Proceedings of the 9th WSEAS International Conference on Applied Mathematics, Istanbul, Turkey, May 27-29, 2006 (pp356-361)

An Improved Version of Backpropagation Algorithm with Effective Dynamic Learning Rate and Momentum

MAMMADAGHA MAMMADOV
Statistics, Faculty of Science
T.C. Anadolu University
26470 Eskisehir
TURKEY

ENGIN TAS
Statistics, Faculty of Science and Literature
Afyon Kocatepe University
03200 Afyon
TURKEY

Abstract: - An improvement of backpropagation algorithm with momentum is introduced. Local quadratic approximation of the error function is performed at every stage of the learning process and the Hessian matrix of the quadratic error function is approximated [1]. Efficient learning rate and momentum factor is determined at every stage of the learning process by means of maximum and minimum eigenvalues of the Hessian matrix. The effective performance of this new approach is demonstrated on three examples.

Key-Words: - Multilayer neural networks; Backpropagation algorithm with momentum; Optimization; Momentum; Supervised learning; Local quadratic approximation

1 Introduction

Learning algorithms in multi layer feed forward neural networks based on the minimization problem of the error function [2, 3]. Backpropagation (BP) algorithm which becomes popular by the work [4], takes its origin from the gradient descent method in numeric optimization. Afterwards, using different class of numerical optimization methods, various first and second order algorithms have been developed for training of neural networks [2, 3]. One simple and important modification of BP algorithm is to add momentum term to the gradient descent formula [5]. This effectively adds inertia to the motion through weight space and smoothes out the oscillations [3]. The inclusion of momentum generally leads to a significant improvement in the performance of gradient descent and introduce a second parameter μ whose value needs to be chosen, in addition to that of the learning rate parameter η. One obvious problem is to choose learning rate η and momentum factor μ efficiently and automatically in BP with momentum (BPM) algorithm.

There have been numerous studies on the stability and the convergence speed of the BPM algorithm [12-19]. (e.g. Jacobs 1988; Fahlman 1989; Silva and Almeida 1990; Le Cun et al. 1993; Hagiwaro and Sato 1995, Kamarathi and Pittner 1999, Phlansalkar and Sastry 1994, Yu and Chen 1997).

Qian, 1999 demonstrates an analogy between the convergence of the momentum algorithm and the movement of Newtonian particles in a viscous medium. By utilizing a discrete quadratic approximation to this continuous system, Qian also derives the conditions for stability of the algorithm for a quadratic function. Torii and Hagan (2002) analysis the effect of momentum on the stability and speed of convergence of the steepest descent algorithm applied to quadratic functions. Amit Bhaya (2004) establishes various connections between the CG algorithm and the BPM acceleration for a quadratic error function.

A modified BPM algorithm is proposed based on the results obtained in section 2. Main principle in here is to apply efficient learning rate and momentum factor using local approximation of the error function and fitting Hessian at every occurring weight point.

2 Stability and Convergence Speed of the BPM Algorithm for Quadratic Error Functions

Consider the gradient descent with momentum algorithm

$$x_{t+1} - x_t = -(1-\mu)\eta H x_t + \mu(x_t - x_{t-1}) + (1-\mu)\eta b \qquad (1)$$

for the minimization of the following quadratic error function

$$E(x) = \frac{1}{2} x^T H x - b^T x + c \qquad (2)$$

where μ is the momentum factor, η is the learning rate, H is an $n \times n$ symmetric positive definite matrix, b is an n-dimensional vector and c is a given constant. Gradient of the quadratic function E

at point x is $\nabla E(x) = Hx - b$.

Applying the orthogonal transformation $x' = Q^T x$ (Q is a matrix which is formed by orthonormal eigenvectors of H), (2) can be rewritten in coordinates as [6, 7]

$$x'_{i,t+1} = [1+\mu-(1-\mu)\eta\kappa_i]x'_{i,t} - \mu x'_{i,t-1} + (1-\mu)\eta b'_i \;,\; i=\overline{1,n} \quad (3)$$

where k_i, $i=\overline{1,n}$ the eigenvalues of symmetric and positive definite matrix H. Then the coordinates of vector x are obtained by the linear combination of the coordinates of x'. Including the dummy equation $x'_{i,t} = x'_{i,t}$, we can write (3) in matrix form:

