Iraqi Journal of Statistical Science (20) 2011
The Fourth Scientific Conference of the College of Computer Science & Mathematics
pp [237-245]

Scaled Fletcher-Revees Method for Training Feed forward Neural Network

Dr.Basher M. Khalaf [*] Dr.Khalil K. Abbo[**]

Abstract

The training phase of a Back-Propagation (BP) network is an unconstrained optimization problem. The goal of the training is to search an optimal set of connection weights in the manner that the error of the network out put can be minimized. In this paper we developed the Classical Fletcher-Revees (CFRB) method for non-linear conjugate gradient to the scaled conjugate gradient (SFRB say) to train the feed forward neural network. Our development is based on the sufficient descent property and pure conjugacy conditions. Comparative results for (SFRB), (CFRB) and standard Back-Propagation (BP) are presented for some test problems.

طريقة فليتجر-ريفييز بمعلمة لتعليم الشبكة العصبية ذات التغذية الأمامية

المستخلص

تعد تعليم شبكة BP احدى مسائل الامثلية غير المقيدة.إن الهدف من تعليم الشبكة هو البحث عن اوزان مثلى بحيث ان الخطاء الناتج من اخراج الشبكة يصبح اقل ما يمكن. في هذا البحث طورت خوارزمية فليتجر–ريفييز التقليدية الى طريقة التدرج الترافق بمعلمة (SFRB) لتعليم الشبكات العصبية ذات التغذية الامامية، استندنا في تطوير هذه الخوارزمية الى خاصية الانحدار الكافي وشرط الترافق. قورنت الخوارزمية المقترحة مع بعض الخوارزميات المعروفة في المجال نفسه لثلاثة انواع من مسائل الاختبار.

1.Introduction

The Back Propagation (BP) algorithm is perhaps the most widely used supervised training algorithm for multi-layered Feed Forward Neural Networks (FFNN), [6 , 15].

The BP learns a predefined set of output example pairs by using a two-phase propagate adapts cycle. After an input pattern has been applied as a stimulus to first layer of network units, it is propagated through each

[*]Prof.\ **College of Education\ University of Mosul**
[**]Asistant. Prof\ **College of Computers Sciences and Math.\ University of Mosul**

Received:1/10 /2011 Accepted: 21 /12 / 2011

layer until an output is generated [1], this output pattern is then compared to the target output and an error signal is computed for each output unit, the signals are then transmitted backward from the output layer to each unit in the intermediate that contributes directly to the output, each unit in the intermediate layer receives only a portion of the total error signal based roughly on the relative contribution the unit made to the original output. This process repeats layer by layer until each unit in the network has received an error signal that describes it is relative contribution to the total error [18]. Mathematically the standard training problem of a neural network reduces to finding a set of weights w to minimize the error function E defined as the sum of the squared errors in the output [3]

$$E(w,b) = \tfrac{1}{2}\sum_{i=1}^{n_l}(T_i - x_i^l)^2 \qquad (1)$$

Where x^l is a function of w (the weight vector), b (the bias) and T is the target through the equations of the forward pass. This cost function measures the squared error between the desired and actual output vectors.

2.The Standard Backpropagation (SBP) Algorithm

Lets diagram the network as

$$x^{\circ} \xrightarrow{w^1,b^1} x^1 \xrightarrow{w^2,b^2} ,\ldots\ldots\ldots\ldots \xrightarrow{w^l b^l} x^l$$

Where $x^l \in R^{n_l}$ for all $l = 0,\ldots\ldots\ldots,L$ and w^l is an $n_l * n_{l-1}$ matrix for all $l = 1,\ldots,L$, b^l is the bias for each $l = 1,\ldots,L$ there are $L+1$ layers of neurons, and L hidden layers, we would like to change the weights w and biases b so that the actual output x^l becomes closer to the desired output d.

The Backpropagation algorithm consists the following steps.

1- Forward pass. The input vector x° is transformed into the output vector x^l, by evaluating the equation

$$x_i^l(k) = f(u_i^l) = f\left(\sum_{j=1}^{n_{l-1}} w_{ij}^l x_j^{l-1} + b_i^l\right)$$

For l=1 to L, and k is index of iteration usually called epoch

2- Error computation .The difference between the desired output d and actual output x^L is computed

$$\delta_i^l(k) = f'(u_i^l)(T_i - x_i^l) \qquad (2)$$

3- Backward pass. The error signal at the output units is propagated Backwards through the entire network, by evaluating

$$\delta_j^{l-1}(k) = f'(u_j^{l-1})\sum_{i=1}^{n_l}\delta_i^l w_{ij}{}^l \ , \quad \text{from l=L to 1} \qquad (3)$$

4- Learning updates. The weights and biases are updated using the results of the forward and backward passes. Compute the gradient

vector $g_k = \nabla E(k) = \frac{\partial E}{\partial w_{ij}^l(k)}$ and learning rate α_k and update the weights and biases and set $d_k = -g_k$, $\alpha \in (0,0.5)$

$$w_{ij}^l(k+1) = w_{ij}^l(k) + \alpha\ d_k \qquad (4)$$

$$b_i^l(k+1) = b_i^l(k) - \alpha\ \frac{\partial E}{\partial b_i^l} \qquad (5)$$

k=k+1; go to step(1).

