OPTIMIZATION STRATEGIES AND BACKPROPAGATION NEURAL NETWORKS

V.P. Plagianakos
University of Patras, Department of Mathematics,
University of Patras Artificial Intelligence Research Center–UPAIRC,
GR-265 00, Patras, Greece.

G.D. Magoulas
University of Athens, Department of Informatics,
University of Patras Artificial Intelligence Research Center–UPAIRC,
GR-157 71, Athens, Greece.

M.N. Vrahatis
University of Patras, Department of Mathematics,
University of Patras Artificial Intelligence Research Center–UPAIRC,
GR-265 00, Patras, Greece.

1 Summary

Adaptive learning rate algorithms try to decrease the error at each iteration by searching a local minimum with small weight steps, which are usually constrained by highly problem-dependent heuristic learning parameters. Based on the idea of the decrease of the error function at each iteration we suggest monotone learning strategies that guarantee convergence to a minimizer of the error function without using highly problem-dependent heuristics. Furthermore, we introduce the idea of nonmonotone learning that provides fast, stable and reliable training and we test both approaches on simulation experiments.

2 Introduction

The goal of neural network training is to iteratively update the network weights to minimize the learning error. The special case of batch training of a Feedforward Neural Network (FNN) is consistent with the theory of unconstrained optimization and can be viewed as the minimization of the batch error measure E, defined as the sum-of-squared-differences error function over the entire training set; that is to find a minimizer $w^* = (w_1^*, w_2^*, \ldots, w_n^*) \in \mathbb{R}^n$, such that:

$$w^* = \min_{w \in \mathbf{R}^n} E(w). \tag{1}$$

The rapid computation of such a minimizer is a rather difficult task since, in general, the number of network weights is large and the corresponding nonconvex error function possesses multitudes of local minima and has broad flat regions adjoined with narrow steep ones.

The widely used batch Back–Propagation (BP) [24] is a first-order training algorithm, which minimizes the error function using the steepest descent method [8]:

$$w^{k+1} = w^k - \eta \nabla E(w^k), . \tag{2}$$

The gradient vector $\nabla E(w)$, where its ith component $\partial_i E(w)$ denotes the partial derivative of $E(w)$ with respect to the ith variable w_i, is usually computed by the back–propagation of the error through the layers of the FNN (see [24]). The constant η is heuristically chosen and defines the learning rate. Appropriate learning rates help to avoid convergence to a saddle or maximum point. In practice, a small constant learning rate is chosen ($0 < \eta < 1$) in order to secure the convergence of the BP algorithm and to avoid oscillations in the directions where the error surface is steep. However, this approach considerably slows down the training process since, in general, a small learning rate may not be appropriate for all the portions of the error surface.

3 Monotone Learning Strategies

It does seem to be common sense to require that the error function is monotonically reduced at each iteration, i.e. $E(w^{k+1}) < E(w^k)$. This can be achieved by employing adaptive learning rate strategies: (i) start with a small learning rate and increase it exponentially, if successive epochs reduce the error, or rapidly decrease it, if a significant error increase occurs [3, 26], (ii) start with a small learning rate and increase it, if successive epochs keep gradient direction fairly constant, or rapidly decrease it, if the direction of the gradient varies greatly at each epoch [5] and (iii) for each weight an individual learning rate is given, which increases if the successive changes in the weights are in the same direction and decreases otherwise [11, 17, 22, 25].

Note that all the above mentioned strategies try to decrease the error by searching a local minimum with small weight steps. These steps are usually constrained by problem-dependent heuristic learning parameters. The use of heuristics enforces the monotone decrease of the learning error and secures the converge of the training algorithm to a minimizer of E. However, enforcing monotonic error reduction using inappropriate values for the critical heuristic learning parameters, can considerably slow down the rate of training, or even lead to divergence and to premature saturation [12, 23]; there is a trade-off between convergence speed and stability of the training algorithm.

