Iraqi Journal of Statistical Sciences (25) 2013
The 6th Scientific Conference of the College of Computer Sciences & Mathematics
pp [118-132]

Conjugate Gradient Algorithm Based on Meyer Method for Training Artificial Neural Networks

Dr. Khalil K. Abbo* Dr.Hind H. Mohammed**

Abstract

The difference between the desired output and the actual output of an multi-layers feed forward neural network produces an error value can be expressed it as a function of the network weights. Therefore training the network becomes an optimization problem to minimize the error function.

This search suggests a new formula for computing learning rate based on Meyers formula to modify conjugate gradient algorithm (MCG) for training the FFNN. Typically this method accelerate the method of Fletcher–Reeves (FRCG) and Polak–Ribere (PRCG) when using it to solve three different types problems well known in the artificial neural network (namely, XOR problem, function approximation, and the Monk1 problem) with 100 simulations.

خوارزمية تدرج مترافق مستندة على طريقة Meyers لتدريب الشبكات العصبية الاصطناعية

الملخّص

إن الإخلاف بين الناتجِ المطلوبِ والناتجِ الفعليِ للشبكة العصبية الاصطناعية متعدّدة الطبقات ينتج عنه قيمةَ خطأ يمْكِنُ التعبير عنه بدلالة أوزان الشبكة وبذلك تصبح مسألة تدريب الشبكة من مسائل الامثلية التي تبحث في تقليل دالة الخطأ اقل ما يمكن. يهتم هذا البحثِ بتحسين الآلية التي يتم بموجها تقليل دالة الخطأ حيث تم اقتراح صيغة جديدة لحساب عامل تعليم مستندة على صيغة Meyers لتطوير خوارزمية تدرج مترافق لتدريب الشبكات العصبية الاصطناعية ذات التغذية الامامية. عمليا عجلت هذه الطريقة تدريب الشبكة عند مقارنتها بطريقتي (FRCG) و(PRCG) لحَلّ ثلاث انواع مختلفة من المسائل المشهورةِ في الشبكات العصبيةِ الإصطناعيةِ (وهي تقريب الدوال، مسألة XOR ومسألة Monk1) حيث تم تنفيذها لـ100 محاكاة.

* Assist Prof / Dept. Mathematics / Computer Sciences and Mathematics College / Mosul University

** Lecturer/ Dept. Mathematics / Computer Sciences and Mathematics College / Mosul University

Received Date 9/10/2013 Accept Date 21/11/2013

1. Introduction

Learning systems, such as multilayer feed forward neural networks (MLFFNN) are parallel computational models comprised of densely interconnected, adaptive processing units, characterized by an inherent propensity for learning from experience and also discovering new knowledge. Due to their excellent capability of self-learning and self-adapting, they have been successfully applied in many areas of artificial intelligence [Bishop (1995), Haykin (1994), Hmich, etal (2011), Takeuchi, etal.(2003) and Wu, etal.(1995)] and are often found to be more efficient and accurate than other classification techniques[Lerner, etal. (1999)].

The operation of the feed forward neural networks(FNN) is usually based on the equations:

$$net_j^l = \sum_{i=1}^{N_{L-1}} w_{i,j}^{l-1,i}\, x_j^{l-1} + b_j^l \;,\; O_j^l = f(net_j^l) \qquad \ldots\ldots\ldots(1)$$

Where $f(\cdot)$ is the activation function , net_j^l is the sum of the weight inputs for the j-th node in the l-th layer (j=1,2,…, N_l), $w_{i,j}$ is the weights from the i-th neuron to the j-th neuron at the $l-1, l-$th layer ,respectively, b_j^l is the bias of the j-th neuron at the l-th layer and x_j^l is the output of the j-th neuron which belongs to the l-th layer. The problem of training a neural network is to iteratively adjust its weights, in order to minimize a the difference between the actual output of the network and the desired output of the training set [Rumelhart, etal.(1986)]. Actually finding such minimum is equivalent to find an optimal minimization of the error function which defined by:

$$E(w) = \frac{1}{2}\sum_{j=1}^{P}\sum_{i=1}^{M}(O_i^{(j)} - T_i^{(j)})^2 \qquad \ldots\ldots\ldots(2)$$

The variables T_i and O_i are the desired and the actual output of the i-th neuron, respectively. The index j denotes the particular learning pattern. The vector w is composed of all weights in the net.