Where k is the current iteration usually called epoch, and $w(0) = w_0$, $b(0) = b_0$ are initial weights and biases respectively and $\alpha > 0$ is the learning rate (step-size). We see from step(4) that the SBP algorithm uses the Steepest Descent (SD) search direction i e ($d_k = -g_k$, $\forall\, k$) with fixed step-size ($\alpha = 0.3$ say) or (learning rate) α in order to perform the minimization of the error function E. The inefficiency of (SD) is due to the fact that the minimization directions and learning are chosen poorly, if the first step-size does not lead directly to the minimum SD will zig- zag with many small steps [2 , 10]

The backpropagation search direction d_k is usually augmented with a momentum term [10]

$$d_{k+1} = -g_{k+1} + \beta_k s_k \qquad (6)$$

This extra term is generally interpreted as to avoid oscillations, adding the momentum term is wise when the values α_k and β_k are well chosen. One method which chooses these parameters is known as Conjugate Gradient (CG) method.

In this work we modify the BP algorithm in two ways, the first way instead of using constant learning rate we use line search procedure to compute the learning rate α_k such that the Wolfe conditions (given later) hold, the second way is to modify the search directions d_{k+1} to the Scaled Fletcher-Reeves conjugate gradient algorithm (SFRBP say).

The plan of this paper is as follows: In section 2 we present the proposed (SFRBP) training algorithm. Section 3 contains our numerical examples and results.

2. The Proposed Method

2.1 Conjugate Gradient (CG)

In Conjugate gradient methods the basic idea for determining the search direction d_k in step(4) Eq.(4) is the linear combination of the negative gradient at the current iteration with the previous search direction namely

$$d_{k+1} = -g_{k+1} + \beta_k d_k \qquad , d_1 = -g_1 \qquad (7)$$

In the literature there have been proposed several choices for defining the scalar parameter β_k which give rise to distinct conjugate gradient methods[4, 12]. The most famous ones were proposed by Fletcher-Reeves (FR) [7] defined as:

$$\beta^{FR} = \frac{g_{k+1}^T g_{k+1}}{g_k^T g_k} \tag{8}$$

The convergence analysis [12, 13] of this method is usually based on mild conditions which refer to the Lipschtz and boundedness assumption and is closely connected with sufficient descent property

$$g_k^T d_k < -c\|g_k\|^2 \quad , \ c \in (0,1) \tag{9}$$

Hagar and Zhang [8] presented an excellent survey on conjugate gradient methods. As a learning acceptability criterion we will apply the standard Wolfe conditions that is we suggest that the learning rate α_k can be computed along the search direction d_k by Wolfe line search conditions :

$$E(w_k + \alpha_k d_k) - E(x_k) \le \delta \alpha_k g_k^T d_k \tag{10}$$

$$g_{k+1}^T d_k \ge \sigma \, g_k^T d_k \tag{11}$$

Where d_k is a descent direction and $0 < \delta < \sigma < 1$

2.2 The proposed Scaled FR Training algorithm (SFRBP)

Multilayer networks typically use sigmoid transfer functions in the hidden layers [9]. These functions are often called ' squashing ' functions, since they compress an infinite input range into a finite output range. Sigmoid functions are characterized by the fact that their slop must approach zero as input gets large [18]. This causes a problem when using SD to train Multilayer network with sigmoid functions, since the gradient can have a very small magnitude and therefore cause small changes in the weights and biases even though the weights and biases are far from their optimal values [17]. In this section we introduce a new scaled Fletcher-Reeves (SFRBP) algorithm to train a multilayer feed forward neural networks. The prupose of the SFRBP training algorithm is to eliminate these harmful effects of the magnitudes of the partial derivatives. know consider the scaled search direction of the form:

$$d_{k+1} = -\theta_{k+1} \, g_{k+1} + \beta_k d_k \qquad , d_1 = -g_1 \tag{12}$$