$$\mathbf{x}_{i,t+1} = P_i \mathbf{x}_{i,t} + d_i \;,\; \mathbf{x}_{i,t} = \begin{pmatrix} x'_{i,t-1} \\ x'_{i,t} \end{pmatrix},\; i = 1,2,...,n \quad (4)$$

where $P_i = \begin{pmatrix} 0 & 1 \\ -\mu & 1+\mu-(1-\mu)\eta\kappa_i \end{pmatrix}$ is a 2×2 matrix, $d_i = \begin{bmatrix} 0 \\ (1-\mu)\eta b_i \end{bmatrix}$ is a two-dimensional vector ($i = 1,2,...,n$). The linear dynamic system given by (3) or (4) is stable if the magnitudes of eigenvalues of P_i matrix is smaller than one [8]. Thus a relation is set upped between the stability problem of backpropagation with momentum (BPM) algorithm (1) and the magnitudes of eigenvalues of P_i matrix.

We can write the corresponding characteristic equation for finding the eigenvalues of P_i ($i = 1,2,...,n$) matrix:

$$|P_i - \lambda I| = \begin{vmatrix} -\lambda & 1 \\ -\mu & 1+\mu-(1-\mu)\kappa_i - \lambda \end{vmatrix} = 0, \quad ,.$$

$i = 1,2,...,n$

Thus we have that the λ eigenvalues of P_i matrix are the roots of the following quadratic equations [6, 9]:

$$\lambda^2 - [(1+\mu)-(1-\mu)\eta\kappa_i]\lambda + \mu = 0,\;\; i = 1,2,...,n \quad (5)$$

Therefore the stability problem of (1) gradient descent algorithm becomes the examination of (5). Roots of (5) correspond to any κ eigenvalue of H matrix, can be calculated as

$$\lambda = \frac{[(1+\mu)-(1-\mu)\eta\kappa] \pm \sqrt{[(1+\mu)-(1-\mu)\eta\kappa]^2 - 4\mu}}{2} \quad (6)$$

Let us take the quadratic function on the left-hand side of (5) (with respect to λ):

$$\varphi(\lambda) = \lambda^2 - [(1+\mu)-(1-\mu)\eta\kappa]\lambda + \mu \quad (7)$$

Discriminant of this quadratic form $D = [(1+\mu)-(1-\mu)\eta\kappa]^2 - 4\mu$, or if we write according to the degrees of momentum factor μ then

$$D(\mu) = (1+\eta\kappa)^2\mu^2 - 2(1+\eta^2\kappa^2)\mu + (1-\eta\kappa)^2 \quad (8)$$

In the case of $D<0$, the roots of (5) are conjugate complex numbers and their magnitudes are constants that equal to $|\lambda| = \sqrt{\mu}$. The sign of function $D(\mu)$ is determined as [7]

$$D(\mu)\begin{cases} <0 & , \quad S(\eta\kappa) < \mu < 1 \\ =0 & , \quad \mu = 1 \; veya \; \mu = S(\eta\kappa) \\ >0 & , \quad \mu > 1 \; veya \; \mu < S(\eta\kappa) \end{cases} \quad (9)$$

where $S(\eta\kappa) = \frac{(1-\eta\kappa)^2}{(1+\eta\kappa)^2}$. For a given matrix H $S(\eta\kappa)$ is a function of η variable. $S(\eta\kappa)$ as a function of $\eta\kappa$ have the following properties: $S(\eta\kappa)$ decreases from 1 to 0 in the interval $0 \le \eta\kappa \le 1$ and takes the minimum value 0 at $\eta\kappa = 1$. This function increases if $\eta\kappa > 1$. $S(\eta\kappa)$ is convex in $0 \le \eta\kappa \le 2$, and concave in $(2, +\infty)$. $\eta\kappa = 2$ is the turning point (see fig. 1).

Teorem [7] Assume that η is the learning rate and κ_i, $i = 1,2,...,n$ are the eigenvalues of the symmetric positive definite H matrix. If $0 < \eta\kappa_i \le 2$, $i = 1,2,...,n$ then the BPM algorithm (1) is stable for any momentum factor μ in the range (0,1) else if $\max_i \eta\kappa_i > 2$ then (1) is stable for any momentum factor μ in the range $\max_i \frac{\eta\kappa_i - 2}{\eta\kappa_i + 2} < \mu < 1$ (Proof of the theorem is not given here).

