ISSN: 2348 - 2117

International Journal of Engineering Technology and Computer Research (IJETCR)
Available Online at www.ijetcr.org
Volume 2, Issue 4, Page No. 01-05

A General Multilayer Perceptrons Feed Forward Neural Network Algorithm for Learning Capability

Dr. Akash Saxena[1*], Jitendra Joshi[2]

[1] Research Supervisor, Jayoti Vidyapeeth Women's University, Jaipur, Rajasthan, India

[2] Ph.D. Scholar, Jayoti Vidyapeeth Women's University, Jaipur, Rajasthan, India

E-mail:lect.jitendra29@gmail.com

ABSTRACT
A multi layer perceptrons feed forward neural network consists of layers of interconnected artificial neurons, all these are interchangeable. The number of layers in a neural network is the number of layers of perceptrons. The feed forward network is important artificial neural network structure [1]. Presently this is fast learning algorithm of perceptrons training and that's algorithm for perceptrons learning capability. The overall computation approach of this algorithm use for exploring the perceptrons training and learning capability in multilayer network [2].

Keywords: *Multilayer, Perceptrons, Feed Forward, Neural Network, Learning.*

I. INTRODUCTION

The multilayer perceptrons feed forward network is composed of a hierarchy of processing unit or layers. That is organized in a series of two or more mutually exclusive sets of neurons or layers. The first input layer this layer serves as a holding site for the inputs that is applied to the neural network [3]. The last output layer is the point at which the overall mapping of the network input is available. The internal layer is hidden layer between these two extremes extend out zero or more layers.

Everyone unit in the hidden layer and in the output layer processes its weighted input to produce an output. The input layer neurons are linked to hidden layer neurons. The weights on these links are referred to as input-hidden layer weights. The connections coming out of an input layer have weights associated with them. A weight going to hidden layer D_h from input layer B_j would be labeled w_{hj}. The hidden layer neurons and the corresponding weights are referred to as output-hidden layer weighted.

A multilayer perceptrons feed forward neural network leaning capability follows of the constraints and assumptions of learning features [4]. The development of this algorithms that is represent of fast learning of perceptrons training and perceptrons learning capability. The overall computation approach of this algorithm use for exploring the perceptrons training and learning capability in multilayer feed forward neural network [5].

II. MULTILAYER PERCEPTRONS FEED FORWARD NEURAL NETWORK

A neural network consists of layers of interconnected artificial neurons. A neuron in a neural network is sometimes called units or layer, all these are interchangeable. A multilayer feed forward neural network is an interconnection of perceptrons in which data and calculations flow in a single direction, from the input data to the outputs. The number of layers in a neural network is the number of layers of perceptrons [6].

The most common network structure we will deal with is a network with one layer of hidden units, so for the rest of these notes, we will make the assumption that we have exactly one layer of hidden units in addition to one layer of input units and one layer of output units [7]. This structure is called multilayer because it has a layer of processing units in addition to the output units. These networks are called feed-forward network because the output from one layer of neurons feeds forward into the next layer of neurons.

There are never any backward connections, and connections never skip a layer. Typically, the layers are fully connected, meaning that all units at one layer are connected with all units at the next layer [8]. So, this means that all input units are connected to all the units in

the layer of hidden units, and all the units in the hidden layer are connected to all the output units.

Usually, determining the number of input units and output units is clear from application. However, determining the number of hidden units is a bit of an art form, and requires experimentation to determine the best number of hidden units. Too few hidden units will prevent the network from being able to learn the required function, because it will have too few degrees of freedom [9]. Too many hidden units may cause the network to tend to over fit the training data, thus reducing generalization accuracy. In many applications, some minimum number of hidden units is needed to learn the target function accurately, but extra hidden units above this number do not significantly affect the generalization accuracy, as long as cross validation techniques are used. Too many hidden units can also significantly increase the training time [10].

Each connection between nodes has a weight associated with it. In addition, there is a special weight is w_0 that feeds into every node at the hidden layer and a special weight is z_0 that feeds into every node at the output layer. These weights are called the bias, and set the thresholding values for the nodes [11]. Initially, all of the weights are set to some small random values near zero. The training of our network will adjust these weights using the learning capability algorithm that we will describe so that the output generated by the network matches the correct output.

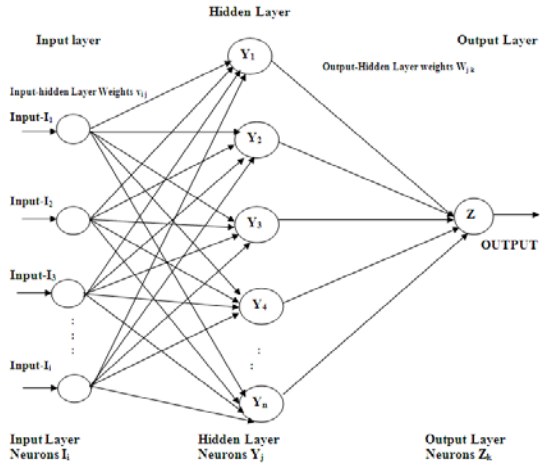

Figure1.1: Multilayer Perceptrons Feed Forward Neural Network

III. PERCEPTRONS PROCESSING AT THE LAYERS

Everyone unit in the hidden layer and in the output layer processes its weighted input to produce an output. This can be done slightly differently at the hidden layer, compared to the output layer. These are working design figure as below.

