An Accelerated Learning Algorithm for Multilayer Perceptron Networks

Alexander G. Parlos, *Senior Member, IEEE*, Benito Fernandez, *Member, IEEE*, Amir F. Atiya, *Member, IEEE*, Jayakumar Muthusami, and Wei K. Tsai, *Member, IEEE*

***Abstract*—An accelerated learning algorithm (ABP—adaptive back propagation[1]) is proposed for the supervised training of multilayer perceptron networks. The learning algorithm is inspired from the principle of "forced dynamics" for the total error functional. The algorithm updates the weights in the direction of steepest descent, but with a learning rate a specific function of the error and of the error gradient norm. This specific form of this function is chosen such as to accelerate convergence. Furthermore, ABP introduces no additional "tuning" parameters found in variants of the backpropagation algorithm. Simulation results indicate a superior convergence speed for analog problems only, as compared to other competing methods, as well as reduced sensitivity to algorithm step size parameter variations.**

I. Introduction

As WIDELY REPORTED in the literature, there are some shortcomings associated with the backpropagation (BP) learning algorithm. Specifically, the primary shortcomings are the uncertainty associated with finding a global minimum of the error function and the excessively long training times required to obtain acceptable errors. The second shortcoming of the BP learning, its slow convergence to a (local or global) minimum, is the main topic addressed in this study.

It is common practice in the variable structure control community (an area of nonlinear control), to force the dynamics of a nonlinear system to behave in a certain way (to be attracted by a sliding surface in the state space), by the use of feedback [1]. Inspired by this idea, the network weight update rule is chosen such that the dynamics of the error functional are forced to behave in a certain manner.

II. Description of the Learning Algorithms

For a given training set, the basic mechanism behind most supervised learning rules is the updating of the ANN weights and the bias terms, until the mean-squared-error between the output predicted by the network and the desired output (target) is less than a prespecified tolerance. Let N be the number of examples in the training set and M be the number of network output nodes. Define the total error and the error for example k as

$$E \equiv \frac{1}{N}\sum_{k=1}^{N} E(k) \tag{1}$$

$$E(k) \equiv \frac{1}{M}\sum_{i=1}^{M}(y_{[i]}(k) - \hat{y}_{[i]}(k))^2, \tag{2}$$

where $y_{[i]}(k), \hat{y}_{[i]}(k)$ are the ith network output and target for example k, respectively.

In the next two subsections the standard BP found in the literature [2], and the quickprop (QP) algorithms [3] are briefly summarized.

A. Standard Back Propagation Weight Update Rule

Let us lump all of the network weights and biases into one vector $\mathbf{w}$. In the BP algorithm, small randomly generated weight values are chosen as the starting point for the learning iterations. The weights are then updated iteratively in a steepest descent manner. One way to iterate the weights is to cycle through all examples before updating the weights, resulting in the so-called *batch* update, as follows [2]:

$$\mathbf{w}(\text{new}) = \mathbf{w}(\text{old}) - \eta\frac{\partial E}{\partial \mathbf{w}}, \tag{3}$$

where η denotes the learning rate. The vector of the error gradients in the right-hand-side of (3), includes the partial of the total error with respect to each variable in the vector $\mathbf{w}$. Alternatively, the weights can be updated after each example presentation, resulting in the so-called *individual* update scheme, as follows:

$$\mathbf{w}(\text{new}) = \mathbf{w}(\text{old}) - \eta\frac{\partial E(k)}{\partial \mathbf{w}}. \tag{4}$$

One of the major problems encountered during implementation of the BP learning rule is the proper choice and update of the learning rate η to allow convergence, while keeping the number of required iterations at a reasonable number. There have been a number of *ad hoc* variations to this algorithm reported in the literature which in some instances result in significant reduction in the number of required iterations [3]. One of the main reasons for investigating the possibility of an ABP type learning rule is the desire to reduce the sensitivity of the learning on the learning rate, without adding more "tuning" parameters.

Manuscript received September 27, 1991; revised April 20, 1993. Funded by the U.S.D.O.E. Grant DE-FG07-89ER12893.

A. G. Parlos and J. Muthusami are with the Department of Nuclear Engineering, Texas A&M University, College Station, TX, USA.

B. Fernandez is with the Department of Mechanical Engineering, The University of Texas at Austin, Austin, TX, USA.

A. F. Atiya is with the Department of Electrical Engineering, Cairo University, Cairo, Egypt.

W. K. Tsai is with the Department of Electrical and Computer Engineering, University of California at Irvine, Irvine, CA, USA.

