A NEW METHOD IN NEURAL NETWORK SUPERVISED TRAINING WITH IMPRECISION

G.D. Magoulas†, *M.N. Vrahatis*‡ *and G.S. Androulakis*‡

† *Department of Electrical & Computer Engineering, University of Patras, GR-261.10, Patras, Greece.*
E-mail : magoulas@ee-gw.ee.upatras.gr

‡ *Department of Mathematics, University of Patras,GR-261.10 Patras, Greece.*
E-mail vrahatis|gsa@math.upatras.gr

ABSTRACT

We propose a method that proceeds solely with the minimal information of the error function and gradient which is their algebraic signs and takes minimization steps in each weight direction. This approach seems to be practically useful especially when training is affected by technology imperfections and environmental changes that cause unpredictable deviations of parameter values from the designed configuration. Therefore, it may be difficult or impossible to obtain very precise values for the error function and the gradient of error during training.

1. INTRODUCTION

The literature on supervised training of feed forward neural networks (FNN) includes all algorithms which consider supervised training as the unconstrained minimization of an objective function. The square error over a finite set of input–desired output patterns, containing T representative pairs, is usually taken as the function to be minimized:

$$E = \sum_{t=1}^{T} \sum_{j=1}^{N_L} \left[\sigma\Big(\sum_{i=0}^{N_{L-1}} w_{ij}^{L-1,L} y_{i,t}^{L-1} \Big) - d_{j,t} \right]^2 , \quad (1)$$

where $w_{ij}^{L-1,L}$ is the connection weight from the ith neuron at the $(L-1)$ layer to the jth neuron at the L output layer, $y_{i,t}^{L-1}$ denotes the output of the ith neuron belonging to the $(L-1)$ layer at pattern t, $w_{0j}^{L-1,L}$ denotes the bias of the jth neuron at the Lth layer associated with zero input, σ is a nonlinear activation function, such as the well known sigmoid function $\sigma(net_j) = (1+e^{-net_j})^{-1}$ and $d_{j,t}$ is the desired response of an output neuron at the input pattern t. Each pass through the entire training set to compute E is called an *epoch*. The minimization of E corresponds to updating the weights by epoch, named *batch training*.

The most popular supervised training method, named backpropagation algorithm (BP) [1] updates the weights using the steepest descent [2] thus it suffers from a slow convergence rate and often yields suboptimal solutions. Alternatively, training methods originating from the field of numerical analysis such as steepest descent [3, 4, 5, 6], nonlinear conjugate gradients [7] and second derivative based methods [8, 9] have been proposed.

These training algorithms require precise error function and gradient values. However, the fact that neural networks are simulated on computers with finite accuracy bounds implies that it may be difficult or impossible to obtain very precise values for the error function and the gradient [10]. On the other hand, the training algorithm introduced in this contribution proceeds solely with the minimal information of the error function and gradient which is their algebraic signs.

In Section 2 several sources of imprecision encountered in supervised training of FNNs are discussed. In Section 3 our training method is derived, while, in Section 4, the corresponding algorithm is presented. In Section 5 we present some simulation results and we finally end, in Section 6, with conclusions and a discussion of further research.

2. IMPRECISION INCURRED IN NEURAL NETWORK TRAINING

Software–based FNNs while exploiting the advantage of training by examples, are directly affected by numerical imprecision; a common problem encountered in numerical simulations. The arithmetic operations required in the numerical simulations of the BP affect the accuracy of the result. All of these operations can be severely impacted by imprecision, especially for problems that are ill–conditioned even when a high precision is used [11]. Moreover, using minimization methods for training FNNs derivative calculations as well as one–dimensional subminimization (for the case of nonlinear conjugate gradient methods) and approximations of the inverse Hessian (for the case of quasi–Newton and variable metric methods) are required. A detailed analysis on the sources of imprecision involved to this kind of computations is presented in [12, 8].

A crucial factor of imprecision, as pointed in [10], is the evaluation of the activation function $\sigma(net_j)$. This function is calculated using a polynomial approximation which implies that numerical accuracy constraints are introduced in the calculation of the error value.