Back propagation (BP) algorithm is the most widely used to train multilayer feed forward neural networks. The standard back propagation algorithm adjusts the weight vector w using steepest descent with respect to E such that :

$$w_{k+1} = w_k - \alpha\; g_k \;,\; g_k = \nabla E(w_k) \qquad \ldots\ldots\ldots(3)$$

Where the constant α is the learning rate belongs to the interval (0,1) and w_k is a vector representing the weights at iteration (epoch) step k. Since the steepest descent method has slow convergence rate and the search for the global minimum often becomes trapped at a poor local minimum, implies that the back propagation algorithm takes unendurable

time to adapt the weights between the units in the network. For this reason many researches proposed to improve this algorithm; several researches are based on new adaptive learning rate [Abbo & Tatal (2011), Abbo & Mohammed (2013) ,Jarmo, etal. (2003), Johansson, etal.(1990), Kostopoulos, etat.(2004), Plagianakos, et.al.(1998), Sabeur & Farhat (2008), Zulhadi, etal.(2010)]. The other introduce the momentum term [Daniel,etal.(1997), Huajin,etal.(2011)], others use the alternative cost functions or dynamic adaptation of the learning parameters [Shahla,etal.(1997), Steven & Narciso (1999)]. Many apply special techniques of initialization of weights [Nguyen & Widrow (1990)]. Most of them apply the higher order gradient optimization routines to minimize the appropriately error function [Amir, etal.(2005), Livieris & Pintelas(2008), Mollar (1993), Rumelhart,etal. (1986), Abbo & Mohammed (2012)], the multivariable function that depends on the weights of the network. However there is still the problem of accelerating the learning process. Especially when large training sets and large network are used. The neural networks training can be formulated as minimization a non-linear unconstrained optimization problem [Livieris, et.at (2009)].

This search is organized as follows. Section 2 a short description of the conjugate gradient algorithms. section 3 presents the proposed BP algorithm (MBP algorithm say). Section 4, repots our experimental results which are compared of the proposed method with FRCG and PRCG methods through three different types of problems.

2. Conjugate Gradient Methods

Conjugate gradient (CG) methods [Livieris & Pintelas (2008)] are among the most commonly and efficient used methods for large scale optimization problems due to their speed and simplicity. In general, conjugate gradient methods play an important role for efficiently training neural networks due to their simplicity and their very low memory requirements, since they don’t require the evaluation of the Hessian matrix neither the impractical storage of an approximation of it. In the literature there is a variety of conjugate gradient methods [Birgin & Martinez(1999), Moller(1993), Livieris & Pintelas(2011) and Abbo (2010)] that have been intensively used for neural network training in several applications [Daniel, etal. (1997) and Zoutendijk (1970)].

The main idea for determining the search direction is the linear combination of the negative gradient vector at the current iteration with the previous search direction. The way to determine the search direction can be expressed as follows:

$$d_1 = -g_1$$
$$d_{k+1} = -g_{k+1} + \beta_k d_k \qquad \ldots\ldots\ldots(4)$$

Conjugate gradient methods differ in their way of defining the multiplier β_k. The most famous approaches were proposed by Fletcher–Reeves (FR), Polak–Ribere (PR) and Hestenes–Stifel (HS):

$$\beta^{FR} = \frac{g_{k+1}^T g_{k+1}}{g_k^T g_k}, \beta^{PR} = \frac{g_{k+1}^T y_k}{g_k^T g_k}, \beta^{HS} = \frac{g_{k+1}^T y_k}{d_k^T y_k} \qquad \ldots\ldots\ldots(5)$$

The conjugate gradient methods using β^{FR} update were shown to be globally convergent [AL-Baali (1999)]. However the corresponding methods using β^{PR} or β^{HS} update are generally more efficient ever without satisfying the global convergence property. In the convergence analysis and implementations of CG methods, one often requires the inexact lien search such as the Wolfe line search. The standard Wolfe line search requites α_k satisfying:

$$\mathrm{E}(\mathrm{w_k} + \alpha_k d_k) \le \mathrm{E}(\mathrm{w_k}) + \rho \alpha_k g_k^T d_k \qquad \ldots\ldots\ldots(6)$$

$$\mathrm{g}(\mathrm{w_k} + \alpha_K d_k)^{\mathrm{T}} d_k \ge \sigma \mathrm{g_k^T d_k} \qquad \ldots\ldots\ldots(7)$$

or strong Wolfe line search:

$$\mathrm{E}(\mathrm{w_k} + \alpha_k d_k) \le \mathrm{E}(\mathrm{w_k}) + \rho \alpha_k g_k^T d_k \qquad \ldots\ldots\ldots(8a)$$

$$\left|g_{k+1} d_k\right| \le -\sigma g_k d_k \qquad \ldots\ldots\ldots(8b)$$

where $0 < \rho < \sigma < 1$

Moreover, an important issue of CG algorithms is that when the search direction (4) fails to be descent (by Descent, we mean $g_k^T d_k < 0 \ \forall k$) directions we restart the algorithm using the negative gradient direction to grantee convergence . A more sophisticated and popular restarting is the Powell restart.

