Hybrid Conjugate Gradient as Neural Network using new line search technique

Dr. Shatha Abdullh Mohammed
Dept. of Computer Scince / Al-Hadbaa University

The 1st Scientific Annual Conference for College of Basic Education (23-24/May/2007)

Abstract:

In this paper a conjugate gradient algorithm was devised as a neural network, based on a middle algorithm, according to which the appropriate learning rate for this network was calculated. Accordingly, a new neural network with high speed and of a supervised type was obtained.

This mathematical style proposed as a new neural network proved to be efficient with regard the results for different English letters

---

التدرج المترافق كشبكة عصبية هجينه باستخدام خط بحث جديد

د.شذى عبد الله محمد
قسم علم الحاسوب / كليه الحدباء الجامعة

ملخص البحث :

في هذا البحث تم استحداث أسلوب خوارزمية التدرج المترافق كشبكة عصبية والمعتمدة على خوارزمية وسطيه يتم بموجبها حساب نسبه التعلم الملائمة لهذه الشبكة وبالتالي يتم الحصول على شبكه عصبيه جديدة ذات سرعة عالية ومن نوع.Supervised

وان هذا الأسلوب الرياضي المقترح كشبكه عصبيه جديدة قد اثبت كفاءة عالية من حيث الحسابات المعتمدة ولإدخالات مختلفة من الحروف الإنكليزية.

1. Introduction :

This paper introduces the modification for one application of Artificial Intelligence, which is neural network, by using classical techniques.

The Neural networks are supposed to reproduce the same mechanisms as in the brain. They are able to solve the problematical problems in real time. They are widely used for excellent results for pattern recognition; also it's capable to learn the underlying mechanics of time series.

A neural network is made of nodes, called, by analogy with the brain, neurons, connected by weighted edges, called synapses. Typically, the nodes are divided into different layers; there is an input layer ,output layer, and hidden layers in between. That forms the topology of the neural network. During a training phase, weights over the synapses are setup in order to pass knowledge through the network.

Where typically one subgroup makes independent computations Neural network (NN): is a group of processing elements where typically one subgroup makes independent computations and passes the results to a second subgroup [17].

Artificial neural network models have been studied for many years in the hope of achieving human-like performance in the fields of difficult real world problems such as pattern recognition, speech and image recognition, and so on[10].

These models are composed of many interconnected non-linear computational elements that operate in parallel and connected together via weights that are typically adapted during use to improve performance [8].

Artificial neural network models or simply neural networks are known by many names such as connectionist models, and neuromorphic systems. Whatever the name, all these models attempt to achieve good performance via dense interconnection of simple computational elements [8][9].

Artificial neural systems function as parallel distributed computing networks. Their most basic characteristic is their architecture. Only some of the networks provide instantaneous responses. Other networks need time to respond and are characterized by their time-domain behavior, which we often refer to as dynamics. Neural networks also differ from each other in their learning modes.

There are a variety of learning rules that establish when and how the connecting weights change. Finally, network exhibit different speeds and efficiency of learning .As a result, they also differ in their capability to accurately respond to the cues presented at the input [19].

Instead of performing a program of instructions sequentially as in VonNeuman computer, neural network models explore many competing

Hypotheses simultaneously using massively parallel network composed of many computational elements connected by links with variable weight [8].

While a digital computers memory is measured in bytes, a neural network "memory" is judged by interconnection; likewise, while the speed of a digital computer is expressed in instructions per second, the neural network speed is measured in interconnections per second.

There are many types of neural networks, but all have three things in common: node (processing unit) characteristics, the connections between them (network topology), and the learning rule. These three aspects together constitute the neural network paradigm [18].

2. The learning rule:

The formalism and its elegant, precise mathematical definition characterize the McCulloch-Pitts model of a neuron. The following paragraphs include the description of that using rule which is applied to recognition. There is some neural network with given architecture. Which find weights for inputs of neurons to train the network for wanted output [15].

