Journal of Computer Science 5 (11): 849-856, 2009
ISSN 1549-3636
© 2009 Science Publications

A Modified Conjugate Gradient Formula for Back Propagation Neural Network Algorithm

[1]Abbas Y. Al Bayati, [2]Najmaddin A. Sulaiman and [3]Gulnar W. Sadiq
[1]Department of Numerical Optimization, University of Mosul, Mosul, Iraq
[2]Department of Numerical Analysis, Salahaddin University, Erbil, Iraq
[3]Department of Operational Research, University of Sulaimani, Sulaimani, Iraq

Abstract: Problem statement: The Conjugate Gradient (CG) algorithm which usually used for solving nonlinear functions is presented and is combined with the modified Back Propagation (BP) algorithm yielding a new fast training multilayer algorithm. **Approach:** This study consisted of determination of new search directions by exploiting the information calculated by gradient descent as well as the previous search direction. The proposed algorithm improved the training efficiency of BP algorithm by adaptively modifying the initial search direction. **Results:** Performance of the proposed algorithm was demonstrated by comparing it with the Neural Network (NN) algorithm for the chosen test functions. **Conclusion:** The numerical results showed that number of iterations required by the proposed algorithm to converge was less than the both standard CG and NN algorithms. The proposed algorithm improved the training efficiency of BP-NN algorithms by adaptively modifying the initial search direction.

Key words: Back-propagation algorithm, conjugate gradient algorithm, search directions, neural network algorithm

INTRODUCTION

Gradient based methods are one of the most widely used error minimization methods used to train back propagation networks. The BP training algorithm is a supervised learning method for multi-layered feed-forward neural networks[1]. It is essentially a gradient descent local optimization technique which involves backward error correction of the network weights. Despite the general success of BP in learning the neural networks, several major deficiencies are still needed to be solved. First, the BP algorithm will get trapped in local minima especially for non-linearly separable problems[2]. Having trapped into local minima, BP may lead to failure in finding a global optimal solution. Second, the convergence rate of BP is still too slow even if learning can be achieved. Furthermore, the convergence behavior of the BP algorithm depends very much on the choices of initial values of connection weights and the parameters in the algorithm such as the learning rate and the momentum.

Improving the training efficiency of neural network based algorithm is an active area of research and numerous papers have been proposed in the literature. Early days BP algorithm saw further improvements. Later, as summarized by Bishop[3] various optimization techniques were suggested for improving the efficiency of error minimization process or in other words the training efficiency. Among these are methods of Fletcher and Powell[4] and the Fletcher-Reeves[5] that improve the conjugate gradient method of Hestenes and Stiefel[6] and the family of Quasi-Newton algorithms proposed by Huang[7].

It has been recently shown that a BP algorithm using gain variation term in an activation function converges faster than the standard BP algorithm[8-10]. However, it was not noticed that gain variation term can modify the local gradient to give an improved gradient search direction for each training iteration. This fact allow us to develop and investigate several important CG-formulas in order to improve the rate of convergence of the proposed type of algorithms.

This study suggests that a simple modification to the search direction can also substantially improve the training efficiency of almost all major (well Known and developed) optimization methods. It was discovered that if the search direction is locally modified by a gain value used in the activation function of the corresponding node, significant improvements in the convergence rates can be achieved irrespective of the optimization algorithm used. Furthermore the proposed algorithm is robust, easy to compute and easy to

Corresponding Author: Abbas Y. Al Bayati, Department of Numerical Optimization, University of Mosul, Mosul, Iraq

implement into several well known nonlinear test problems as will be shown later.

The remaining of the study is organized as follows: Materials and methods are proposed first. The outlines and the implementation of the proposed algorithm with CG algorithm have been discussed second. Experimental results are discussed after that and finally, the concluding remarks with short discussion for further research are listed later on.

