ISSN: 2456-2955
Volume 3, Number 2
UNIVERSAL JOURNAL OF COMPUTERS & TECHNOLOGY

Artificial Neural Networks

Hasanah Tulloh

College of Engineering and Computer Science, Florida Atlantic University

tullohhusana@gmail.com

ABSTRACT

Proposed in this paper is a novel fast-convergence algorithm applied to neural networks (ANNs) with a learning rate based on the eigenvalues of the associated Hessian matrix of the input data. That is, the learning rate applied to the backpropagation algorithm changes dynamically with the input data used for training. The best choice of learning rate to converge to an accurate value quickly is derived. This newly proposed fast-convergence algorithm is applied to a traditional multilayer ANN architecture with feed-forward and backpropagation techniques. The proposed strategy is applied to various functions learned by the ANN through training. Learning curves obtained using calculated learning rates according to the novel method proposed are compared to learning curves utilizing an arbitrary learning rate to demonstrate the usefulness of this novel technique. This study shows that convergence to accurate values can be achieved much more quickly (a reduction in iterations by a factor of hundred) using the techniques proposed here. This approach is illustrated in this research work with derivations and pertinent examples to illustrate the method and learning curves obtained.

Indexing terms/Keywords

Artificial neural networks; Backpropagation algorithm; Eigenvalues; Hessian Matrix; Learning rate

Academic Discipline And Sub-Disciplines

Engineering and Computer Science

SUBJECT CLASSIFICATION

Neural Networks

TYPE (METHOD/APPROACH)

Computational research

INTRODUCTION

In short, the scope of this study is to develop an ANN that converges more quickly to an accurate result than traditional ANNs.

An ANN has its origin in the field of biology. The biological neural network consists of billions of interconnected neurons. An ANN is a mathematical model evolved as a computational tool based on the image of the biological neural complex as shown by Neelakanta and De Groff and elaborated in [1]. During supervised training, the neural network takes in relevant data. The desired output is sought comparable to a supervisory teacher standard. A weight set is then determined based on the interconnections between the neuronal layers. That is, the artificial neural network refers to a mathematical technique that maps from an input space to an output space. The goal of supervised training is to update the weights iteratively to minimize the error which is the difference between the actual output vector of the network and the desired output vector.

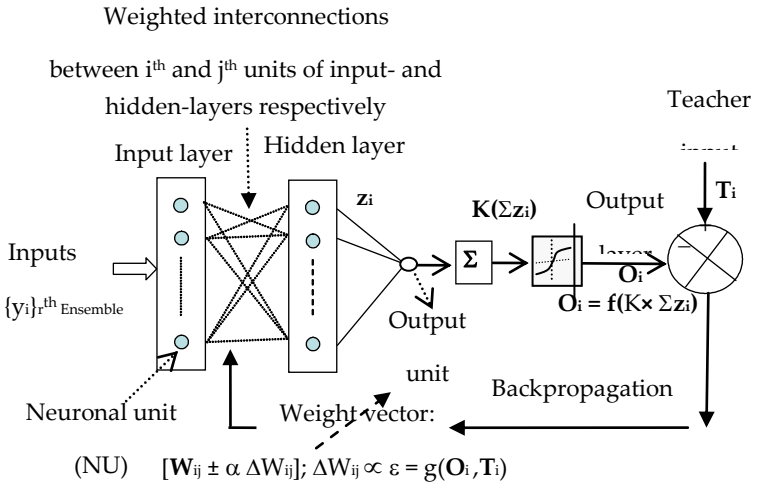


Fig. 1. Test ANN MFLP architecture constructed with I = 9 input neuronal units (NU), one hidden-layer (with J = 9, NUs) and one output-unit. The topology includes supervised learning and backpropagation.