$$\left|g_{k+1}^T g_k\right| \ge 0.2 \left\|g_{k+1}\right\|^2 \qquad \ldots\ldots\ldots(9)$$

Where, $\| \ \|$ denotes to the Euclidean norm. Other important issue for the CG methods is that the search directions generated from equation (4) are conjugate if the objective function is convex and line search is exact i.e:

$$d_i^T G d_j = 0, \ \forall \ i \ne j \qquad \ldots\ldots\ldots(10)$$

Where, G is the Hessian matrix for the objective function . Dai and Lioa in [Dai and Liao (2001)] showed that the equation (10) can be written as follows:

$$d_{k+1}^T y_k = 0 \qquad \ldots\ldots\ldots(11)$$

which is called pure conjugacy condition and generalize to the

$d_{k+1}^T y_k = -t\, g_{k+1}^T s_k , t > 0$, $s_k = w_{k+1} - w_k$ ……..(12)

for general objective function with inexact line search.

3. Suggested Conjugate Gradient Algorithm:

When a sequence or an iterative process is slowly converging a convergence acceleration process has to be used , Aitken's process is the most well-known convergence acceleration for linearly converging sequence. Abbo and Mohammed in [Abbo & Mohammed(2012)] suggested a new conjugate gradient (NACG) algorithm to train neural network based on Aitken's process which guarantees sufficient descent with Wolfe line search .This algorithm summarize as follows:

The Algorithm NACG

Step1:Initialize w_1 and choose σ, ρ such that $0 < \rho < \sigma < 1$, $\gamma \in (0,1), E_G , \varepsilon > 0$ and K_{max} , set $k = 1$.

Step 2: Calculate the error function value E_k and its gradient g_k .

Step 3: IF $(E_k < E_G) or \|g_k\| < \varepsilon$,set $w^* = w_k$ and $E^* = E_k$, return goal is meet and stop .

Step 4: compute the descent direction :

If $k = 1$ then, $d_k = -g_k$ go to step 6

Else

$$\beta_{k-1}^{NA} = \frac{\gamma\, g_k^T g_k + g_{k+1}^T y_k}{y_k^T d_k}$$ and then compute:

$d_k = -g_k + \beta_{k-1}^{NA} d_{k-1}$.

Step5:compute the learning rate α_k by line search procedure, such the standard Wolfe conditions (6) and (7) .

Step 6: update the weights

$$w_{k+1} = w_k + \alpha_k d_k$$

and set $k = k + 1$.

Step 7: If $k > k_{max}$ return Error goal not meet and stop else go to step (2).

3.1 Accelerated with Meyer Process

The convergence acceleration process transforming the slowly converging sequence in to a new one which, under some assumptions converges faster to the same limit [Clade & Michela (2007)].

In Mathematics and Computer Science, there exists a large number of iterative algorithms whose goal is usually to reach a solution to a problem within a certain tolerance within a given number of iterations. Iterating means going over a pattern of steps and procedures that can sometimes be complex and sometimes take a substantial amount of time even for fast modern computers.[David and William (1982), Meyers A. and Mathews & Fink (1999)]

As we know, Aitken's process is the most well-known convergence acceleration for linearly converging sequence and there are three equivalent forms for this method as follows:

$$w_{k+3} = w_k + \frac{(w_{k+1} - w_k)^2}{(w_{k+1} - w_k) - (w_{k+2} - w_{k+1})} \qquad \ldots\ldots(13a)$$

$$w_{k+3} = w_{k+1} + \frac{(w_{k+2} - w_{k+1})(w_{k+1} - w_k)}{(w_{k+1} - w_k) - (w_{k+2} - w_{k+1})} \qquad \ldots\ldots(13b)$$

$$w_{k+3} = w_{k+2} + \frac{(w_{k+2} - w_{k+1})^2}{(w_{k+1} - w_k) - (w_{k+2} - w_{k+1})} \qquad \ldots\ldots(13c)$$

where $k = 0,1,2,...$

unfortunately, Aitken acceleration need to store the vectors w_{k+2}, w_{k+1} and w_k at every iteration. Which is prompting to suggest new ways to escape from disadvantage. One of these manners is Meyer method. The steps of this method can be illustrated by the following way:

