A Literature Review on Methodology & Fundamentals of Development of Mathematical Model through Simulation of Artificial Neural Network

Mr. P. A. Chandak [#1], Ms. A. R. Lende[#2], Mr. J. P. Modak 3[#3]

#1 Asst. Professor, DMIETR, Wardha, 9860032565.
#2 Ex- Asst. Professor, DMIETR, Wardha, 9860032565.
#3 Professor and Dean (R&D), PCE, Nagpur.

ABSTRACT

The Artificial neural network is most forthcoming tool to be used for future forecasting, prediction, decision making, etc. This document evaluates the fundamentals of artificial neural network, its parameters and explains the terminologies involved in it.
The article tries to extend perception towards mathematics involved in a single neuron, a single layer multi-neuron and multilayer multi-neuron network of a feed forward network.
Going through the various terms of artificial neural network the documentary discusses the methodology to emerge with a mathematical model that could perform the same function as the neural network.
This mathematical model may perhaps useful in some of the applications like incorporation of artificial intelligence to a machine, decision making in case of hazard situations on shop floors, limiting conditions at production plants, etc.

Key words: MATLAB, Artificial Neural Network (ANN), Mathematical Modelling.

Corresponding Author: Mr. P. A. Chandak

1. INTRODUCTION TO ARTIFICIAL NEURAL NETWORK

The artificial Neural Network (ANN) is detailed in three sub titles namely fundamentals, classes & training and verification as below.

1.1.1. Neural Network

It is the most simplified model of the brain. It is analogous to call it as a function approximator which transforms input into output with best of its ability.

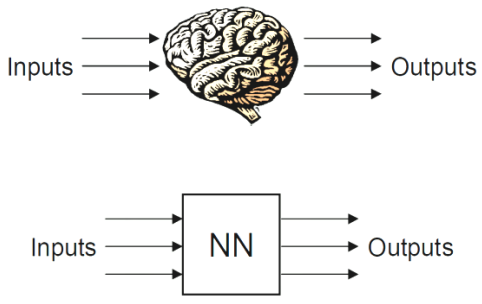

Figure 1: Exploring Function of Brain [3]

1.1.2. Artificial Neural Network

The model of a biological neuron is as shown in figure 2 and artificial neuron in figure 3. It includes cell body analogous to neuron and synapse as the information carrier analogous to weights in case of artificial neural network.

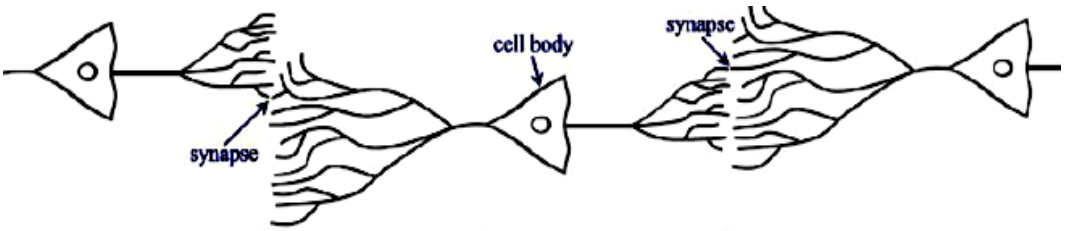

Figure 2: Basic Diagram illustrating Biological Neuron[3]

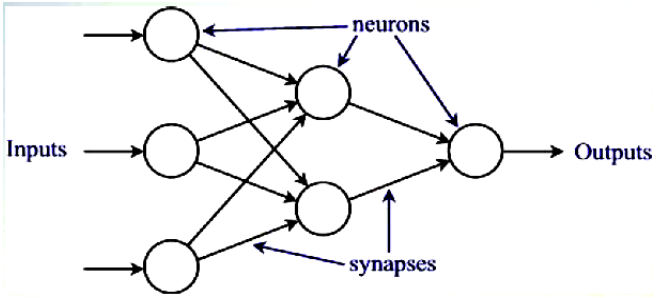

Figure 3: Basic model of Artificial Neuron [3]

The artificial neural network is composed of many neurons like biological neuron in a structured way and tries to perform the same function as biological neuron. The input vectors are provided to the network and output of the network is compared with the desired target vectors.