MATERIALS AND METHODS

First, a novel approach for improving the training efficiency of gradient descent method (BP algorithm) which was presented by Nawi *et al.*[12] are discussed. Their method modified the initial search direction by changing the gain value adaptively for each node. Following their iterative method for changing the initial search direction using a gain value:

Algorithm[12]: Initialize the weight vector with random values and the vector of gain values with one. Repeat the following steps 1, 2 and 3 on an epoch-by-epoch basis until the given error minimization criteria are satisfied.

Step 1: By introducing gain value into activation function, calculate the gradient for weight vector by using Eq. 6 and gradient for gain value by using Eq. 9.
Step 2: Calculate the gradient descent on error with respect to the weights and gains values.
Step 3: Use the gradient weight vector and gradient of gain calculated in step 1 to calculate the new weight vector and vector of new gain values for use in the next epoch.

In general, the objective of a learning process in neural network is to find a weight vector w which minimizes the difference between the actual output and the desired output. Namely:

$$\min_{w \in R^n} E(w) \tag{1}$$

Suppose for a particular input pattern o^0 and let the input layer is layer 0. The desired output is the teacher pattern $t = [t_1 \dots t_n]^T$ and the actual output is o_i^L, where L denotes the output layer. Define an error function on that pattern as:

$$E = \frac{1}{2}\sum_i \left(t_i - o_i^L\right)^2 \tag{2}$$

The overall error on the training set is simply the sum, across patterns, of the pattern error E. Consider a multilayer feed Forward Neural Network (FNN)[1] has one output layer and one input layer with one or more hidden layers. Each layer has a set of units, nodes, or neurons. It is usually assumed that each layer is fully connected with a previous layer without direct connections between layers which are not consecutive. Each connection has a weight. Let o_i^s be the activation of the ith node of layer s and let $o^s = \left[o_1^s \dots o_n^s\right]^T$ be the column vector of the activation values in the layer s and the input layer as layer 0. Let w_{ij}^s be the weight on the connection from the ith node in layer s −1 to the jth node in layer s and let $w_{ij}^s = \left[w_{1j}^s \dots w_{nj}^s\right]^T$ be the column vector of weights from layer s −1 to the jth node of layer s. The net input to the jth node of layer s is defined as $net_j^s = \left(w_j^s, o^{s-1}\right) = \sum_i w_{j,i}^s o_i^{s-1}$ and let $net^s = \left[net_1^s \dots net_n^s\right]^T$ be the column vector of the net input values in layers. The activation of a node is given by a function of its net input:

$$o_j^s = f\left(c_j^s net_j^s\right) \tag{3}$$

Where:
f = Any function with bounded derivative
c_j^s = A real value called the gain of the node

Note that this activation function becomes the usual logistic activation function if $c_j^s = 1$.

By introducing "gain variation" term in this activation function, then only updating formulas for δ_1^s are changed while others are the same as the standard back propagation. To simplify the calculation, taken from the Eq. 2 we then can perform gradient descent on E with respect to w_{ij}^s. The chain rule yields:

$$\begin{aligned}\frac{\partial E}{\partial w_{ij}^s} &= \frac{\partial E}{\partial net^s} \cdot \frac{\partial net^{s+1}}{\partial o_j^s} \cdot \frac{\partial o^s}{\partial net_j^s} \cdot \frac{\partial net^s}{\partial w_{ij}^s} \\ &= \left[-\delta_1^{s+1} \dots -\delta_n^{s+1}\right] \begin{bmatrix} w_{1j}^{s+1} \\ \vdots \\ w_{nj}^{s+1} \end{bmatrix} .f'\left(c_j^s net_j^s\right) c_j^s . o_j^{s-1}\end{aligned} \tag{4}$$

where, $\delta_j^s = -\frac{\partial E}{\partial net_j^s}$. In particular, the first three factors of (4) indicate that:

$$\delta_1^s = \left(\sum_i \delta_i^{s+1} w_{i,j}^{s+1}\right) f'\left(c_j^s net_j^s\right) c_j^s \quad (5)$$