2.1Sigmoid Function:

The delta learning rule is supervised turning, and its only valid for continuous activation function $\sigma = \psi(net)$, which offer the possibility of more sophisticated learning algorithms. This monotonically increasing and continuous function satisfying [19]:

$net \in R, \psi(net) \in (-1,1)$, $\lambda > 0$.

$$\psi(net) = \frac{2}{1+\exp(-\lambda(net))} - 1 = \frac{2}{1+\exp(-\lambda(net))} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (1)$$

Where

$$net = w^T x = \sum_{i=1}^{n} w_i x_i$$

2.2 The Derivative of Sigmoid Function:

The Derivative of $\psi(net)$, is denoted by $\psi'(net)$. And it's computed as follows:

$$\sigma = \psi(net) = \frac{2}{1+\exp(-net)} - 1 \Rightarrow \psi'(net) = \frac{1}{2}(1-\sigma^2)$$

This Derivative is given in that general form, without proving therefore we can produce the following steps for proving it:

Since $\sigma = \psi(net)$ then:

First step: by taking the square for the two sides for this equation,

$$\sigma^2 = \left(\frac{2}{1+\exp(-net)} - 1\right)^2$$

$$= \left(\frac{2}{1+\exp(-net)}\right)^2 - 2\left(\frac{2}{1+\exp(-net)}\right) + 1$$

$$= \left(\frac{4}{(1+\exp(-net))^2}\right) - \left(\frac{4(1+\exp(-net))}{(1+\exp(-net))^2}\right) + 1$$

$$= \left(\frac{-4\exp(-net)}{(1+\exp(-net))^2}\right) + 1$$

By subtracting one from this equation:

$$\frac{2\exp(-net)}{(1+\exp(-net))^2} = \frac{1}{2}(1-\sigma^2) \quad \text{…….. (2.a)}$$

Second step: take the derivative of $\sigma = \psi(net)$

$$\sigma = \frac{2}{1+\exp(-net)} - 1 \Rightarrow \sigma = \frac{1-\exp(-net)}{1+\exp(-net)}$$

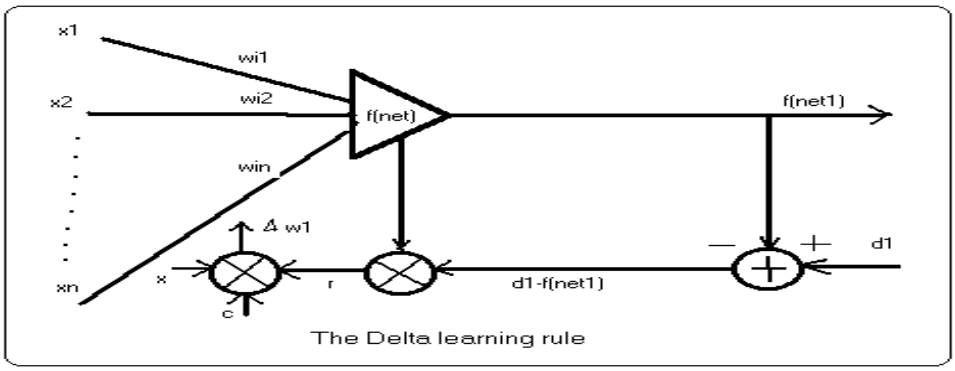
$$\psi'(net) = \frac{2\exp(-net)}{(1+\exp(-net))^2} \quad \text{…….. (2.b)}$$

From (2.a), and (2.b), the desired result is concluding.

2.3 Delta learning rule:

The delta learning rule was introduced only recently for natural network training, which is parallels the discrete perceptron training rule, therefore its called continuous perceptron training rule [17][19]. The suggestions of approaches to error reduction learning, was emergence of networks with activation function.

Therefore the delta learning rule was once of these methods. It's only valid for continuous activation function offer the possibility of more sophisticated learning algorithms.



The delta learning rule is supervised turning, where the learning signal is equal:

$$\gamma = [d_i - f(net)]f'(net)$$

Where d_i is the desired output value, $f(net)$ is the actual output value. Where the delta learning rule is adopted the concept of an error surface, this error surface represents cumulative error over a data set as a function of network weights.

