IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 18, Issue 5, Ver. II (Sept - Oct. 2016), PP 34-38
www.iosrjournals.org

Abalone Age Prediction using Artificial Neural Network

Khadija Jabeen[1], K.Ishthaq Ahamed[2]
[1](*M.tech Scholar, CSE Department, G.Pulla Reddy College of Engineering and Technology, Kurnool, Andhra Pradesh, India)*
[2](*Faculty of CSE, CSE Department, G.Pulla Reddy College of Engineering and Technology, Kurnool, Andhra Pradesh, India)*

Abstract: *Artificial Neural Networks are the intelligent computation systems that can be used to solve various challenging problems such as compression, optimization, classification, pattern recognition and prediction. In this paper, a feed forward multi layer perceptron (MLP) network is used to predict the age of abalone on the basis of various physical attributes. Levenberg-Marquardt back propagation training algorithm (trainlm) is used to train the neural network. The results of experiment show that by increasing the number of hidden layers of the MLP network, the calculated error rate keeps decreasing for each corresponding target value. Thus, the declining error rate indicates that MLP network with Levenberg-Marquardt based Backpropagation algorithm acts as best tool in predicting the age of abalone.*

Keywords: *Abalone age, prediction, Artificial Neural Network, MLP network, Levenberg-Marquardt Backpropagation algorithm*

I. Introduction

Over the last few years neural networks have seen an explosion of interest and are being successfully applied across an extraordinary range of problem domains, in areas as diverse as finance, medicine, engineering, geology and physics [1]. Their approach to solve the problems is different when compared to conventional computers. Neural networks process information in a way similar to human brain. This property allows the ANN to distribute the knowledge in the entire network structure; there is no element with specific stored information [2].They are composed of small interconnected processing elements known as neurons that work in parallel to solve a particular problem. The most important feature of these networks is their adaptive nature, where 'learning by example' replaces 'programming' in solving problems [5].

Artificial Neural Networks (ANN) can create their own organization or representation of the information they receive during learning time. Below fig-1 represents a simple neural network structure:

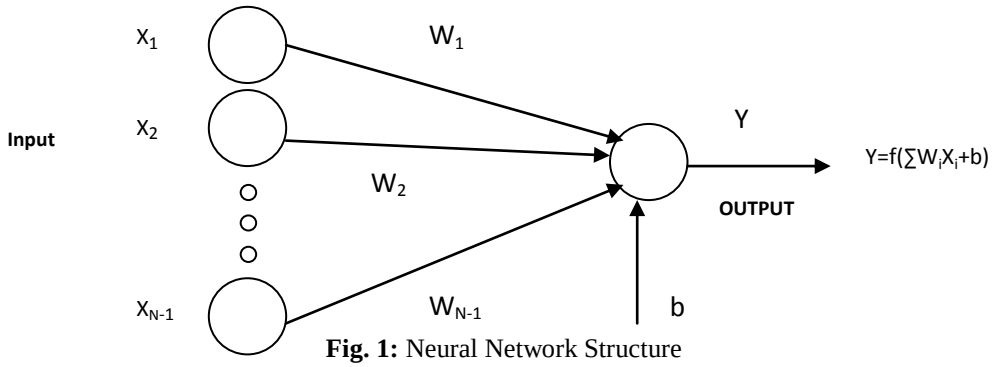

Fig. 1: Neural Network Structure

The structure of neural networks has three types of layers: an **input layer** that receives data from external sources, **hidden layer** that performs computation on the basis of function provided, and an **output layer** that generates output based on the input provided [3]. They have capability to extract the required concept from complex data. In terms of information provided for analysis, a trained neural network acts as an expert system. Neural networks can be used to learn and identify correlated patterns between the input data and corresponding target values. These networks are interesting from a statistical perspective due to their potential use in prediction problems. [5]

A network, an activation rule, and a learning rule are the three major components of a neural network. The network contains set of nodes that are connected via directed links. A numeric activation is associated with each node in the network at time t. The state of the network at a particular time is represented by the overall pattern of activation [6]. Each node in the network follows activation rule to update its activation level. Learning

rule is used to know how the weights on connections should be modified. The processing elements of neural network are generally arranged into a sequence of layers with several connections between them.