1.1.3. Application Areas of Neural Network

The applications of Artificial Neural Networks are widely spread. Those can be basically classified in three categories as

Classification: Pattern recognition, feature extraction, image matching, clustering data.

Noise Reduction: Recognize patterns in the inputs and produce noiseless outputs.

Prediction: Extrapolation based on historical data like fitting function or time series prediction.

1.1.4. Abilities of Neural Network

The basic abilities of the neural networks are

Ability to learn

NN's figure out how to perform their function on their own and determine their function based only upon sample inputs.

Ability to generalize

It. produces reasonable outputs for inputs it has not been taught how to deal with.

1.1.5. Working of Neural Networks

The output of a neuron is a function of the weighted sum of the inputs plus a bias. The function of the entire neural network is simply the computation of the outputs of all the neurons which is an entirely deterministic calculation.

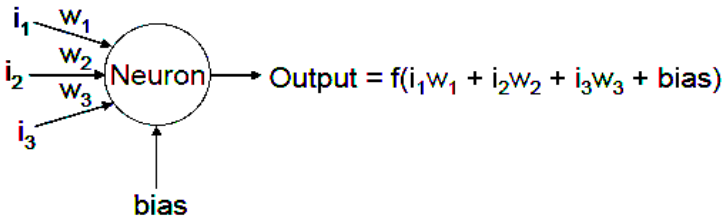

Figure 4: Basic mathematical model of Artificial Neuron [3]

The symbol 'W' stands for weights and 'f' stands for activation function. The activation function is used to calculated output response of neuron [2]. For neurons in one layer same activation function is used. There are linear as well as nonlinear activation functions.

1.1.6. Weights in the Neural Network

Weight is information used by the network to solve the problem [3]. The weights in a neural network are the most important factor in determining its function. Training is the act of presenting the network with some sample data and modifying the weights to better approximate the desired function.

1.1.7. Types of Training

There are two main types of training

Supervised Training

In this training, the neural network is supplied with inputs and the desired outputs and response of the network to the inputs is measured. The weights are modified to reduce the difference between the actual and desired outputs.

Unsupervised Training

In this type of training, neural network is supplied with only inputs and neural network adjusts its own weights so that similar inputs cause similar outputs The network identifies the patterns and differences in the inputs without any external assistance.

1.2 Classes [1]

Important ANN Parameters are

- Network Topology
- Number of Layers in the network
- Number of Neurons
- Learning Algorithm
- Training Methods
- Activation functions/ Transfer Functions used
- Type of Networks

- Performance function

1.2.1 Network Topology [3]

Neurons in the network are interconnected in certain ways which is called as topology [3]. The various topologies used are as follows [3].

- Multilayer feed forward: Generally used for Historical data prediction.
- Bilayer feedforward/ backward: Generally used in continuous learning.
- Monolayer Heterofeedforback: Generally used in time series type network.

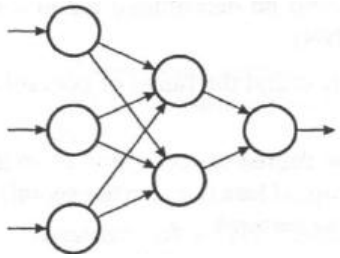

Figure5: Multilayer Feed Forward [2]

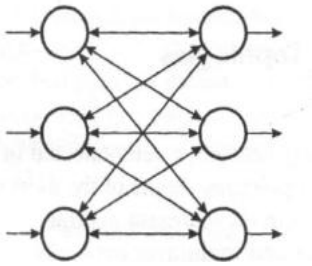

Figure6: Bilayer Feed Forward [2]

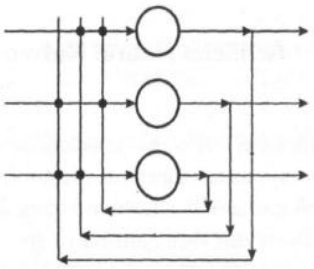

Figure 7: Monolayer Hetero feedback [2]

1.2.2 Number of Hidden Layers in Neural Network

For linear problems one hidden layer is sufficient. While two or more layers are required when the function is discontinues or non-linear. Higher the number of hidden layers more memory and time gets consumed. Hence optimum hidden layers must be used in order to avoid inaccuracy and conjunction in network.