In fig. 1, for the stability of (2) iterative process, variation interval of μ with respect to $\eta\kappa$ is demonstrated geometrically.

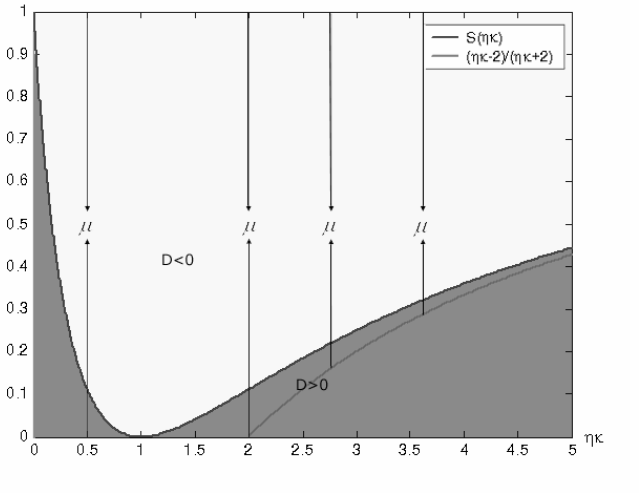

Figure I.

Note 1. From the proof of the theorem [7] it can be seen that the following statements are true:
If $0<\eta\kappa<1$ and $D(\mu)>0$ then the appropriate roots of (5) settle in (0,1).
If $1<\eta\kappa\le 2$ and $D(\mu)>0$ then the appropriate roots of (5) settle in (-1,0).
If $\eta\kappa>2$ and $D(\mu)>0$ then the appropriate roots of (5) settle in (-1,0).
Note 2. The theorem given above can be expressed in short:
Assume that η is the learning rate and $\kappa_i, i=1,2,...,n$ are the eigenvalues of the symmetric positive definite matrix H, then the BP with momentum algorithm given by (1) is stable for the momentum factors in the range

$$\max\{0\ ,\ \max_i \frac{\eta\kappa_i-2}{\eta\kappa_i+2}\}<\mu<1$$

Note 3. From the proof of the theorem, it can be seen that: While the momentum factor μ changes in $-1<\mu<1$, $\max_i \frac{\eta\kappa_i-2}{\eta\kappa_i+2}<\mu<1$ is the necessary and the sufficient condition for algorithm (1) to be stable.
As explained in [9] the convergence speed of the algorithm depends on the magnitudes of λ eigenvalues – the smaller the magnitude the faster the convergence – when λ eigenvalues are complex. This shows that for a given learning rate, the choice $\mu=\max_i \frac{(1-\eta\kappa_i)^2}{(1+\eta\kappa_i)^2}=\max_i S(\eta k_i)$ provides a better convergence in general. In fact, a better choice of learning rate η should shrink the magnitudes of λ eigenvalues more. We propose to determine $\eta=\eta^0$ from the following minimax problem:

$$\max_i \frac{(1-\eta^0\kappa_i)^2}{(1+\eta^0\kappa_i)^2}=\min_{0<\eta}\max_i \frac{(1-\eta\kappa_i)^2}{(1+\eta\kappa_i)^2}$$

Assume that the eigenvalues of the symmetric positive definite H matrix are ordered in this way:

$$0<k_n\le k_{n-1}\le ...\le k_2\le k_1,$$

where k_n is the smallest and k_1 is the largest eigenvalue. In this case, the plots of $S_i(\eta)=\frac{(1-\eta\kappa_i)^2}{(1+\eta\kappa_i)^2}$, $i=1,2,...,n$ function comparisons of η are illustrated in fig. 2.
As can be seen from fig. 2, the function $\mathcal{S}(\eta)=\max_i S_i(\eta)=\max_i \frac{(\eta\kappa_i-1)^2}{(\eta\kappa_i+1)^2}$ can be defined as