In the special case of FNN applications with a very large number of patterns, the errors involved because of the imprecision in the computation of the gradient of the batch error measure may be comparable to the gradient itself. In general, the rounding off error is one serious source of imprecision in the error function value E and its gradient. When this type of error is generated by overflow, mostly occurred during the calculation of the term net_j^l, is not crucial due to the characteristics of the sigmoid neuron model. However the underflow error that occurs during the calculation of the backpropagating error signal [1] denoted by δ_j is very crucial and can lead to nonconvergence and saturation [5]. Despite the fact that the error associated with neuron j can be significant, δ_j may become negligible and rounded to zero if the derivative of the activation function $\sigma'(net_j^l)$ is very

small. In this case weight adaptation is not possible although there is a large value of error.

A similar to saturation phenomenon occurs when second derivative based training methods are used. In this case, problematic situation occurs when the Hessian is not positive define, as well as when it is ill–conditioned or singular (see [9, 8] for simulations on these kind of problems).

Moreover, in various small and large scale FNN applications the error surface has flat regions. This results in the evaluation of imprecise gradient values which affects all training methods that use first derivatives in case we are far from the minimum.

Imprecision is also encountered when the partial derivative of E with respect to the nth weight is evaluated using the forward–difference approximation:

$$\nabla_n E(w) = [E(w + pert \cdot e_n) - E(w)]/pert, \quad (2)$$

where *pert* is a small quantity and e_n denotes the nth unit vector. This approach has been used by several researchers as an alternative to the generation of derivatives using the backpropagation chan rule [1] because only forward operations of the FNN can give the weight updates. As reported in [12], truncation error as a consequence of the neglected terms in the Taylor series, condition or cancellation error due to imprecise values of E and rounding off errors are intoduced in this case.

The above mentioned problematic situations can be handled, at least in part, by developing training algorithms that can take into consideration that the error function and gradient values are known only imprecisely. In the next section, a new training method eminently suitable to work under imprecise conditions is presented. If a method is capable of converging when imprecise values are used then computational effort can be saved by avoiding the extra work required to compute precise function and gradient values.

3. THE NEW TRAINING METHOD

The new training method is derived from a recently proposed method for unconstrained optimization [13]. This optimization method has an improved performance when compared with nonlinear conjugate gradient methods and BFGS on many well known test cases (see [13] for comparative results).

To simplify the notation, the weights are written as vector $w = (w_1, w_2, \ldots, w_{n-1}, w_n, w_{n+1}, \ldots, w_N)^\top$ defining a point in the N–dimensional real Euclidean space $\mathbb{R}^N$, where N denotes the total number of weights and biases in the network.

In order to find a proper weight vector w^* so that the FNN exhibits a desired behavior we want a sequence of weight vectors $\{w^k\}_0^\infty$ that converges to a minimizer of the error function E. In the new training method an element of this sequence of weight vectors can be obtained by solving an one–dimensional equation for each component of the weight vector as the following one:

$$\begin{aligned} &E(w_1^{k+1}, \ldots, w_{n-1}^{k+1}, w, w_{n+1}^k, \ldots, w_N^k) - \\ &E(w_1^{k+1}, \ldots, w_{n-1}^{k+1}, w_n^k, w_{n+1}^k, \ldots, w_N^k) = 0. \end{aligned} \quad (3)$$

Solving the above equation for w and keeping all the other components of the weight vector in their constant values we get $\hat{w}_n$ as a solution. Then, the weight vector $(w_1^{k+1}, \ldots, w_{ni-1}^{k+1}, \hat{w}_n, w_{n+1}^k, \ldots, w_N^k)$ possesses the same error function value with the vector $(w_1^{k+1}, \ldots, w_{n-1}^{k+1}, w_n^k, w_{n+1}^k, \ldots, w_N^k)$, i.e., these points belong to the same contour line of the function E.

Assuming that the Hessian of the error function E at w^* is positive define, which means that E curves up from w^* in all directions, any point which belongs to the line that connects the points $(w_1^{k+1}, \ldots, w_{n-1}^{k+1}, \hat{w}_n, w_{n+1}^k, \ldots, w_N^k)$ and $(w_1^{k+1}, \ldots, w_{n-1}^{k+1}, w_n^k, w_{n+1}^k, \ldots, w_N^k)$ possesses smaller error value than these points. To find such a point we update at the $(k+1)$st iteration the nth component w_n^k using the following relation:

$$w_n^{k+1} = w_n^k + \gamma_k(\hat{w}_n - w_n^k), \quad \gamma_k \in (0,1). \quad (4)$$

Obviously the above procedure handles N–dimensional problems using reduction to simpler one–dimensional equations like Eq. (3). It can be considered as an *nonlinear successive overrelaxation* (nonlinear SOR) method [2]. For convergence properties of the above method see [13].

