the final stage. Starting teaching process with low parameter value enables to determine weights coefficients signs.

df(e)/de represents derivative of neuron activation function.

CHAPTER-4 RESULTS AND DISCUSSION

4.1. Implementation in Matlab 7.0.1

The training and testing of the feed forward neural network for distinguishing the linear and nonlinear faults is done using back propagation algorithm in Matlab 7.0.1.The first step in training a feed forward network is to create the network object. The function newff creates a feed forward network. Before training a feed forward network, the weights and biases must be initialized. The newff command will automatically initialize the weights. The function sim simulates a network. sim takes the network input p, and the network object net, and returns the network outputs a. Once the network weights and biases have been initialized, the network is ready for training. The network can be trained for function approximation (nonlinear regression), pattern association, or pattern classification. In this thesis training is being done for pattern classification. The training process requires a set of examples of proper network behavior -network inputs p and target outputs t. During training the weights and biases of the network are iteratively adjusted to minimize the network performance function net.performFcn. The default performance function for feed forward networks is mean square error mse - the average squared error between the network outputs a and the target outputs t.

The implementation of back propagation algorithm updates the network weights and biases in the direction in which the performance function decreases most rapidly - the negative of the gradient. One iteration of this algorithm can be written as

$$x_{k+1} = x_k - \alpha_k g_k$$

Where x_k a vector of current weights and biases is, g_k is the current gradient, and α_k is the learning rate. RMS values of third and fifth harmonic components of voltage and current at feeder are found out at each location of the fault. Half of the data obtained like this is used for training the neural

.twork by using back propagation algorithm and the remaining half is used ›r testing it. In the present case the neural network is trained to give an ›utput value of 1 for linear fault and 0 for nonlinear fault. We have adapted 3 ıayer neural network consisting of 4 neuron in each hidden layer and 2 neuron in output layer and log sigmoid function is used in all the layers. Learning rate of 0.565 is used and a momentum factor of 0.585 is used in the training phase of the neural network. Percentage of training and testing samples classified by different back propagation algorithms of the neural network are shown below in table 4.1.

Table 4.1 percentage classification for different Back propagation algorithm:

s.no	Back propagation algorithm used	Percentage of samples classified in training	Percentage of samples classified in testing
1.	Trainlm	100	100
2.	Traingdm	66.85	65.78
3.	Trainr	95.72	95.18
4.	Trainb	81.28	82.88
5.	Traingd	62.57	61.5
6.	Traingda	87.70	86.1
7.	Traingdx	90.375	87.7
8.	Trainrp	100	96.8
9.	Trainbr	100	99.47
10.	Trainoss	99.47	98.4
11.	Trainbfg	98.4	95.73
12.	Traincgf	95	95
13.	Traincgp	98.4	98.4
14.	Traincgb	96.26	93.6
15.	Trainscg	98.93	98.39