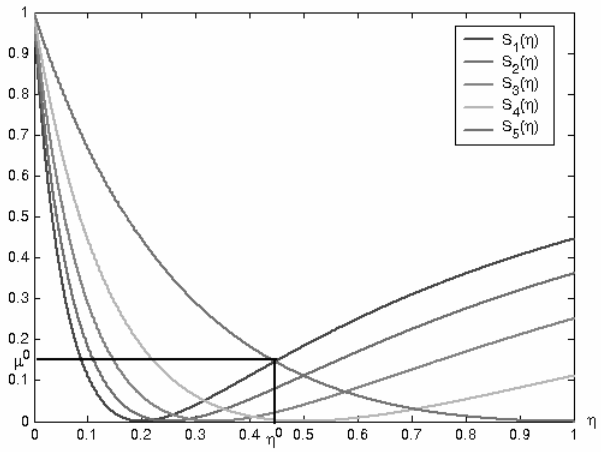

Figure II.

$$\mathcal{S}(\eta)=\begin{cases} S_n(\eta) & ,\ 0\le\eta\le\eta_{1,n} \\ S_1(\eta) & ,\ \eta\ge\eta_{1,n}\end{cases}\ ,\ \text{or}$$

$$\mathcal{S}(\eta)=\begin{cases} \frac{(1-\eta k_n)^2}{(1+\eta k_n)^2} & ,\ 0\le\eta\le\eta_{1,n} \\ \frac{(1-\eta k_1)^2}{(1+\eta k_1)^2} & ,\ \eta\ge\eta_{1,n}\end{cases}$$

In this case, it is easy to see that the solution point of the problem $\min_\eta \mathcal{S}(\eta)$ is $\eta^0=\frac{1}{\sqrt{k_1k_n}}$. Thus, a good

convergence speed is achieved by taking the learning rate and the momentum factor as

$$\eta = \eta^0 = \frac{1}{\sqrt{k_1 k_n}} \tag{10}$$

$$\mu = \mu^0 = S(\eta^0) = \frac{\left(\sqrt{\frac{k_1}{k_n}} - 1\right)^2}{\left(\sqrt{\frac{k_1}{k_n}} + 1\right)^2} \tag{11}$$

For faster convergence of the gradient descent with momentum algorithm, (10) and (11) formulas are proposed for determining the learning rate and momentum factor respectively. Results obtained from the experiments support this proposal.

3. MODIFIED BPM (MBPM)

Consider the gradient descent with momentum algorithm for BPM. Assume that $x \in R^n$ is the weight vector and $E(x)$ is the error function representing the total squared errors for the whole input pattern set. Since sigmoid shaped activation functions are used in hidden layers, the error function $E(x)$ is generally nonlinear. Backpropagation is a gradient descent algorithm for finding the minimum of the error function $E(x)$; hence it can be written as the following:

$$x_{t+1} - x_t = -(1-\mu)\eta \nabla_x E(x_t) + \mu(x_t - x_{t-1})$$

Keeping the above idea as the basis, we propose to modify BPM algorithm so as to work with dynamic dynamic efficient values of learning rate and momentum factor. To achieve this, at every step of the algorithm, we consider the right hand side of (BPM) as a search direction for the minimum of the local quadratic approximation of the error function $E(x)$. If point the search direction with dx_t at step t, we have

$$dx_t = -(1-\mu)\eta_t \nabla_x E(x_t) + \mu_t (x_t - x_{t-1}), \tag{12}$$

where η_t is the dynamic learning rate and μ_t is the dynamic momentum factor at step t which is determined by (10) and (11) respectively. In order to obtain these values, we have to compute Hessian matrix at every step. Since this computation is expensive, instead of using exact Hessian we can use an approximation which is given as [1]

$$H_t = H_{t-1} + \frac{pp^T}{p^T dx_{t-1}} + \frac{vv^T}{v^T (x_t - x_{t-1})}, \tag{13}$$

where $p = \nabla E_x(x_{t-1})$, $v = \nabla E_x(x_t) - \nabla E_x(x_{t-1})$ and dx_{t-1} is the previous search direction. And weight vector x is updated according to the following

$$x_{t+1} = x_t + \alpha_t dx_t, \tag{14}$$

where α_t is found by the line search to minimize $E(\alpha) = E(x_{t-1} + \alpha dx_{t-1})$.

Modified BPM can now be described as follows.