A different monotone learning strategy that does not applies heuristics consists in accepting a positive learning rate η^k along the search direction φ^k if it satisfies the *Wolfe conditions*:

$$E(w^k + \eta^k \varphi^k) - E(w^k) \leq \sigma_1 \eta^k \langle \nabla E(w^k), \varphi^k \rangle, \tag{3}$$

$$\langle \nabla E(w^k + \eta^k \varphi^k), \varphi^k \rangle \geq \sigma_2 \langle \nabla E(w^k), \varphi^k \rangle, \tag{4}$$

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 reduced sufficiently and the second prevents the learning rate from being too small. It can be shown that if φ^k is a descent direction and E is continuously differentiable and bounded below along the ray $\{w^k + \eta \varphi^k \mid \eta > 0\}$, then

there always exist learning rate satisfying (3)–(4) [15, 6]. Relation (4) can be replaced by

$$E(w^k + \eta^k \varphi^k) - E(w^k) \geq \sigma_2 \eta^k \langle \nabla E(w^k), \varphi^k \rangle, \tag{5}$$

where $\sigma_2 \in (\sigma_1, 1)$ (see [6]).

An alternative strategy has been proposed in [20]. It is applicable to any descent direction φ^k and uses two parameters $\alpha, \beta \in (0, 1)$. Following this approach the learning rate is $\eta^k = \beta^{m_k}$, where $m_k \in \mathbb{Z}$ is any integer such that

$$E(w^k + \beta^{m_k} \varphi^k) - E(w^k) \leq \beta^{m_k} \alpha \langle \nabla E(w^k), \varphi^k \rangle \tag{6}$$

$$E(w^k + \beta^{m_k - 1} + \varphi^k) - E(w^k) > \beta^{m_k - 1} \alpha \langle \nabla E(w^k), \varphi^k \rangle. \tag{7}$$

All the above strategies must be combined with tuning subprocedures generating learning rates that satisfy conditions (3)–(4) or (6)–(7) in order to guarantee global convergence.

The strategy based on Wolfe's conditions provides an efficient and effective way to ensure that the error function is globally reduced sufficiently. In practice, the condition (4) or (5) generally is not needed because the use of a backtracking strategy avoids very small learning rates. A simple backtracking strategy to tune the length of the minimization step, so that it satisfies conditions (3)–(4) at each epoch, is to decrease the learning rate by a reduction factor $1/q$, where $q > 1$ [16]. This has the effect that the learning rate is decreased by the largest number in the sequence $\{q^{-m}\}_{m=1}^{\infty}$, so that the condition (3) is satisfied. 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 satisfy the condition (3). Thus, when seeking to satisfy (3) it is important to ensure that the learning rate is not reduced unnecessarily so that the condition (4) is not satisfied. Since, in training the gradient vector is known only at the beginning of the iterative search for a new weight vector, the condition (4) cannot be checked directly (this task requires additional gradient evaluations at each epoch), but is enforced simply by placing a lower bound on the acceptable values of the learning rate. This bound on the learning rate has the same theoretical effect as the condition (4) and ensures global convergence [6]. The value $q = 2$ is usually suggested in the literature [1] and indeed it was found to work without problems in the experiments (see [14]).

4 Nonmonotone Learning Strategies

Although monotone learning strategies provide an efficient and effective way to ensure that the error function is reduced sufficiently, they possess the disadvantage that no information is stored and used that might accelerate convergence [7]. To alleviate this situation we propose a nonmonotone learning strategy that exploits the accumulated information regarding the M most recent values of the error function, to accelerate the convergence. The following condition is used to formulate the new approach and to define a criterion of acceptance of any weight iterate [9]:

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

where M is a nonnegative integer, $0 < \sigma < 1$, and η^k indicate the learning rate in the kth epoch. The above condition allows an increase in the function values without affecting the global convergence properties as has been proved theoretically in [9, 21] and experimentally in [18, 19]. Experiments indicate that the choice of the parameter M is critical for the implementation and depends on the problem. The following procedure provides an elegant way to dynamically adapt the value of M at each epoch:

$$M^k = \begin{cases} M^{k-1} + 1, & \Lambda^k < \Lambda^{k-1} < \Lambda^{k-2}, \\ M^{k-1} - 1, & \Lambda^k > \Lambda^{k-1} > \Lambda^{k-2}, \\ M^{k-1} \quad , & \text{otherwise}, \end{cases} \tag{9}$$

where Λ^k is the local estimation of the Lipschitz constant [13] at the kth iteration, defined as:

$$\Lambda^k = \frac{\left\| \nabla E(w^k) - \nabla E(w^{k-1}) \right\|}{\|w^k - w^{k-1}\|}. \tag{10}$$

If Λ^k is increased for two consecutive epochs, the sequence of the weight vectors approaches a steep region, so we decrease the value of M, to avoid overshooting a possible minimum point. Reversely, when Λ^k is decreased for two consecutive epochs, the method possibly enters a valley in the weight space, so we increase the value of M. This allows the method to accept larger learning rates and move faster out of the flat region. Finally, when the value of Λ^k has a rather random behavior (increasing and decreasing for consecutive epochs), we leave the value of M unchanged.