1.2.3 Learning Algorithm

It is the mathematical tool that outlines the method and speed of the network. It characteristically starts error functions or energy function. Every learning algorithm calculates error in different ways and the weights are adjusted accordingly. Objective of learning is to reach to minimum error that corresponds to a set of weights. The various learning algorithms are [3][4]

- Delta Learning
- Gradient descent weight / bias learning
- Hebb weight learning
- Kohonen weight learning
- LVQ1 weight learning

1.2.4 Network Training styles

The network has to be trained for the set of input and output variables in order to avail prediction of the database. The training is an iterative process through which the weight matrix and bias matrix of input and output layer is updated until the network provides satisfactory results. There are various methods through which the weights and biases are updated while training. Each method has its significant application and could not be useful for each case. Hence best of the learning style has to be adopted to get minimum complications and best results.

The various training styles are [4],

- Batch training with weight and bias learning rule
- BFGS quasi-Newton backpropagation
- Bayesian regularization
- Cylindrical order incremental update
- Powell-Beale conjugate gradient backpropagation
- Fletcher-Powell conjugate gradient backpropagation
- Polak-Ribiere conjugate gradient backpropagation
- Gradient descent backpropagation
- Gradient descent backpropagation with momentum and adaptive learning rule backpropagation
- Levenberg-Marquardt backpropagation

1.2.5 Types of Transfer/Activation Functions

Selection of transfer function depends on layer number, input output relationship. The various types of transfer functions are

The figure 8 illustrates various transfer functions used with their symbols. If observed thoroughly the curve generated by each transfer function with the variation of input parameter is made known. The linear transfer functions like “purelin”, “satlin” are generally used at the final layer of multilayer networks. The transfer functions like “logsig”, “tansig” are used at the middle layers of the multilayer networks.

While, if one 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).

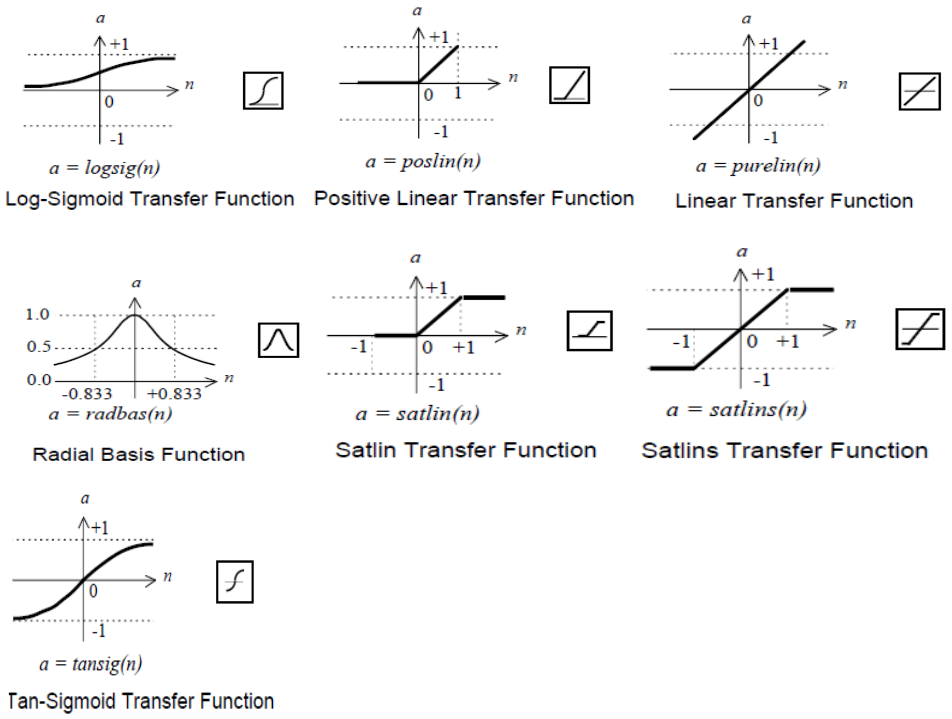

Figure 8: Transfer functions and their symbols [4]

