NONMONOTONE LEARNING RULES FOR BACKPROPAGATION NETWORKS

V.P. Plagianakos
University of Patras,
Department of Mathematics,
U.P. Artificial Intelligence
Research Center–UPAIRC,
GR-26500 Patras, Greece.

G.D. Magoulas
University of Athens,
Department of Informatics,
U.P. Artificial Intelligence
Research Center–UPAIRC,
GR-15771 Athens, Greece.

M.N. Vrahatis
University of Patras,
Department of Mathematics,
U.P. Artificial Intelligence
Research Center–UPAIRC,
GR-26500 Patras, Greece.

ABSTRACT

In this paper we study nonmonotone learning rules, based on an acceptability criterion for the calculated learning rate. More specifically, we impose that the error function value at each epoch must satisfy an Armijo-type criterion, with respect to the maximum error function value of a predetermined number of previous epochs. To test this approach, we propose two training algorithms with adaptive learning rates that employ the above mentioned acceptability criterion. Experimental results show that the proposed algorithms have considerably improved convergence speed, success rate, and generalization, when compared with other classical neural network training methods.

1. INTRODUCTION

The efficient supervised training of Feedforward Neural Networks (FNNs) is a subject of considerable ongoing research and numerous algorithms have been proposed to this end. The special case of batch training of an FNN is consistent with the theory of unconstrained optimization, since the information from all the training set is used.

Let us consider the family of gradient based training algorithms having the iterative form

$$w^{k+1} = w^k + \eta^k d^k, \qquad k = 0, 1, 2, \ldots \tag{1}$$

where w^k is the current weight vector, d^k is a search direction, and η^k is the learning rate. Various choices of the direction d^k give rise to distinct algorithms. A broad class of methods uses the search direction $d^k = -\nabla E(w^k)$, where the gradient $\nabla E(w)$ can be obtained by means of backpropagation of the error through the layers of the network. In practice, a small constant learning rate is chosen ($0 < \eta < 1$) in order to secure the convergence of the BP training algorithm and to avoid oscillations in a direction where the error function is steep. It is well known that this approach tends to be inefficient.

2. NONMONOTONE BACKPROPAGATION TRAINING

While no learning rate adaptation strategy will always be optimal, it does seem to be common sense to require that $E(w^{k+1}) < E(w^k)$. It must be noted that this simple condition does not guarantee global convergence for general functions.

The use of learning rate adaptation strategies which enforce monotonicity can considerably slow the rate of training, or even lead to divergence and to premature saturation [6, 15], in cases where inappropriate values for the critical heuristic learning parameters are used. To alleviate these problems the usage of monotone line search strategies has been suggested [7, 8]. A strategy of this kind consists in accepting a learning rate η^k along the direction d^k if it satisfies the *Wolfe conditions*:

$$E(w^k + \eta^k d^k) - E(w^k) \leq \sigma_1 \eta^k \langle \nabla E(w^k), d^k \rangle, \tag{2}$$

$$\langle \nabla E(w^k + \eta^k d^k), d^k \rangle \geq \sigma_2 \langle \nabla E(w^k), d^k \rangle, \tag{3}$$

where $0 < \sigma_1 < \sigma_2 < 1$ and $\langle \cdot, \cdot \rangle$ stands for the usual inner product in $\mathbb{R}^n$. The first inequality ensures that the error is sufficiently reduced at each iteration and the second prevents the learning rates from being too small.

Although Wolfe's approach provides an efficient and effective way to ensure that the error function is globally reduced sufficiently, it possesses the disadvantage that no information is stored and used that might accelerate convergence [3].

Thus, we use a different approach that exploits the accumulated information regarding the M most recent values of the error function to accelerate convergence [4]. The following condition is used to formulate the proposed approach and to define a criterion of acceptance of any weight iterate [4]:

$$E\left(w^k - \eta^k \nabla E(w^k)\right) - \max_{0 \leq j \leq M} E(w^{k-j}) \leq -\sigma\, \eta^k \left\| \nabla E(w^k) \right\|^2, \tag{4}$$

0-7803-5682-9/99/$10.00©1999 IEEE.

where M is a nonnegative integer and $0 < \sigma < 1$. The above condition allows an increase in the function values without affecting the global convergence properties as has been proved in [14].

Next, we propose an algorithm model that employs the above acceptability criterion. The kth iteration of the algorithm consists of the following steps:

1: Compute the new learning rate η^k using any learning rate adaptation strategy.