As we noted that the iterative formula (5) for δ_j^s is the same as standard BP[11] except for the appearance of the value gain. By combining (4) and (5) yields the learning rule for weights:

$$\Delta w_{ij}^s = \eta \frac{\partial E}{\partial w_{ij}^s} = \eta \delta_j^s o_j^{s-1} \quad (6)$$

where, η is a small positive constant called 'step length' or 'learning rate' and the search direction or gradient vector is $d = \Delta w_{ij}^s = g$. In this approach, at step n is the calculation for gradient of error $g^{(n)}$ is modified by including the variation of gain value to yield:

$$d^{(n)} = \Delta w_{ij}^{(n)}\left(c_j^{s(n)}\right) = g^{(n)}\left(c_j^{s(n)}\right) \quad (7)$$

The gradient descent on error with respect to the gain can also be calculated by using the chain rule as previously described; it is easy to compute as:

$$\frac{\partial E}{\partial c_j^s} = \left(\sum_i \delta_i^{s+1} w_{i,j}^{s+1}\right) f'\left(c_j^s net_j^s\right) net_j^s \quad (8)$$

Then the gradient descent rule for the gain value becomes:

$$\Delta c_j^s = \eta \delta_j^s \frac{net_j^s}{c_j^s} \quad (9)$$

At the end of each iteration the new gain value is updated using a simple gradient based method as given by the formula:

$$c_j^{new} = c_j^{old} + \Delta c_j^s \quad (10)$$

For more details of this method and its implementations[12].

CG-type method: In order to overcome the difficulties of the standard delta rule learning technique, where this turning through the steepest descent direction. Which causes the zigzag problematic, where it increases the turning time. To improve the global rate of convergence of the standard delta rule learning technique, we have suggested the new delta rule based on CG type method. The CG method is an iterative optimization approach, the gradient is n-components vector and the gradient direction is called the steepest ascent or descent (the Fletcher-Reeves method tries to exploit the fact that for a quadratic function of n variables, n linear searches along the mutually conjugate directions will locate the minimum). But this direction has a local property and not a global one. Thus any method which makes use of the gradient vector, can be expected to give the minimum point. The steps of CG algorithm are:

Algorithm (CG-based NN-algorithm):

- Select the initial weights randomly with small values and set p = 1
- Select the next training pair from the training set [T_p] apply the input vector [x_p] to the network input and specify the desired output vector. Then calculate the actual outputs of the network by using these two formulae:

$$net_{pj}^{s+1} = \sum_{i=1}^{ns} W_{ij}^s out_i^s + bias_j^{s+1}$$

$$out_{pj}^{s+1} = f\left(net_{pj}^{s+1}\right) = \frac{1}{1 + e^{-\beta net_{pj}^{L+1}}}$$

Where:

net_{pj}^{s+1}	=	Summation of multiple weight with inputs for j
out_{pj}^{s+1}	=	The actual output of the network j by using function
$x_j = out_{pj}^{\phi}$	=	It represents input patterns, p number of pattern and j is number of net

$$net_j = x_1 w_{1j} + x_2 w_2 j + ... + x_n w_{nj} = \sum_{i=1}^{n} x_i w_{ij}$$

And:

net_j	=	The result of summation of j node
x_i	=	Input pattern
w_{ij}	=	Weight between node j and i

- Calculate the errors between the actual output of the network and the desired output by using these steps

Start with x_i. If (i = 1) then $d_1 = -g_1$. Obtain (μi) using line search and update the object (x) using the formula: $x_{i+1} = x_i + \mu_i d_i$. Increment i by 1 and calculate the gradient vector g_n to obtain β_i. Update the direction d_i using the formula $d_{i+1} = -g_{i+1} + \beta_i d_i$ and find the corresponding step size (μ_i) by updating the object (x) to minimize the value of object (x) and adjust the weights of the network in a way that minimizes the error.

When (x_{i+1}) is minimum terminate the iterations; if not, continue to repeat step (3) for each vector in the training set until the total error for the entire set is acceptable low, where d_i is the update direction of the object at the iteration and d_i, the initial direction, is chosen to be the steepest descent direction, μi is the step size along the direction d_i; β_i is chosen to insure that the direction vector is conjugate to all previous directions and g_i is the gradient of the activation function of the iteration. There is a number of different choices for β_i.

