A Nonmonotone Backpropagation Training Method for Neural Networks

V.P. Plagianakos, D.G. Sotiropoulos and M.N. Vrahatis
University of Patras, Department of Mathematics,
Division of Computational Mathematics & Informatics,
GR-265 00, Patras, Greece.
e-mail: vpp|dgs|vrahatis@math.upatras.gr.

Abstract

A method that improves the speed and the success rate of the backpropagation algorithm is proposed. This method adapts the learning rate using the Barzilai and Borwein [IMA J.Numer. Anal., 8, 141–148, 1988] steplength update for gradient descent methods. The learning rate is automatically adapted at each epoch, using the weight and gradient values of the previous one. Additionally, an acceptability criterion for the learning rate has been enforced. More specifically, we impose that the error function value of each new epoch must satisfy an Armijo-type criterion, with respect to the maximum error function value of a predetermined number of previous epochs. Experimental results show that the proposed method has considerably improved convergence speed, and for the chosen test problems, outperforms other well-known training methods.

1 Introduction

Artificial Neural Networks (ANN's) have been used widely in many application areas and have shown their strengths in solving hard problems in artificial intelligence. Many different models of neural networks have been proposed, but multilayered feedforward ANN's are the most common. In order to train an ANN supervised training is probably the most frequently used technique. The training process is an incremental adaptation of connection weights that propagate information between simple processing units called neurons. Neurons are arranged in layers and there are connections between neurons in one layer to the following. Let us consider a feedforward ANN whose l-th layer contains N_l neurons, for $l = 1, \ldots, M$. The network is based on the following equations:

$$net_j^l = \sum_{i=1}^{N_{l-1}} w_{ij}^{l-1,l} y_i^{l-1}, \qquad y_j^l = f(net_j^l),$$

where net_j^l is for the j-th neuron in the l-th layer ($j = 1, \ldots, N_l$), the sum of its weighted inputs. The weights from the i-th neuron at the $(l-1)$ layer to the j-th neuron at the l-th layer are denoted by $w_{ij}^{l-1,l}$, y_j^l is the output of the j-th neuron that belongs to the l-th layer, and $f(net_j^l)$ is the j-th's neuron activation function.

The training process can be realized by minimizing the error function E defined by :

$$E = \frac{1}{2} \sum_{p=1}^{P} \sum_{j=1}^{N_M} \left(y_{j,p}^M - t_{j,p} \right)^2 = \sum_{p=1}^{P} E_p, \tag{1}$$

where $\left(y_{j,p}^M - t_{j,p} \right)^2$ is the squared difference between the actual output value at the j-th output layer neuron for pattern p and the target output value; p is an index over input-output pairs. This function also provides an error surface over the weight space.

To simplify the formulation of the equations let as use a unified notation for the weights. Thus, for a feedforward ANN with a total of n weights, w denotes a column weight vector with components $w_1, w_2, \ldots, w_n$ and it is defined in the n–dimensional real space $\mathbb{R}^n$; E represents the batch error measure defined as the sum-of-squared-differences error function over the entire training set; $\nabla E(w)$ defines the gradient vector of the error function E at w.

In this work we propose and study the behaviour of a NonMonotone Back Propagation (NMBP) method that uses the Barzilai and Borwein steplength update [2] and is combined with a generalization of the Armijo's rule as a steplength acceptance criterion.

The remaining of the paper is organized as follows: In Section 2 we propose a training algorithm with adaptive learning rate and describe the acceptability criterion. Experiments and simulation results are presented in Section 3. The final section contains concluding remarks and a short discussion for further work.

