Minimization Algorithm for Training Feed Forward Neural Network

Khalil K. Abbo **Zena T. yaseen**
Department Of Mathematic
College of Computer Science and Mathematics
University of Mosul

Received 24 / 04 / 2011 **Accepted** 10 / 07 / 2011

الملخص

في هذا البحث اقترحنا نسبة عامل تعليم جديد α التي تحسن خوارزمية (BP) Backpropagation التقليدية. أشتقاق α استند على تقريب دالة الخطأ E لدالة تربيعية, في جوار النهاية الصغرى المحلية لمتجه الأوزان. الخوارزمية المقترحة Spectral Backpropagation (SBP). نتائج الاختبار لها أظهرت ان (SBP) تحسن بعض الطرائق التقليدية المستخدمة في هذا المجال.

Abstract

In this paper we suggested a new learning rate α, which improves the classical Backpropagation algorithm (BP). The derivatation of α are based on approximating the error function E to the quadratic one in sufficiently small neighborhood for the optimal weight vector. The suggested algorithm (Spectral Backpropagation SBP say) is tested and the experimental results show that the SBP learning strategy improves the considered methods.

1- Introduction

The batch training of a feed forward Neural network (FNN) is consistent with the theory of unconstrained optimization [5] and can be viewed as the minimization of the function E; that is to find a minimizer $w^* = (w_1^*, ..., w_n^*) \in R^n$ Such that:

$$w^* = Min_{w \in R^n} E(w) \qquad (1)$$

where E is the batch error measure defined as

$$E = \frac{1}{2}\sum_{p=1}^{P} \sum_{j=1}^{N_M} (O_{j,p}^M - T_{j,p})^2 = \frac{1}{2}\sum_{p=1}^{P} E_p \qquad , E \in C^2$$

where $(O^{M}_{j,p} - T_{j,p})^2$ is the squared difference error between the actual output value at jth output layer neuron for pattern p and the target output value. The scalar p is index over input-output pairs

The widely used batch Back Propagation (BP) ,[13] is a first order neural network training algorithm, which minimizes the error function using the steepest descent (SD) method [4]:

$$w_{k+1} = w_k - \alpha\, g_k \qquad (2)$$

where k indicates iterations (k=0,1,…) and $g_k = \nabla E(w_k)$, the gradient vector is usually computed by the BP of the error through the layers of the FNN (see [9]) and α is a constant heuristically chosen learning rate or (step length). Appropriate learning rates help to avoid convergence to a saddle point or a maximum. In practice a small constant learning rate is chosen $0 < \alpha < 1$ [12] in order to secure the convergence of the BP training algorithm and to avoid oscillation in a direction where the error function is steep. It is well known that this approach tens to be inefficient [11]. For difficulties in obtaining convergence of BP training algorithm utilizing a constant learning rate see [7]. On the other hand, there are theoretical results that guarantee the convergence when the learning rate is a constant [12]. In this case the learning rate is proportional to the inverse of the Lipschitz constant i. e

$$\| g_{k+1} - g_k \| \le L \| w_{k+1} - w_k \|, \ L > 0 \qquad (3)$$

$$L = \frac{y_k^T y_k}{s_k^T y_k} \qquad (4)$$

where $y_k = g_{k+1} - g_k$ and $s_k = w_{k+1} - w_k$

2- proposed spectral Learining

An interesting new idea is the choice of step length that are proposed by [1] for the steepest descent (SD) method for unconstrained optimization The key element for derivation of our new algorithm is based on the following theorem .

Theorem (1) [3]:

The general function behaves like a quadratic function in a sufficiently small neighborhood of w_k^*. As a consequence of theorem (1) the following relation hold

$$y_k = G_k\, s_k \qquad (5)$$

Note that equation (5) true only on small neighborhood of w^* and G_k is the Hessian matrix of the error function, therefore we may use the Barziui Browein approximation to the G_k [8] i.e.

$$G_k = \frac{s_k^T y_k}{s_k^T s_k} * I_{n*n} \qquad (6)$$

for the unconstrained optimization problem given in equation (1), as we know a necessary condition for the point w^* be an optimal solution is

$$g(w^*)=0 \qquad (7)$$

This is a system of non-linear equations which must be solved to get the optimal solution w^*. In order to fulfill this optimality condition the following continuous gradient flow reformulation of the problem is suggested [6]. Solve the following system of ordinary differential equation:

$$\frac{dw(t)}{dt} = g(w(t)) \qquad (8)$$

with initial condition:

$$w(0)=w_0 \qquad (9)$$

The solution of the system (8) with initial condition (9) is convergence to optimal solution which is minimum of the function given in (1) according to the following theorem see [2] .