Where θ_{k+1} is parameter. In the conjugate gradient algorithms a search is performed along conjugate directions that is the search directions d_{k+1} satisfies the equation [5]

$$d_{k+1}^T y_k = 0 \tag{13}$$

Assume that the search directions d_{k+1} defined by equation (2.6) satisfies the sufficient descent condition (9), then

$$d_{k+1}^T g_{k+1} = -\theta_{k+1} g_{k+1}^T g_{k+1} + \frac{g_{k+1}^T g_{k+1}}{g_k^T g_k} g_{k+1}^T d_k \le -c\, g_{k+1}^T g_{k+1}, \quad c \in (0,1) \tag{14}$$

Solve the above equation for θ_{k+1} to get

$$\theta_{k+1} = \frac{d_k^T g_{k+1} + c\, g_k^T g_k}{g_k^T g_k}, \quad c \in (0,\infty) \tag{15}$$

To find the value of c we use the equation (13), then

$$d_{k+1}^T y_k = -(\frac{d_k^T g_{k+1} + c\, g_k^T g_k}{g_k^T g_k}) g_{k+1}^T y_k + \frac{g_{k+1}^T g_{k+1}}{g_k^T g_k} y_k^T d_k = 0$$

Then

$$c = \frac{d_k^T y_k \; g_{k+1}^T g_{k+1} - d_k^T g_{k+1} \; g_{k+1}^T y_k}{g_k^T g_k} \tag{16}$$

From equations (16) and (15) we have

$$\theta_{k+1} = \frac{d_k^T y_k \; g_{k+1}^T g_{k+1}}{g_k^T g_k \; y_k^T g_{k+1}} \tag{17}$$

Therefore the search direction for the new scaled FR algorithm is

$$d_{k+1} = -\theta_{k+1} \; g_{k+1} + \frac{g_{k+1}^T g_{k+1}}{g_k^T g_k} d_k \tag{18}$$

Where θ_{k+1} is defined in the equation (17).

New Scaled FR Backpropagation (SFRBP) Algorithm

The steps (1), (2) and (3) are the same as SBP algorithm and step(4) changes to the following form

Step(4):

1. Initialization: use Nguyen widrow method to initialize the weights and Biases and set k=1, $gaol = err$, $\varepsilon > 0$ and compute $g_k = \nabla E(w_{ij}^l(k)$ And $E(w^l(k))$, $1 = 1,..,L$, $d_k = -g_k$, $\alpha_k = 1/\|g_k\|$
2. If $E(w_k^l) < err$ or $\|g_k\| < \varepsilon$ stop $w_{ij}^l(k)$, $l = 1,..,L$ is optimal else goto 3
3. learning rate computation: compute α_k by line search procedure such that Wolfe conditions (10) and (11) are satisfied and update the weights and biases according to the
   $w_{ij}^l(k+1) = w_{ij}^l(k) + \alpha_k d_k$, $\quad b_i^l(k+1) = b_i^l(k) + \alpha_k d_k$
4. Direction computation: compute g_{k+1}, β_k^{FR}, θ_{k+1} and set $d = -\theta_{k+1} g_{k+1} + \beta_k^{FR} d_k$, if Powell restart [13] is satisfied then set $d_{k+1} = -\theta_{k+1} g_{k+1}$

Else $d_{k+1} = d$; k=k+1 go to 2

3. Experimental Results

In this section a computer simulation has been developed to study the performance of the learning algorithms, the simulation has been carried out using MATLAB. The following table lists the algorithms that are tested and the acronyms we use to identify them

Acronym	Algorithm
CBP	**traingd- constant learning rate Backpropagation**
CGFR	**traincgf- Fletcher-Revees conjugate gradient with Powell restart.**
CGPR	**traincgp- Polak-Rebier conjugate gradient with Powell restart.**
SFRB	**our development algorithm (scaled FR algorithm)**

Toolbox default values for the heuristic parameters of the above algorithms are used unless stated otherwise. The algorithms were tested using the same initial weights, initialized by the Nguyen-Widrow method [11] and received the same sequence of input patterns. The weights of the network are updated only after the entire set of patterns to be learned has been presented.

For each of the test problems, a table summarizing the performance of the algorithms for simulations that reached solution is presented. The reported parameters are: min the minimum number of epochs listed in the first column, max the maximum number of epochs listed in the second column, mean the mean value of epochs listed in the third column , Tav the average of total time in the fourth column and finally succ in the last column. The succeeded simulations out of (100) trials within the error function evaluations limit. If an algorithm fails to converge within the above limit, it is considered that it fails to train the FFNN, but its epochs are not included in the statical analysis of the algorithms, one gradient and one error function evaluations are necessary at each epoch.

Problem 1 (XOR problem).