First let us rewrite (13b) by replacing w_{k+2} by $f(w_{k+1})$ and w_{k+1} by $f(w_k)$. It shows the Aitken formula in another form:

$$w_{k+3} = w_{k+1} + \frac{(f(w_k) - w_k)}{(f(w_k) - w_k) - (f(w_{k+1}) - w_{k+1})}(f(w_{k+1}) - w_{k+1}) \quad \ldots..(14)$$

Let us now construct a sequence $\{y_n\}_{n \in N}$. y_0 is the iteration guess start value, chosen sometimes randomly or with a rough estimation depending the problem you are trying to solve. y_1 is the first iteration and defined for the original sequence by $y_1 = f(y_0)$ then let's define $a_0 = 1$. We get:

$$y_1 = f(y_0) = y_0 + f(y_0) - y_0 = y_0 + a_0(f(y_0) - y_0)$$

Let's apply (14) to y_0 and y_1 to determine y_2:

$$y_2 = y_1 + \frac{(f(y_0) - y_0)}{(f(y_0) - y_0) - (f(y_1) - y_1)}(f(y_1) - y_1)$$

If we define:

$$a_1 = \frac{(f(y_0) - y_0)}{(f(y_0) - y_0) - (f(y_1) - y_1)}$$

and use the fact that $a_0 = 1$ then:

$$y_2 = y_1 + a_0\, a_1 (f(y_1) - y_1)$$

In general, Meyers iterative Aitken formula become as:

$$y_{i+1} = y_i + (\prod_{i=0}^{i} a_i)(f(y_i - y_i) \qquad \ldots\ldots(15a)$$

$$a_i = \frac{(f(y_{i-1}) - y_{i-1})}{(f(y_{i-1}) - y_{i-1}) - (f(y_i) - y_i)}, \text{ for } i \geq 1 \text{ and } a_1 = 1 \ldots(15b)$$

It is very important to realize here that it has not been demonstrated that $\{y_n\}_{n \in N}$ is convergent, nor that it respects the necessary conditions to be accelerated by the Aitken formula. Although in practice the Meyers formula works really well in many cases and its convergence is in many case very impressive, it might be necessary to verify that it applies to the problem at hand. Nevertheless the advantage of this method is that it does not require storing all previous values of the sequence and it does not require more use of the function f .

3.2 Derivation of The Proposed Learning Method

In this section we present a new CG algorithm (MCG) by simple multiplicative modification of the learning rate. The idea is to modified the learning rate of the following form :

$$w_{k+1} = w_k + \gamma\, \alpha_k\, d_k$$
$$d_k = -g_k + \beta_{k-1}^{NA}\, d_{k-1} \qquad \ldots\ldots (16)$$

by using Meyers' method (equation 15) together with Wolfe condition (6 and 7), where γ is the learning rate which is used in a classical BP and it has constant value, $\alpha_k \in (0,1)$ is the relaxation parameter (computed by Meyers method) and $d_k = -g_k$, $k = 0$.

3.3 MCG Algorithm

Step1:Initialize w_1 and choose σ, ρ such that $0 < \rho < \sigma < 1$, $\gamma \in (0,1), E_G, \varepsilon > 0$ and $\mathrm{K}_{\max}$, set $\mathrm{k} = 1$.

Step 2: Calculate the error function value E_k and its gradient g_k .

Step 3: IF $(E_k < E_G)\, or \left\| g_k \right\| < \varepsilon$,set $w^* = w_k$ and $E^* = E_k$, return goal is meet and stop .

Step 4: compute the descent direction :

If $k=1$ then, $d_k=-g_k$ and $\alpha_k=1$ then go to step 6

Else

$$\beta_{k-1}^{NA}=\frac{\gamma\, g_k^T g_k+g_{k+1}^T y_k}{y_k^T d_k}$$ and then compute: $d_k=-g_k+\beta_{k-1}^{NA} d_{k-1}$.

Step5: compute the learning rate α_k using equations(15a,b) then use the backtracking to satisfy the standard Wolfe conditions (9) and (10) .

Step 6: update the weights

$$w_{k+1}=w_k+\alpha_k d_k$$

and set $k=k+1$.

Step 7: If $k>k_{\max}$ return Error goal not meet and stop else go to step (2).

4.Experimental Results

In the following section, we will present experimental results in order to study and evaluate the performance of our proposed conjugate gradient algorithm MCG in three classical artificial intelligence problems(XOR problem, Continuous Function Approximation and Monk1 problem).

In particular, we investigate the performance of gradient methods with Fletcher-Reeves update (FRCG) and Polack-Ribiere update (PRCG); (equations 8) then, we compare them with our method MCG. All conjugate gradient methods have been implemented with the Wolfe line search conditions (6) and (7). The implementation has been carried out by using Matlab (2007a) and the Matlab Neural Network Toolbox.