1.2.6 Types of Networks [4]

- Multilayer feedforward network
  Generally used for prediction based on historical data
- Dynamic network
  Used for prediction of time varying patterns like financial market, sorting, fault detection, etc
- Pattern network
  Used for pattern recognition
- Self Organizing and Learning Vector Quantization Nets
  Used for prediction of clustering type of data

1.2.7 Performance Function

The performance of the network is observed through the types of errors as given below

- Mean squared error performance function(mse)
  Error can be calculated as follows. [4]

$$mse = \frac{1}{N}\sum_{i=1}^{N}(e_i)^2 = \frac{1}{N}\sum_{i=1}^{N}(t_i - a_i)^2$$

- Mean absolute error performance function(mae)
  Error can be calculated as follows. [4]

$$mae = \frac{1}{N}\sum_{i=1}^{N}(e_i) = \frac{1}{N}\sum_{i=1}^{N}(t_i - a_i)$$

- Sum squared error performance function(sse)
  Error can be calculated as follows. [4]

$$sse = \sum_{i=1}^{N}(e_i)^2 = \sum_{i=1}^{N}(t_i - a_i)^2$$

1.3 Training and Verification

The set of all known samples is broken into three orthogonal (independent) sets:

- Training set

It is a group of samples used to train the neural network and it is mostly chooses 70% of the available sample set.

- Testing set

It is a group of samples used to test the performance of the neural network used to estimate the error during training and approximately 15% of the available sample set is selected for testing.

- Validation set

It is a group of samples used to validate the performance after training. Approximately 15% of the available sample set is used for validation of network after training. Until the network is validated training is continued.

The network is generated by choosing the ANN parameters to the desired value. Then the training of the network is carried out for the desired goal until validation. Number of networks can be tried to attain the desired goal. Its performance is observed through various plots like regression, etc.

2. Mathematical Modelling

2.1 Mathematics Involved in Architecture of Neural Network

2.1.1 Architecture of Simple Neuron

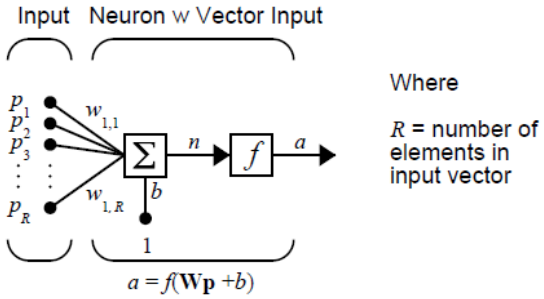

Figure 9: Simple neuron model [4]

The simple neuron as in figure 9 can be extended to handle inputs that are vectors. A neuron with a single R-element input vector is shown below. Here the individual input elements p_1, p_2,... p_R are multiplied by weights $w_{1,1}$, $w_{1,2}$, ... $w_{1,R}$ and the weighted values are fed to the summing junction. Their sum is simply Wp, the dot product of the (single row) matrix W and the vector p.
The neuron has a bias b, which is summed with the weighted inputs to form the net input n. The net input n is the argument of the transfer function.

$$n = w_{1,1}p1 + w_{1,2}p_2 + ... + w_{1,R}p_R + b$$

This expression can, of course, be written in MATLAB® code as

$$n = W*p + b$$

Where,
W = Weight matrix
p = Input vector and
b = Bias Vector

The simple model of a neuron can also be shown as in figure 10.