A point on surfaces of this error represents each possible network weight configuration. In order to get learning algorithm, which find the direction on this surface which most rapidly reduces the error. Therefore it is called gradient descent learning. Since the gradient is a measure of slope, as a function of direction, from a point on surface**.**

The error E is depending on the concept of leas mean square errors between d_i and σ_i

$$E_i = \frac{1}{2}(d_i - \sigma_i)^2$$

The component of error gradient is:

$$\nabla E = \frac{\partial E}{\partial w_{ij}} = -(d_i - \sigma_i)\psi'(w_i^T x)x_j \qquad , j = 1,..,n \quad \text{.....(3.a)}$$

The changing in the weight ($\Delta w_i = w_{i+1} - w_i$) to be negative gradient direction it's equivalent to minimization of the error.

$\Delta w_i = -c\nabla E$ …….. (3.b)

By substituting (3.a) in (3.b) to get the delta learning rule, which has the following form:

$\Delta w_i = c(d_i - \sigma_i)\psi'(w_i^T x)x_j \quad , j = 1,..,n$ ……. (3)

Where c is constant, which controlling the learning rate. This is called learning constant. Where the value of it determines how much the weight values move in a single learning episode, if c have large value more quickly the weighs move toward an optimal value, while if the value of c is too large then the algorithm may overshoot the minimum or oscillate. But if c have smaller value, that is less prone to this problem, but do not allow the system to learn as quickly [19].

The optimal value of the learning rate is modified with a momentum factor, which is a parameter adjusted for particular application through experiment.

Outline of standard delta learning rule(S):

The following outlines express the steps, which was produced it for using standard delta learning rule. The initial weight vector is generating randomly, and others computed typically. Where the stopping criteria, use the compare between two vectors $w_{(i)new}$, w_i , $\forall i$.

Step (1): start with i=0, $X = (x_1,...,x_n)$, $W = (w_1,...w_n)$, where x_i , w_i are Vectors. Where W is randomly weight, X is input matrix.

Step (2): Do {i++

$net_i = w_i^T x_i$

$$w_i^* = w_i + (-c)\nabla E$$

$$w_{(i)new} = w_i^*$$

} While ($i \neq n$)

Step (3): check if $\|W_{new} - W\| < \varepsilon, \forall i$, then stop.

Else set $w = w_{new}$, go to Step (2).

3. Hybrid Conjugate Gradient learning rule:

The hinterland was mad between standard delta learning rule and conjugate gradient method, in order to improve the global rate of convergence of the standard delta rule learning technique and to overcome the difficulties of the standard delta rule learning technique, where this turning through the steepest descent direction, which cause the zigzagging problematic, where it is increase the iterations and turning time.

The conjugate gradient method is improved in order to avoid the difficulties of the zigzagging method which is very slow in practice [6][11] .The first CG method was published by Hestenes-Stiefel in 1952 for solving a system of linear algebraic equations, then Fletcher & Reeves in 1964 were the first which used this technique to minimize a nonlinear function of several variables [5].

The descent type's minimization algorithms construct a sequence of iterations, each of which modifies the independent variable vector from the previous iteration.

In general the introducing of the classical CG method in optimization technique, which uses the starting point with the negative gradient (-g), to get the minimum point of the unconstrained problem.

Existing methods for the solution of the general unconstrained minimization problem are commonly classified as gradient or non-gradient according to whether or not they make explicit use of derivatives of the objective function. In the class of gradient method we use the gradient vector g, where f is differentiable function .The conjugate gradient method CG is an effective method for symmetric positive definite systems. It is the oldest and one of the best known of the non-stationary methods. The method proceeds by generating vector sequence of iterate $\{x_i\}$ (which is successive approximation to the solution). The idea of this generating is the concept of conjugate two vectors, in order to determine new directions $(d_i, d_2, ..., d_n)$ of search using information related to the gradient of a quadratic function. This equivalent to find x with $\nabla f(x) = 0$. In such away that successive search directions are conjugate with respect to positive definite Hessian matrix G. At each stage i the direction d_i is obtained by combining linearly the gradient at x_i and $(-g_i)$ and the set of directions $\{d_1, d_2, ..., d_{i-1}\}$, (where d_i is conjugate to each element of this set).