For each test problem, we present a table summarizing the performance of the algorithms for (100) simulations that reached solution within a predetermined limit of epochs. The parameters used in all tables are as follows: Min the minimum number of epochs, Mean, the mean value of epochs, Max the maximum number of epochs, Tav the average of total time, FcEv the number of function evaluations and Succ. The simulation succeeded out of 100 trials within predetermined error limit.

Worth mentioning, for each algorithm, the networks has resaved the same input samples and the same initial weights and if an algorithm fails to converge within the above limit considered, then it fails to train the FFNN, and its epochs are not included in the statically analysis of the algorithm.

4.1 Problem (1):XOR Problem

The XOR problem is considered as one of the well-known test function to train neural network. This function maps two binary inputs to a single binary output. As it is well known, this function is not linearly separable. The network architectures for this binary classification problem consists of two hidden layers with 2 and 3 neurons, respectively, with one neuron in the output layer. The termination criterion is set to $E \leq 0.001$ within the limit of 1000 epochs.

Table (1) summarize the average performance of the presented algorithms for the XOR problem. Clearly that FRCG and PRCG algorithms exhibit excellent probability (93%) of successful training for network architectures. Thus, computational cost is probably the most appropriate indicator for measuring the efficiency of the algorithms. Therefore the FRCG algorithm exhibit, the best performance, since it reports the least average number of epochs to converge, time and the number of function evaluations.

Algorithms	Min	Mean	Max	Tav	FcEv	Succ
FRCG	**4.0**	**55.226**	**648.0**	**0.71874**	**114.19**	**93%**
PRCG	3.0	66.151	1000.0	1.0014	155.46	93%
MCG	5.0	23.628	159.0	0.63406	64.349	86%

Table(1): Comparative the Results of the XOR Problem with Fixing Initial Weights

Finally, Figure (1) present the performance profiles of our proposed conjugate gradient algorithm (MCG) together with FRCG and PRCG with using XOR problem. The interpretation of this Figure show that our proposed method (MCG) is the best algorithm with respect to the number of epochs corresponding with other algorithms.

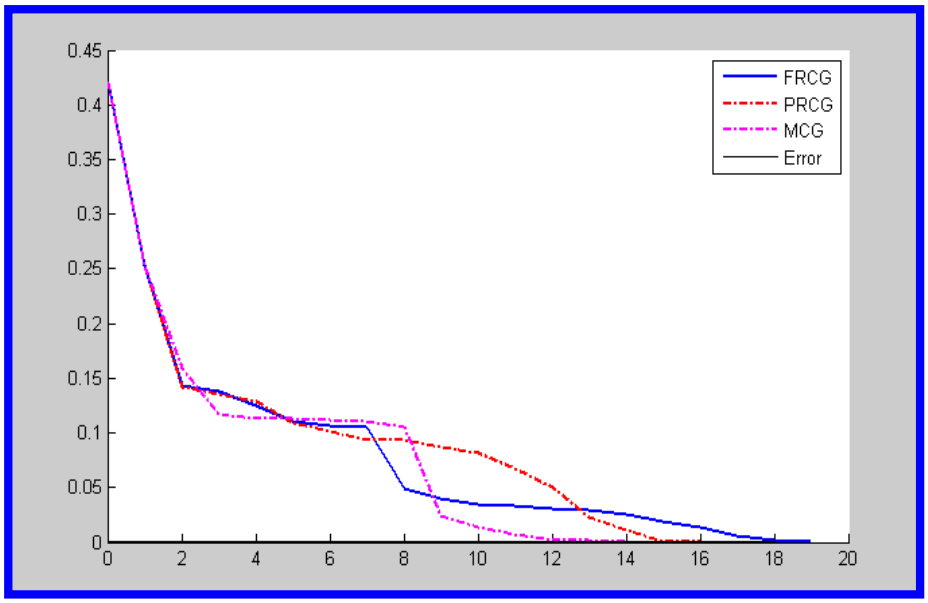

Figure (1):XOR Performance Comparison for Training a FFNN Using FRCG,PRCG and MCG