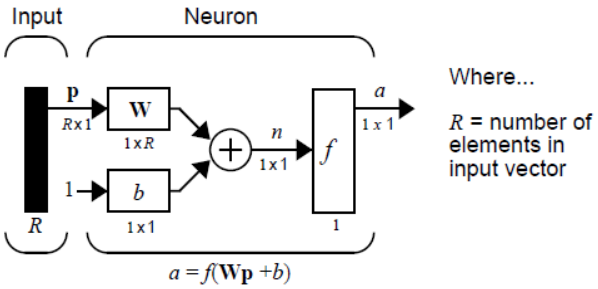

Figure 10: Simplified neuron model [4]

Here the input vector p is represented by the solid dark vertical bar at the left. The dimensions of p are shown below the symbol p in the figure as Rx1. (Note that a capital letter, such as R in the previous sentence, is used when referring to the size of a vector.) Thus, p is a vector of R input elements. These inputs post multiply the single-row, R-column matrix W. As before, a constant 1 enters the neuron as an input and is multiplied by a scalar bias b. The net input to the transfer function f is n, the sum of the bias b and the product Wp. This sum is passed to the transfer function f to get the neuron's output a, which in this case is a scalar. If there were more than one neuron, the network output would be a vector.
A layer of a network is defined in the figure 61. A layer includes the weights, the multiplication and summing operations (here realized as a vector product Wp), the bias b, and the transfer function f. The array of inputs, vector p, is not included in or called a layer.
As with the "Simple Neuron", there are three operations that take place in the layer: the weight function (matrix multiplication, or dot product, in this case), the net input function (summation, in this case) and the transfer function. Each time this abbreviated network

notation is used, the sizes of the matrices are shown just below their matrix variable names. These notations allow to understand the architectures and follow the matrix mathematics associated with them.

2.1.2 Architecture of Simple Neuron

Two or more of the neurons shown earlier can be combined in a layer, and a particular network could contain one or more such layers. First consider a single layer of neurons.

- A Network of Single Layered Neurons

A one-layer network with R input elements and S neuron is as follows.

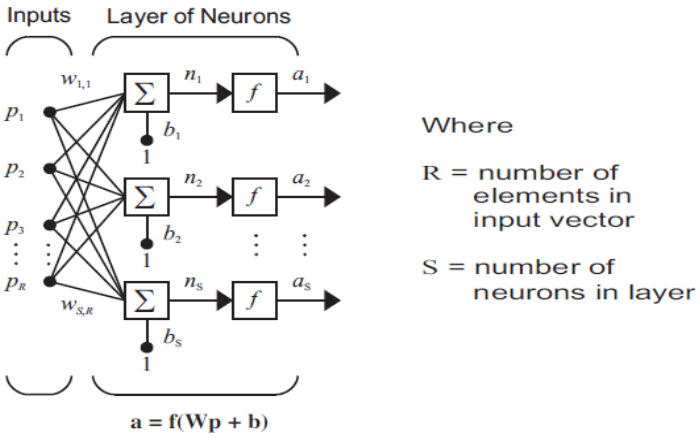

Figure 11: Architecture of a layer with R input elements and S Neurons [4]

In this network, each element of the input vector p is connected to each neuron input through the weight matrix W. The i^{th} neuron has a summer that gathers its weighted inputs and bias to form its own scalar output n(i). The various n(i) taken together form an S-element net input vector n. Finally, the neuron layer outputs form a column vector a. The expression for **a** is shown at the bottom of the figure. Observe that it is common for the number of inputs to a layer to be different from the number of neurons (i.e., R is not necessarily equal to S). A layer is not constrained to have the number of its inputs equal to the number of its neurons.

A single (composite) layer of neurons having different transfer functions could be created simply by putting two of the networks shown earlier in parallel. Both networks would have the same inputs, and each network would create some of the outputs.