2 The NonMonotone BackPropagation Training Method

2.1 Description of the NMBP Training Method

Our aim is to minimize the error function E

$$\min E(w), \qquad w \in \mathbb{R}^n, \tag{2}$$

where $E \in C^2$. The gradient $\nabla E(w)$ can be obtained by means of back-propagation of errors through the layers. Let the family of gradient training algorithms having the iterative form

$$w^{k+1} = w^k + \eta_k d_k, \qquad k = 0, 1, 2, \ldots \tag{3}$$

where w^k be the current point, d_k be a search direction, and η_k be the steplength. Various choices of the direction d_k give rise to distinct algorithms. A broad class of methods uses $d_k = -\nabla E(w^k)$ as a search direction. The widely used gradient-based training algorithm, named batch back–propagation (BP), minimizes the error function using the following steepest descent method with constant, heuristically chosen, learning rate η:

$$w^{k+1} = w^k - \eta \nabla E(w^k). \tag{4}$$

Clearly, the behaviour of any algorithm depends on the choice of the steplength no less than the choice of the search direction. It is well known that pure gradient descent methods with fixed learning rate tend to be inefficient (see [10]). This happens, for example, when the search space contains long ravines that are characterized by sharp curvature across them and a gently sloping floor [11].

In [2] a gradient method has been proposed where the search direction is always the negative gradient direction, but the choice of the steplength (learning rate) is not the classical choice of the steepest descent method. The motivation for this choice is that it provides two–point approximation to the secant equation underlying quasi-Newton methods [10]. This yields the iteration:

$$w^{k+1} = w^k - \eta_k \nabla E(w^k), \tag{5}$$

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. The key features of this method are the low storage requirements and the inexpensive computations. Moreover, it does *not* guarantee descent in the error function E.

Our experiments in a previous work [6], show that this property is valuable in neural network training, because very often the method escapes from local minima and flat valleys where other methods are trapped. The only implementation problem of the above method, when applied to train an FNN, is that the update formula (6) potentially gives large values for the learning rate η_k. In this case, the method may overshoot the minimum $w^\star$ or possibly diverge. To tackle the above problem we had introduced a parameter μ, called maximum growth factor, which controled the growth of the learning rate. The side effect of parameter μ is that sometimes decreases the speed of convergence, because it is constant during the training process and does not take into account the local shape of the error function and the current direction.

While no learning rate acceptance rule 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 that the sequence of weight vectors $\{w^k\}_{k=0}^{\infty}$ will converge to a minimizer w^* of E. Additionally, the use of a learning rate selection rule which enforces monotonicity can considerably slow the rate of convergence. A well-known monotone line search method, given by Armijo in [1], uses as an acceptability criterion for the learning rate, the following condition:

$$E\left(w^k - \eta_k \nabla E(w^k)\right) - E(w^k) \leq -\sigma\ \eta_k \left\|\nabla E(w^k)\right\|^2 \tag{7}$$

where $0 < \sigma < 1$, ensures that using this η_k, the error function is reduced during the training process. It comes as no great surprise that a direct combination of the (5) gradient method using (6) as learning rate, with the Armijo's condition (7) has some disadvantages. In particular, monotone line search forcing decrease at every epoch will destroy some of the local properties of the method and may reduce it to the back-propagation, which is known for its slow convergence.

A nonmonotone line search for Newton-type methods has been proposed in [4] and some computational advantages of this technique for large scale unconstrained minimization problems have been pointed out in [7]. They suggest that the learning rate η_k can be computed along the search direction d_k by an Armijo-type nonmonotone line search, which does not enforce $E(w^{k+1}) < E(w^k)$, but uses a learning rate acceptability criterion which can be viewed as a generalization of the Armijo's rule [1]. More specifically imposes that the error function value E of each new epoch must satisfy the Armijo's condition with respect to the maximum error function value of a prefixed number M of previous epochs. Formally, the condition is given by:

$$E\left(w^k - \eta_k \nabla E(w^k)\right) - \max_{0 \leq j \leq M} E(w^{k-j}) \leq -\sigma\ \eta_k \left\|\nabla E(w^k)\right\|^2 \tag{8}$$

where M is a nonnegative integer. The above condition allows an increase in the function values without affecting the global convergence properties as has been proved in [7]. This condition is preferred in order to accept the learning rate given by (6) as frequently as possible.

In the next subsection, the NMBP training algorithm is summarized.