Note that the local approximation of the Lipschitz constant is easily obtained, without any additional error function and gradient evaluations. Obviously, M has to be positive. Thus, if Relation (9) gives a non positive number, the value of M is set to $M = 1$. Moreover, in practice, we have to enforce an upper limit to the values of M, as well. Experimental results indicate that $M = 10$, is a good choice for the maximum value of M.

Next, we propose a high-level description of an algorithm 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 weights $w^{k+1} = w^k - \eta^k \nabla E(w^k)$.
3: If the acceptability condition (8) 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 at Step 4 is to decrease the learning rate by a reduction factor $1/q$, as suggested in the previous section.

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 (GE) is, in general, less than the number of error function evaluations (FE).

5 Simulations

Both the monotone and the nonmonotone learning strategies can be incorporated in any training algorithm. In this section, we present experimental results and compare the performance of three training algorithms with and without the use of the monotone and the nonmonotone learning strategies.

5.1 The BPM algorithm

In [11, 24] a simple, heuristic, strategy for accelerating the BP algorithm has been proposed. They have incorporated a momentum term in the steepest descent method as follows:

$$w^{k+1} = w^k - (1-m)\eta\nabla E(w^k) + m(w^k - w^{k-1}),$$

where m is the momentum constant. One drawback with the above scheme is that if m is set to a comparatively large value gradient information from previous epochs is more influential than the current gradient information in updating the weights. One solution is to increase the learning rate, however in practice this approach frequently proves ineffective and leads to instability or saturation. Thus, if m is increased it may be necessary to make a compensatory reduction in η to maintain network stability. In the experiment reported we alleviate this problem by combining BPM with the proposed learning strategies. We set η and m to a set of high value, i.e. $\eta = 1.2$ and $m = 0.9$ and check the performance of the BPM with and without the learning strategies. The modified BPM method is called NMBPM.

To test the use of the proposed learning strategies on the BPM, a 64-6-10 FNN (444 weights, 16 biases) is trained to recognize 8×8 pixel machine printed numerals from 0 to 9 in helvetica italic [13]. The network is based on neurons of the logistic activation model. The termination condition for all algorithms tested is an error value $E \leq 10^{-1}$.

Detailed results regarding the training performance of the algorithms are presented in Table 1, where μ denotes the mean number of gradient or error function evaluations required to obtain convergence, σ the corresponding standard deviation, Min/Max the minimum and maximum number of gradient or error function evaluations, and % denotes the percentage of simulations that converge to a global minimum. Obviously, the use

Table 1: Comparative results for the numeric font learning problem

Algorithm	Gradient Evaluation			Function Evaluation			Success
	μ	σ	Min/Max	μ	σ	Min/Max	%
BPM	560.2	684.9	239/3962	560.2	684.9	239/3962	39
NMBPM	565.9	429.1	289/2823	571.6	428.9	295/2827	97

of nonmonotone strategies has significantly improve the convergence rate of the BPM method.

5.2 The BPVS algorithm

In [13] learning rate adaptation using a Lipschitz constant estimation has been proposed. The corresponding algorithm was named BPVS and applies a monotone learning strategy. The learning rate update formula is:

$$\eta^k = \frac{\left\| w^k - w^{k-1} \right\|}{2 \left\| \nabla E(w^k) - \nabla E(w^{k-1}) \right\|}. \tag{11}$$

To show the effect of the proposed learning strategies on the performance of the BPVS we apply them on a 16-8-12 FNN (224 weights, 20 biases) trained to classify 120 texture patterns to 12 texture types. The network is based on neurons of logistic activations with biases, and the weights and biases were initialized with random numbers from the interval $(-1, 1)$. The texture patterns have been obtained by applying the co-occurrence method [10] to 12 digitized images of size 512×512 taken from the Brodatz's album [4]. The termination condition is a classification error $CE < 3\%$ [13]; that is the network classifies correctly 117 out of the 120 patterns.