For the goal described here, a multilayered feedforward perceptron (MLFP) made of an input layer, a single hidden layer and a single output is used (Figure 1). The values addressed at the input units progress via interconnected inner layers and the summed output is squashed by a nonlinear sigmoid to assume a limited level; and, the sigmoid-compressed output, O is finally compared against a teacher/supervisory (reference) value, T (representing the desired output objective). The resulting error, denoted as ε, the mean-squared value of (O – T), is then sensed and applied to the interconnection weights by a backpropagation gradient algorithm.

The backpropagation (BP) algorithm adopted facilitates typically a steepest-descent based gradient that modifies the weight vector values (either to increase or decrease), when the error function is applied to the interconnection weights, W_{ij}.

The backpropagation algorithm is a powerful tool for training feedforward neural networks. However, since it applies the steepest descent method to update the weights, it suffers from a slow convergence rate and may yield suboptimal solutions [2]. In this work, a procedure is used that increases the rate of convergence. Applying this speed-up technique results in the number of iterations required to train the net being much smaller. The objective of this learning process is to adjust the weights of the network so as to minimize the average squared error, ε. For a net of size L (L is number of patterns contained in the training set), where T is the teacher (desired) output, O is the network output, and y is the network input, the squared error (cost) function for the rth ensemble input, ε, is given by [3]:

$$\left[\varepsilon = g(O_i, T_i)_r = \frac{1}{2L}\sum_{i=1}^{L}\left(T_i - O_i\right)_r^2\right]_r \tag{1}$$

where $[\mathbf{O}]_i = [f(K\times\Sigma\mathbf{z_i})]$, $(\Sigma\mathbf{z_i})$ depicts the sum of the outputs from the hidden neuronal units and, K is a prescribed linear scaling-constant on the computed sum. Further, $[f(K \times \Sigma\mathbf{z_i})]$ implies a squashing transfer function imposed on the linearly-scaled sum, $(K \times \Sigma\mathbf{z_i})$ so that, the resulting output remains limited (typically between ± 1). Among various possibilities of choosing this squashing function, f(.), depicting a simple hyperbolic tangent (sigmoidal) function [1] is adopted in the present study.

The outputs from the hidden neuronal units depict the weighted value of the inputs, namely, $\{z\}_i = \{w\}_{ij} \times \{y\}_i$, where $\{w\}_{ij}$ corresponds to the matrix $[\mathbf{W}_{ij}]$ denoting the [I × J] weight matrix of interconnection between i^{th} and j^{th} units of input- and hidden-layers. Further, relevant to each iteration of the backpropagated error, corresponding change in the value of w, is specified by the following relation:

$$w(new) = w(old) \pm \mu\frac{d\varepsilon}{dw} \tag{2}$$

with μ being a constant of proportionality; and, $d\varepsilon/dw$ denotes the gradient that facilitates a proportionate change on the existing error-value. Implicitly, for a given ensemble r, ε is a function of: (i) Input set $\{y_i\}$; (ii) interconnection weights, $\{w_{ij}\}$; (iii) the summed values of $\{z_i\} = \{w_{ij}\} \times \{y_i\}$ and, (iv) $O_i = f(K \times \Sigma z_i)$. Using the relevant constituents of the error function ε as above, the gradient indicated can be written explicitly as:

$$\left[\frac{d\varepsilon}{dw}\right]_r = \pm\frac{1}{I}\sum_{n=1}^{I}\left[\left(T_i - O_i\right) \times y_i\right]_r \tag{3}$$

Factorizing, equation (3) can be further rewritten as follows:

$$\left[\frac{d\varepsilon}{dw}\right]_r = \pm\left[\frac{1}{I}\sum_{i=1}^{I}\left(T_i \times y_i\right) + \frac{1}{I} \times w_{ij}\sum_{i=1}^{I} y_i \times y_i\right]_r \tag{4}$$

which explicitly denotes the measurable gradient value of the backpropagated error. This value, when applied to interconnections iteratively would modify the associated weighting coefficients as per equation (2); hence, the corresponding output, O_i (being compared with the teacher value T_i) would increase or decrease, so as to reduce the objective error-function towards zero (or a prescribed stop criterion).