In this study, Fletcher and Reeves has been used. It is one of the most commonly used and its formula is:

$$\beta i = \frac{g_{i+1}^T g_{i+1}}{g_i^T g_i} \text{ (Fletcher and Reeves)} \quad (11)$$

The other type of β_i is Al Assady and Al Bayati and Polak and Ribere. The formula of β_i of these two types are:

$$\beta i = \frac{g_{i+1}^T y_i}{d_i^T g_i} \text{ (Al Assady and Al Bayati)} \quad (12)$$

$$\beta i = \frac{y_i^T g_{i+1}}{g_i^T g_i} \text{ (Polak and Ribere)} \quad (13)$$

Following we are going to investigate and develop a new formula for the extended FR formula which is based on the non-quadratic models, this will increase the rate of convergence of the In fact, these quantities are well-known and derived with respect of quadratic models, certainly the FR formula has a standard FRCG-method. Now most of the currently used optimization methods use a local quadratic representation of the objective function. But the use of quadratic model may be inadequate to incorporate all the information so that more general models than quadratic are proposed as a basis for CG algorithms.

It is shown that CG methods with $g_{i+1}^T g_{i+1}$ in the numerator of β_i has strong global convergence theorems with exact and inexact line searches but has poor performance in practice. On the other hand the CG methods with $g_{i+1}^T y_i$ in the numerator of β_i has uncertain global convergence for general non-linear functions, but has good performance in practice. Therefore CG methods has been frequently modified and improved by many authors, for example[14-16] have proposed further modifications of the conjugate gradient methods which are based on some non-quadratic models.

Here we generalize the FRCG by considering more general function than quadratic which we call it as quasi-sigmoid function, our goal is to preserve and increase the convergence properties of FR algorithm and to force its performance in practice.

A generalized FRCG algorithm (based on non-quadratic model): If q(x) is a quadratic function defined by:

$$q(x) = \frac{1}{2} x^T G x + b^T + c \quad (14)$$

Where:

G = n×n symmetric and positive definite matrix
b = Constant vector in R^n and c is constant

Then we say that f is Defined as a nonlinear scaling of q(x) if the following conditions hold:

$$f(x) = F(q(x)),\ q > 0 \quad (15)$$

And:

$$\frac{dF}{dq} > 0 \quad (16)$$

The following proportions are immediately derived from the above conditions:

- Every contour line of q(x) is a contour line of f
- If $x^\bullet$ is minimize of q(x) then it's also a minimize of f

In this area there are various published works. The special polynomial case:

$$F(q(x)) = \varepsilon_1 q(x) + \frac{1}{2}\varepsilon_2 q^2(x) \quad (17)$$

where, ε_1, ε_2 scalars are has been investigated by[16]. A rational model has been developed by[15] where:

$$F(q(x)) = \frac{\varepsilon_1 q(x) + 1}{\varepsilon_2 q(x)},\ \varepsilon_1 q(x) < 0 \quad (18)$$

Another rational model was considered by[14] where:

$$F(q(x)) = \frac{\varepsilon_1 q(x)}{1-\varepsilon_2 q(x)},\ \varepsilon_1 > 0, \varepsilon_2 < 0 \quad (19)$$

A new extended CG-formula: Here we consider the new model function defined by:

$$f(x) = F(q(x)) = \frac{q(x)}{1+e^{-q(x)}} \quad (20)$$

We call it as quasi sigmoid function where q>0 with further assumption that:

$$\frac{dF}{dq} > 0 \quad (21)$$

From Eq. 20 we have:

$$\frac{dF}{dq} = F' = \frac{\left(1+e^{-q}\right)+qe^{-q}}{\left(1+e^{-q}\right)^2} = f*(1+\frac{1}{q}-\frac{f}{q})$$

$$\therefore F' = f*(\frac{1+q-f}{q}) \quad (22)$$

Return to Eq. 20 and solve it for q assuming that $e^{-q} = 1-q+\frac{1}{2}q^2+r$ where the remainder $r = \sum_{n=3}^{\infty}\frac{(-1)^n}{n!}q^n(1-\frac{1}{n+1}q)$. It is clear that $r<0$ and set $\sigma = r+1$.