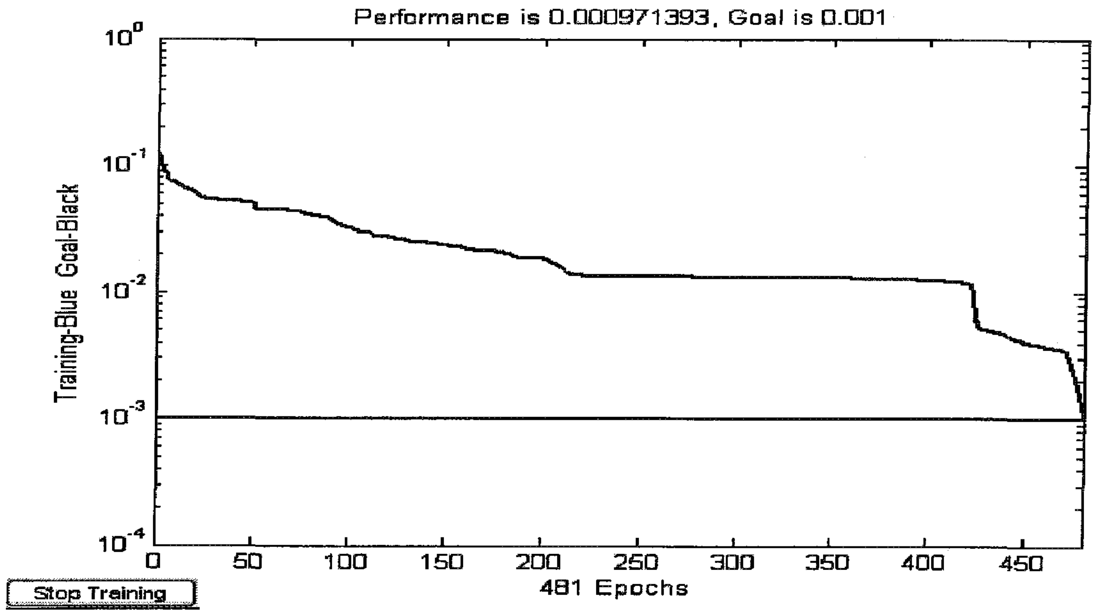


Figure 4.1 Training the feed forward network with trainlm algorithm

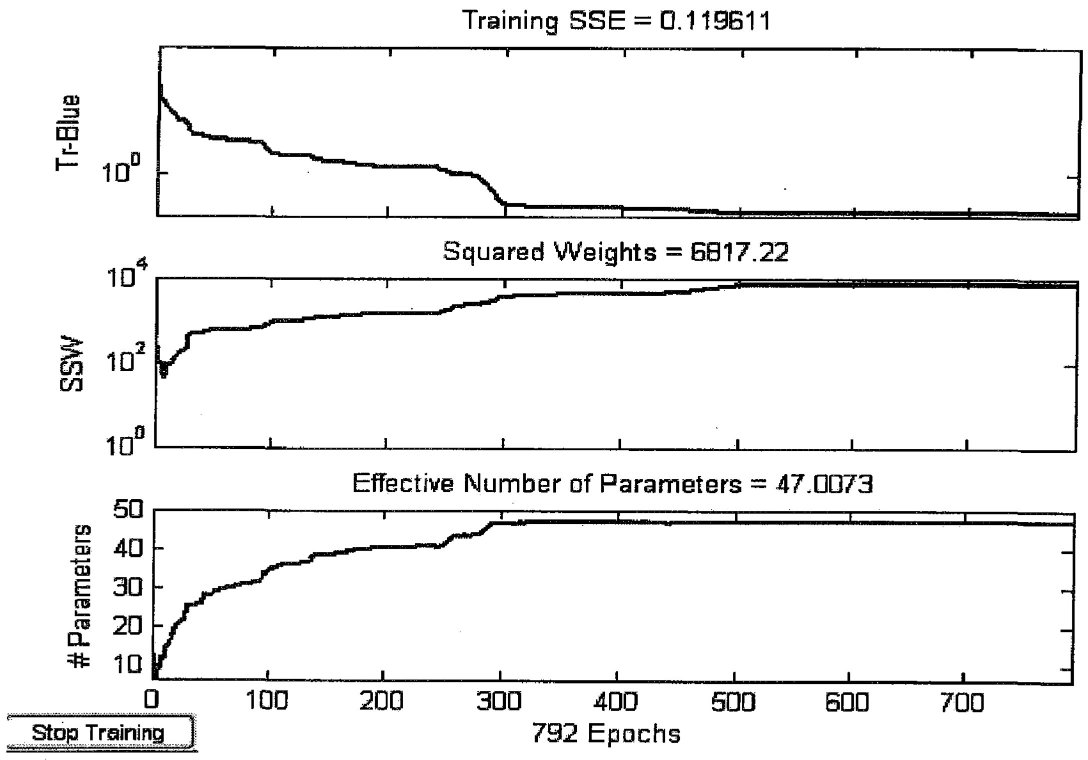


Figure 4.2 Training the feed forward network with 'trainbr' algorithm

Performance curves due to training the feed forward network by the back propagation algorithms with trainlm and trainbr are shown in figure 4.1 and

4.2. The percentage classification for different combination of neuron in 2 hidden layer are given in table 4.2.

Table 4.2 percentage classification for different combination of neuron:

s.no	Back propagation algorithm used	No. of neurons in first hidden layer	No. of neurons in second hidden layer	Percentage of samples classified in training	Percentage of samples classified in testing
1.	Trainlm	4	8	100	97.86
		6	8	100	98.4
		8	8	100	98.5
		8	6	100	95.2
		4	6	100	97.33
		6	4	100	95.2
		6	6	93.58	93.58

Percentage of samples classified during training and testing of the three layer network for different combination of transfer functions used by the neuron are shown below in table 4.3.