The above procedure implies the general class of *steepest-descent method* in updating the weights; however, it suffers from a slow convergence rate; and as such, it may yield only suboptimal solutions as indicated in [2]. Therefore, an alternative method is suggested in the present study towards ANN training schedule, which increases the rate of convergence. With this proposed speed-up technique, the number of iterations required to train the net becomes significantly smaller as indicated in the following section.

2. FAST CONVERGENCE DURING TRAINING

Equation (4) can be rewritten as:

$$\left[\frac{d\varepsilon}{dw}\right]_r = \pm\left[\rho + \left([H]w_{ij}\right)\right]_r \tag{5}$$

The derivative of ε is zero at the minimum so we can solve for the optimum weight,

$$w_{opt} = \frac{\rho}{[H]} \tag{6}$$

The learning rate that takes us directly to the minimum of (1) is equal to the inverse Hessian matrix. Note from equation (4) above and applying the same analysis to multidimensional case, the Hessian can be defined as a matrix which consists of the average over all inputs of yy^T,where y^T is the transpose of y. The Hessian signifies the shape of the cost surface. The eigenvalues of H are a measure of the steepness of the surface along the curvature directions. A large eigenvalue would signify steep curvature and that a small learning rate is needed. That is, the learning rate should be proportional to 1/eigenvalue. Since we will use a single learning rate for all the weights, a learning rate is chosen that will not cause divergence along the steep directions (large eigenvalue directions). Thus, a learning rate is chosen that is on the order of $1/\lambda_{max}$ where λ_{max} is the largest eigenvalue of the Hessian matrix.

In this research work, the novel approach of calculating the learning rate by finding the largest eigenvalue of the Hessian matrix at the input of the ANN is employed. It is shown that convergence towards accurate output prediction is realized quickly.

2.1 Generation Of The Inputs

The data used to train the neural network comes from the waveforms (or physical processes) that the ANN is to learn. Therefore, to generate the inputs:

- Use a sufficiently high sampling rate to generate the samples.
- Use these sample points to generate 100 training sets of 9 inputs, the nine inputs being 9 sample points on the generated wave.
- Randomize the input by +/- 0.25.
- Normalize the randomized input by dividing by the maximum value.

2.2 Training The ANN

- Initially a set of uniformly distributed random weights (-1 to 1) is used. For 9 inputs and 9 hidden neurons (Figure 1), one requires a total weight matrix of 9 x 9. Zero bias input is assumed. The output of a neuron is calculated using the formula $z_i = \sum W_{ij} y_i$ where i (and j) ranges from 1 to 9. Then the following algorithm is followed:
- Multiply and calculate z_i for each of the 9 neurons output of the hidden layer (Figure 1). The result will then be an output vector of a 9 x 1 matrix with each entry corresponding to the output of one neuron.
- Apply the nonlinear (hyperbolic tangent) activation function to each of the outputs thus generating another vector of 9 x 1 size.
- Sum all the elements in this vector and compare the result with the teacher value.
- Calculate the error using ENTF (or mean-squared error ,MSE).
- Adjust the weights proportional to the value of ENTF and the learning coefficient.
- Repeat steps above till the error ENTF reduces to less than 0.001.
- Store the final weight values in an array
- Repeat steps above for all the 100 sets of inputs. 100 sets of weight values (9 x 9) arrays are obtained after this step.
- Average the weight values to obtain one set of 9 x 9 array. Use these weights for testing.

2.3 The Testing Phase

- Randomly generate 9 sets of 9 points on the given function.
- Randomize and normalize the test inputs.
- Calculate the outputs of each neuron by using the weights obtained after the training phase.
- Apply the activation function to each of the outputs and sum the outputs.
- Calculate the ENTF/MSE using the desired value being the teacher value.