Then:

$$f = \frac{q}{\frac{1}{2}q^2 - q + \sigma}$$

And:

$$q = (1+\frac{1}{f}) + \sqrt{(1+\frac{1}{f})^2 - 2\sigma} \quad (23)$$

Use (21) and (23) to compute F^t using only function values:

$$F' = f*(\frac{2-f+\frac{1}{f}+a}{1+\frac{1}{f}+a}) \quad (24)$$

where, $a = \sqrt{(1+\frac{1}{f})^2 - 1}$ assuming $\sigma = \frac{1}{2}$

Now define ρ_i as follows:

$$\rho_i = \frac{F_i'}{F_{i+1}'} \quad (25)$$

Then we can compute the value of ρ_i using function value at two points x_{k+1} and x_k from Eq. 24:

$$\rho_i = \frac{f_i(2-f_i+\frac{1}{f_i}+a_1)}{(1+\frac{1}{f_i}+a_1)} * \frac{1+\frac{1}{f_{i+1}}+a_2}{f_{i+1}(2-f_{i+1}+\frac{1}{f_{i+1}}+a_2)} \quad (26)$$

Where:

$$a_1 = \sqrt{(1+\frac{1}{f_i})^2 - 1}$$

$$a_2 = \sqrt{(1+\frac{1}{f_{i+1}})^2 - 1}$$

Our objective is to investigate an extended FRCG method applied to function F(q(x)) obtained by non-linear scaling of q(x). The extended FRCG method can be done by modifying the search directions as follows:

$$\tilde{d}_1 = \tilde{g}_1 \quad (27)$$

And for i≥0:

$$\tilde{d}_{i+1} = -\tilde{g}_{i+1} + \rho_i \tilde{\beta}_i \tilde{d}_i \quad (28)$$

Where:

$$\tilde{\beta}_i = \frac{\tilde{g}_{i+1}^T \tilde{g}_{i+1}}{\tilde{g}_i^T \tilde{g}_i} = \frac{\rho_i\, g_{i+1}^T \tilde{g}_{i+1}}{g_i^T g_i} \quad (29)$$

Where:

$$\tilde{g} = \nabla F(q(x)) = \frac{\partial f}{\partial x} = \frac{dF}{dq}\frac{\partial q}{\partial x} = F'g$$

$\bar{d}_i$ is the search direction applied to F(q(x)) and $\tilde{y}_i = \tilde{g}_{i+1} - \tilde{g}_i$.

We can show that the original FRCG method and extended FRCG algorithm defined in (27-29) generates the same set of directions and same sequence of points $\{x_i\}$by using the following theorem:

Theorem 1: Given an identical starting point $x_0 \in R^n$.

The method of FRCG applied to f(x) = q(x) and the extended (ERFCG) method defined by (27-29) and applied to f(x) = F(q(x)) with ρ_i defined by (26) generate identical set of directions and identical sequence of points $\{x_i\}$.

Proof: The prove is by induction for k = 0 we have:

$$\tilde{d}_1 = -\tilde{g}_1 = -\frac{dF}{dq}\frac{\partial q}{\partial x} = -F_1' g_1 = F_1' d_1$$

Suppose: $\tilde{d}_i = F_i' d_i$, to prove for i+1

$$\begin{aligned}\tilde{d}_{i+1} &= -\tilde{g}_{i+1} + \rho_i \tilde{\beta} \tilde{d}_{i-1} \\ &= F_{i+1}'[- g_{i+1} + \frac{g_{i+1}^T g_{i+1}}{g_i^T g_i} d_i] \\ &= F_{i+1}' d_{i+1}\end{aligned}$$

Hence the two methods generates the same set of directions.