Table 4.3 percentage classification for different transfer function used by neuron:

s.no	Transfer function used by neuron in first hidden layer	Transfer function used by neuron in second hidden layer	Transfer function used by neuron in output layer	Percentage of samples classified in training	Percentage of samples classified in testing
1.	Logsig	Logsig	Logsig	100	100
2.	Tansig	Tansig	Tansig	100	97.86
3.	Purelin	Purelin	Purelin	82.89	83.96
4.	Hardlim	Hardlim	Hardlim	42.24	42.24
5.	Logsig	Logsig	Purelin	100	100
6.	tansig	Tansig	purelin	100	97.3

CHAPTER-5 CONCLUSIONS

The values of feeder voltage and feeder current are obtained for two kinds of fault cases (i.e. linear and nonlinear) by simulating the model of High Impedance Fault system. The values of third and fifth harmonics are obtained by applying the Fast Fourier Transform. RMS values of these harmonics are used to train the Multilayer Perceptron Neural Network for classification of these two types of faults. It consists of two hidden layer and one output layer. Each hidden layer consists of four neuron and one output layer consists of two neuron. This network is trained by using the Back propagation algorithm. Many types of back propagation algorithms are tested and it's found that trainlm and trainbr are classifying these two kinds of faults more perfectly compared to other algorithms. As well as for selecting the no of neuron the network is tested for different number of neuron in each layer and it's found that the network consisting of four neuron in each hidden layer is performing well. The network is tested for different transfer function and it's found that its performance is good when log-sigmoid transfer function is used in all the 3 layers or when tan-sigmoid transfer function is used by the neuron in two hidden layer and linear transfer function is used by neuron in output layer.

REFERENCES

1. B. M. Aucoin, B. Don Russell, "Distribution High Impedance Fault Detection Utilizing High Frequency Current Components", IEEE Trans. on PAS, Vol. PAS-101, No. 6, June 1982,pp. 1596-1606.

2. Emanuel, A.E., Cyganski, D., Orr, J.A., Shiller, S.,Gulachcnski, E.M., "High Impedance Fault Arcing on Sandy Soil in 15 kV Distribution Feeders: Contributions to The Evaluation of the Low Frequency Spectrum," IEEE Transactions on Power Delivery, Volume 5, Issue 2, April 1990 pp 676 -686

3. Aucoin, B. Michael, and Jones, Robert H., "High Impedance Fault Detection Implementation Issues", IEEE Transactions on Power Delivery, Vol. 11, No. I, January, 1996, pp.139-148.

4. A.M. Sharaf , L.A. Snider , K. Debnath , "A Third Harmonic Sequence ANN Based Detection Scheme For High Impedance Faults", Proceedings of the ISEDEM , Singapore,1993, pp. 802-806.

5. L. A. Snider, Yuen Yee Shan, "The Artificial Neural Networks based Relay Algorithm for Distribution System High Impedance Fault Detection ", Proceedings of the 4th International Conference on Advances in Power System Control, Operation and Management, APSCOM-97, Hong Kong, November 1997 , pp 100-106.

6. L.A. Snider, Y.S. Yuen "The Artificial Neural-Networks based Relay Algorithm for the Detection of Stochastic High Impedance Faults", ELSEVIER Transactions on Neuro computing, 1998, pp 243-254.

7. T. M. Lai, L.A. Snider, E. Lo, C.H. Cheung and K.W. Chan "High Impedance Faults Detection Using Artificial Neural Network", Proceedings of the 6th International Conference on Advances in Power System Control, Operation and Management, APSCOM , Hong Kong, November 2003, pp 821-826.

8. A.M. Sharaf, L.A. Snider, K. Debnath, "A Neural Network based Back Error Propagation Relay Algorithm for Distribution System High Impedance Fault Detection", Proceedings of the ISEDEM, Singapore, pp.613-620.

9. Adel M. Sharaf ,Guosheng Wang , "High Impedence Fault Detection Using Low Order Pattern Harmonic Detection ", IEEE trans. pp 883-886.

10.Adel M. Sharaf ,Guosheng Wang ,"High Impedance Fault Detection using Feature-Pattern Based Relaying ", IEEE trans. pp 222- 226.

11.A. M. Sharaf ,R M. El-Sharkawy, H. E. A. Talaat ,M. A. L. Badr "Novel Alpha - Transform Distance Relaying Scheme " , IEEE trans. pp 754-757.

12.A.M. Sharaf, L.A. Snider, K. Debnath, "A Neural Network Based Relaying Scheme for Distribution System High Impedance Fault Detection", IEEE trans. pp 321-326.