2: Update the weight vector $w^{k+1} = w^k - \eta^k \nabla E(w^k)$.

3: If the learning rate acceptability condition (4) is fulfilled store w^{k+1} and terminate; otherwise go to the next step.

4: Use a tuning technique for η^k and return to Step 3.

Remark 1.: A simple technique to tune η^k in Step 4 is to decrease the learning rate by a reduction factor $1/q$, where $q > 1$ [11]. This has the effect that each learning rate is decreased by the largest number in the sequence $\{q^{-m}\}_{m=1}^{\infty}$. We remark here that the selection of q is not critical for successful learning, however it has an influence on the number of error function evaluations required to obtain an acceptable weight vector. Thus, some training problems respond well to one or two reductions in the learning rates by modest amounts (such as 1/2) and others require many such reductions, but might respond well to a more aggressive learning rate reduction (for example by factors of 1/10, or even 1/20). On the other hand, reducing η^k too much can be costly since the total number of epochs will be increased. The value $q = 2$ is usually suggested in the literature [1] and indeed it was found to work without problems in the experiments reported in the paper.

Remark 2.: The above model constitutes an efficient method to determine an appropriate learning rate without additional gradient evaluations. As a consequence, the number of gradient evaluations is, in general, less than the number of error function evaluations.

3. LEARNING RATE ADAPTATION STRATEGIES

In this section we briefly describe two recently proposed learning rate adaptation strategies which can be successfully used for nonmonotone backpropagation training.

1) Learning rate adaptation using Lipschitz constant estimation: In [7] an approach that exploits the local shape of the error surface using a Lipschitz constant estimation has been proposed. The corresponding algorithm was named BPVS. The local estimation of the Lipschitz constant is defined for the kth iteration as:

$$\mathrm{A}^k = \frac{\left\|\nabla E(w^k) - \nabla E(w^{k-1})\right\|}{\|w^k - w^{k-1}\|}, \tag{5}$$

and the learning rate is $\eta^k = 0.5/\mathrm{A}^k$.

In order to eliminate the possibility of using an unsuitable local estimation of the Lipschitz constant we use the learning rate tuning technique of Remark 1. This version of the BPVS that provides nonmonotone training is named NMBPVS.

2) Learning rate adaptation using the Barzilai and Borwein formula: In a previous work [12] we have proposed a neural network training algorithm called BBP, which is based on the Barzilai and Borwein [2] learning rate update formula, where η^k for the kth epoch is given by:

$$\eta^k = \frac{\langle \delta^{k-1}, \delta^{k-1} \rangle}{\langle \delta^{k-1}, \psi^{k-1} \rangle}, \tag{6}$$

where $\delta^{k-1} = w^k - w^{k-1}$, $\psi^{k-1} = \nabla E(w^k) - \nabla E(w^{k-1})$ and $\langle \cdot, \cdot \rangle$ denotes the standard inner product. Our experiments in [12, 13], show that very often the method escapes from local minima and flat valleys where other methods are trapped. In order to secure the convergence of the method, even when the above formula gives unsuitable learning rates, we use the learning rate tuning technique of Remark 1. We call this modified training algorithm NMBBP.

4. EXPERIMENTS AND RESULTS

In this section, we evaluate the performance of our algorithms (BPVS, NMBPVS, BBP, and NMBBP) and compare them with the batch versions of BP, momentum BP (MBP) [5], and adaptive BP (ABP) [17], from the Matlab Neural Network Toolbox version 2.0.4. For the NMBPVS and NMBBP algorithms, we have fixed the values of $M = 10$ and $\sigma = 10^{-5}$. The algorithms have been tested using the same initial weights, initialized by the Nguyen–Widrow method [16], 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, *s.d.* the standard deviation, and *succ.* the simulations succeeded out of 1000 trials within the error function evaluations limit.

We must also note that for the BP, BPM and, ABP one gradient and one error function evaluation are necessary at each epoch, while the number of error function evaluations (FE) of BPVS, NMBPVS, BBP, and

NMBBP is, in general, larger than the number of gradient evaluations (GE), due to the learning rate acceptability criterion we use. As a consequence even when our algorithms fail to converge within the predetermined limit of *function evaluations*, their number of gradient evaluations is smaller than the corresponding number of the other methods. Keeping in mind that a gradient evaluation is more costly than an error function evaluation (see for example [9], where Moller suggests to count a gradient evaluation three times more than an error function evaluation), one can understand that our methods require fewer floating point operations and are actually much faster. From the above discussion it is clear why in the tables there are two lines for the BPVS, NMBPVS, BBP, and NMBBP algorithms; the first one indicates the statistics for the FE and the second for the GE.