Our proposed algorithm known as (NEW) begins the minimization process with an initial estimate w_0 and an initial search direction as:

$$d_0 = -\nabla E(w_0) = -g_0 \tag{30}$$

Then, for every epoch by using our proposed method in Eq. 7 the search direction at (n+1)th iteration is calculated as:

$$d_{n+1} = -\frac{\delta E}{\delta w_{n+1}}(c_{i,n+1}) + \rho_n \beta_n (c_{i,n}) d_n (c_{i,n}) \tag{31}$$

where the scalar β_n is to be determined by the requirement that d_n and d^{n+1} must fulfill the conjugacy property[3] and (ρ_n) is defined by Eq. 26.

Outlines of the new algorithm:

Step 1: Initializing the weight vector randomly, the gradient vector $g_0 = 0$ and gain value as one. Let the first search direction $d_0 = g_0$. Set $\beta_0 = 0$, epoch = 1 and n = 1. Let N_t is the number of weight parameters Select a Convergence Tolerance CT.

Step 2: At step n, evaluate gradient vector $g_n(c_n)$ with respect to gain vector c_n and calculate gain vector.

Step 3: Evaluate $E(w_n)$. If $E(w_n)<CT$ then STOP training ELSE go to step 4.

Step 4: Calculate a new search direction: $d_n = -g_n(c_n) + \rho_{n-1}\beta_{n-1}d_{n-1}$, where ($\rho_{n-1}$) is defined in Eq. 25 for I = n-1.

Step 5: For the first iteration, check if n>1 THEN with the function of gain, update:

$$\beta_{n+1} = \rho_{n+1}\frac{g_{n+1}^T(c_{n+1})g_{n+1}(c_{n+1})}{g_n^T(c_n)g_n(c_n)} \text{ ELSE go to step 6}$$

Step 6: If [(epoch +1)/Nt] = 0 THEN 'restart' the gradient vector with $d_n = g^{n-1}(c_{n-1})$ ELSE go to step 7.

Step 7: Calculate the optimal value for learning rate η^* by using line search technique such as:

$$E(w_n + \eta_n^* d_n) = \min_{\lambda \ge 0} E(w_n + \eta_n d_n)$$

Step 8: Update $w_{n+1} = w_n - \eta_n^* d_n$, where d_n is a descent direction.

Step 9: Evaluate new gradient vector $g_{n+1}(c_{n+1})$ with respect to gain value (c_{n+1}).

Step 10: Calculate new search direction:

$$d_{n+1} = -g_{n+1}(c_{n+1}) + \rho_{n-1}\beta_{n-1}(c_n)d_n$$

Step 11: Set n = n+1 and go to step 2.

RESULTS

The performance criterion used in this research focuses on the speed of convergence, measured in number of iterations and CPU time. The test problems used to verify our new proposed algorithm are taken from the open literature by Prechelt[11]. The numerical results have been carried out on a Pentium IV using MATLAB version 7.0. On our test problems, four algorithms have been computed. The first algorithm is the standard CG with Fletcher-Reeves update (traincgf). The other algorithm is a standard CG algorithm computed from the NAG library (CGFR). The third one is the Nawi *et al*.[12] NN-algorithm computed by using their algorithm with their computer program listed in[12]. Finally, the new proposed ECG based NN algorithm (NEW) which is implanted by modifying NN-algorithm. This algorithm uses a major change in calculating the gradient step by in forcing the value of the new parameter ρ_i which is given in Eq. 25 of this study. We have noticed that if the value of this parameter is one, then, this new algorithm will coincide with Nawi *et al.*[12] algorithm but for positive values the new algorithm produces better numerical results because of the use of the non-quadratic model in the derivation of the CG-formula To compare the performance of the proposed algorithm with respect to others: standard optimization algorithms from the (MATLAB NN-toolbox) network parameters such as network size and architecture (number of nodes, hidden layers etc), values for the initial weights and gain parameters were kept same. For our test problem the neural network had one hidden layer with five hidden nodes and sigmoid activation function was used for all nodes. All algorithms were tested using the same initial weights that were initialized randomly from range [0, 1] and received the input patterns for training in the same sequence.