3.0 METHODOLOGY AND DETERMINATION OF THE LEARNING RATE

As referred to above, in the training phase, arbitrary weights between -1 and 1 are assigned to the links between the input layer and the hidden layer. The error (ε_r) between the actual value and the output of the ANN is then computed. The internal weights in the neural network are then adjusted to reduce the error to a predetermined value. That is, the gradient-based method is employed to adjust the weights proportional to the magnitude of the error, the sign of the error, and the learning rate according to the equation:

$$w(new) = w(old) + |w(old)|\, sign(\varepsilon_r)(\text{learning rate}) \tag{7}$$

Iterations are repeated towards convergence. The error, ε_r, versus the number of iterations is plotted to get the learning curve.

This gradient-based method of adjusting the weights is applied successively to all of the hundred input sets, each time using the converged weight matrix $\{W_{ij}\}$ on the previous data set.

This study compares convergence in number of iterations using an arbitrary learning rate (α = 0.001) versus the learning rate computed as ($1/\lambda_{max}$). That is, for each of the hundred input sets, the Hessian eigenvalue λ is computed. The largest of the Hessian eigenvalues is then chosen as λ_{max} , and this largest eigenvalue is used to compute the learning rate.

In summary, the supervised training proposed here for the test MFLP architecture is based on specifying a Hessian matrix of the relevant data applied on the input neurons interconnected to equal number of neurons of a hidden layer (I = J in Figure 1). That is, in Figure 1, there are I=9 input units y= $\{y_1, y_2, \ldots y_I\}$,. The Hessian matrix corresponds to $y^T y$ which is an I x I (9 x 9 in the present example) square matrix; and, this Hessian matrix can be put into a diagonal form [HD]. Because of the symmetry of the Hessian matrix, it has a unique, single eigen-value, λ_{II} in the diagonal form as shown below (all other eigenvalues are zero):

$$[\mathbf{HD}] = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \lambda_{II} \end{bmatrix} \tag{8}$$

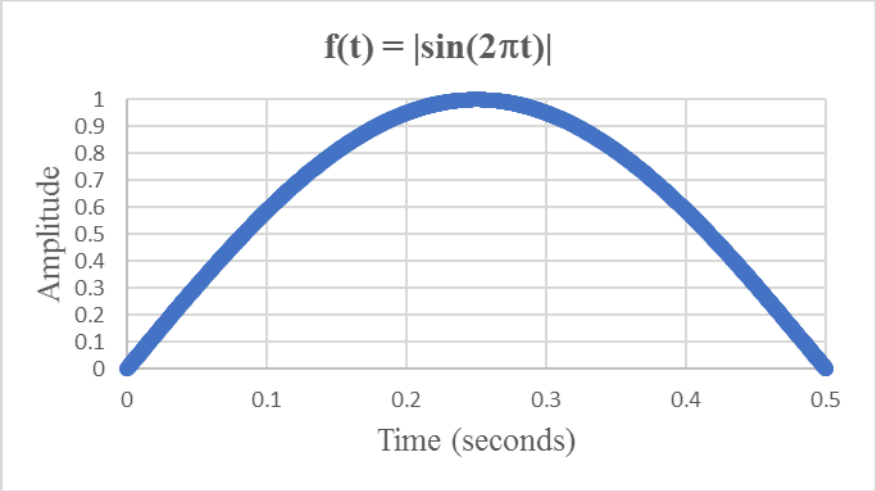
In the proposed backpropagation model, the learning-rate (α) applied on the test ANN, would correspond to the largest, single eigen-value, λ_{II} of the Hessian matrix as above so that, a fast convergence towards a desired extent of output prediction is feasible; as such, in the present study, this algorithmic suite is prescribed to the test ANN.