Theorem (2) [2]**:** Consider that w^* is a point satisfying (7) suppose that: $G = \nabla^2 E/(w^*)$ is positive definite, if the initial point w_0 is close enough to w^*, then w(t) is the solution to the (8) and tends to w^* as $t \to \infty$.

Theorem (3) [2]**:**

Let w(t) be the solution of (8), for fixed $t_0 \geq 0$ if $g(w(t)) \neq 0$ for all $t>t_0$. then E(w(t)) is strictly decreasing with respect to t for all $t > t_0$.

for proof of theorem (1) and (2) see [2].

As we have seen solving the unconstrained optimization problem (1) has been reduced to that of integration of the ordinary differential equation (8) with initial condition (9). One simple algorithms for solving (8) and (9) is the following [2]:

$$\frac{w_{k+1}(t) - w_k(t)}{h_k} = -[(1-\gamma_k)g_k + \gamma_k g_{k+1}] \qquad (10)$$

Where $h_k = t_{k+1} - t_k$, $0 = t_0 < t_1 < \ldots\ldots < t_k < \ldots$ and $\gamma \in [0,1]$ is scalar. If $\gamma = 0$ the above discretization is the explicit forward Euler's scheme on the other hand where $\gamma = 1$ we have used the implicit backward Euler's scheme. But

$$g_{k+1}=g_k+G_k s_k \qquad (11)$$

from (10) and (11) we get:

$$\frac{w_{k+1} - w_k}{h_k} = -[g_k + \gamma_k G_k s_k]$$

Or $(w_{k+1} - w_k) + h_k \gamma_k G_k(w_{k+1} - w_k) = -h_k g_k$

$\to$ $[I + h\gamma_k G](w_{k+1} - w_k) = -h_k g_k$

Or $w_{k+1}=w_k - h_k[I+h\gamma_k G_k]^{-1} g_k \qquad (12)$

The method based on the algorithm (12) has quite good performance if G_k is positive definite and have desirable feature but not recommended for practical use, the major drawback of the algorithm is computing $[I + h\ \gamma_k\ G_k]^{-1}$. At each iteration and also there is no specified value of h. However one can deduce a simple implementation of the algorithm given in (12) with preserving useful theoretical features as follows:

Since *E (w)* is continuously differentiable therefore the gradient vector g_k is Lipshitz continuous that satisfies equation (3), without loss of generality we may take $h_k = L_k$ i.e.:

$$h_k = \frac{y_k^T y_k}{s_k^T y_k} \qquad (13)$$

From (6), (12) and (13) we get:

$$\frac{1}{h_k}(w_{k+1} - w_k) = -[I + \gamma_k h_k G_k]^{-1} g_k$$

Or $$d_k = -\left[1 + \gamma_k \frac{y_k^T y_k}{s_k^T y_k}\frac{s_k^T y_k}{s_k^T s_k}\right]^{-1} g_k$$

$$\therefore d_k = -\frac{s_k^T s_k}{s_k^T s_k + \gamma_k\, y_k^T y_k} g_k \qquad (14)$$

therefore we can adjust the weight vector according to the following equation:

$$w_{k+1} = w_k - \alpha_k g_k \qquad (15)$$

where: $$\alpha_k = \frac{s_k^T s_k}{s_k^T s_k + \gamma_k\, y_k^T y_k} \qquad (16)$$

We summarize the above algorithm (15-16) (the specral step size SBP) as follows:

Step (1): Initialization: number epochs k = 1, $\gamma_k \in (0,1)$, error goal = eg, weight vector = w_k stopping criteria = ε, $g_k = \nabla E(w_k)$ $\alpha_k = \dfrac{1}{\| g_k \|_2}$.

Step (2): Check for convergence:

If $\|g_k\|_2 < \varepsilon$ or $E(w_k) < eg$, stop w_k is optimal else:

$w_{k+1} = w_k - \alpha_k g_k$ and go to step (3).

Step (3): Compute $g_{k+1} = \nabla E(w_{k+1})$, $s_k = w_{k+1} - w_k$

$y_k = g_{k+1} - g_k$, $E(w_{k+1})$, $\alpha_{k+1} = \dfrac{s_k^T s_k}{s_k^T s_k + \gamma_k\, y_k^T y_k}$

step (4): k = k+1 go to step (2).

3- Experiments and Results:

A computer simulation has been developed to study the performance of the learning algorithms. The simulations have been carried out using MATLAB version 5.4. The performance of the specral-step size BP (SBP) has been evaluated and compared with batch versions of BP, constant learning BP (CBP) known as (traingd) see Appendix, in the neural net work toolbox, adaptive BP (ABP) (traingda) and BP with momentum MBP (traingdx). 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 [10] 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, mean the mean value of epochs, max the maximum number of epochs, Tav the average of total time and succ. 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 FNN, 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.