IEEE Log Number 9211019.

[1]U.S. Patent Pending, June 1991.

B. Quick Prop Weight Update Rule

Fahlman [3] has developed an algorithm called the "quick-prop," which presently appears to be one of the fastest algorithms reported in the literature. In this algorithm, if t denotes the number of training epochs, presentation of the entire training set, then the update of the ith component of the vector $\mathbf{w}$ is given by $\Delta w^i(t) \equiv w^i(\text{new}) - w^i(\text{old})$. The algorithm follows the same procedure as in the standard BP, with the exception that following each epoch, a copy of the vector

$$\left(\frac{\partial E(t-1)}{\partial \mathbf{w}}\right)$$

is saved.

As reported by Fahlman [3], two major assumptions are made in the development of the QP algorithm: (1) the error vs. weight curve for each weight is assumed to be a convex parabola, and (2) the change in the slope of the error curve, with respect to each weight, is not affected by all the other weights that are changing at the same time. The weight update rule for QP implemented in this study is given by the following expression:

$$\Delta w^i(t) = \frac{S^i(t)}{S^i(t-1) - S^i(t)}\Delta w^i(t-1) + \eta S^i(t). \quad (5)$$

where $S^i(t)$ and $S^i(t-1)$ are the current and previous values of the ith component of the error gradient vector $\frac{\partial E}{\partial \mathbf{w}}$.

The method appears to work best with the introduction of a new parameter, μ, called the "maximum growth factor." The weight changes are not allowed to be greater in magnitude than μ times the previous weight change for a specific weight. The second term, a gradient descent term based on the current error gradient and some learning rate η, is added only when two consecutive error gradients have the same sign, otherwise it is omitted. Initialization of the learning process is accomplished by considering only the second term in (5), because the first term makes no contributions during the first iteration. Furthermore, the second term assists in the restart of certain weights which even though they had no change in their values during one iteration, they exhibit nonzero error gradient in the following iteration. Additional details regarding the QP algorithm can be found in [3] and [4].

C. Adaptive Backpropagation Weight Update Rule

The proposed weight update rule is also a steepest descent algorithm, i.e. the weights are updated in the negative direction of the gradient. The difference is that the learning rate is a specific function of the error and of the error gradient, chosen such as to accelerate network convergence. This is the motivation for the name ABP.

The algorithm uses the *batch* update mode. The weights are iterated as follows:

$$\mathbf{w}(\text{new}) = \mathbf{w}(\text{old}) - \rho(E)\frac{\partial E}{\partial \mathbf{w}}\Bigg/\left\|\frac{\partial E}{\partial \mathbf{w}}\right\|^2. \quad (6)$$

where $\rho(E)$ is some function of the error E. There are several choices for the form of $\rho(E)$. The ones considered thus far include:

$$\rho(E) \equiv \begin{cases} \eta. \\ \eta E. \\ \eta \tanh\left(\frac{E}{E_0}\right). \end{cases} \quad (7)$$

where η and E_0 are constant, non-negative numbers, representing the algorithm step size parameter, and the error normalization factor, respectively. Each choice in (7) results in a different decay mode for the total error functional.

A typical choice for the aforementioned learning rate functional is $\rho(E) = \eta E$, resulting in the following weight update law:

$$\mathbf{w}(\text{new}) = \mathbf{w}(\text{old}) - \eta E\frac{\partial E}{\partial \mathbf{w}}\Bigg/\left\|\frac{\partial E}{\partial \mathbf{w}}\right\|^2. \quad (8)$$

Equation (8) is equivalent to the choice of a linear "sliding surface" in Sliding Mode Control [1]. Because the update is in the negative direction of the gradient, under certain mild assumptions, such as a small enough step size parameter, η, the ABP is guaranteed to reach at least to within a neighborhood of a local minimum. The proof of this fact is similar to the convergence proof of the well-known steepest descent technique, found in many optimization theory books. It should be noted that the update law of (8) does not require any second-order derivative information, and it should not be confused with Hessian-based algorithms such as Newton's method. Furthermore, (8) may appear similar to parameter estimation algorithms such as the normalized least-mean-squares (NLMS). However, even though such methods employ normalized update rules, they utilize no (modeling) error gradient information.