Default values were used for the heuristic parameters, of the above algorithms, unless stated otherwise. For the purpose of comparison, all tested algorithms were fixed with the values of learning rate = 0.3 and momentum term = 0.4. The initial value used for the gain parameter was one. The results of all these four algorithms will be presented as Table 1 and 2 which summarize the performance of the algorithms for simulations that have reached solution. All algorithms were trained with 100 trials, if an algorithm fails to converge, it is considered that it fails to train the FNN, but its epochs, CPU time and generalization accuracy are not included in the statistical analysis of the algorithms.

Test problem (1) Fisher[13]:

Iris classification problem: This is a classical classification dataset made famous by Fisher[13], who used it to illustrate principles of discriminate analysis.

Table 1: Comparisons of four different algorithms for test problem (1)

Algorithms	NOE	CPU	CT
traincgf	69	5.54×10^{-2}	3.8071
CGFR	39	4.90×10^{-2}	1.9146
Nawi *et al.*[12]	29	4.94×10^{-2}	1.4232
NEW	25	4.19×10^{-2}	1.2097

Note: NOE: Number Of Epochs; CPU: Time in seconds per epochs CT: Time of Convergence

Table 2: Comparisons of four different algorithms for test problem (2)

Algorithms	NOE	CPU	CT
Traincgf	71	5.34×10^{-2}	3.7883
CGFR	65	5.11×10^{-2}	3.3060
Nawi *et al.*[12]	39	4.00×10^{-2}	1.5503
NEW	33	3.40×10^{-2}	2.8101

This is perhaps the best-known database to be found in the pattern recognition literature. Fisher's study is a classic in the field and is referenced frequently to this day. The selected architecture of the FNN is (4-5-3-3) with target error was set as (0.01) and the maximum epochs to (2000).

Table 1 shows that the proposed algorithm reached the target error after only about (25) epochs as opposed to the standard CGFR at about (39) epochs and clearly we see that there is an improvement ratio, nearly (3.6), for the number of epochs compare to NN-toolbox and almost (3.681) for the convergence time. The following main computer program used with Nawi *et al.*[12] routine is:

```
% main learning program
P = [0 1 2 3 4 5 6 7 8 9 10];
    T   =  [0 1 2 3 4 3 2 1 2 3 4];
    net =  Newff(minmax(P),[5 1],{'tansig' 'purelin'});
    Y   =  Sim(net,P); plot(P,T,P,Y,'o')
           net.trainParam.epochs = 50;
    net =  Train(net,P,T);
    Y   =  Sim(net,P);
```

Test problem (2) Fisher[13]:

Winconsin breast cancer problem: This dataset was created based on the ‘breast cancer Wisconsin’ problem dataset from[12]. This problem tries to diagnosis of breast cancer by trying to classify a tumor as either benign or malignant based on cell descriptions gathered by microscopic examination. The selected architecture of the FNN is (9-5-2-2). The target error is set as to (0.015) and the maximum epochs to (2000).

In Table 2, it is worth noticing that the performance of the new algorithm since it take only (33) epochs to reach the target error compare to Nawi *et al.*[12] (39) epochs and to CGFR at about (65) epochs and worst for traincgf that need about (71) epochs to converge. Still the proposed algorithm outperforms others three algorithms with a considerable improvements.

DISCUSSION

In this study, we have introduced neural network algorithm based on several optimization update. The new algorithm is compared with three well known standard CG and NN algorithms using the Iris Classification Problem and Winconsin Breast Cancer Problem. Our numerical results indicate that the new technique has an improvements of about (10-15%) NOE; CPU and CT tools.