II. Neural Network Architecture

The structure of a neural network should be an easy and simplified one. The artificial neural networks are mainly classified in to two types: feedforward and feedback neural networks. A feedforward neural network is also known as Multi-Layer Perceptron (MLP) network. In this network, the signal travels only in one direction where as in a feedback network the signal travels in both the directions as loops [4].

2.1 Feedforward Neural Network: (MLP)

The Multi-Layer Perceptron or feed-forward neural network is perhaps the most popular network architecture in use today. The units each perform a biased weighted sum of their inputs and pass this activation level through an activation function to produce their output, and the units are arranged in a layered feed forward topology. The network thus has a simple interpretation as a form of input-output model, with the weights and thresholds (biases) the free parameters of the model. [1]

A multi-layer perceptron has two distinctive features:

1. A nonlinear activation function is included in each neuron model of the network.
2. A high degree of connectivity is exhibited by the network.

Generally, it may not be sufficient to use one neuron, even with many inputs. We might need five to ten neurons that operate in parallel, which is called as a "layer". Multi-layer feed forward networks are characterized by directed layered graphs. They consist of an input layer, one or more hidden layers and an output layer. On a layer-by-layer basis, the input signal moves through the network in forward direction [7 & 8]. These networks can be trained through variants of backpropagation algorithm in a supervised manner. The following fig-2 represents Multi-Layer Perceptron architecture:

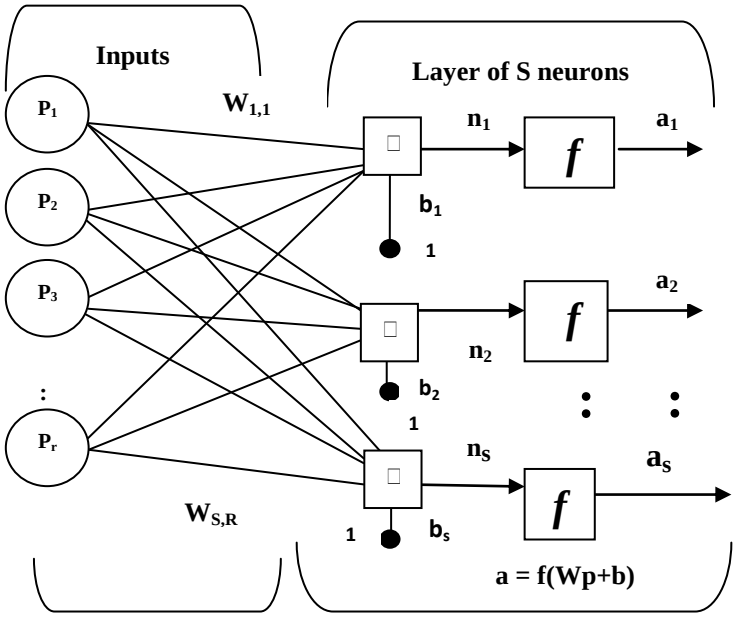

Fig. 2: Multi-Layer Perceptron Architecture

The layer consists of summing units, activation functions, bias **b,** weight matrix **W**, and output vector **a**. Each component of the input **P** is connected to each neuron through weight matrix **W**. Each neuron has an activation function ***f***, bias $\mathbf{b}_s$ and an output $\mathbf{a}_s$. The number of inputs to a layer can be different from the number of neurons. The neurons in a layer can have different activation functions by combining two of networks each network can produce some outputs. The MLP network is one kind of a family of Feed-forward neural networks, where data flows into the network through the input layer, passes through the hidden layer and finally flows out of the network through the output layer, without any direct data-flow loop. The network therefore has a simple interpretation as a form of input output model, with network weights as free parameters. [2]

III. Levenberg-Marquardt Back propagation Algorithm

Levenberg-Marquardt algorithm was developed by Kenneth Levenberg and Donald Marquardt in order to provide numerical solution to the problem of minimizing a non-linear function [9]. Due to its fast and stable convergence, in artificial neural networks it is used to train small-and medium size problems.