There are several advantages in using ABP, the major one being the dependence of the learning rate on the instantaneous value of the error functional E. This allows one to dictate the dynamics of the functional resulting in "faster" convergence. However, there are disadvantages which must be considered when using ABP. The single most serious disadvantage of this approach appears to be the possibly "jumpy" behavior of the weights as a local or a global minimum is approached. Assuming that the weight vector is not close to a zero-gradient point (for instance a local or global minimum), the change in error after an iteration can be expressed as

$$\Delta E \equiv E(\text{new}) - E(\text{old}) \approx -\rho(E). \quad (9)$$

which is obtained by retaining first order terms in the Taylor series expansion for the error functional. Hence, by an appropriate choice of $\rho(E)$ one can control in some way the convergence speed. The choice $\rho(E) = \eta E$ seems reasonable because for large errors big steps are required to approach a weight vector corresponding to a (local or global) optimal error. Once the weight vector approaches such a point (i.e. the error becomes smaller), only smaller steps are required. Assuming E is not very close to a minimum, the aforementioned choice of $\rho(E)$ results to a convergence rate linear in the total error functional.

Numerically, it has been observed that the error functional behavior in the vicinity of a local minimum has an interesting

characteristic. It is known that convergence of the standard BP close to a minimum is very slow. This is because the reduction in the error, approximated as

$$E(\text{new}) - E(\text{old}) \approx -\eta \left\| \frac{\partial E(k)}{\partial \mathbf{w}} \right\|^2 \tag{10}$$

is very small as a result of a small error gradient in the vicinity of a (local) minimum. In the ABP because of the division by the square of the norm of the error gradient, the weight update steps appear to become bigger in the vicinity of a local minimum, possibly preventing entrapment. An empirical explanation to this appears to be that the error does not approach zero around a local minimum, whereas the ratio of the two gradients approaches infinity, resulting in large "jumps" in the values of the weights and consequently in the value of the error. This may be desirable when approaching a local minimum, because the possibility of entrapment may be reduced. The situation appears to be different, though, when approaching the global minimum, because it is desired to stay in this vicinity. Again, empirically, it has been found that by choosing an appropriate number of hidden layer nodes, the error functional (contained in the right-hand-side of (8)) appears to become smaller in the vicinity of the global minimum, observing smoother convergence of the error to some final value. This behavior has been numerically confirmed during the simulation studies presented in the following section.

III. Simulation Results

A comparison between the new algorithm and three competing algorithms: the standard BP algorithm with both individual and batch update, and the QP algorithm, has been performed. Furthermore, only analog problems have been considered in this comparison. This means training the neural network to map analog (real-valued) inputs to analog outputs. The reason is that these are the types of problems at which ABP is advantageous. For binary problems, such as the parity problem or the encoder problem, ABP was found to be as fast as the BP algorithm. ABP is therefore not intended for use in binary problems, but rather for analog-type problems, such as the ones encountered in typical control and signal processing applications. Furthermore, it has been observed that ABP performs best when used in networks containing linear discriminatory functions in their output layers.

The numerical experimentation procedure is similar to that performed in some of the other comparison papers, such as Fahlman [3], and Thomas [4]. The procedure is as follows: A randomly generated initial weight configuration is chosen and kept constant, while the learning rate (or any other tuning parameter) is tuned to obtain the fastest possible convergence. Then nine additional initial weight settings (also generated randomly) are chosen, and each of the algorithms is executed using these sets of weights, and the algorithm step size parameter obtained from the tuning procedure using the first set of weights. Obviously, the best step size parameter is different for each of the algorithms tested. The target is to obtain a normalized RMS error ($\sqrt{E}$ divided by the standard deviation of the target samples) of 5% in less than 100 000 iterations. If an algorithm does not reach the targeted RMS error value within the allowed iterations, it is considered divergent.

TABLE I
Sinusoidal Function Approximation Using a 1–15–1 Network

Update rule	Trials	η	Converged trials	Average of converged trials	Standard deviation
BP (batch)	10	0.0028	10	25844	4217
BP (ind.)	10	0.011	10	7336	1461
QP ($\mu = 1.7$)	10	0.009	10	2521	2011
ABP	10	0.05	10	968	363

TABLE II
Sinusoidal Function Approximation Using a 1–30–1 Network

Update rule	Trials	η	Converged trials	Average of converged trials	Standard deviation
BP (batch)	10	0.0006	10	41360	13048
BP (ind.)	10	0.003	10	8265	1375
QP ($\mu = 1.7$)	10	0.03	10	2456	2574
ABP	10	0.05	10	1056	202

TABLE III
A Simple Two-Dimensional Function Approximation Using a 2–10–1 Network

Update rule	Trials	η	Converged trials	Average of converged trials	Standard deviation
BP (batch)	10	0.0015	10	6713	1833
BP (ind.)	10	0.011	10	2225	2052
QP ($\mu = 1.5$)	10	0.008	10	3424	2760
ABP	10	0.011	10	516	223

A. One-Dimensional Problem

In this example a one-dimensional mapping, $y = \sin(x)$, is used. The training set consists of 25 samples of the function in the range from -2π to 2π. Tables I and II show the results for a 1-15-1 and a 1-30-1 network, respectively.