1. Choose initial weight vector x_0.
2. If first iteration,
   calculate $E(x_0)$ and $\nabla E_x(x_0)$,
   set initial search direction to negative gradient, $dx_0 = -\nabla E_x(x_0)$,
   set initial Hessian to the unit matrix, $H = I$.
3. After first iteration,
   calculate change in gradient, $v = \nabla E_x(x_t) - \nabla E_x(x_{t-1})$
   calculate new Hessian approximation by (13)
   calculate η, μ using (10), (11) respectively,
   set the new search direction dx_t by (12)
4. Check the search direction whether it is a descent direction or not.
   If it is not a descent direction
   set Hessian to the unit matrix, $H = I$,
   set search direction to negative gradient
   end if
5. Update the weights by (14)
6. Check for the stopping criteria. Repeat steps 3-6.

4. Simulation Results

In this section the convergence behaviour of the proposed BP training algorithm with efficient learning rate and momentum (MBPM) is compared with well-known BP training algorithms such as gradient descent (GD), gradient descent with adaptive learning rate (GDA), gradient descent with momentum (GDM), gradient descent with adaptive learning rate and momentum (GDX), scaled conjugate gradient (SCG) and a quasi Newton based method (BFG). Detailed explorations of these algorithms can be found in neural

Table I. Simulation Results.

		BFG	MBPM	SCG	GDX	GDM	GDA	GD
XOR	**Epochs**	16	30	15	420	1960	173	1966
	MSE	2.48E-06	5.41E-06	4.09E-06	0.016136	0.032367	7.47E-06	0.03237
	Time	0.20287	0.31594	0.17574	2.0475	8.8521	0.85706	8.4043
Two Spirals	**Epochs**	506	3580	3324	10000	10000	10000	10000
	MSE	0.22656	0.0076954	0.0044271	0.048915	0.52471	0.07582	0.52518
	Time	8.2365	60.836	42.038	72.212	73.627	79.026	71.244
L-T	**Epochs**	25	57	41	500	8929	1136	8858
	MSE	7.49E-06	8.53E-06	8.30E-06	9.63E-06	9.15E-05	9.83E-06	9.17E-05
	Time	0.24168	0.52452	0.33071	2	40	4.5788	38

networks literature.Learning rate parameter is chosen as 0.01 for GD and the momentum parameter is chosen as 0.9 for GDM and GDX. Initial values of the weights are calculated by Nguyen-Widrow initialization method as in Matlab toolbox. Once the initial weights are determined then all algorithms use these initial weights as startup weights. Therefore all algorithms take the same initial weights at the startup of each learning process. For all tests reported, a tangent sigmoid transfer function was used for hidden nodes and a linear transfer function for the output nodes of a fully connected feed forward neural network. The learning processes terminate when the iterations are over a fixed number of epochs or the mean squared error (MSE) is less than a small threshold.

Epoch numbers, mean squared errors (MSE) and running times are considered in evaluating the convergence performance of the algorithms. All codes for the simulations were written with Matlab and performed on an IBM Thinkpad / Centrino 1.5 Ghz mobile.

XOR problem. The first test was performed with the exclusive-or (XOR) problem, which is the most popular benchmark for neural network training. The network architecture used for this problem consisted of two inputs, one hidden layer with three units and one output unit. The network mapped each of the four pairs of input patterns into the corresponding output target value.

Two spirals problem. For the second training task we selected the "two spirals separation problem" examined in [10] The difficulty of this problem has been demonstrated in many attempts to solve the problem with backpropagation and several elaborated modifications. The input pattern set consists of the pairs of coordinates describing the points of two intertwined spirals in the $x-y$-plane. The network is trained to discriminate points lying on these two separate spirals. The membership of an input point to one or the other spiral was indicated by the target values 0 and 1,
respectively. Since the BP algorithm is unable to locate more than suboptimal solutions of the two spirals problem for networks with one hidden layer [11], a network with two hidden layers was used. The network was built up of an input layer of two units, each one representing a coordinate, a first hidden layer with ten units, a second hidden layer with two units and an output layer with one unit.

L-T problem. The third experiment conducted a simple L-T letter recognition task. The network had nine input units, two hidden units and a single output unit, and was trained to recognize the letters L and T. Each input pattern was a 3×3 pixel binary image of a letter. The training set was formed by eight patterns, all four orientations for
each letter. The letters L and T were indicated by the target values 0.05 and 0.95, respectively, for
the output unit.