CONCLUSION

In this research, new fast learning algorithm for neural networks which are based on CG updates with adaptive gain training algorithm (NEW) is introduced. The proposed algorithm improved the training efficiency of BP-NN algorithms by adaptively modifying the search direction. The initial search direction is modified by introducing the gain value. The proposed algorithm is generic and easy to implement in all commonly used gradient based optimization processes. The simulation results showed that the proposed algorithm is robust and has a potential to significantly enhance the computational efficiency of the training process.

REFERENCES

1. Rumelhart, D.E., G.E. Hinton and R.J. Williams, 1986. Learning Internal Representations by Error Propagation. In: Parallel Distributed Processing, Rumelhart, D.E. and J.L. McClelland (Eds.). MIT Press, ISBN: 0-262-68053-X, pp: 318-362.
2. Gori, M. and A. Tesi, 1992. On the problem of local minima in back-propagation. IEEE Trans. Patt. Anal. Mach. Intel., 14: 76-86. http://doi.ieeecomputersociety.org/10.1109/34.107014
3. Bishop, C.M., 1995. Neural Network for Pattern Recognition. Oxford University Press, ISBN: 10: 0198538642, pp: 504.
4. Fletcher, R. and M.J.D. Powell, 1963. A rapidly convergent descent method for minimization. Comput. J., 6: 163-168. http://comjnl.oxfordjournals.org/cgi/content/abstract/6/2/163
5. Fletcher, R. and C.M. Reeves, 1964. Function minimization by conjugate gradients. Comput. J., 7: 149-160. DOI: 10.1093/comjnl/7.2.149
6. Hestenes, M.R. and E. Stiefel, 1952. Methods of conjugate gradients for solving linear systems. J. Res. Natl. Bureau Stand., 49: 409-435. http://nvl.nist.gov/pub/nistpubs/jres/049/6/V49.N06.A08.pdf
7. Huang, H.Y., 1970. A unified approach to quadratically convergent algorithms for function minimization. J. Optim. Theor. Appli., 5: 405-423. DOI: 10.1007/BF00927440
8. Thimm, G., P. Moerland and E. Fiesler, 1996. The Interchangeability of learning rate and gain in back propagation neural networks. Neural Comput., 8: 451-460. DOI: 10.1162/neco.1996.8.2.451
9. Maier, H.R. and G.C. Dandy, 1998. The effect of internal parameters and geometry on the performance of back-propagation neural networks. Environ. Model. Software, 13: 193-209. DOI: 10.1016/S1364-8152(98)00020-6
10. Eom, K., K. Jung and H. Sirisena, 2003. Performance Improvement of Back propagation algorithm by automatic activation function gain tuning using fuzzy logic. Neurocomputing, 50: 439-460. DOI: 10.1016/S0925-2312(02)00576-3
11. Prechelt, L., 1994. A set of neural network bencmark problems and benchmarking rules. Technical Report 2194. http://page.mi.fu-berlin.de/prechelt/Biblio/1994-21.pdf
12. Nawi, N.M., M.R. Ransing and R.S. Ransing, 2006. An improved learning algorithm based on conjugate gradient methods for back propagation neural network. Proc. Word Acad. Sci. Eng. Technol., 4: 46-54. http://www.waset.org/ijci/v4/v4-1-6.pdf
13. Fisher, R.A., 1936. The use of multiple measurements in taxonomic problems. Ann. Eugen., 7: 179-188. http://www.mendeley.com/c/85931226/Fisher-1936-The-use-of-multiple-measurements-in-taxonomic-problems/
14. Al Bayati, A., 1993. A new non-quadratic model for unconstrained nonlinear optimization. Mutah. J. Res. Stud., 8: 131-155.
15. Tassopoulus, A. and C. Story, 1984. A variable-metric method using a nonquadratic model. J. Optim. Theor. Appli., 43: 383-393. DOI: 10.1007/BF00934462
16. Boland, W.R., E.R. Kamgnia and J.S. Kowallik, 1979. A conjugate gradient optimization method invariant to non-linear scaling. J. Optim. Theor. Appli., 27: 221-230. http://www.springerlink.com/index/T514NN45788701V4.pdf