2.2 The NMBP training algorithm

A high-level description of the NMBP method is outlined below:
Step 0: Initialize by setting the number of epochs $k = 0$, the weights w^0, the learning rate to an arbitrary value η_0, the error goal ε, the $\sigma \in (0, 1)$ and $M \geq 1$.
Step 1: Compute the weight vector w^1 according to relation (5) using η_0 as the learning rate and set $k = k + 1$.
Step 2: Compute the new learning rate η_k using the relation (6).
Step 3: Update the weight vector w^{k+1} according to the relation (5).

Step 4: If the learning rate acceptability condition(8) is fullfilled go to Step 6.
Step 5: Set $\eta_k = \eta_k/2$ and go to Step 3.
Step 6: Check if $E(w^{k+1}) > \varepsilon$, set $k = k + 1$ and go to Step 2; otherwise get the final weight vector w^*, and the corresponding value of E.
Remark: Steps 4–5 constitute an efficient method to determine an appropriate learning rate without additional gradient evaluations. As a consequence, the number of gradient evaluations, in general, is less than the number of error function evaluations.

3 Experiments and Results

A computer simulation has been developed to study the performance of the learning algorithms. The simulations have been carried out on a Pentium 133MHz PC IBM compatible using MATLAB version 5.01. The performance of the NMBP algorithm has been evaluated and compared with the batch versions of BP, momentum BP (MBP) [5], and adaptive BP (ABP) [12], from Matlab Neural Network Toolbox version 2.0.4. Toolbox default values for the heuristic parameters, of the above algorithms, are used unless stated otherwise. For the NMBP algorithm, we have fixed the values of $M = 10$ and $\sigma = 10^{-5}$. Especially for the XOR problem we have studied the performance of the NMBP method for various values of M and we give a table summarizing these simulations. The algorithms were tested using the same initial weights, initialized by the Nguyen–Widrow method [9], 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. 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 statistical analysis of the algorithms. This fact clearly favours BP, MBP and ABP that require too many epochs to complete the task and/or often fail to find the minimum $w^\star$.

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 NMBP is, in general, larger than the number of gradient evaluations (GE), due to the acceptability criterion. As a consequence even when NMBP fails to converge within the predetermined limit of *function evalutions*, its 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 [8], where suggests to count a gradient evaluation three times more than an error function evaluation), one can understand that the NMBP method requires fewer floating point operations and actually is much faster, when compared with the other methods. From the above discussion it is clear why in the tables there are two lines for the NMBP method; the first one indicates the statistics for the FE and the second for the GE.

3.1 The Exclusive-OR Problem

The first test problem we will consider is the exclusive-OR (XOR) Boolean function problem, which has historically been considered a good test of a network model and learning algorithm. The XOR function maps two binary inputs to a single binary output. This simple Boolean function is not linearly separable (i.e. it cannot be solved by a simple mapping directly from

the inputs to the output), and thus requires the use of extra hidden units to learn the task. Moreover, it is sensitive to initial weights as well as to learning rate variations and presents a multitude of local minima with certain weight vectors. A 2-2-1 FNN (six weights, three biases) has been used to train the XOR problem. The network is based on hidden neurons of logistic activations with biases and on a linear output neuron with bias. For the BP and MBP algorithms the learning rate is chosen to be 0.1 instead of the default value 0.01 to accelerate their convergence, since – for this problem – they converge slowly with the default value. The termination criterion is $E \leq 0.01$ within 1000 error function evaluations. The results of the simulation are shown in Table 1.

Table 1: Results of simulations for the XOR problem

Algorithm	min	mean	max	s.d.	succ.
BP	32	185.43	947	175.16	61.4%
MBP	26	197.01	973	189.08	60.5%
ABP	20	119.94	993	189.14	67.6%
NMBP	7 7	107.01 96.32	999 571	237.25 122.58	73.7%

In Table 2 we have summarized the performance of the NMBP method for the XOR problem and various choises of M. The statistics shown are for the error function evaluations and it is clear that the original monotone Armijo's condition (for $M = 1$), has excibited the worst performance, while for values of $M > 3$ the performance of the method have been improved significantly.