The input vector elements enter the network through the weight matrix W.

$$\mathbf{W} = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,R} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,R} \\ & & & \\ w_{S,1} & w_{S,2} & \cdots & w_{S,R} \end{bmatrix} \quad [4]$$

It may be observed the row indices on the elements of matrix W indicate the destination neuron of the weight, and the column indices indicate which source is the input for that

weight. Thus, the indices in $w_{1,2}$ say that the strength of the signal from the second input element to the first (and only) neuron is $w_{1,2}$. The S neuron R input one-layer network also can be drawn in abbreviated notation as,

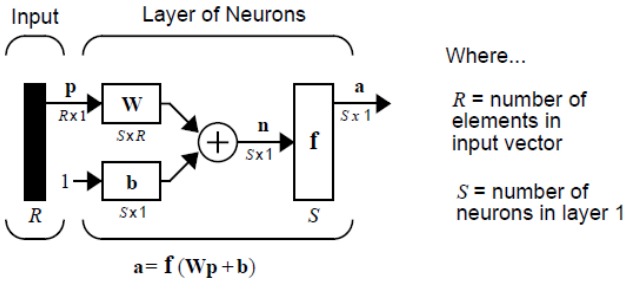

Figure 12: Simplified Architecture of a layer with R input Elements and S Neurons [4]

Here p is an R length input vector, W is an SxR matrix, and a & b are S length vectors. As defined previously, the neuron layer includes the weight matrix, the multiplication operations, the bias vector b, the summer, and the transfer function blocks.

- Inputs and Layers

To describe networks having multiple layers, the notation must be extended. Specifically, it needs to make a distinction between weight matrices that are connected to inputs and weight matrices that are connected between layers. It also needs to identify the source and destination for the weight matrices. We will call weight matrices connected to inputs input weights; we will call weight matrices connected to layer outputs layer weights. Further, superscripts are used to identify the source (second index) and the destination (first index) for the various weights and other elements of the network. To illustrate, the one-layer multiple input network shown earlier is redrawn in abbreviated form below.

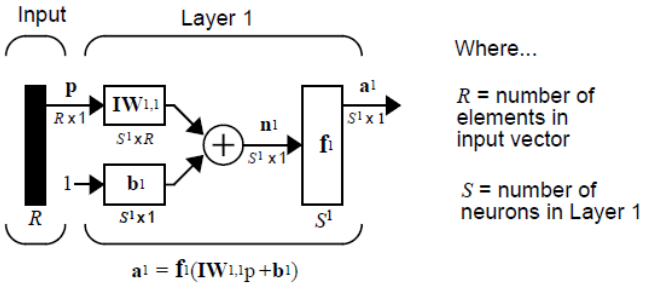

Figure 13: Architecture of a layer with R input Elements and S Neurons [4]

As one can see, the weight matrix connected to the input vector p is labeled as an input weight matrix ($IW_{1,1}$) having a source 1 (second index) and a destination 1 (first index). Elements of layer 1, such as its bias, net input, and output have a superscript 1 to say that they are associated with the first layer.

- Multiple Layers of Neurons

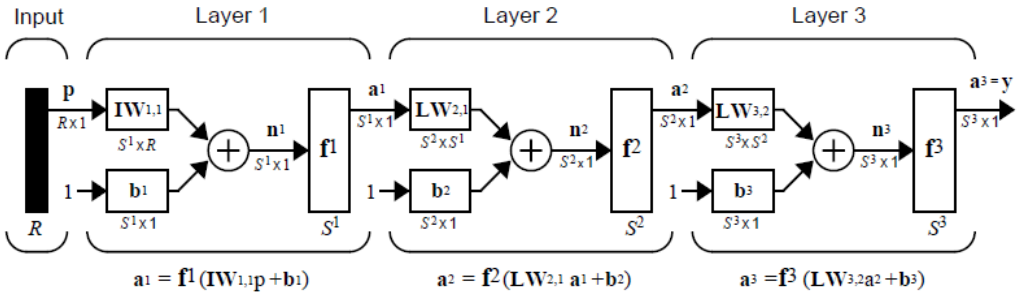

Figure 14: Architecture of Multilayer feedforward Network [4]

"Multiple Layers of Neurons" uses layer weight (LW) matrices as well as input weight (IW) matrices.