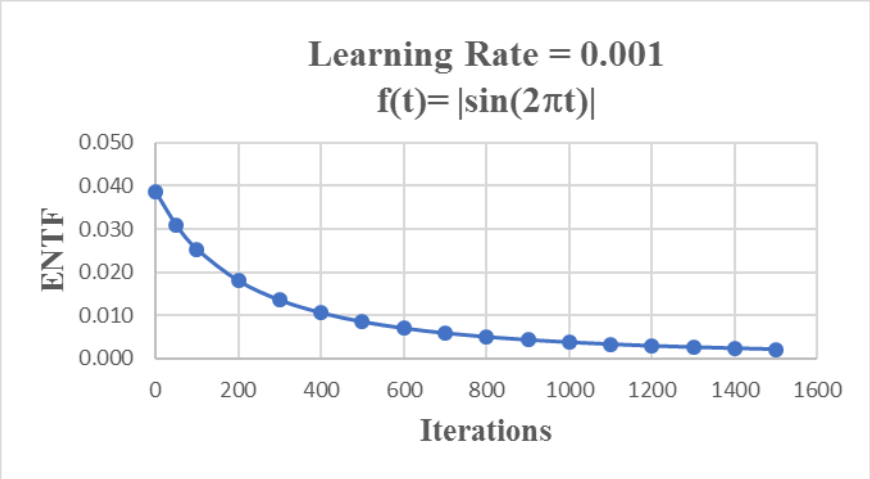
4. RESULTS AND DISCUSSIONS

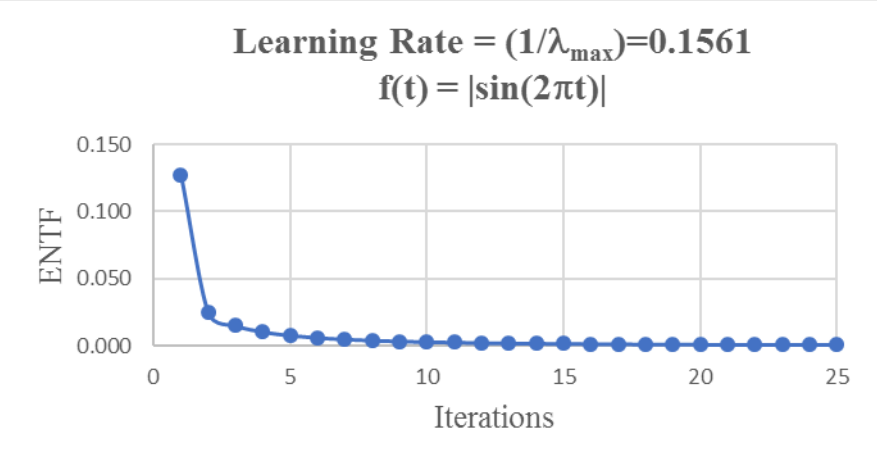
Following are the learning curve plots of several functions.