The coefficient of the linear combination is chosen in such away that d_i satisfy this property; therefore the direction d_i is computed as follows:

$$d_{i+1} = -g_{i+1} + \beta_{i+1} d_i \quad \text{..... (4)}$$

Where β_{i+1} is the conjugate coefficient, and it has several values which are introduced from others as follows:

$$\beta_{i+1} = g_{i+1}^T y_i / d_i^T y_i \quad \text{..... [7]..... (4.a)}$$

$$\beta_{i+1} = g_{i+1}^T g_{i+1} / g_i^T g_i \quad \text{..... [5]..... (4.b)}$$

$$\beta_{i+1} = g_{i+1}^T g_{i+1} / g_i^T y_i \quad \ldots\ldots\ldots\ldots [12]\ldots\ldots\ldots ... \quad (4.c)$$

$$\beta_{i+1} = -g_{i+1}^T g_{i+1} / d_i^T g_i \quad \ldots\ldots\ldots\ldots [4]\ldots\ldots\ldots ... \quad (4.d)$$

$$\beta_{i+1} = -g_{i+1}^T y_i / d_i^T g_i \quad \ldots\ldots\ldots\ldots [1]\ldots\ldots\ldots ... \quad (4.e)$$

$$\beta_{i+1} = g_{i+1}^T g_i / g_{i+1}^T g_{i+1} \quad \ldots\ldots\ldots\ldots [16]\ldots\ldots\ldots ... \quad (4.f)$$

Where $y_i = g_{i+1} - g_i$, if the function is quadratic and the exact line search is used all these formulas are identical [3][14], or approximately identical [16].

But when they are applied to arbitrary functions, employing inexact line searches the algorithm already differ.

Therefore the general conjugate gradient method to find minimum of unconstrained Optimization problems have the following outlines [3]:

Step (1): start with x_0 , ε , n

Step (2): do {

$$i = 1, d_i = -g_i$$

Step (3): Compute $x_{i+1} = x_i + \lambda_i d_i$, where λ_i is obtained from the Line search procedure.

Step (4): Find the new direction $d_{i+1} = -g_{i+1} + \beta_{i+1} d_i$; where β_{i+1} is The conjugally coefficient which is computed using Last formulas.

Step (5): Check for convergence if $\|g_{i+1}\| \le \varepsilon$ then stop.

} While (stopping criteria is not satisfying (i = n))

Otherwise set i = i + 1, then go to step (3).

The CG methods have generally the following properties: -

- In the conjugate gradient method the only operation involving G is multiplication of G with a vector.
- The conjugate gradient method is iterative for large problem.
- The CG method can be generalized to a minimization method for general (non-quadratic) smooth function [2].
- $d_i^T G d_j = 0, 0 < j < i$," Conjugate Condition ".
- $g_i^T g_j = 0, 0 < j < i$," Orthogonality Condition ".
- $d_i^T g_i = -g_i^T g_i = -\|g_{i+1}\| < 0$, "Descent Condition ".
- The quadratic termination property with Exact Line Search.

Lemmas:

1. If starting from some point x_0 in direction d till reached the minimum $x_1 = x_0 + \lambda d$ where λ is the minimum of $f(x_0 + \lambda d)$, then $(d^T G(x_1 - x^*) = 0)$.
2. $[d_1, d_2, ..., d_i] = [g_0, g_1, ..., g_{i-1}]$
$= [G(x_0 - x^*), G^2(x_0 - x^*), ..., G^i(x_0 - x^*)]$
3. $g_j^T G\ d_i = 0, 0 < i < j$.
4. If $d_1, d_2, ..., d_i$ are G-conjugate, then $d_1, d_2, ..., d_i$ are linearly independent.