There are many methods to train a neural network such as steepest descent algorithm. Many enhancements were made to steepest descent algorithm but due to its slow convergence it proved to be an inefficient algorithm [9]. This slow convergence of steepest descent method can be improved drastically by Gauss-Newton algorithm as it provides faster convergence and finds proper step sizes for each direction [9]. But this improvement cannot be expected in many of the cases as Gauss-Newton algorithm is divergent in nature.

The Levenberg-Marquardt algorithm (trainlm) combines the steepest descent method and Gauss-Newton algorithm. Levenberg-Marquardt algorithm inherits the advantage of speed and stability from Gauss-Newton algorithm and steepest descent method respectively [9]. Levenberg-Marquardt is considered as one of most efficient training algorithms because it performs a combined training process. This algorithm was designed to approach second-order training speed without computing Hessian matrix. [10]

Levenberg-Marquardt algorithm switches between steepest descent and Gauss-Newton algorithm during the process of training as it combines best of both the algorithms. If the scalar μ is zero, it is Newton's method using approximate Hessian matrix. If μ is large, it becomes gradient descent with small step size. Newton's method is more accurate and faster near error minimum [10]. So after each successful step μ is decreased i.e; reduction in performance function and it is increased only when tentative step increases the performance function. The backpropagation algorithm can be outlined as follows [11]:

Step 0: Small random values are used to initialize weights and learning rate.
Step 1: When stopping condition is false perform the steps 2 to 9.
Step 2: For each training pair perform the steps 3 to 8.
Step 3: The input signal x_i (i=1 to n) from each input unit is sent to the hidden unit.
Step 4: To calculate net input, each hidden unit z_j (j=1 to p) sums its weighted input signals.

$$z_{inj} = v_{0j} + \sum_{i=1}^{n} x_i v_{ij}$$

By applying activation functions over z_{inj} the output of the hidden unit can be calculated as:

$$z_j = f(z_{inj})$$

and then the output from the hidden unit is sent to input of output layer units.

Step 5: Calculate net input for each output unit y_k (k=1 to m):

$$y_{ink} = w_{0k} + \sum_{j=1}^{p} z_j w_{jk}$$

and then activation function is applied to calculate the output signal.

$$y_k = f(y_{ink})$$

Step 6: A target pattern related to input training pattern is received by output unit y_k and the error correction term is calculated as:

$$\delta_k = (t_k - y_k) f'(y_{ink})$$

Based on calculated error term, the change in weights and bias can be updated as:

$$\Delta w_{jk} = \alpha \delta_k z_j; \ \Delta w_{0k} = \alpha \delta_k$$

Step 7: Every hidden unit z_j sums its delta inputs from output units as follows:

$$\delta_{inj} = \sum_{k=1}^{m} \delta_k w_{jk}$$

To calculate the error term δ_{inj} gets multiplied with derivative of $f(z_{inj})$ as follows:

$$\delta_j = \delta_{inj} f'(z_{inj})$$

Based on the computed value of δ_j, the change in weights and bias are updated as:

$$\Delta v_{ij} = \alpha \delta_j x_i; \Delta v_{0j} = \alpha \delta_j$$

Step 8: The bias and weights are updated by each output unit as follows:

$$w_{jk}(new) = w_{jk}(old) + \Delta w_{jk}$$
$$w_{0k}(new) = w_{0k}(old) + \Delta w_{0k}$$

The bias and weights updated by each hidden unit are as follows:

$$v_{ij}(new) = v_{ij}(old) + \Delta v_{ij}$$
$$v_{0j}(new) = v_{0j}(old) + \Delta v_{0j}$$

Step 9: Check for stopping condition. It may be number of epochs reached or the actual output matching nearer to target output.