13.T.M. Lai, L. A. Snider (Senior Member- IEEE), E. Lo (Member- IEEE), and D.Sutanto (Senior Member- IEEE) "High-Impedance Fault Detection Using Discrete Wavelet Transform and Frequency Range and RMS Conversion", IEEE Transactions on Power Delivery, Vol. 20, No. 1, January 2005, pp 397-407.

14.T. M. Lai, L.A. Snider, E. Lo "Wavelet transform based relay algorithm for the detection of stochastic high impedance faults", ELSEVIER Transactions on Electric Power Systems Research , pp. 626–633.

15.M. M. Eissa (SMIEEE), G.MA. Sowilam, A.M. Sharaf(SMIEEE) "A New Protection Detection Technique for High Impedance Fault Using Neural Network" IEEE trans. pp 146-151.

16.Howard Demuth, Mark Beale "Artificial Neural Network Matlab Users Guide" Version 4.

17.Back Propagation Learning Algorithm, www.wilkipedia.org.

APPENDIX

Parameters

The parameters used in the simulation shown in figure 2.3 are shown below.

Source voltage (Vs) = 25 kv.

Source inductance (Ls) = 3 mh to 7mh.

Source resistance (Rs) = 0.3 to 0.7 ohm.

Frequency = 314 rad/sec

Resistance of the line (R_line) = 0.25 ohm/ km

Inductance of the line (L_line) = 0.99472 mh/km

Capacitance of the line (C_line) = 0.01117 microfarad /km

Load resistance (R_load) = 200 ohm.

Inductance of load (L_load) = 0.2 Henry.

Fault inductance (Lf) = 3 mh to 7 mh.

In case of linear fault fault resistance (Rf) = 20 to 30 ohm.

Incase of non linear fault the fault resistance is a function of current and is given by Rf = Rf0 (1+alfa (If/If0) ^ beta) where Rf0=20 to 30 ohm, alfa = 0.6, beta = 2.0.

DATA

The data obtained by simulation for training and testing the neural network for linear fault is shown below.

	X	v3	v5	i3	i5
lineardata= [	0.025	4.131	2.514	0.09834	0.02402
	0.025	3.953	2.085	0.1005	0.02543
	0.025	1.6	0.7916	0.09831	0.0257
	0.025	2.269	0.3301	0.1505	0.07113
	0.05	6.576	4.784	0.07748	0.007527
	0.075	5.674	8.168	0.08715	0.02525
	0.10	10.44	10.63	0.04726	0.03885
	0.125	8.425	11.50	0.009257	0.09639
	0.15	9.579	16.34	0.01831	0.1067
	0.175	11.59	16.99	0.0199	0.1055
	0.20	3.715	3.367	0.04891	0.008923
	0.225	8.626	12.99	0.01805	0.04045
	0.25	2.043	3.526	0.04884	0.01721
	0.275	7.821	12.63	0.01131	0.03882
	0.30	10.95	18.79	0.01254	0.07266
	0.3	17.67	28.95	0.04296	0.1252
	0.3	11.87	16.73	0.04627	0.05117
	0.3	13.99	21.9	0.01883	0.08308
	0.325	7.351	12.49	0.01229	0.03414
	0.35	19.41	30.54	0.04162	0.1133
	0.375	19.41	32.06	0.04211	0.1125
	0.40	23.63	39.67	0.05597	0.1321
	0.425	24.01	39.01	0.0531	0.1256
	0.45	34.02	56.35	0.08645	0.177
	0.475	37.82	63.04	0.09785	0.1922
	0.50	26.66	43.85	0.0536	0.1193
	0.5	23.75	39.69	0.04815	0.111
	0.5	38.65	64.43	0.08826	0.1761