Any one of the well known one–dimensional rootfinding methods [2] can be employed to solve Eq. (3). Here we use the bisection method because it is the only method that can be applied to imprecise problems.

Specifically, in order to compute a root $\hat{w}_n$ of Eq. (3) in the interval $[a_n, b_n]$, we use the bisection method which has been modified to the following simplified version [14]:

$$\hat{w}_n^{p+1} = \hat{w}_n^p + C\,\mathrm{sgn}\Big(E(z^p) - E(z)\Big)/2^{p+1}, \quad (5)$$

where $p = 0, 1, \ldots$ indicates iterations, sgn defines the well known sign function, $z^p = (w_1^{k+1}, \ldots, w_{n-1}^{k+1}, w_n^p, w_{n+1}^k, \ldots, w_N^k)$, $z = (w_1^{k+1}, \ldots, w_{n-1}^{k+1}, w_n^k, w_{n+1}^k, \ldots, w_N^k)$ and $C = \mathrm{sgn}\Big(E(z^0) - E(z)\Big)h_n$ where $h_n = b_n - a_n$, and $z^0 = (w_1^{k+1}, \ldots, w_{n-1}^{k+1}, a_n, w_{n+1}^k, \ldots, w_N^k)$.

To minimize the function E by applying the above sequence we choose the endpoints a_n and b_n in such a way that, at the left endpoint a_n the nth component of the gradient has negative value, or, at the right endpoint b_n the nth component of the gradient has positive value. In order that this condition is fulfilled, we choose these endpoints by the following relation:

$$a_n = w_n^k - \frac{1}{2}\Big\{1 + \mathrm{sgn}\,\nabla_n E(z)\Big\}h_n, b_n = a_n + h_n. \quad (6)$$

The number of iterations η of the sequence (5) which are required to obtain an approximate root $\hat{w}_n$ such that $|\hat{w}_n - w_n^*| \leq \delta$, for some $\delta \in (0,1)$, is given by the following relation:

$$\eta = \lceil \log_2(h\,\delta^{-1}) \rceil, \quad (7)$$

where the notation $\lceil \cdot \rceil$ is the ceiling of the real number quoted.

It is evident from the sequence (5) that the only computable information required by this method is the algebraic signs of the function E. Also, the gradient values in Relation (6) can be achieved by using Relation (2). In this case the sign can be accurately obtained by comparing the relative size of the function value. So, rounding and quantization errors, causing, in simulations, imprecise function values, cannot affect its convergence as long as the signs are preserved. In addition, storage requirements regarding gradient are minimized. Furthermore, Sequence (5) is a global convergence method, it always converges within the given interval, it is optimal [15],

i.e., it possess asymptotically the best rate of convergence and it has a known behavior concerning the number of iterations required to obtain a root with a predetermined accuracy (see Relation (7)).

4. TRAINING WITH IMPRECISION ALGORITHM – TIA

A high level description of the batch version of the new algorithm named TIA is outlined below.

Initialization: Randomly initialize the weight vector. Define stepsizes h_n in each weight direction, the relaxation coefficients γ, ζ; the maximum number of epochs (ME); the predetermined desired accuracies δ and ε.

Recursion: For $k = 1, \ldots, ME$ (epochs); and for $n = 1, \ldots, N$ (components of the weight vector w).

1. Present all patterns to the network and compute the output of all nodes.
   (a) Calculate E using Relation (1).
   (b) Define the interval $[a_n, b_n]$ where the bisection seeks the root that minimizes E using Eq. (6).
   (c) Find the parameter $\hat{w}$ that satisfies Eq. (3) by applying the sequence (5) in (a_n, b_n) within accuracy δ.
   (d) If $\hat{w} \geq b - \delta$, set $y^0 = (w_1^{k+1}, \ldots, w_{n-1}^{k+1}, w_n^k, w_{n+1}^k, \ldots, w_N^k)$ and go to Step 5; otherwise continue.
   (e) Set $w_n^{k+1} = w_n^k + \gamma\left(\hat{w} - w_n^k\right)$.
   (f) If $n < N$, increment n and begin recursion.
2. Check the convergence criterion $\|w^{k+1} - w^k\| \leq \varepsilon$. In case it is true, the weights remain unmodified so *Terminate*; otherwise go to next step.
3. Set $w^{k+1} = w^k + \zeta(w^{k+1} - w^k)$.
4. If $\operatorname{sgn}\left(E(w^{k+1}) - E(w^k)\right) \leq 0$, increment k and reset n; otherwise set $y^0 = w^k$ and continue.
5. Apply Armijo's algorithm utilizing the starting value y^0 and take its output value y^{MAR}.
6. Set $w^{k+1} = y^{MAR}$, increment k and reset n.