The network shown above has R1 inputs, S^1 neurons in the first layer, S^2 neurons in the second layer, etc. It is common for different layers to have different numbers of neurons. A constant input 1 is fed to the bias for each neuron.

Note that the outputs of each intermediate layer are the inputs to the following layer. Thus layer 2 can be analyzed as a one-layer network with S^1 inputs, S^2 neurons, and an S^2xS^1 weight matrix W^2. The input to layer 2 is a^1; the output is a^2. Now that all the vectors and matrices of layer 2 have been identified, it can be treated as a single-layer network on its own. This approach can be taken with any layer of the network.

The layers of a multilayer network play different roles. A layer that produces the network output is called an output layer. All other layers are called hidden layers. The three-layer network shown earlier has one output layer (layer 3) and two hidden layers (layer 1 and layer 2).

The architecture of a multilayer network with a single input vector can be specified with the notation $R\text{-}S^1\text{-}S^2\text{-}...\text{-}S^M$, where the number of elements of the input vector and the number of neurons in each layer are specified.

2.2 Generation of Mathematical Model for a Trained Network [4]

As discussed earlier (in figure 14) each layer has an output which becomes input for the successive layer. The output of each layer is governed by a function, a weight matrix and a bias vector which can be represented as

$$\text{Output of layer 1} = a_1 = f^1 (IW_{1,1}.p + b_1)$$

.....Equation 1

Where $IW_{1,1}$ = Input layer (layer1) weight matrix
p = input Vector for layer 1
b_1 = bias vector for layer 1

In the same way,

$$\text{Output of layer 2} = a_2 = f^2 (LW_{2,1}. a_1 + b_2)$$

.....Equation 2

Where $LW_{2,1}$ = layer weight matrix for layer 2
a_1 = input Vector for layer 2
b_2 = bias vector for layer 2

In addition,

$$\text{Output of layer 3} = a_3 = f^3 (LW_{3,2}. a_2 + b_3)$$

.....Equation 3

Where $LW_{2,1}$ = layer weight matrix for layer 3
a_3 = input Vector for layer 3
b_3 = bias vector for layer 3

Hence from above three equations that is Equation1, Equation2, Equation3 it can be concluded as [4]

$$\text{Output of layer 3} = a_3 = f^3 (LW_{3,2}. f^2(LW_{2,1}. f^1(IW_{1,1}.p + b_1) + b_2) + b_3)$$

..... Equation 4

Therefore, it is possible that the above equation will give the same performance as the neural network if all the above unknown variables are known.

3. CONCLUSION

So far the researchers are using the neural network itself for prediction of data which was eccentrically acquiring a computational tool like central processing unit in order to process the network through specific software. For the machine tools like automobile, small scale production units, etc it is unfavourable to use such processors to implement artificial intelligence as it adds lot to the primary cost of the tool. Hence this equation could open the limitations to large extent in desire of artificial intelligent machine tools at common man's door.

The above mentioned equation could play a vital role if used to predict the target for seen and unseen data. This will assist the machine tool itself to take the decision accordingly in favour of desired output. With the help of linear electronic circuit, microprocessors, etc a controller may be designed that will process the equation timely and give digital signals to an analogue output converter. This analogue output could then be used to govern value of one or more input parameters to get desired output of the machine tool.

4. ACKNOWLEDGEMENTS

The authors of this paper would like to thank mathworks.com for development of neural networks user guide which help us much in expansion of idea about this article.

5. REFERENCES

[1] Limin Fu, "Neural Networks in Computer intelligence",Tata McGraw-Hill Education, 2008.

[2] S. N. Shvanandam, Introduction to Neural Network using Matlab 6.0, McGraw Hill publication.

[3] Stamtios V. Kartaplopoulos, Understanding Neural Networks and Fuzzy Logics, IEEE Press.

[4] Mathworks.com, Neural Network Toolbox TM 7 User's Guide, R2010a.