0.5	14.63	24.02	0.01476	0.05889
0.525	39.56	65.90	0.09131	0.1776
0.55	18.42	29.52	0.02505	0.07048
0.575	10.60	17.19	0.002381	0.03056
0.60	30.25	50.45	0.0538	0.1151
0.625	26.43	43.94	0.04139	0.09561
0.65	35.17	58.7	0.06363	0.1292
0.675	103.1	170	0.223	0.3771
0.70	3.951	3.902	0.01763	0.002303
0.725	38.5	63.95	0.06684	0.1316
0.75	11.53	20.15	0.006667	0.02454
0.775	34.83	57.81	0.05041	0.1024
0.80	36.21	59.8	0.0535	0.1085
0.8	23.99	39.53	0.02622	0.06413
0.825	29.17	48.11	0.03929	0.08274
0.85	43.74	72.74	0.06544	0.1257
0.875	65.76	108.4	0.1051	0.1877
0.90	67.23	109.7	0.1073	0.1898
0.925	14.63	24.67	0.009181	0.03005
0.95	32.16	53.62	0.03689	0.07733
0.975	11.96	19.62	0.002975	0.02179
0.975	13.71	22.52	0.004955	0.02658
0.975	10.8	17.53	0.002566	0.01818
0.975	12.29	20.18	0.03285	0.0227
0.975	13.59	22.29	0.004783	0.02636
0.01	2.604	1.012	0.1067	0.03138
0.02	3.21	1.16	0.1129	0.04712
0.03	1.968	2.113	0.08831	0.02579
0.04	4.779	3.702	0.07083	0.02779
0.06	3.383	6.495	0.06361	0.04318
0.07	3.342	6.568	0.06085	0.04272
0.08	6.661	5.17	0.07833	0.01515
0.09	3.893	7.437	0.02157	0.07673
0.11	11.33	9.968	0.04029	0.03143
0.12	8.654	12.12	0.01721	0.09892

0.13	9.376	15.24	0.02287	0.1012
0.14	11.47	10.41	0.05337	0.005422
0.16	12.26	20.08	0.03523	0.1349
0.17	6.739	9.872	0.03885	0.01974
0.18	12.48	18.99	0.01961	0.09893
0.19	11.94	11.82	0.0286	0.02724
0.2	6.39	10.51	0.01651	0.04577
0.2	4.95	8.818	0.02953	0.02607
0.2	2.844	5.154	0.04261	0.005797
0.2	6.088	10.04	0.02001	0.04226
0.2	10.61	17.69	0.01295	0.09919
0.21	15.59	26.45	0.04656	0.1439
0.22	4.722	2.848	0.05885	0.03004
0.23	19.36	30.61	0.06619	0.1708
0.24	11.08	18.75	0.02446	0.1011
0.26	15.67	22.7	0.02616	0.1006
0.27	20.38	34.41	0.06436	0.1615
0.28	16.66	28.15	0.04398	0.1266
0.29	3.801	6.12	0.03552	0.003625
0.31	17.29	29.01	0.03794	0.1132
0.32	17.61	29.3	0.04903	0.1287
0.33	14.41	24.03	0.02728	0.09222
0.34	22.42	36.66	0.06248	0.1483
0.36	24.67	41.4	0.07061	0.1593
0.37	15.96	26.59	0.02782	0.0886
0.38	4.879	4.732	0.03349	0.01004
0.39	11.28	16.3	0.005322	0.04534
0.4	23.73	39.91	0.05336	0.1285
0.4	18.9	31.36	0.03767	0.1035
0.4	18.8	30.67	0.03662	0.1012
0.4	21.63	35.44	0.04515	0.1155
0.4	12.43	19.32	0.008569	0.0547
0.41	28.86	48.02	0.07642	0.1638
0.42	22.81	37.51	0.05057	0.1209
0.43	7.508	6.819	0.02838	0.006265

0.44	26.24	43.62	0.06482	0.1424
0.46	3.877	4.198	0.02629	0.005054
0.47	36.63	60.8	0.09231	0.1845
0.48	3.414	2.749	0.02594	0.005779
0.49	16.86	26.03	0.01839	0.06498
0.51	10.6	15.82	0.004049	0.03029
0.52	39.09	64.07	0.09253	0.1815
0.53	21.96	36.2	0.03649	0.09036
0.54	13.51	22.93	0.009543	0.0456
0.56	24.25	40.61	0.0443	0.1007
0.57	15.52	24.85	0.0137	0.0503
0.58	32.09	53.4	0.06349	0.1312
0.59	28.35	47.23	0.05037	0.1101
0.61	68.06	112.8	0.1548	0.2754
0.62	32.19	53.23	0.06041	0.1245
0.63	18.89	30.94	0.025	0.0666
0.64	10.55	14.14	0.005704	0.0195
0.64	3.82	5.139	0.01956	0.003523
0.64	28.13	46.74	0.04781	0.1033
0.64	13.77	21.24	0.007319	0.03696
0.64	35.28	58.11	0.06363	0.129
0.64	23.12	37.88	0.03376	0.08089
0.66	14.58	23.79	0.01243	0.04456
0.67	38.35	63.97	0.06851	0.1352
0.68	8.045	12.12	0.007213	0.01569
0.69	24.94	41.82	0.03347	0.07877
0.7	30.33	50.54	0.04717	0.1007
0.7	31.9	52.82	0.05123	0.1076
0.7	36.14	60.07	0.05945	0.121
0.7	55.5	92.25	0.1044	0.1909
0.7	21.36	35.12	0.0253	0.06568
0.71	20.23	32.94	0.02134	0.06012
0.72	28.06	46.69	0.04234	0.09329
0.73	19.37	32.88	0.0186	0.05363
0.74	76.73	127.2	0.1443	0.253