Table 2: Comparative results for the texture classification problem

Algorithm	Gradient Evaluation			Function Evaluation			Success
	μ	σ	Min/Max	μ	σ	Min/Max	%
BPVS	544.8	274.1	294/2227	519.5	257.7	283/2101	100
NMBPVS	471.7	116.6	273/888	463.7	113.7	268/870	100

5.3 The BBP algorithm

In [19], we have proposed a training algorithm called BBP, based on the Barzilai and Borwein steplength update [2]. BBP applies a monotone learning strategy and uses the following formula to evaluate the learning rate at the kth epoch:

$$\eta^k = \frac{\langle \delta^{k-1}, \delta^{k-1} \rangle}{\langle \delta^{k-1}, \psi^{k-1} \rangle}, \tag{12}$$

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.

Next, we test the proposed learning strategies on a function approximation problem. We call the modified method NMBBP. The continuous function $f(x) = \sin(x)\cos(2x)$ is approximated by a 1-15-1 FNN (thirty weights, sixteen biases). 20 input/output pairs are taken, scattered in the interval $[0, 2\pi]$ and the termination condition is $E \leq 0.1$. The FNN is based on hidden neurons with logistic activations and on a linear output neuron. Comparative results are exhibited in Table 3. There is a remarkable improvement of the BBP algorithm when the nonmonotone learning strategy 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 3: Results of simulations for function approximation

Algorithm	Gradient Evaluation			Function Evaluation			Success
	μ	σ	Min/Max	μ	σ	Min/Max	%
BBP	186.4	111.3	27/502	186.4	111.3	27/502	79.6
NMBBP	158.2	114.7	26/625	241.7	195.1	29/988	92.2

6 Conclusion

In this paper we have presented new learning strategies for generating monotone and nonmonotone learning rules for neural network training. The simulation results were satisfactory and suggest that the use of the proposed strategies, can significantly accelerate the convergence of the training algorithms. Additionally, the use of difficult to tune problem-dependent heuristic parameters is unnecessary; the modified methods proved to be robust against phenomena such as oscillations due to poorly chosen heuristics parameters.

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] R. Battiti, Accelerated backpropagation learning: two optimization methods, Complex Systems, 3, 331–342, (1989).

[4] P. Brodatz, Textures - a Photographic Album for Artists and Designers, Dover, New York, (1966).

[5] L.W. Chan and F. Fallside, An adaptive training algorithm for back–propagation networks, Computers Speech and Language, 2, 205–218, (1987).

[6] J.E. Dennis and R.B. Schnabel, Numerical Methods for Unconstrained Optimization and Nonlinear Equations, Prentice-Hall, Englewood Cliffs, NJ, (1983).

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

[8] P.E. Gill, W. Murray, and M.H. Wright, Practical Optimization, Academic Press, NY, (1981).

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

[10] R. Haralick, K. Shanmugan, and I. Dinstein, Textural features for image classification, IEEE Trans. System, Man and Cybernetics, 3, 610–621, (1973).

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

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

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

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

[15] Nocedal J., Theory of algorithms for unconstrained optimization, Acta Numerica, 199–242, (1992).

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

[17] M. Pfister and R. Rojas, Speeding-up backpropagation - A comparison of orthogonal techniques, in Proc. of the Joint Conference on Neural Networks, Nagoya, Japan, 517–523, (1993).

[18] V.P. Plagianakos, M.N. Vrahatis, and G.D. Magoulas, Nonmonotone Methods for Backpropagation Training with Adaptive Learning Rate, In: Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN'99), Washington D.C., to appear, (1999).

[19] 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).

[20] E. Polak, Optimization: Algorithms and Consistent Approximations, Springer-Verlag, NY, (1997).

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

[22] M. Riedmiller and H. Braun, A direct adaptive method for faster backpropagation learning: the Rprop algorithm, Proceedings of the IEEE International Conference on Neural Networks, San Francisco, CA, 586–591, (1993).

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

[24] D.E. Rumelhart, G.E. Hinton, and R.J. Williams, Learning Internal Representations by Error Propagation. In D. E. Rumelhart, and J. L. McClelland, eds., Parallel Distributed Processing: Explorations in the Microstructure of Cognition, 1, pp. 318–362. MIT Press, Cambridge, Massachusetts, (1986).

[25] F. Silva and L. Almeida, Acceleration techniques for the back–propagation algorithm, Lecture Notes in Computer Science, 412, 110–119. Springer-Verlag, Berlin, (1990).

[26] 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).