3.1 Problem (1): (SPECT Heart Problem):

This data set contains data instances derived from Cardiac Single Proton Emission Computed Tomography (SPECT) images from the university of Colorado [8]. 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(1) summarizes the result of all algorithms i e for 100 simulations the minimum epoch for each algorithm are listed in the first column (Min), the maximum epoch for each algorithm are listed in the second column, third column contains (Tav) the average of time for 100 simulations and last columns contains the percentage of succeeds of the algorithms in 100 simulation.

Table(1): Results of simulations for the Heart problem

Algorithms	Min	Max	Mean	Tav	Succ
CBP	fail	--	--	--	0.0%
ABP	124	828	563.09	2.82s	71.65%
MBP	114	444	262.12	1.63s	67.34%
SBP	119	571	298.40	1.82s	70.13%

3.2 Problem (2):

Continuous function Approximation:

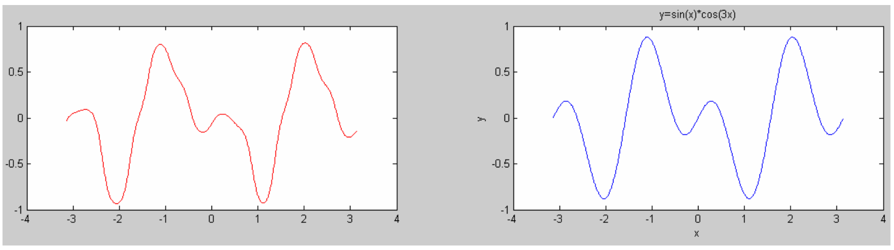
The second test problem we consider is the approximation of the continuous trigonometric function:

f(x)=sin(x)*cos(3x).

The network architectures for this problem is 1-15-1 FNN (thirty weights, sixteen biases) is trained to approximate the function f(x), where $x \in [-\pi,\pi]$ and the network is trained until the sum of the squares of the errors becomes less than the error goal 0.1. The network is based on hidden neurons of logistic activations with biases and on a linear output neuron with bias. Comparative results are shown in table (2). Figure (1) shows performance of SBP.

Table(2): Results of simulations for the function approximation problem

Algorithms	Min	Max	Mean	Tav	Succ
CBP	907	1965	1050.0	12.35s	46%
ABP	77	170	112.2	1.79s	100%
MBP	97	108	101.9	1.61s	100%
SBP	78	158	102.6	1.72s	100%



(a) Approximated function **(b) Target function**

Figure(1); Performance of SBP

Appendix

1- traingd: is matlab function (in the matlab toolbox) utilize steepest descent direction with constant step-size to minimize error function E (training the network) known as standard Backpropagation.

2- traingda: is matlab function (in the matlab toolbox) utilize steepest descent direction with adaptive step-size to minimize error function E (training the network) known as standard Adaptiv Backpropagation.

3- traingdx: is matlab function (in the matlab toolbox) utilize steepest descent direction with momentum and computes step-size by line search procedure to minimize error function E or (training the network).

References

1) Abbo. K. (2007). "Modifying of Barzilai and Borwein Method for solving large scale unconstrained optimization problem", Iraqi J. of statical science Vo. (11). No.(7).

2) Andrei N. (2003). "Gradient flow algorithm for unconstrained optimization", Research Institute for informatics center for advanced Modeling optimization 8-10 Aver-escu Avenue, Bucharest.

3) Fletcher R. (1987). "Practical Methods Of Optimization", 2nd Edition. John Wilely Chichester.

4) Gill P., Murrag W. and Wright M. (1981). "Practical Optimization", Academic Press, NY.

5) Johansson. E. Dowla F. and Goodman G. (1990). "Back propagation learning for Multi-Layer Feed–forward Neural" Networks using the Conjugate Gradient method Lawrence, Livermore National Laboratory . preprint UCRL-JC-104850.

6) Khalaf, B. and Al-Wagih, K.(2001). "Parallel shooting Method for unconstrained numerical optimization", Raf. J. Sci. Vol(12). No.(2).

7) Kuan G. and Hornik K. (1991). "Convergente of Learning algorithms with constant learning rates", IEEE Trans. Neural Networks, (2).

8) 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.

9) Moller. F. (1993). "A scaled conjugate gradient algorithm for fast supervised learning", Neural networks. (6).

10) 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).

11) Nocedol J. (1992). "Theory of algorithms for unconstrained optimization", Acta Numerica, Vol. (199), No. (242).

12) 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.

13) Rumethart D., Hinton G. and Williams R (1986). "Learning Internal representations by Error propagation in Parallel Distributions Processing"; Exploration in the Microstructure of Cognition, Eds. D. Rumelhart .J.L. Mc Cleland, MIT press Cambridge. MA.