ISSN: 2456-2955
Volume 3, Number 2
UNIVERSAL JOURNAL OF COMPUTERS & TECHNOLOGY

4.1 f(t) = |sin(2πt)|

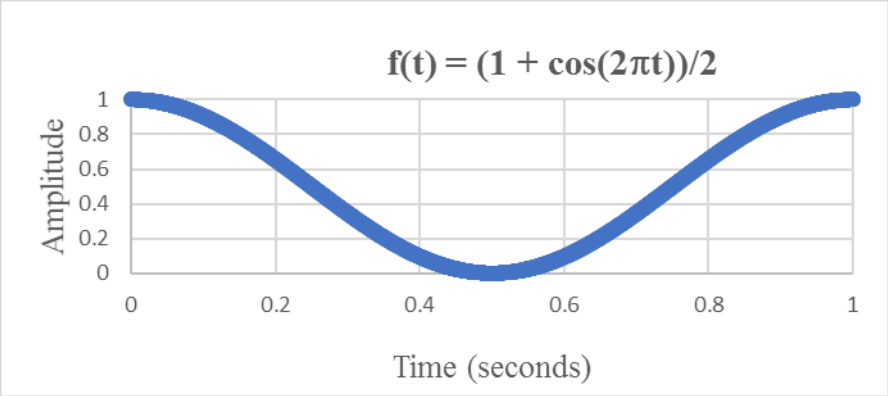


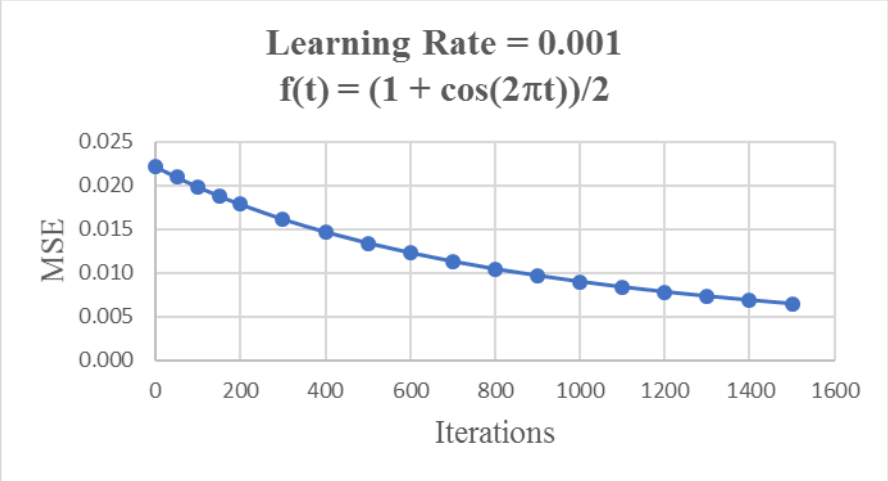


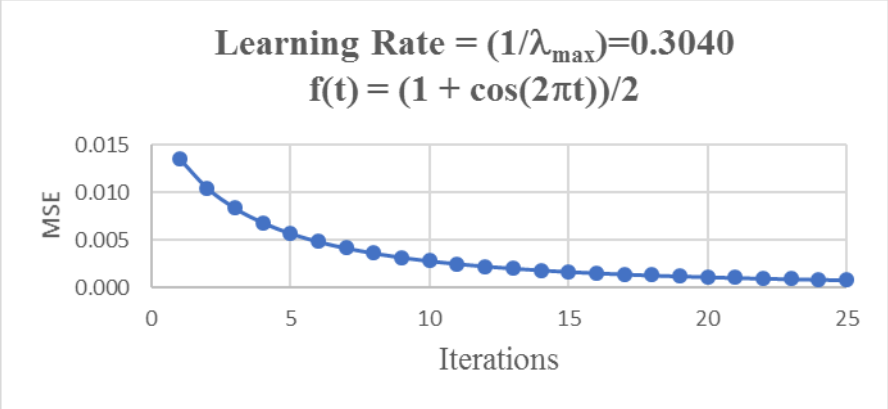


ISSN: 2456-2955
Volume 3, Number 2
UNIVERSAL JOURNAL OF COMPUTERS & TECHNOLOGY

4.2 f(t) = (1 + cos(2πt))/2







.5. CONCLUDING REMARKS

Pursuant to a summary of details on the study performed as outlined above, relevant to the major queries posed earlier, the conclusive remarks and response can be listed as follows:

(1) It was found that the convergence rate can be speeded up by first calculating an appropriate value for the learning rate based on $\alpha = 1/\lambda_{max}$, where λ_{max} is the maximum eigenvalue of the corresponding Hessian matrices of the input data.

(2) In this novel work, the standard Backpropagation algorithm for training feedforward neural networks has been improved based on the concept of using efficient learning rate based on λ_{max} .

(3) Experimental results show that the new algorithm offers much higher speed of convergence than the conventional algorithm.

(4) The improvement presented here can be considered as a valuable and viable alternative to using arbitrary learning rate.

REFERENCES

1. Neelakanta, P.S., De Groff, D., 1994. *Neural Network Modeling: Statistical Mechanics and Cybernetic Perspectives.* CRC Press, Boca Raton.

2. Margoulas, Androulakis, Vrahatis, 1999. Improving the convergence of the backpropagation algorithm using learning rate adaptation methods. *Neural Computation* (November 1999),1769-1796.

3. Haykin, S.,1994. *Neural Networks: A Comprehensive Foundation.* Macmillan Publishing Company, Englewood Cliffs, NJ.

.