0.76	7.918	13.09	0.00669	0.01837
0.77	48.58	80.67	0.08167	0.1526
0.78	115.7	190.3	0.224	0.3764
0.79	50.23	83.43	0.08308	0.155
0.81	39.78	65.66	0.05971	0.1179
0.82	41.19	67.66	0.06123	0.1203
0.83	33.43	54.26	0.04419	0.0927
0.84	48.16	79.08	0.07303	0.1383
0.86	36.41	60.13	0.05336	0.1063
0.87	68.04	112.3	0.1085	0.1929
0.87	15.82	26.96	0.009322	0.03271
0.87	15.58	24.41	0.009251	0.0337
0.87	40.7	67.58	0.05732	0.113
0.87	26.94	43.3	0.02944	0.06851
0.87	23.01	38.74	0.02406	0.05844
0.88	39.7	65.83	0.05774	0.1127
0.89	15.69	25.91	0.01122	0.03912
0.91	33.69	55.27	0.04417	0.09067
0.92	39.65	66.11	0.05171	0.1017
0.93	12.01	20.1	0.004893	0.02259
0.94	8.568	12.96	0.005122	0.01248
0.96	23.62	39.06	0.022222	0.05497
0.97	12.14	19.68	0.002164	0.02239
0.98	8.705	13.83	0.004656	0.01248
0.99	3.73	3.791	0.01454	0.003956]

Where x = fault location if whole distance = 1.0 P.U.

v3 = R.m.s value of third harmonic component of voltage at fault point

v5 = R.m.s value of fifth harmonic component of voltage at fault point

i3 = R.m.s value of third harmonic component of fault current

i5 = R.m.s value of fifth harmonic component of fault current

Similarly the data obtained by simulation for training and testing the neural network for non-linear fault is shown below.

0.025	231	147.9	34.24	13.38
0.025	166.5	107.3	34.62	13.68

0.025	167	107.3	34.5	13.62
0.025	135.8	85.49	34.65	13.74
0.05	196.3	111.6	28.91	9.95
0.075	167.2	80.41	24.26	7.366
0.10	140.4	66.28	20.30	5.388
0.125	114.5	54.54	16.93	3.963
0.15	91.54	49.72	14.09	2.953
0.175	77.94	33.75	11.64	2.143
0.20	57.68	28.69	9.626	1.631
0.2	34.18	15.27	9.7	1.591
0.2	50.41	22.61	9.663	1.602
0.2	57.71	25.49	9.572	1.593
0.225	47.49	16.31	7.93	1.18
0.25	31.16	29.65	6.564	0.9538
0.275	36.74	5.989	5.403	0.5842
0.30	17.4	24.28	4.474	0.5438
0.325	6.038	29.65	3.701	0.4308
0.35	6.209	38.23	3.066	0.3599
0.375	5.726	36.61	2.544	0.2787
0.40	17.64	51.61	2.116	0.2799
0.425	12.64	39.51	1.766	0.1852
0.45	19.63	48.2	1.48	0.1964
0.475	6.326	3.02	1.298	0.0661
0.50	48.49	91.36	1.009	0.2821
0.5	4.606	18.19	1.071	0.06281
0.5	2.558	8.744	1.103	0.05488
0.5	4.278	5.646	1.102	0.05362
0.525	43.17	80.76	0.8706	0.2468
0.55	38.94	72.45	0.7418	0.1996
0.575	26.26	50.23	0.6501	0.1298
0.60	38.91	70.4	0.54	0.1726
0.625	38.10	68.17	0.4727	0.1592
0.65	12.09	24.36	0.4543	0.05143
0.675	24.89	46.18	0.3694	0.09109
0.70	40.56	70.47	0.2945	0.1471