IV. Experimental results

Abalone are a very common type of sea snails that are considered to be a delicacy and popular in jewellery making. Their life span ranges from 1 to 29 years. Its age can be determined by counting number of layers in its shell. This is a time consuming process as it requires to cut the sample of the shell, stain it and count number of rings through a microscope. Instead we used a statistical approach of neural networks to predict the age of abalone. Here we predict the age of abalones based on the given physical characteristics.

The multi-layer feedforward neural network (MLP) is fed with 8 input variables and its corresponding 1 target sequence. After giving the input/target sequence to the network it is trained through Levenberg-Marquardt backpropagation algorithm. Each time the network is trained, the number of hidden layers are increased from 2 to 10. In each cycle, the error values are calculated by subtracting obtained output values from the given target values or records.

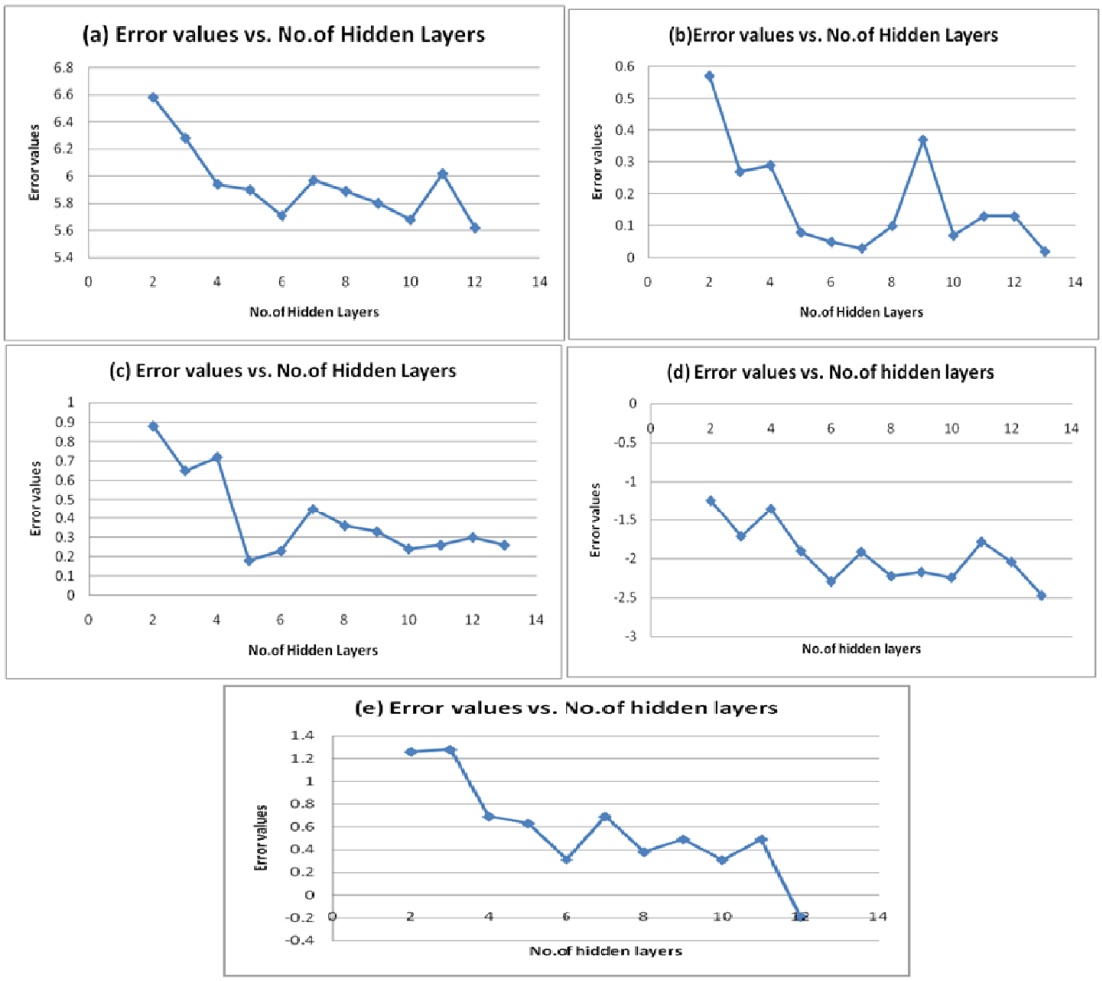

Fig. 3: Declining error value against increasing hidden layers for different target values

In Figure 3(a), for the target record 15 with two hidden layers, the error value is 6.58. But as the hidden layers increase from 2 to 12 the error value falls to 5.62. This decline in the error values indicates that obtained output value is nearing to match the target value.