B. Two-Dimensional Problems

In order to demonstrate the effectiveness of ABP for other, potentially more complex problems, a relatively smooth two-dimensional function is considered. The function has the form $z = f(x, y) = 0.05 + 0.1x^2 \cos(y + 3) + 0.5xye^{-(y^2-1)}$. The training set consists of an 11×11 grid from $[-2, 2] \times [-2, 2]$. Tables III and IV show the simulation results using 2-10-1 and 2-20-1 networks, respectively.

A highly irregular two-dimensional function has also been considered for testing. The function is $z = f(x, y) = \sin(x)\sin(y)$, and it is sampled by an 11×11 grid from $[-\pi, \pi] \times [-\pi, \pi]$, thus allowing one entire cycle in each dimension, with two positive peaks and two negative peaks. The simulation results using 2-10-1 and 2-20-1 networks are presented in Tables V and VI, respectively. It is worth noting that in Table V ABP appears to perform more poorly than both

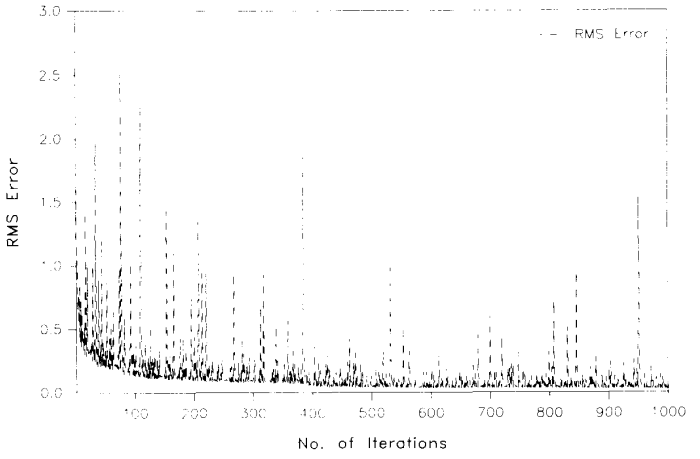

Fig. 1. ABP error profile during a training session.

TABLE IV
A SIMPLE TWO-DIMENSIONAL FUNCTION APPROXIMATION USING A 2–20–1 NETWORK

Update rule	Trials	η	Converged trials	Average of converged trials	Standard deviation
BP (batch)	10	0.0016	10	5621	3182
BP (ind.)	10	0.011	10	556	219
QP ($\mu = 1.5$)	10	0.006	10	659	244
ABP	10	0.012	10	307	70

TABLE V
AN IRREGULAR TWO-DIMENSIONAL FUNCTION APPROXIMATION USING A 2–10–1 NETWORK

Update rule	Trials	η	Converged trials	Average of converged trials	Standard deviation
BP (batch)	10	0.0011	7	56510	28298
BP (ind.)	10	0.0084	6	10703	7842
QP ($\mu = 1.4$)	10	0.008	3	19707	17954
ABP	10	0.0028	9	21548	13322

QP and BP (Ind.). Nevertheless, it should be kept in mind that only trials with less than 100 000 iterations are included in the averaging presented in these tables. Because fewer trials have converged for both the QP and the BP (Ind.) algorithms, these methods reveal lower average number of iterations for the converged trials.

Tables I through VI demonstrate the improved performance of the ABP algorithm, in terms of speed and chance of convergence, for the class of problems analyzed in this study. For complex problems, the comparative improvement in the performance of ABP appears more apparent. For example, for the two-dimensional example problem $z = \sin(x)\sin(y)$, one of the most difficult problem investigated in this study, the ABP acceleration is more pronounced. It is also observed that for the ABP algorithm the standard deviation is comparatively small, when an appropriately size network is chosen. Furthermore, the ABP algorithm does not introduce any additional tuning parameters, as it is the case with the parameter μ of the QP algorithm, requiring more trials for acceptable results.