5. Conclusion

In this paper, we focused on improving the standard BPM algorithm for training feedforward neural

networks. Local quadratic approximation of the error function allows us to determine optimum parameters for learning. We propose a new speed-up algorithm called MBPM based on the concept of using efficient learning rate and momentum factor at every stage. MBPM does not contain any user-dependent parameters whose values are crucial for the success of MBPM. Experimental results (table I) with the MBPM algorithm show that this algorithm offers much higher speed of convergence than the variants of conventional BP algorithm (GD,GDA,GDM,GDX). Consequently, the improvement presented can be considered as a valuable and viable alternative to existing methods. On the other hand, when compared with second-order methods such as BFG and SCG, MBPM shows considerable performance.

References:

[1] Gill, P. E., W. Murray, and M. H. Wright, *Practical Optimization*, New York: Academic Press, 1981.

[2] Haykin S., *Neural Networks (2nd ed.)*. Upper Saddle River. NJ:Prentice-Hall, 1999.

[3] Bishop C. M., *Neural networks for pattern recognition*. Oxford Univ. Press, 1995.

[4] Rumelhart D. E., Hinton G. E., and Williams R. J., *Learning representations by back-propagating errors*, Nature, vol. 323, pp. 533-536, 1986.

[5] Plaut, D.; Nowlan, S. ve Hinton, G.E. *Experiments on learning by backpropagation*. Technical Report CMU-CS-86-126, Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1986.

[6] Qian N., *On the momentum term in gradient descent learning algorithms*, Neural Networks, vol. 12, pp. 145-151, 1999.

[7] Mammadov M., Tas E., *Stability and Convergence Speed of Gradient Descent With Momentum Training Algorithm*, IV. Statistics Congress, Antalya, May 2005.

[8] Brogan W. L., *Modern Control Theory*. Englewood Cliffs, NJ: Prentice-Hall, 1991.

[9] Torii M. and Hagan M. T., *Stability of steepest descent with momentum for quadratic functions*, IEEE Transactions on Neural Networks, vol. 13, pp.752-756, 2002.

[10] Lang, K. J., Witbrock, M. J., *Learning to tell two spirals apart.*, Proceedings of the 1988 Connectionist Models Summer School, (pp. 52-59). Pittsburgh, PA: Morgan Kaufmann, 1989.

[11] Baum, E. B., Lang, K. J., Constructing hidden units using examples and queries. In R. P. Lippmann & J. E. Moody & D. S. Touretzky (Eds.), (pp. 904-910). Advances in neural information processing systems, 3. San Mateo, CA: Morgan Kaufmann. 1991.

[12] Jacobs, R. A. *An adaptive least square algorithm for the efficient training of artificial neural networks*. IEEE Transactions on Circuits and Systems. 36, 1092-1101. 1988.

[13] Fahlman, S.E. *Faster-learning variations on back-propagation: an empirical study*. Proceedings of the 1988 Connectionist Models Summer School, (pp. 38-51). Pittsburgh, PA: Morgan Kaufmann. 1989.

[14] Silva, F. M., Almeida, L. B. In L. B. Almeida & C. J. Wellekens (Eds.), *Accelaration techniques for the backpropagation algorithm*, (pp. 110-119). Lecture Notes in Computer Science, 412. Berlin: Springer.

[15] LeCun, Y., Denker, J.S., Solla, S.A. *Optimal Brain Damage*. In D.S. Touretzky (Ed.), Advances in neural information processing systems 2 (NIPS*89) (pp. 598-605). Denver, CO: Morgan Kaufman.

[16] Hagiwara M. and Sato A., *Analysis of momentum term in backpropagation*, IEICE Trans. Inform. Syst., vol. E78-D, no. 8. Aug. 1995.

[17] Kamarthi S. V. and Pittner S., *Accelerating neural network training using weight extrapolations*, Neural Networks, vol. 12, pp. 1285-1299, 1999.

[18] Phansalkar V. V., Sastry P.S., *Analysis of the backpropagation algorithm with momentum*, IEEE Trans. Neural Networks, vol. 5, May 1994.

[19] Yu X. H. and Chen G. A., *Efficient backpropagation learning using optimal learning rate and momentum*, Neural Networks, vol. 10, pp. 517-527, 1997.