Outlines of Hybrid CG-delta learning rule (H):

The following steps describe the Hybrid Conjugate Gradient learning rule. Which interlard between Conjugate Gradient and standard delta rule:

Step (1): start with i=0, $\chi = (\chi_1,...\chi_n)$, $w = (w_1,...w_n)$, where χ_i , w_i are Vectors. Where W is randomly weight matrix, X is input matrix.

Step (2): Do {i=1

$$net_i = w_i^T x_i$$

$$\nabla E_i = (d_i - \psi_i(net_i))\psi'(net_i)x_i$$

$$D_i = -1 * \nabla E_i$$

$$w_i^* = w_i + c * D_i$$

Step (3): do {i++, $w_{(i)new} = w_i^*$

$$net_i = w_i^T x_i$$

$$\beta_{[16]} \quad ,\nabla E_i = (d_i - \psi_i(net_i))\psi'(net_i)x_i, \beta_i = \frac{\nabla E_{i+1}^T \nabla E_i}{\nabla E_{i+1}^T \nabla E_{i+1}}$$

$$D_i = -1 * \nabla E_i + \beta_i * D_{i-1}$$

$$w_i^* = w_i + c * D_i$$

} While ($i \neq n$)

} While (($\|\nabla E_n\| \geq \varepsilon$) or ($\|w_i^* - w_i\| > \varepsilon, \forall i$))

4. New Hybrid CG-delta learning rule (NH)

In this algorithm we reformulate the Hybrid Conjugate Gradient learning rule by applying middle search to compute the best value of coefficient learning c, which is novel line search [13]. In order to get superiority of learning.

In order to scaling the network with optimal weight in lowest number of iterations. Where in standard delta learning rule the learning coefficient c, was assumed to be arbitrary constant

The following steps describe this new Hybrid Conjugate Gradient learning rule, which based on novel line search.

Step (1): start with i=0, $\chi = (\chi_1,...\chi_n)$, $w = (w_1,...w_n)$, where χ_i , w_i are Vectors. Where W is randomly weight matrix, X is input matrix.

Step (2): Do {i=1

$$net_i = w_i^T x_i$$

$$\nabla E_i = (d_i - \psi_i(net_i))\psi'(net_i)x_i$$

$$D_i = -1 * \nabla E_i$$

$w_i^* = w_i + c * D_i$, Where c is obtained from line search procedure, as following.

Step (2.1): starting with $\in = 0.00001$, let $f(w)$ be any Uni. model function.

$$\text{Set} \begin{Bmatrix} w_1 = a & , & f_1 = f(w_1) & \\ w_2 = T_0 & , & f_2 = f(w_2) & ,0 < T_0 < 1 \end{Bmatrix}$$

Step (2.2): if $f_1 \le f_2 \to T_0 = T_0 / 3$ and go to step (2.1)

Else go to step (3).

Step (2.3):

1st: Set i=i+1

2nd: $f_i = f(w_i)$, $w_i = 2 * T_0$

3rd: if $f_{i-1} \le f_i$, go to step (4)

else set $T_0 = 2 * T_0$, go to 1st:

Step (2.4): the steps (2.1),(2.2) and (2.3) all give the following:

$$y_1 = w_{i-2} \quad , f_{y_1} = f_{i-2}$$

$$y_3 = w_{i-1} \quad , f_{y_3} = f_{i-1}$$

$$y_5 = w_i \quad , f_{y_5} = f_i \text{ , let } f_s = f_{y_3}$$

Step (2.5): the following formulas describe two more points in the interval of uncertainty:

$$y_2 = y_3 - \left| 1 - \frac{f_{y_3}}{f_{y_5}} \right| * \left[1 - \frac{y_3}{y_5} \right], \qquad f_{y_2} = f(y_2)$$

$$y_4 = y_3 + \left| 1 - \frac{f_{y_3}}{f_{y_1}} \right| * \left[1 - \frac{y_1}{y_3} \right], \qquad f_{y_4} = f(y_4)$$

Therefore getting five computed points, which are: $y_1,\ y_2,\ y_3,\ y_4,\ y_5$

Resorting these points by increasing order, S.T. $y_1 < y_2 < y_3 < y_4 < y_5$ and $f_{y_1},\ f_{y_2},\ f_{y_3},\ f_{y_4},\ f_{y_5}$ are their corresponding function values.