Termination: Get the final weights and the corresponding error value.

Note that Step (1d) of the above algorithm detects if the bisection does not converge. This can happen only when Eq. (5) converges to the right endpoint b_n or when Bolzano's criterion is not fulfilled. Furthermore, Steps (5) and (6) apply the well–known Armijo's method [13] which belongs to the class of steepest descent methods for unconstrained minimization and, under some suitable assumptions, it always converges to a local minimum.

5. SIMULATION RESULTS

TIA was applied to several problems and a number of simulations were performed to evaluate the new algorithm and compare its performance with several popular batch training methods. Parameters γ and ζ were 0.5 and 1 respectively for all simulation experiments. The results of the simulations that have reached solution are reported in the following tables where the corresponding parameters indicate: **m.n.** mean number of epochs; **st.d.** standard deviation of epochs; **m.n.e.** mean value of obtained error; and the last row in each Table contains the mean number of algebraic signs evaluations and the corresponding standard deviation for TIA. The gradient was computed using the finite difference approximation for TIA and backpropagation chain rule [1] for the other methods.

Table 1. Results for the XOR problem

Method	m.n.	st.d.	m.n.e.	Success
BP	561.042	550.403	0.0396	24 %
MBP	511.667	525.051	0.0395	21 %
ABP	233.357	332.848	0.0285	28 %
TIA	12.775	8.238	0.0235	40 %
	171.075	207.483		

Table 2. Results for the numeric font problem

Method	m.n.	st.d.	m.n.e.	Success
BP	9280.2	13177	0.01996	46 %
ABP	1111.1	525.34	0.01675	55 %
TIA	4.285	1.3801	0.00389	100 %
	3695.57	4682.58		

5.1. Classification of the four XOR patterns

The four XOR [1] patterns are classified into $\{0, 1\}$ using a simple FNN architecture consisting of two linear, unity–gain input neurons with biases set to zero, and three neurons (two hidden, one output) of sigmoid activation model with biases to be learned. The weights for all methods are initialized in $[-10, 10]$. The stepsize for the BP, the momentum BP (MBP) [3], and the adaptive BP (ABP) [4] is set to the classical value of 0.75. The goal is an $E < 0.04$ within 600 epochs and 1000 simulations have run. The results are shown in Table 1.

5.2. Learning of the 8×8 dot numeric font

An FNN is used as a simple numeral recognizer with an 8×8 pixel input and a 10 bit one–hot output representing zero through nine. The network has 64 input neurons of unity–gain, non bias, and 16 neurons (6 hidden, 10 output) of sigmoid activation model with biases. The stepsize for the BP and the ABP is set to 1.5 [16]. The weights for all methods are initialized in $[-1, 1]$. The goal is an $E < 0.02$ and 1000 simulations have been performed. The results are shown in Table 2.

5.3. Inaccurate measure of the error

In this simulation the performance of the TIA in the presence of measurement noise will be demonstrated. The classification of the XOR patterns is used here and is corrupted by measurement noise. The noise is assumed to have a uniform random distribution from the interval $[-1, +1]$. The noise contributes to an imprecise value of E and would be expected to result in miscalculations of the parameter updates. The results are summarized in Table 3. The results for TIA here are worse when compared with the results of Subsection 5.1. This is due to the fact that the signs required in Eqs. (5) and (6) are not correct.

Table 3. Training with imprecise error value

m.n.	st.d.	m.n.e.	Success
44.6	25.102	0.00143	30 %

Table 4. Training with weight variations

m.n.	st.d.	m.n.e.	Success
44.115	28.187	0.00945	44 %

Table 5. Function approximation problem

m.n.	st.d.	m.n.e.	Success
608.277	489.992	0.4975	100 %
46994.5	29714.8		

5.4. Robustness against weight variations

The effects of weight variations are simulated by introducing a factor into the weight vector so that the actual weights w are related to the ideal weights w' by $w = w'(1 + noise)$ where *noise* is a uniform random distribution from the interval $[-1, +1]$ leading in up to 100% weight vector variation. The actual weights w are used in the weight update calculations. The goal is an $E < 0.04$ within 600 epochs and weights are initialized in $[-10, 10]$. The results of TIA for the XOR problem are shown in Table 4. BP, MBP and ABP have not converged with such large weight variations.