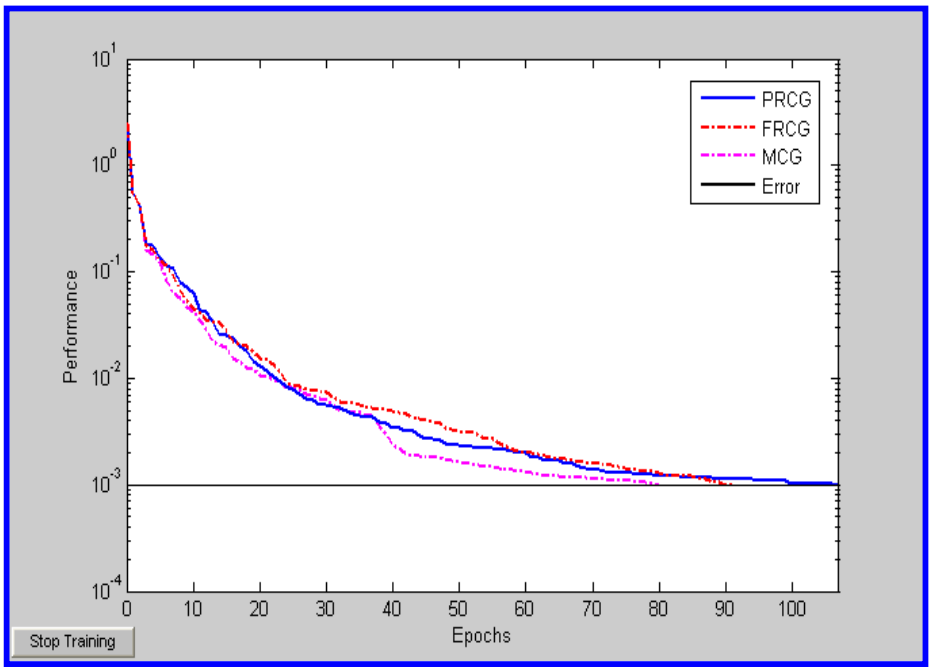

Figure (2):Function Approximation Performance Comparison for Training a FFNN Using FRCG, PRCG and MCG

4.2 Problem (2): Continuous Function Approximation

The second test problem is the Continuous Function Approximation. We consider the approximation of the continuous trigonometric function as:

$$f(x) = sin(x) * cos(3x)$$, where $x \in [-\pi, \pi]$

The network architecture for this problem is 1-15-1 FNN (thirty weights, sixteen biases) is trained to approximate the function and the network is trained until the sum of the squares of the errors becomes less than the error goal 0.001 within the limit of 2000 epochs. The activation function of the hidden neurons is the logistic function with biases and a linear function in the output neuron with bias.

Tables (2) present the performance comparison of the algorithms FRCG, PRCG and MCG for the continuous function approximation problem. All algorithms exhibit excellent probability (100%) of successful training for network architectures using the same initial weights. Thus, computational cost is probably the most appropriate indicator for measuring the efficiency of the algorithms. The new method (MCG) improved the result of FRCG and PRCG, since The MCG significantly outperforms all algorithms in terms of the average number of epochs and the number of function evaluations.

Algorithms	Min	Mean	Max	Tav	FcEv	Succ
FRCG	52	89.41	187	0.8925	148.08	100%
PRCG	59.0	95.76	224.0	1.0636	181.79	100%
MCG	**43.0**	**78.08**	**165.0**	**1.6291**	**130.85**	**100%**

Table(2): Comparative the Results of the Function Approximation Problem with Fixing Weights

Graphically, Figure (2) present the performance profiles of our proposed conjugate gradient algorithm (MCG) together with FRCG and PRCG by using function approximation problem. This Figure show that our proposed method (MCG) is the best algorithm with respect to the number of epochs corresponding with other algorithms.

4.3 Monk1 Problem

The Monk1 problem [Thrun,et.al.,1991] is a collection of three binary classification problems relying on the artificial robot domain, in which robots are described by six different attributes. These benchmarks are made of a numeric base of examples and of a set of symbolic rules.

Monk-1 consists of 124 patterns which were selected randomly from the data set for training, while the remaining 308 were used for the generalization testing.