Let $f_B = \min\left\{ f_{y_1},\ f_{y_2},\ f_{y_3},\ f_{y_4},\ f_{y_5} \right\}$ and let B its corresponding point.

Take the points A and C in its neighborhood, which is satisfying:

$A < B < C \Rightarrow f_A \geq f_B \leq f_C$ Now can get w_0 as following

$$w_0 = \sqrt{\frac{A^2 + B^2 + C^2}{3}}\ ,\quad f_0 = f(w_0)$$

Therefore four points was getting, $w_0,\ A,\ B,\ C$ and these corresponding function values

$f_0 = f(w_0),\ \ f_A = f(A),\ \ f_B = f(B),\ \ f_C = f(C)$

Again choose three appropriate points and set them as $y_1 < y_3 < y_5$, S.T.

$f_{y_1} \geq f_{y_3} \leq f_{y_5}\ \ , f_{y_1} = f(y_1),\ \ f_{y_3} = f(y_3),\ \ \leq f_{y_5} = f(y_5)$

Step (2.6): if $\left| \frac{f_s}{f_0} - 1 \right| \leq \in$ then stop with $w^* = y_3$

Else set $f_{ss} = \min\left\{ f_s,\ \ f_0 \right\},\ \ f_s = f_{ss}$, and go to step (2.5).

Step (3): do {i++, $w_{(i)new} = w_i^*$

$$net_i = w_i^T x_i$$

$$\beta_{SFR} \quad ,\nabla E_i = (d_i - \psi_i(net_i))\psi'(net_i)x_i, \beta_i = \frac{\nabla E_{i+1}^T \nabla E_i}{\nabla E_{i+1}^T \nabla E_{i+1}}$$

$$D_i = -1 * \nabla E_i + \beta_i * D_{i-1}$$

$$w_i^* = w_i + c * D_i$$

} While ($i \neq n$)

} While (($\|\nabla E_n\| \geq \varepsilon$) or ($\|w_i^* - w_i\| > \varepsilon, \forall i$))

5. Numerical results:

All three algorithms described in this paper:

1. Standard delta learning rule (S).
2. Hybrid CG-delta learning rule (H).
3. New Hybrid CG-delta learning rule (NH).

The complete results are given in table (1) and table (2). These computations are getting from learning the three algorithms on the English liters, which are given in the appendix.

These algorithms are coded using C++ language with numerical results using the personal Pentium IV computer. The comparison of the methods occurred depending on using the number of iterations and total time of learning for the suggestion neural networks.

Table(1): comparisons between Standard delta rule and hybrid methods

Inputs	**N=2**	**N=14**	**N=26**	**Total**
methods	**NOI(Time)**	**NOI(Time)**	**NOI(Time)**	
Standard delta rule	356(1.430)	617(2.420)	880(3.355)	1853(7.205)
Hybrid_ Delta	328(0.715)	656(1.320)	704(1.375)	1688(3.410)
New Hybrid Delta	359(2.060)	453(2.300)	465(2.500)	1277(6.860)