Table 2: Results of simulations for various values of M (XOR problem)

M	min	mean	max	s.d.	succ.
1	7	55.45	474	86.47	63.7%
2	7	55.24	495	84.59	64.2%
3	7	58.08	510	88.43	63.5%
4	7	73.35	540	99.39	74.8%
5	7	74.81	535	94.78	72.5%
6	7	85.34	562	111.46	74.8%
7	8	87.74	560	111.38	73.2%
8	6	83.94	565	111.82	73.2%
9	8	91.71	583	120.79	72.4%
10	7	96.32	571	128.36	73.7%

3.2 The Font Learning Problem

In this simulation we consider the font learning problem. We present to the network 26 matrices with the capital letters of the English alphabet. Each letter has been defined in terms of binary values on a grid of size 5×7. For this problem a 35-30-26 FNN (1830 weights, 56 biases) has been used. The network is based on hidden neurons of logistic activations with biases and on a linear output neuron with bias. The performance of the methods has been measured and the Table 3 shows the results. In order to improove the performance of the methods with fixed learning rate (i.e. BP and MBP), the weights have been initialized following the Nguyen–Widrow method but afterwards the weights and biases of the output layer have been multiplied by 0.01. NMBP method has also performed very well without this scaling and exhibited a success rate of 100.0%, when the other methods always failed to converge. Additionally, the error function evaluations

Table 3: Results of simulations for the font problem

Algorithm	min	mean	max	s.d.	succ.
BP	1098	1561.84	1999	202.81	76.8%
MBP	1142	1519.06	1931	169.32	4.9%
ABP	1119	1773.06	1999	168.88	37.2%
NMBP	113 73	182.01 119.39	309 193	33.47 20.47	100.0%

limit has been increased to 2000, since (as indicated by the minimum number of epochs needed to train the FNN) all the methods – except the proposed one, which had maximum number of error function evaluations 309 – failed to complete the task within the standard limit we have used in the previous test problem (XOR).

3.3 Continuous function approximation

The third test problem we consider is the approximation of the continuous trigonometric function, $f(x) = \sin(x)\cos(2x)$. In this problem we consider the performance of the methods to form a look-up table for the function mentioned above. In this case a 1-15-1 FNN (thirty

Table 4: Results of simulations for function approximation

Algorithm	min	mean	max	s.d.	succ.
BP	328	706.65	998	175.56	13.8%
MBP	332	699.18	993	174.76	13.7%
ABP	166	628.03	994	216.77	26.9%
NMBP	29 26	241.65 158.24	988 625	195.11 114.71	92.2%

weights, sixteen biases) is trained to approximate the function $f(x)$, where the input values are scattered in the interval $[0, 2\pi]$ and the network is trained until the sum of the squares of the errors (over 20 input/output sets) 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 exhibited in Table 4. Once again, the NMBP method exhibited the best performance and had a excellent success rate 92.2%.

4 Concluding Remarks

A simple technique for the adaptation of the learning rate for batch training FNNs is proposed. It is shown that this adaptation improves the convergence speed in several classical test problems. Our experimental results clearly show that the proposed method outperforms the classical training algorithms (BP, MBP and ABP). It runs much faster, performs less floating point operations, has improved average number of epochs, and better convergence rates. Further work must be done in order to evaluate the generalization performance of NMBP on well-known classification benchbarks, and test it on bigger and more complex real-life learning tasks.

Acknowledgements

We would like to thank Prof. J. Barzilai for sending us a reprint of his work.

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] S. Fahlman, An empirical study of learning speed in back-propagation networks, Technical Report CMU-CS-88-162, Carnegie Mellon University, Pittsburgh, PA 15213, September 1988.

[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] V.P. Plagianakos, D.G. Sotiropoulos and M.N. Vrahatis, An Improved Backpropagation Method with Adaptive Learning Rate, to be appear in: *Procceeding of the 2nd Intern. Confer. on: Circuits, Systems and Computers*, (1998).

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

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

[9] 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).

[10] E. Polak, Optimization: Algorithms and Consistent Approximations, Springer-Verlag, 1997.

[11] D.E. Rumelhart and J.L. McClelland(eds), Parallel distributed processing: explorations in the microstructure of cognition, Vol. 1: Foundations, MIT Press, 1986.

[12] 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).