TABLE VI
AN IRREGULAR TWO-DIMENSIONAL FUNCTION APPROXIMATION USING A 2–20–1 NETWORK

Update rule	Trials	η	Converged trials	Average of converged trials	Standard deviation
BP (batch)	10	0.0007	10	28610	12325
BP (ind.)	10	0.052	7	4239	3228
QP ($\mu = 1.3$)	10	0.008	8	6604	11014
ABP	10	0..012	10	1083	280

The RMS error resulting from the ABP learning versus the number of iterations, for one trial of the last example, is shown in Fig. 1. It is observed that the error fluctuates on its way down. Despite this jumpy behavior, not a single analog problem was encountered where the ABP failed to converge while the standard BP or the QP algorithms succeeded. For the same trial case, the BP algorithm exhibits a smooth, monotonic descent in the error, whereas the QP algorithm exhibits initial fluctuations, followed by smoother decay of the error function. The RMS errors resulting from both the BP and the QP algorithms, however, decay at a slower rate than the equivalent error when using the ABP algorithm. Furthermore, a study investigating the sensitivity of the ABP algorithm to variations in the algorithm step size parameter, η, during a trial for the last example is depicted in Fig. 2. Again, no simulation was allowed to continue beyond 100 000 iterations, and trials not converged to desired accuracy within this number of iterations are indicated with 100 000 on the vertical axis of Fig. 2. Furthermore, trials that were interrupted because of overflow errors have been omitted from this figure. As anticipated, the ABP algorithm appears more robust to changes in the step size parameter, allowing convergence for wider variations in its values.

IV. SUMMARY AND CONCLUSION

An accelerated learning algorithm is proposed for the supervised training of multilayer networks. The basic principle of

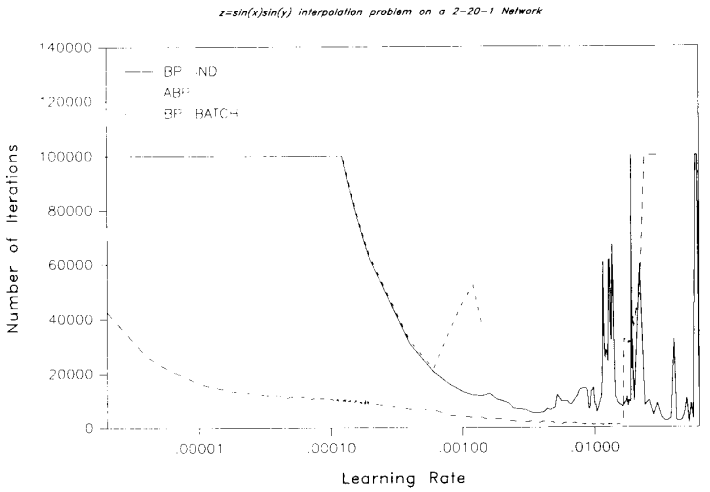

Fig. 2. Sensitivity of learning algorithms to various learning rates.

the algorithm is inspired from the theory of variable structure systems. The rate of change of the network weights is chosen such that the error functional to be minimized is forced to "decay" in a certain mode. The weight update rule requires knowledge of only the error gradient terms found in the standard BP algorithms, and it does not necessitate retention of any information from the previous update step, as it is the case with the BP with momentum and the QP algorithms. There is, however, a disadvantage in using ABP, namely the possibly "jumpy" behavior when approaching local minima. Thus far, though, no numerical problems have been encountered during implementation of the ABP. The simulations performed indicate that the new method is considerably faster for analog problems, and it is more likely to converge than the standard BP algorithm, as well as the competing QP algorithm. Additionally, numerical studies demonstrate that the ABP algorithm exhibits less sensitivity to variations in the algorithm step size parameter.

ACKNOWLEDGMENT

The authors would like to acknowledge Dr. S. E. Fahlman of Carnegie Mellon University, for providing a programmed version of the Quickprop algorithm, which was modified for use in this study.

REFERENCES

[1] J. J. Slotine and S. S. Sastry, "Tracking control of non-linear systems using sliding surfaces, with applications to robot manipulators," *International Journal of Control*, Vol. 38, No. 2, 465–492, 1983.

[2] D. Rumelhart, G. Hinton and R. Williams, "Learning Internal Representation by Error Propagation," in *Parallel Distributed Processing*, Vol. I, MIT Press, Cambridge, MA, 1986.

[3] S. E. Fahlman, "Faster Learning Variations on Back-Propagation: An Empirical Study," *Proceedings of the 1988 Connectionist Model Summer School*, pp. 38–51, Carnegie Mellon University, also Report CMU-CS-88-162, June 1988.

[4] T. R. Thomas and T. L. Brewster, "Experiments in Finding Neural Network Weights," *LANL Report*, LA-11772-MS, DE90 007696, April 1990.