In Figure 3(b), for the target record 10 with two hidden layers, the error value is 0.57. But as the hidden layers increase from 2 to 10 the error value falls to 0.07. This decline in the error values indicates that obtained output value is nearing to match the target value.

In Figure 3(c), for the target record 10 with two hidden layers, the error value is 0.88. But as the hidden layers increase from 2 to 10 the error value falls to 0.24. This decline in the error values indicates that obtained output value is nearing to match the target value.

In Figure 3(d), for the target record 7 with two hidden layers, the error value is -1.24. But as the hidden layers increase from 2 to 12 the error value falls to -2.04. This decline in the error values indicates that obtained output value is nearing to match the target value.

Similarly in Figure 3(e), for the target record 9 with two hidden layers, the error value is 1.26. But as the hidden layers increase from 2 to 12 the error value falls to -0.19. This decline in the error values indicates that obtained output value is nearing to match the target value.

V. Conclusion And Future Work

In this paper we have presented a Multi-Layer Perceptron Network which is trained through Levenberg-Marquardt backpropagation algorithm for predicting the age of abalone. The number of hidden layers in the network are increased in order to match the target and obtained output values. From the above experimental results, we observe that as the number of hidden layers increases in neural network architecture, the error values for each target record decreases gradually which results in accurate performance of this network to predict the age of abalone. Thus, from the results it is concluded that Multi-Layer Perceptron Network is one of the effective tools in abalone age prediction.

As part of future work, the error rate for each target record can be reduced in less number of epochs by making enhancements to the training algorithm.

References

[1] A.B.Karthick Anand Babu, 2012. *Design and Development of Artificial Neural Network Based Tamil Unicode Symbols Identification System*. IJCSI International Journal of Computer Science Issues, *Vol. 9, Issue 1, No 2*, January 2012.

[2] S. Haykin, *Neural networks: A comprehensive foundation* (2nd Edn., New Jersey: Prentice-Hall, 1999).

[3] Ms.Sonali, B.Maind and Ms.Priyanka Wankar, *"Research paper on basic of artificial neural network"*, International journal on recent and innovation trends in computing and communication, *Vol 2 Issue 1* IJRITCC January 2014.

[4] Jitender Singh Yadav, Mohit Yadav, Ankit Jain, *Artificial Neural Network*. International Journal Of Scientific Research And Education, *Volume 1, Issue 6*, Pages 108-118, 2013, ISSN (e): 2321-7545.

[5] M. H. Hassoun, *Fundamentals of artificial neural networks* (Cambridge: MIT Press, 1995).

[6] Lippmann,R. P. *An Introduction to computing with neural nets* (IEEE ASSP Magazine, 1987) 4-22.

[7] Bengio, y., Frasconi, P., Gori, M., & G.Soda. (1993). Recurrent *neural networks for adaptive temporal processing*. In Proceedings of the 6th Italian workshop on parallel architectures and neural networks wirn93 (pp. 85-117). Vietri (Italy): World Scientific Pub.

[8] Qin He., (1999). *Neural Network and Its Application in IR*. University of Illinois at Urbana-Champaign Spring.

[9] B. M. Wilamowski and H. Yu, *Improved computation for Levenberg Marquardt training*, *IEEE Transactions on Neural Networks*, 21, 930–937, 2010.

[10] Howard Demuth Mark Beale, *Neural Network Toolbox For Use with MATLAB*. User's Guide, Version 4.

[11] SN Sivanandam, SN Deepa, *Principles of Soft Computing* (Wiley India, 2007)**.**