5.5. Continuous function approximation

The task is the approximation of a continuous function, $f(x) = \sin(x)\cos(2x)$ with domain $0 \leq x \leq 2\pi$ and range $-1 \leq f(x) \leq 1$. The function is learned through a set of 20 input–output patterns, which are uniformly chosen and represent the function adequately [7].

BP with line search instead of a fixed stepsize, needs, on the average, 4×10^6 error function evaluations to reach the minimum. BP with a fixed stepsize never found a global minimum due to oscillations; when BP is approaching a global minimum a smaller stepsize is necessary for the algorithm to continue decreasing the error. To overcome this problem the BP with variable stepsize, as proposed by Silva *et al.*, [17] is used. The algorithm converged to the global minimum needing more than two million error function evaluations. The corresponding results for TIA are exhibited in Table 5.

6. CONCLUDING REMARKS

The main feature of the new method in the formulation of the learning problem is the reduction to simple one–dimensional equations for the components $w_1, w_2, \ldots, w_N$ of E. The TIA approximates the derivatives using finite differences but it is capable of giving solutions with more accuracy in the error goal than the BP. It also requires only the algebraic signs of the error function and gradient values to be correct.

Preliminary results suggest that our method copes successfully with on–line training. Also, our method may be of practical interest for implementation of FNN with hardware to carry out the training on–chip. In this case, it may be difficult or impossible to obtain very precise values for the error function and the gradient of error. An analysis and results for these approaches will appear in a future publication.

REFERENCES

[1] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, 1*, (E. Rumelhart and J. L. McClelland, eds.), pp. 318–362, MIT Press, 1986.

[2] J. M. Ortega and W. C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables.* New York: Academic Press, 1970.

[3] R. A. Jacobs, "Increased rates of convergence through learning rate adaptation," *Neural Networks*, vol. 1, pp. 295–307, 1988.

[4] T. P. Vogl, J. K. Mangis, A. K. Rigler, W. T. Zink, and D. L. Alkon, "Accelerating the convergence of the back-propagation method," *Biol. Cyber.*, vol. 59, pp. 257–263, 1988.

[5] A. K. Rigler, J. M. Irvine, and T. P. Vogl, "Rescaling of variables in backpropagation learning," *Neural Networks*, vol. 4, pp. 225–229, 1991.

[6] G. D. Magoulas, M. N. Vrahatis, and G. S. Androulakis, "Effective backpropagation training with variable stepsize," *Neural Networks*, 1996. accepted for publication.

[7] P. P. Van der Smagt, "Minimisation methods for training feedforward neural networks," *Neural Networks*, vol. 7, pp. 1–11, 1994.

[8] R. Battiti, "First– and second–order methods for learning: between steepest descent and Newton's method," *Neural Computation*, vol. 4, pp. 141–166, 1992.

[9] G. D. Magoulas, M. N. Vrahatis, T. N. Grapsa, and G. S. Androulakis, "Neural network supervised training based on a dimension reducing method," *Ann. Math. Artif. Intel.*, 1996. in press.

[10] J. Wray and G. G. R. Green, "Neural networks, approximation theory and finite precision computation," *Neural Networks*, vol. 8, pp. 31–37, 1995.

[11] J. L. Holt and J. Hwang, "Finite precision error analysis of neural network hardware implementations," *IEEE Trans. Comp.*, vol. 42, pp. 281–290, 1993.

[12] P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization.* Academic Press, NY, 1981.

[13] M. N. Vrahatis, G. S. Androulakis, and G. E. Manoussakis, "A new unconstrained optimization method for imprecise function values," *J. Math. Anal. Appl.*, vol. 197, pp. 586–607, 1996.

[14] M. N. Vrahatis, "CHABIS: a mathematical software package for locating and evaluating roots of systems of nonlinear equations," *ACM Trans. Math. Software*, vol. 14, pp. 330–336, 1988.

[15] K. Sikorski, "Bisection is optimal," *Numer. Math.*, vol. 40, pp. 111–117, 1982.

[16] A. Sperduti and A. Starita, "Speed up learning and network optimization with extended backpropagation," *Neural Networks*, vol. 6, pp. 365–383, 1993.

[17] F. Silva and L. Almeida, "Acceleration techniques for the backpropagation algorithm," in *Lecture Notes in Computer Science*, pp. 110–119, Berlin: Springer-Verlag, 1990. 412.