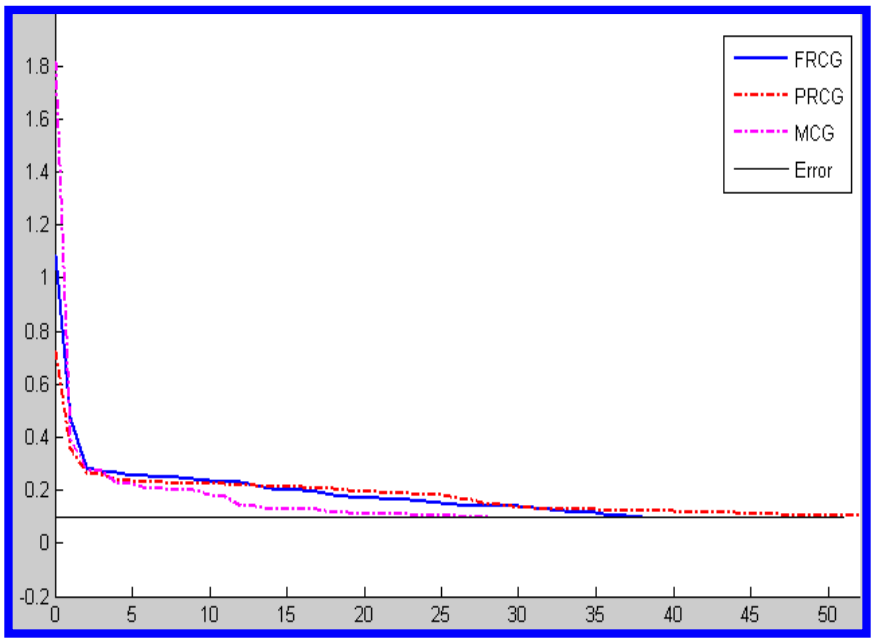

Figure (3):Monk1 Problem Performance Comparison for Training a FFNN Using FRCG, PRCG and MCG

Figure (3) present the performance profiles of our proposed algorithm (MCG) together with FRCG and PRCG with using MONK1 problem. This Figure show that our proposed method (MCG) is the best algorithm with respect to the number of epochs corresponding with other algorithms.

Table (3) summarize the average performance of the presented algorithms for the Monk1 problem. All algorithms exhibit excellent probability (100%) of successful training for network architectures when using the same initial weights. Thus, computational cost is probably the most appropriate indicator for measuring the efficiency of the algorithms.

In general MCG algorithm exhibits the best performance since it report, the least average number of epochs to converge and demonstrate the highest success rate in the case of training a feed forward neural network i.e. MCG is the best.

Algorithms	Min	Mean	Max	Tav	FcEv	Succ
FRCG	21.0	47.93	100.0	0.49831	83.36	100%
PRCG	17.0	36.44	68.0	0.44798	84.19	100%
MCG	**15.0**	**32.9**	**69.0**	**0.98967**	**105.38**	**100%**

Table(3.8): Comparative the Results of the Monk1 Problem with Fixing Initial

References

1. Abbo. K (2010)." Developing of Gradient Algorithms for Solving Unconstrained Non-linear Problems with Artificial Neural Networks". Ph.D. thesis, University of Mosul.

2. Abbo K. and Mohammed .H (2013). " Improving the learning rate of the back propagation by Aitken process". Iraqi Journal of statistical sciences (23),PP[1-10].
3. Abbo K. and Mohammed .H (2013). " Conjugate gradient algorithm based on Aitken's process for training neural networks". Raf. J. of Comp. &Math's., (to appear).
4. Abbo K. and Talal Z. (2011). "Minimization algorithm for training feed forward neural network", J. Of Educ. and Sci. (to appear).
5. AL-Baali M .(1999). "Descent property and global convergence of Fletcher and Reeves method with inexact line search". IMA J. of Numerical Analysis.
6. Amir A., Mohammad B., and Abbas H. (2005). "Modified levenberg-marquardt method for NN training", World Academy of Science, Engineering and Technology 6. 46-48.
7. Birgin E. and Martinez J.(1999). "A spectral conjugate gradient method for unconstrained optimization" Applied Mathematics and Optimization, 43:117-128.
8. Bishop C.M. (1995)." Neural Networks for Pattern Recognition". Oxford.
9. Charalambous C. (1992). " Conjugate gradient algorithm for efficient training of artificial neural networks". IEEE Proceedings, 139(3):301-310.
10. Clade B. and Michela R. (2007). "Generalizations of Aitken's process fo accelerating the convergence of Sequences", J. of Computational and Applied Math. Vol (26). No 2, 171-189.
11. Dai Y and Liao Z (2001). "new conjugacy conditions and related nonlinear conjugate gradient methods", Applied Mathematics and Optimization 43.
12. Daniel S., Vladimir. K and JiE P. (1997). "Introduction to multi-layer feed-forward neural networks.",Chemometrics and Intelligent Laboratory Systems 39 , 43-62.
13. David A. S. and William F. F. (1982)." Numerical Comparisons of Nonlinear Convergence Accelerators". Mathematics of Computation, Vol. 38, No. 158, pp. 481-499
14. Haykin S.(1994)." Neural Networks: A comprehensive foundation". Macmillan College Publishing Company, New York.
15. Hmich A. , Badri A. , and Sahel A.(2011)."Automatic speaker identification by using the neural network". In IEEE 2011 International Conference on Multimedia Computing and Systems (ICMCS), pages1–5.
16. Huajin T., Haizhou L. and Zhang Y. (2011). "Online learning and stimulus-driven responses of neuronsin visual cortex". Cogn Neurodyn 5. Springer:77–85.