0.725	68.16	115.8	0.2269	0.2378
0.75	35.31	61.57	0.2356	0.1206
0.775	115.4	192.2	0.1438	0.3794
0.80	38.07	65.1	0.1736	0.1189
0.825	52.78	89.32	0.1314	0.1563
0.85	4.156	8.887	0.1801	0.00739
0.875	14.87	25.98	0.1518	0.03444
0.90	71.84	120	0.06256	0.1926
0.925	89.55	148.90	0.08222	0.2466
0.95	33.98	57.57	0.07634	0.08393
0.975	2.92	2.881	0.1128	0.00627
0.990	3.099	3.81	0.105	0.00440
0.005	260.5	184.7	38.97	16.86
0.010	253.2	174.3	37.76	15.93
0.015	245.8	164.6	36.56	15.04
0.020	238.6	155.4	35.38	14.2
0.030	223.6	139.5	33.11	12.62
0.035	217.5	132.1	32.03	11.88
0.040	211.6	123.4	30.96	11.21
0.045	203.7	118.1	29.92	10.55
0.055	185.8	106.5	27.91	9.397
0.060	186.4	101.9	26.99	8.79
0.065	177.6	92.06	26.03	8.32
0.070	170	91.51	25.15	7.802
0.080	159.1	86.49	23.46	6.887
0.085	156.9	80.68	22.64	6.483
0.090	142.8	72.31	21.81	6.112
0.095	142.6	75.03	21.07	5.729
0.105	127.1	63.55	19.59	5.078
0.110	123	62.74	18.9	4.77
0.115	127	52.01	18.18	4.475
0.120	119.2	50.03	17.53	4.198
0.130	108.7	51.2	16.31	3.725
0.135	102.6	55.09	15.75	3.532
0.140	95.74	45.18	15.15	3.305

0.145	98.06	48.22	14.61	3.127
0.155	91.67	48.83	13.56	2.777
0.160	83.67	34.7	13.03	2.576
0.165	76.62	40.23	12.58	2.486
0.170	74.37	30.09	12.09	2.295
0.180	77.33	35.19	11.2	2.021
0.185	65.43	30.77	10.8	1.939
0.190	61.16	32.65	10.4	1.857
0.195	57.41	26.05	10	1.73
0.205	54.23	21.12	9.254	1.499
0.210	53.35	29.31	8.919	1.468
0.215	53.81	23.5	8.572	1.329
0.220	56.41	19.44	8.234	1.196
0.230	51.05	16.16	7.625	1.054
0.235	41.94	21.24	7.357	1.072
0.240	41.82	24.44	7.078	1.022
0.245	33.61	37.68	6.827	1.05
0.255	33.02	18.55	6.301	0.8469
0.260	26.44	28.78	6.078	0.8523
0.265	24.62	26.58	5.842	0.7919
0.270	40.47	13.52	5.613	0.6216
0.280	19.18	29	5.204	0.6903
0.285	18.57	27.23	5.012	0.6382
0.290	17.37	26.04	4.829	0.604
0.295	16.5	29.9	4.644	0.5893
0.305	24.33	7.574	4.303	0.4319
0.310	16.09	17.87	4.15	0.4647
0.315	11.41	25.97	3.985	0.4462
0.320	25.04	1.417	3.85	0.3422
0.330	2.27	36.11	3.558	0.4339
0.335	5.326	27.65	3.426	0.3691
0.340	2.521	38.64	3.3	0.3944
0.345	3.08	37.6	3.176	0.3696
0.355	10.39	20.22	2.968	0.2843
0.360	5.353	21.6	2.861	0.2823

0.365	14.26	6.51	2.766	0.2195
0.370	9.65	11.99	2.657	0.2112
0.380	2.583	29.56	2.455	0.2385
0.385	9.184	14.24	2.391	0.1923
0.390	12.59	43.89	2.272	0.2588
0.395	5.138	25.27	2.219	0.206
0.405	7.37	33.63	2.057	0.2126
0.410	20.05	54.25	1.964	0.2716
0.415	15.02	44.49	1.892	0.2103
0.420	21.18	54.55	1.821	0.2502
0.430	0.5472	17.5	1.736	0.1347
0.435	13.69	39.65	1.646	0.1758
0.440	8.229	29.96	1.611	0.1533
0.445	9.375	2.307	1.587	0.08667
0.455	19.03	46.77	1.435	0.1856
0.460	2.13	16.84	1.412	0.08356
0.465	8.117	0.9534	1.389	0.07151
0.470	38.48	77.4	1.266	0.2725
0.480	23.14	50.98	1.2	0.1704
0.485	8.642	26.6	1.19	0.1091
0.490	29.01	59.89	1.118	0.1978
0.495	15.24	36.45	1.107	0.126
0.505	23.86	50.14	1.019	0.1602
0.510	12.47	30.8	1.002	0.09137
0.515	12.29	29.99	0.9707	0.08105
0.520	13.76	32.54	0.9463	0.1034
0.530	26.29	52.73	0.8653	0.1541
0.535	22.59	46.15	0.8396	0.1221
0.540	27.78	54.45	0.81	0.155
0.545	17.14	36.66	0.8089	0.1065
0.555	30.91	58.82	0.7272	0.164
0.560	24.49	48	0.7165	0.1327
0.565	24.68	47.89	0.6951	0.1221
0.570	37.66	69.08	0.6626	0.1877
0.580	21.6	42.29	0.6381	0.1027