4.1 3-Bit Parity

The task is to train a 3-2-1 FNN (eight weights, three biases) to produce the sum, mod 2, of 3 binary inputs – otherwise known as computing the "odd parity" function. The results are summarized in the following table:

Table 1: Results for the 3-Bit Parity problem.

Algorithm	min	mean	max	s.d.	succ.
BP	–	–	–	–	0.0%
MBP	246	485.4	973	195.4	48.0%
ABP	465	599.2	924	103.9	45.0%
BPVS	104 94	314.2 298.1	992 939	176.6 166.2	72.5%
NMBPVS	103 98	292.1 285.5	986 960	161.9 156.8	72.7%
BBP	70 38	330.9 145.2	994 497	200.8 83.8	57.6%
NMBBP	47 37	298.9 181.9	978 601	212.4 118.9	67.9%

Despite the effort we made to choose its learning rate, BP failed to converge within the error function evaluations limit in all the simulations. On the other hand, the MBP and ABP algorithms performed much better, with MBP slightly outperforming ABP. The BPVS, NMBPVS, BBP, and NMBBP algorithms exhibited the best performance, as they had the highest success rates and the least average number of epochs. In this problem the learning rate acceptability criterion does not improve the convergence properties of BPVS, but improves the BBP algorithm significantly.

4.2 Continuous function approximation

An 1-15-1 FNN (thirty weights, sixteen biases) is trained to approximate the continuous function $f(x) = \sin(x)\cos(2x)$, where the input values are scattered in the interval $[0, 2\pi]$. The network is trained until $E \leq 0.1$. The FNN is based on hidden neurons of logistic activations and on a linear output neuron. Comparative results are exhibited in the following table. Once again, our algorithms exhibited the best performance and had the best success rate. Moreover, there is a remarkable improvement of the BBP algorithm when the learning rate acceptability criterion is used; the BBP algorithm has a success rate of 79.6%, while the nonmonotone version NMBBP has a success rate of 92.2%.

Table 2: Results for function approximation.

Algorithm	min	mean	max	s.d.	succ.
BP	328	706.7	998	175.6	13.8%
MBP	332	699.2	993	174.8	13.7%
ABP	166	628.1	994	216.8	26.9%
BPVS	48 44	446.3 417.3	994 943	236.9 220.3	63.4%
NMBPVS	64 64	443.9 429.2	991 960	238.5 228.9	66.8%
BBP	39 27	362.1 186.4	995 502	233.5 111.3	79.6%
NMBBP	29 26	241.7 158.2	988 625	195.1 114.7	92.2%

4.3 The Font Learning Problem

For this problem, 26 matrices with the capital letters of the English alphabet are presented to a 35-30-26 FNN (1830 weights, 56 biases). Each letter has been defined in terms of binary values on a grid of size 5×7. The network is based on hidden neurons of logistic activations and on linear output neurons. In this problem the acceptability criterion we imposed accelerated the convergence of both BPVS and BBP methods, as shown in the following table:

Table 3: Results for the font problem.

Algorithm	min	mean	max	s.d.	succ.
BP	1098	1561.9	1999	202.8	76.8%
MBP	1142	1519.1	1931	169.3	4.9%
ABP	1119	1773.1	1999	168.9	37.2%
BPVS	347 323	688.9 639.3	1238 1156	136.4 128.6	100.0%
NMBPVS	337 322	669.4 633.3	1112 1041	134.9 131.1	100.0%
BBP	167 90	332.6 169.8	758 373	70.4 35.9	100.0%
NMBBP	113 73	182.0 119.4	309 193	33.5 20.5	100.0%

4.4 Generalization Performance

In order to evaluate the generalization of the nonmonotone algorithms NMBPVS and NMBBP, we have tested them on the MONK's problems [10]. These are binary classification tasks which are used as benchmarks for testing the generalization performance of learning algorithms. These problems rely on the artificial robot domain.

We have tested our methods against the well known BP, BP with weight decay (BPWD), and Cascade Correlation (CC) methods. In our simulation we have used the same network topologies as those found in [10] for the BP method. The next table clearly shows that the NMBPVS and NMBBP algorithms are excellent generalizers and manage to correctly classify all the input patterns in all the MONK's problems.

Table 4: Results for the MONK's problems.