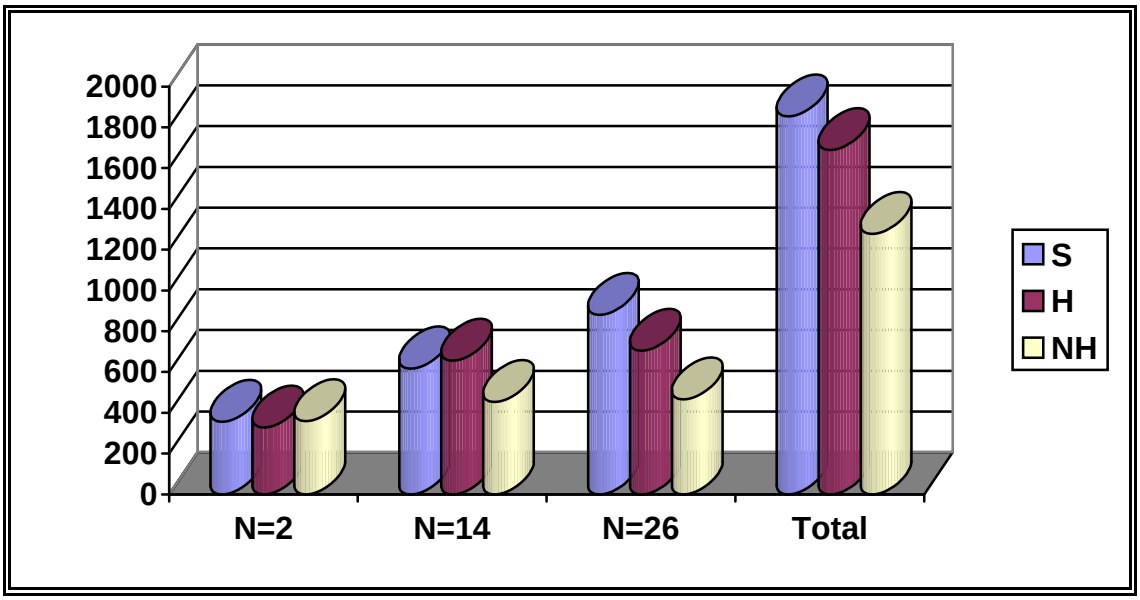
Table(2): improvement ratio Standard delta rule and hybrid methods

	Standard delta rule(S)	Hybrid_ Delta(H)	New Hybrid Delta(NH)
NOI	100%	91.09%	68.91%
Time	100%	47.32%	95.21%

We note that table(1) deals with learning as dimensionality $2 \le n \le 26$. Clearly the performance of the new algorithms Hybrid Delta (H) and New Hybrid Delta (NH) are best compared with the standard delta rule(S).

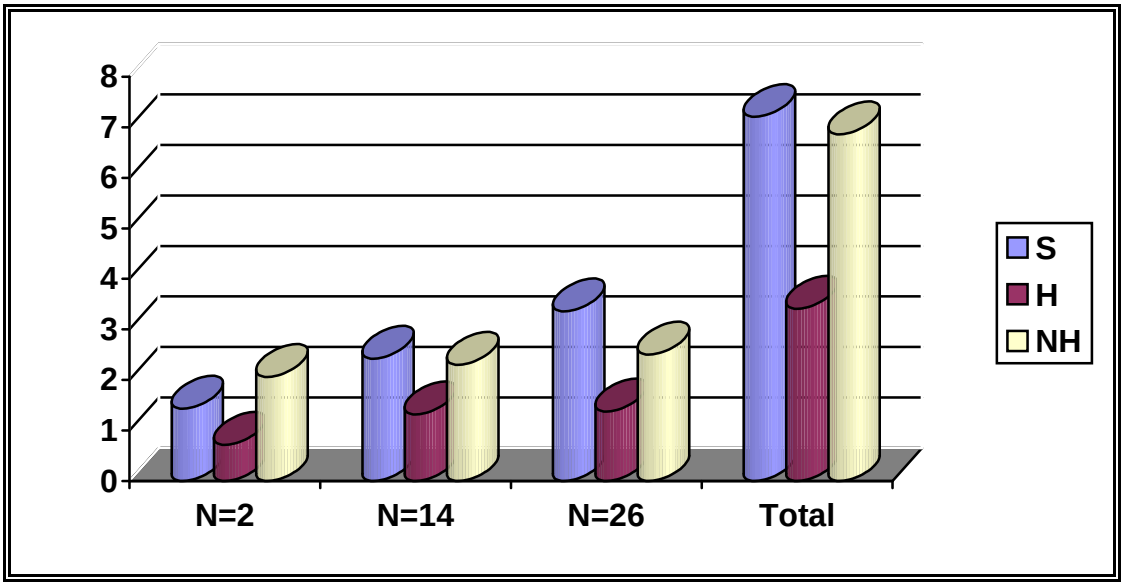
In table (2) is clear that, taking standard delta rule(S) as 100% with respect to number of iterations (NOI) and total time of learning (Time), the Hybrid Delta (H) requires 91.09%NOI and 47.32%Time, while New Hybrid Delta (NH) requires 68.91%NOI and 95.21%Time.