0.58	7.083	15.11	0.6826	0.03956
0.58	40.5	71.83	0.6155	0.1873
0.58	14.94	30.24	0.6644	0.07632
0.585	34.67	63.52	0.6095	0.1709
0.590	4.456	13.45	0.6421	0.0359
0.595	23.84	45.57	0.586	0.1136
0.605	21.8	41.74	0.5574	0.1036
0.610	40.07	71.95	0.5097	0.1751
0.615	29.34	53.95	0.5086	0.131
0.620	29.52	53.9	0.4944	0.1245
0.630	69.85	120.5	0.4013	0.2853
0.635	77.77	133.4	0.3756	0.3117
0.640	22	40.93	0.4588	0.09572
0.645	28.48	51.62	0.4266	0.1136
0.655	2.253	7.902	0.4564	0.008546
0.660	18.29	34.35	0.4124	0.06966
0.665	40.39	70.97	0.3659	0.1567
0.670	24.47	44.47	0.3863	0.0975
0.680	35.74	63.07	0.3432	0.1347
0.685	28.68	51.17	0.3485	0.1079
0.690	32.53	57.39	0.3351	0.1228
0.695	35.92	62.87	0.3157	0.1351
0.705	40.61	69.82	0.3052	0.1513
0.710	26.71	47.35	0.3008	0.0951
0.715	84.45	142.7	0.2219	0.3015
0.720	18.93	33.66	0.3033	0.0575
0.730	10.82	20.69	0.306	0.03405
0.735	10.04	16.37	0.3058	0.01996
0.740	36.91	63.84	0.2483	0.1286
0.745	90.74	152.2	0.1868	0.313
0.755	18.72	33.35	0.255	0.06328
0.760	14.61	26.59	0.2556	0.03734
0.765	18.97	33.7	0.2429	0.06213
0.770	10.41	19.72	0.2483	0.02797
0.780	22.73	39.79	0.2202	0.07181

0.785	50.08	85.3	0.1686	0.1576
0.790	15.62	27.76	0.2213	0.04584
0.795	50.33	85.52	0.164	0.1583
0.805	22.62	39.43	0.1905	0.06335
0.810	35.39	59.67	0.1801	0.11
0.815	18.72	32.55	0.1892	0.057
0.820	65.58	110.4	0.1125	0.1914
0.830	35.21	59.35	0.1541	0.1036
0.835	15.72	26.15	0.1767	0.03625
0.840	39.69	67.57	0.1378	0.1185
0.845	23.36	40.78	0.1572	0.05518
0.855	7.012	6.96	0.1833	0.004531
0.860	22.81	39.03	0.1456	0.06686
0.865	3.04	6.678	0.1741	0.002403
0.870	29.55	50.77	0.1272	0.0719
0.880	19.36	32.48	0.1389	0.05201
0.885	28.94	48.93	0.1153	0.07685
0.890	8.563	14.96	0.1515	0.014
0.895	62.44	104.9	0.07866	0.1736
0.905	53.71	90.1	0.08139	0.1498
0.910	36.42	61.73	0.09142	0.09598
0.915	51.3	85.44	0.08763	0.1429
0.920	2.289	1.255	0.1425	0.0092
0.930	49.02	82.5	0.07021	0.1292
0.935	52.42	88.1	0.06201	0.1356
0.940	47.47	79.61	0.073	0.1256
0.945	41.15	69.26	0.07441	0.1065
0.955	30.09	51.01	0.08051	0.07377
0.960	10.66	18.13	0.1064	0.0201
0.965	20.04	34.12	0.0901	0.04592
0.970	14.71	25.19	0.09512	0.03084
0.980	4.65	7.636	0.1058	0.001792
0.985	4.194	6.614	0.1046	0.00022