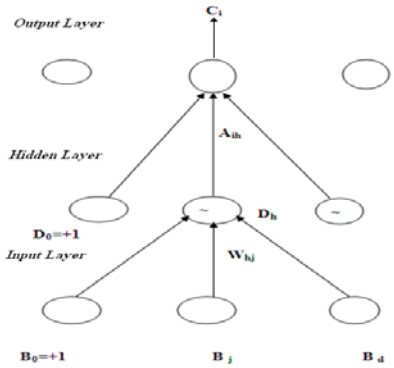

Figure 1.2: Perceptrons Processing at the Layers

A. Input Layer

The input data users provide for network comes through the input units. No processing takes place in an input layer it simply feeds data into the system. The input layer neurons are linked to hidden layer neurons [12]. The weights on these links are referred to as input-hidden layer weights.

The value coming out of an input layer is labeled B_j, for j going from 1 to d, representing d input layer. There is also a special input layer labeled B_0, which always has the value of 1. This is used to provide the bias to the hidden layer.

B. Hidden Layer

The connections coming out of an input layer have weights associated with them. A weight going to hidden layer D_h from input layer B_j would be labeled w_{hj}. The bias input layer, B_0, is connected to the entire hidden layer, with weights w_{h0}. In the training, these bias weights, w_{h0}, are treated like all other weights [13], and are updated according to the perceptrons learning algorithm. Remember, the value coming out of B_0 is always 1.

Each hidden layer calculates the weighted sum of its inputs and applies a thresholding function to determine the output of the hidden layer [14]. The weighted sum of the inputs for hidden layer D_h is calculated as:

$$\sum_{j=0}^{d} W_{hj} B_j$$

C. Output Layer

The hidden layer neurons and the corresponding weights are referred to as output-hidden layer weighted [15]. We start out the same as we did with the hidden layer, calculating the weighted sum. We label the weights going into output layer i from hidden layer h as v_{ih}. Just like the input layer, we also have a bias at the hidden layer. So, each output layer has a bias input from hidden layer D_0, where the input from D_0 is always 1 and the weights associated with that input are trained just like all the other weights.

So, output unit **i** compute the weighted sum of its inputs as:

$$o_i = \sum_{h=0}^{H} Aij\, D_h$$

IV. CONSTRAINTS AND ASSUMPTIONS OF LEARNING

A multilayer perceptrons feed forward neural network leaning capability depends on following features:

- The types of input layer weight values and desired outputs
- The underlying function of layers to be realized
- The allowable number of neural network layers
- The bound on the input number of weight values in a layer
- The allowable of activation function

We should also consider the possibility that the mapping accuracy extend beyond static characterization of learning, in that derivatives of function to be approximated are also well approximated by the neural network [16].

V. FAST LEARNING ALGORITHM OF PERCEPTRONS TRAINING

This algorithm shows a fast learning process for perceptrons training, where weights are adjusted to curtail error whenever the output does not equal to the desired output of given input [17]. This algorithm will be complete in following steps.

Step -1: If the output is accurate then no adjustment of weight is done.

$$W_{hj}^{k+1} = W_{hj}^{k}$$

STEP-2: If the output is one but should have been zero then the weight is down-ward or decreasing on the active input link.

$$W_{hj}^{k+1} = W_{hj}^{k} - \alpha . B_i$$

STEP-3: If the output is zero but should have been one then the weight is up-ward or increasing on the active input link.

$$W_{hj}^{k+1} = W_{hj}^{k} + \alpha . B_i$$

Where:

W_{hj}^{K+1} is the new adjusted weight

W_{hj}^{K} is the previous weight

B_i is the input and ∞ is the learning rate parameter

α small lead to slow and α large lead to fast learning

VI. ALGORITHM FOR PERCEPTRONS LEARNING

It will show the mathematical representation of multilayer perceptrons learning capability in neural network.

STEP-1: Create a perceptrons with (n+1) input neurons I_0, $I_1, I_2 \ldots\ldots\ldots I_n$ where $I_0 = 1$ is the bias input.

Let ***0*** be the output neuron.

STEP-2: Initialize the weight W = (w_0, w_1, $w_2 \ldots\ldots\ldots w_n$) to random weights.

STEP-3: Iterate through the input pattern I_j is the training set using the weight sets, as compute the weighted sum of the inputs net….

$$j = \sum_{i=1}^{n} I_i W_i$$

For each input pattern is…. j.