The selected architecture of the FFNN is the 2-3-1 layers with logsig transfer function in the hidden layers and purelin transfer function in out put layer, the error goal has been set to$1*10^{-5}$, table (3.1) shows the results for all algorithms

Table(3.1) Results of simulations for the XOR problem

Algorithms	Min	Max	Mean	Tav	Succ
CBP	fail	----	------	----	------
CGFR	9	80	49.60	0.0326	100%
CGPR	5	87	27.23	0.0142	86.7%
SFRB	6	82	25.34	0.0136	100%

Problem 2 (**Speed control of DC motor problem**).

The data set for this problem are taken from [14], the selected network is 1-377-1 with logsig transfer function in hidden layer and purelin function in output layer, the goal is 0.001, table (3.2) shows the detailed results

Table(3.2) Results of simulations for the Speed control of DC motor problem

Algorithms	Min	Max	Mean	Tav	Succ
CBP	fail	--	--	--	0.0%
CGFR	166	373	254.50	0.5740	100%
CGPR	75	177	115.60	0.0573	100%
SFRB	78	138	104.93	0.0518	100%

Problem (3): (SPECT Heart Problem):

This data set contains data instances derived from Cardiac Single Proton Emission Computed Tomography (SPECT) images from the university of Colorado [9]. The network architectures for this medical classification problem consists of one hidden layer with 6 neurons and an output layer of one neuron. The termination criterion is set to $E \leq 0.1$ within limit of 1000 epochs, table(3.3) summarizes the result of all algorithms i e for 100 simulations

Table(3): Results of simulations for the Heart problem

Algorithms	Min	Max	Mean	Tav	Succ
CBP	fail	--	--	--	0.0%
CGFR	30	70	31.820	0.0910	100%
CGPR	20	56	31.066	0.0703	100%
SFRB	26	49	30.860	0.0657	100%

REFERENCES

[1] Apostolopulou M., Sotiropoulos D., Livieris I and Pintelas . (2009) A memoryless BFGS neural network training Algorithm 7th IEE International Conference on Industrial Informatics.

[2] Bannas, J., Gilbert, J., Lemarechal, C. and Sagastizable, G. (2006), Numerical Optimization, 2th Eddition. Springer-Verlag Brlin Heidelberg

[3] Battiti R. (1992) First and second order methods for learining : between steepest descent and Newtons method ", Neural computation(4).

[4] Dai, Y. and Yuan,Y. (2010), An extended class of non-linear CG -methods, to appear, fifth International conference of Optim. Hong-Kong

[5] Dai,Y. and Liao, L. (2001), New conjugacy conditions and related non-linear CG methods. Appl. Math optim.,(43). Spring-verlag, New York

[6] Daniel Group (2006), Principles Of Artificial Neural Networks, 2nd edition. Advanced Series on Circuits and Systems. Vol.(6)

[7] Fletcher, R. and Reeves, G. (1964), Function minimization by Gradients. Computer. Journal , Vol.(7).

[8] Hager W. and Zhany H. (2006) A surrey of non-linear conjugate gradient methods Pacific J.optim.2.

[9] Livieris I., Sotiropoulos D. and Pintelas P. (2009). "On Descent spectral CG algorithms for Training Recurrent Neural Network", IEEE Computer Society. 13th Panhellenic Conference on Informatics.

[10] Marco, G. and Alberto, T. (1992), On the problem of local minima in Back propagation, TEEE Transactions on Pattern Analysis and Machine Intelligence (14) 1.

[11] 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).

[12] Nocedal J. and Wright S.(1999) Numerical Optimization. Springer Verlag, New York.

[13] Powell M.D. (1984) Nonconvex minimization Calculations and the conjugate gradient Method Numerical Analysis (Dundee. 1983), Leckux Notes in Mathematics Vol. 1066.

[14] Rakan, K. (2005), Line Injection Technique for Harmonics

Reduction in Three-phase Bridge controlled Rectifier, A Thesis For The Degree of Master Of Science, College of Engineering, University of Mosul

[15] Rumelhart D. and Mcclelland J.(1986) parallel distributed processing explorations in the microstructure of Cogniton. Vol.1: Foundations. MTT press.

[16] Shankar, T. (2008), Neural Networks, Laxmi Publications Prt. Ltd.

[17] Sotiropoulos D., Kostopoulos A. and Grapasa T.(2002). A spectral version of perry´s Gonjugate Gradient method for Neural Netwark Training. 4th GRACM Congress on computational Mechanics. Vol(1).Patra.

[18] Watkins C. and Dayan P. (1992) ' Q-learning'. Machine Learning, Vol(8)