Algorithm	MONK-1	MONK-2	MONK-3
BP	100%	100%	93.1%
BPWD	100%	100%	97.2%
CC	100%	100%	97.2%
NMBPVS	100%	100%	100%
NMBBP	100%	100%	100%

5. CONCLUDING REMARKS

In this paper we presented a new approach for generating nonmonotone learning rules based on an acceptability criterion. We have tested this approach on two recently published training algorithms [7, 12] and the results were satisfactory. The simulation results suggest that the use of the acceptability criterion, can significantly accelerate the convergence of the training algorithms. Moreover, the use of difficult to tune problem-dependent heuristic parameters is unnecessary. The training algorithms we have studied in this paper succeed to converge faster and more times than the other algorithms tested. Moreover, the behavior of the nonmonotone algorithms proved to be robust against phenomena such as oscillations due to large learning rates. Finally, the nonmonotone algorithms exhibited an excellent generalization capability, when tested on the MONK's problems.

References

[1] L. Armijo, Minimization of functions having Lipschitz-continuous first partial derivatives, Pacific J. Math., 16, 1–3, (1966).

[2] J. Barzilai and J.M. Borwein, Two point step size gradient methods, IMA J. Numer. Anal., 8, 141–148, (1988).

[3] A.V. Fiacco and G.P. McCormick, Nonlinear Programming: Sequential Unconstrained Minimization Techniques, SIAM, Philadelphia, (1990).

[4] L. Grippo, F. Lampariello and S. Lucidi, A nonmonotone line search technique for Newton's method, SIAM J. Numer. Anal., 23, 707–716, (1986).

[5] R.A. Jacobs, Increased rates of convergence through learning rate adaptation, Neural Networks, 1, 295–307, (1988).

[6] Y. Lee, S.H. Oh and M.W. Kim, An analysis of premature saturation in backpropagation learning, Neural Networks, 6, 719–728, (1993).

[7] G.D. Magoulas, M.N. Vrahatis, and G.S. Androulakis, Effective back–propagation with variable stepsize. Neural Networks, 10, 69–82, (1997).

[8] G.D. Magoulas, M.N. Vrahatis, and G.S. Androulakis, Improving the convergence of the backpropagation algorithm using learning rate adaptation methods, Neural Computation, accepted for publication.

[9] M.F. Møller, A scaled conjugate gradient algorithm, for fast supervised learning, Neural Networks, 6, 525–533, (1993).

[10] S.B. Thrun, J. Bala, E. Bloedorn, I. Bratko, B. Cestnik, J. Cheng, K. De Jong, S. Dzeroski, S.E. Fahlman, D. Fisher, R. Hamann, K. Kaufmann, S. Keller, I. Kononenko, J. Kreuziger, R.S. Michalski, T. Mitchell, P. Pachowicz, Y. Reich, H. Vafaie, W. Van de Welde, W. Wenzel, J. Wnek and J. Zhang, The MONK's Problems: A performance comparison of different learning algorithms, Technical Report, Carnegie Mellon University, CMU-CS-91-197, (1991).

[11] J.M. Ortega, and W.C. Rheinboldt Iterative Solution of Nonlinear Equations in Several Variables, Academic Press, NY, 1970.

[12] V.P. Plagianakos, D.G. Sotiropoulos and M.N. Vrahatis, Automatic adaptation of learning rate for Backpropagation Neural Networks, Recent Advances in circuits and systems, Nikos E. Mastorakis, ed., World Scientific, (1998).

[13] V.P. Plagianakos, D.G. Sotiropoulos and M.N. Vrahatis, A Nonmonotone Backpropagation Training Method for Neural Networks, Proceedings of the 4th Hellenic–European Conference on Computer Mathematics and its Applications, Athens, (1998).

[14] M. Raydan, The Barzilai and Borwein gradient method for the large scale unconstrained minimization problem, SIAM J. Optim., 7, 26–33, (1997).

[15] A.K. Rigler, J.M. Irvine and T.P. Vogl, Rescaling of variables in backpropagation learning, Neural Networks, 4, 225–229, (1991).

[16] D. Nguyen and B. Widrow, Improving the learning speed of 2-layer neural network by choosing initial values of the adaptive weights, IEEE First International Joint Conference on Neural Networks, 3, 21–26, (1990).

[17] T.P. Vogl, J.K. Mangis, J.K. Rigler, W.T. Zink and D.L. Alkon, Accelerating the convergence of the back-propagation method, Biological Cybernetics, 59, 257–263, (1988).