STEP-4: Compute output Y_J using the step-3 function.

$$Y_j = f(\text{net } j) = \begin{cases} 1 \text{ if net } j >= 0 \\ 0 \text{ if net } j < 0 \end{cases}$$

$$\text{Where} \quad \text{net } j = \sum_{i=1}^{n} I_i W_{ij}$$

STEP-5: Compare the computed output Y_j with the target output Y_j for each input pattern j. If all the input pattern has been classified in the approved manner, then output study the weight and exit.

STEP 6: Otherwise, update the weight as given below

i. If the computed output Y_j is 1 but should have been 0.

ii. Then $W_i = W_{I-} \alpha I_i$, i = 0,1,2,3.....n

iii. If the computed output Y_j is 0 but should have been 1.

iv. Then $W_i = W_{i+} \alpha I_i$, i = 0,1,2,3.....n

Where:

α is the learning factor and is constant.

STEP 7: Now go to step 3.

STEP 8: Exit

VII. CONCLUSION

In this paper, we present a general multilayer perceptrons feed forward neural network algorithm for learning capability. In this paper, the multilayer perceptrons feed forward network is composed of a hierarchy of processing unit. A multilayer feed forward neural network is an interconnection of perceptrons in which data and calculations flow in a single direction, from the input data to the outputs. The number of layers in a neural network is the number of layers of perceptrons. Perceptrons processing completed with three layers as input, hidden and output layer. The multilayer perceptrons feed forward neural network leaning capability depends on features of constraints and assumptions of learning. That is represent of fast learning of perceptrons training and perceptrons learning capability. The overall computation approach of the algorithm use for exploring the perceptrons training and learning capability in multilayer feed forward neural network.

VIII. REFERENCES

[1] R. Callan. The Essence of Neural Networks. Prentice Hall, 2008.

[2] G.F. Luger. Artificial Intelligence. Addison Wesley, 2005.

[3] D.E. Rumelhart and J.L. McClelland. Parallel Distributed Processing: Explorations in the Microstructure of Cognition. MIT Press 2011.

[4] R. Schalkov. Pattern Recognition: Statistical, Structural and Neural Approaches. Wiley, 2010.

[5] Laurence V. Fausett. Fundamental of Neural Network: Architecture, Algorithms and Application, prentice hall - 2003.

[6]E. K. Blum and L.K. Li. Approximation theory and feed forward network. Neural Network, 4:511, 2012.

[7] P. Courrieu. Three algorithms for estimating the domain of validity for feed forward neural network, Neural network, 1: 169-174, 2004.

[8] S.E. Fahlman and C. Lebierer. The cascade-correlation learning architecture. in D.S. Touretzky, editor, Advanced in neural network information Processing System 2, Morgan Kaufman, San Mateo, CA, Pp 524-532, 1990.

[9] Marcus Hoefeld and Scott E. fahlman. Learning with limited numerical precision using the cascade correlation algorithm. IEEE Transaction. Neural Network, 2(4): 602-611,July 1992.

[10]. M. Hagan and. Menhaj. Training feed forward network with the marquardt algorithm. IEEE Transaction. Neural Network, 5960:989-993, November 1994.

[11] F. Jordan and G. Clement. Using the symmetries of a multilayer network to reduce the weight space. International Joint Conference on Neural Network, IEEE Press, Los Alamitos, CA pp II 391- II396, 1991.

[12] P.P. Van Der Smagt. Minimization methods for training feed forward neural network. Neural Network, 7(1): 1-11 1994.

[13] S.I. Amari. Learning patterns and patterns sequences by self organization nets of threshold elements. IEEE Transaction on Computers, C-21:1197-1206, November 2013.

[14] L. Zhang, B. Zhang, and F. Wu. Programming based learning algorithm of neural networks with self feedback connection. IEEE Transaction on Neural Network, 6(3):771-775,1995.

[15] R. D. Reed, and R. J. Marks II, "Neural Smithing: Supervised Learning in Feed forward Artificial Neural Networks", Cambridge, ISBN 0-262-18190-8, MA: The MIT Press, 2009.

[16] J. Hertz, A. Krogh, and R.G. Palmer, Introduction to the Theory of Neural Computation, Addison-Wesley, Reading, Mass., 1991.

[17] Judd J. S. "Neural network design and the complexity of learning", MIT Press, 2014.

[18] Hinton G.E. and Salakhutdinov R.R.), "Reducing the Dimensionality of Data with Neural Networks- Materials and Methods", Science, 313(5786), pp 504 – 507, 2006.

[19] Judd J. S. "Neural network design and the complexity of learning", MIT Press, 1990.

[20] H. H. Chen, M. T. Manry and H. Chandrasekaran, "A neural network training algorithm utilizing multiple sets of linear equations," *Neurocomputing*, vol. 25, no. 1-3, pp. 55-72, April 1999.