The briefly of these two tables are shown in figure(1) and figure(2), where figure(1) produced the comparisons among Standard(S), hybrid(H) and new hybrid(NH) delta rule depending on number of iterations only, we not that with increasing of inputs N, the number of iterations will reduced.



Figure(1):comparisons between Standard delta rule and hybrid Methods depending on NOI

While figure(2) , produced the comparisons among Standard(S), hybrid(H) and new hybrid(NH) delta rule depending on learning time only, we not that with increasing of inputs N, the learning time will reduced.



Figure(2): comparisons between Standard delta rule and hybrid Methods depending on Time

References

1. Al-Assady, N.H. and Al-Bayati, A.Y.,(1986) ***"Conjugate Gradient Methods"***, Technical Research Report, No.(1),School of Computer Studies , Leeds Universities ,U.K.Fpi.
2. Al-Mashhadany, H.K.M.,(1996) ***"Non-Quadratic Models for Unconstrained Optimization Technique"*** , MS.C, Thesis, University of Mosul, September.
3. Beale, E.M.L., (1988) ***"Introduction to Optimization"***, Wiley inter Science Series in Discrete Mathematics and Optimization.
4. Dixon, L.C.W., Spedicato, E. and Szgoo, G.P., (1975) ***"Non Linear Optimization Theory and Algorithms"***, Birkhauser, Boston L-.S.A.
5. Fletcher, R. and Reeves, C.M.,(1964) ***"Function Minimization by Conjugate Gradient"***, Computer Journal Vol. 7,pp.149-154.
6. Ghorbani, A. Ali and Leila Bayar, (1998) ***"A Correlated Back propagation Learning Dynamic Parallel Tangent Optimization Algorithm"***, IASTED, Irbid, Jordan.
7. Hestenes, M.R. and Stiefle, E., (1952) ***"Methods of Conjugate Gradient for Solving Linear Systems"***, Journal of Research of the National Bureau of Standards 49, pp. 409-436.
8. Lippman R.P., (1987) ***"An Introduction to Computing with Neural Nets"*** IEEE ASSP Magazine, Vol.4, No.2, PP.1-22.
9. Luger G.F., Stubblefied, W.A. and Stubblefield, W.A.,(1998) ***"Artificial Intelligence Structure and Strategies for Complex Problems"*** .Third Edition Addison Wesley Longman.
10. Mustafa, K. M. S., (2002) "Multiple Classification System for Remotely Sensed Data ", Ms. Thesis University of Mosul.
11. Nazareth, L. (1986) ***"Conjugate Gradient methods less dependent on conjugacy"***, SIAM review 28, pp.501-512.

12. Polak E. and Ribiere, G., (1969) ***"Note for Convergence Des Methods Direction Conjugate"***, Rev. Fr. Inf. Rec. Op. 16-R.
13. Rao, M. and Chandra, K., (1983)" ***A New Line Search for Optimization Algorithms"***, Proc. Conf. On systems, M, AM.
14. Rao S.S, (1984) ***"Optimization Theory and Applications"***, Wiley Eastern Limited second edition.
15. Rao, V.B. and Rao, H.V.,(1996) ***" C++ Neural Networks and Fuzzy Logic"***, 2nd Edition, New York.
16. Shatha, A. M.,(2004***) " An Investigation for Intelligence and Classical Optimization for Non Linear Problems"*** , Ph.D. Thesis University of Mosul.
17. Valluru, B. Rao and Hayagriva V. Rao (1993) ***"C++ Neural Networks and Fuzzy Logic"***, Henry Holland Company Inc., New York.
18. Wu Jian-Kang, (1997) ***"Neural Networks and Simulation Methods"***, Marcel Decker Inc., New York.
19. Zurada, J.M., (1994) ***"Introduction to Artificial Neural Systems ",*** JAICO Publishing House, Mumbai.

Appendix