17. Jarmo I., Jon K. and Jouni L. (2003). "Differential Evolution Training Algorithm for Feed-Forward Neural Networks", Neural Processing Letters 17: 93–105, Kluwer Academic Publishers. Printed in the Netherlands.
18. Johansson. E. Dowla F. and Goodman G. (1990). "Back propagation learning for Multi-Layer Feed –forward Neural Networks using the Conjugate Gradient method", Lawrence, ivermore National Laboratory. preprint UCRL-JC-104850.
19. Kostopoulos A. Sotiropoulos D. and Grapsa T. (2004). "A new efficient learning rate for Perry's spectral conjugate gradient Training method",1st International Conference ' From Scientific Computing to Computational Engineering'. 1st IC-SCCE.
20. Lerner B., Guterman H., Aladjem M., and Dinstein I.(1999)."A comparative study of neural network based feature extraction paradigms". Pattern Recognition Letters, 20(1):7–14.
21. Livieris I. and Pintelas P. (2008). "A survey on algorithms for training artificial neural networks", Technical Report No. TR08-01, Department of Mathematics University of Patras.
22. Livieris I., Sotiropoulos D., and Pintelas P. (2009). "On descent spectral CG algorithms for training recurrent neural networks", IEEE Computer Society. 13th Panellenic Conference of Informatics, pages 65–69.
23. Livieris I. and Pintelas R. (2011). "An advanced conjugate gradient training algorithm based on a modified secant equation", Technical Report NO. TR11-03.
24. Mathews J., Fink K. (1999). "Numerical Methods Using MATLAB 3rd edition", Prentice Hall.
25. Meyers A.," Accelerating Aitken's convergence accelerator", World Wide Web site at the address http://www.tm.bi.ruhr-uni-bochum.de/profil/mitarbeiter/meyers/aitken.pdf.
26. Moller M. (1993). "A scaled conjugate gradient algorithm for fast supervised learning", Neural Networks,6.
27. Murphy P.M. and Aha D.W.(1994). "UCI repository of machine learning databases". Irvine, CA: University of California, Department of Information and Computer Science.
28. Nguyen D. and Widrow B. (1990). "Improving the learning speed of 2-layer neural network by choosing initial values of the adaptive weights", IEEE First International Jaint Conference on Neural Networks, (3).
29. Plagianakos. V., Sotiropouls D. and Vrahatis. M. (1998). "Automatic adaptation of Learning rate for Back-Propagation Neural networks", Recent advances in circuits and systems, Nikos E. Mastoraksi, ed, world Scientific.

30. Rumethart D. ,Hinton G. and Williams R (1986). "Learning Internal representations by Error propagation", In D. Rumelhart .J. Mc Clelland editors, Parallel Distributions Processing: Exploration in the Microstructure of Cognition. 318-362.
31. Sabeur A. and Farhat F. (2008). "Fast training of multi-layer perceptron with least mean fourth (LMF) Algorithm". International J. of soft computing. 3 (5), 359-367.
32. Shahla K., Ajaya D. and Jyothi N.(1997). "Application of artificial neural networks for the development of a signal monitoring system", Expert Systems, , Vol. 14, No. 2. 69-79.
33. Steven W. and Narciso . (1999). "Heuristic principles for the design of artificial neural networks", Information and Software Technology 41 (2), 109-119.
34. Takeuchi H., Terabayashi Y., Yamauchi K., and Ishii N.(2003)."Improvement of robot sensor by integrating information using neural network". International Journal on Artificial Intelligence Tools, 12(2):139–152.
35. Thrun S. B., Bala J., Bloedorn E., Bratko I., Cestnik B., Cheng J., De Jong K.، Dzeroski S., Fahlman S. E., Fisher D., Hamann R.,
36. Wu C.H., Chen H.L., and Chen S.C. 1995. "Gene classification artificial neural system". International Journal on Artificial Intelligence Tools, 4(4):501–510.
37. Zoutendijk G. (1970). "Non_linear programming, computional methods". Abadie J. ed. Integer and Non_linear Programming, Amsterdam.
38. Zulhadi Z., Nor Ashidi M. and Shahrel A. (2010). "A Study on Neural Network Training Algorithm for Multi face Detection in Static Images". World Academy of Science, Engineering and Technology 62. 170-173.