Sign-methods for Training with Imprecise Error Function and Gradient Values

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

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

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

Abstract

Training algorithms suitable to work under imprecise conditions are proposed. They require only the algebraic sign of the error function or its gradient to be correct, and depending on the way they update the weights, they are analyzed as composite nonlinear Successive OverRelaxation (SOR) methods or composite nonlinear Jacobi methods, applied to the gradient of the error function. The local convergence behavior of the proposed algorithms is also studied. The proposed approach seems practically useful when training is affected by technology imperfections, limited precision in operations and data, hardware component variations 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 the error during training.

Introduction

Software–based feedforward neural networks (FNNs) while exploiting the advantage of training by examples, are directly affected by numerical imprecision; a common problem encountered in numerical simulations.

The most popular supervised training method, named backpropagation algorithm (BP) [12] updates the weights using the steepest descent method [8], 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 gradient descent [6, 7, 11, 19], nonlinear conjugate gradients [14] and second derivative based methods [3] have been proposed. These training algorithms require precise error function and gradient values. However, in many cases it is difficult or impossible to obtain very precise values for the error function and the gradient [23].

The precision of the arithmetic operations required in the numerical simulations of neural networks affects 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 [5]. Moreover, using minimization methods for training FNNs derivative calculations as well as one–dimensional subminimization (in the case of nonlinear conjugate gradient methods) and approximations of the inverse Hessian (in the case of quasi–Newton and variable metric methods) are required. A detailed analysis on the sources of imprecision involved in this kind of computations is presented in [3, 4]. A crucial factor of imprecision is the evaluation of the sigmoid activation function [23]. 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 due to the imprecision in the computation of the batch error measure gradient 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 during the calculation of the sum of the weighted inputs of each neuron, it is not crucial due to the characteristics of

0-7803-5529-6/99/$10.00

the sigmoid neuron model. However, when the error occurs during the calculation of the backpropagating error term [12] is very crucial and can lead to nonconvergence and saturation [11]. Despite the fact that the error associated with a neuron can be significant, the backpropagating error term may become negligible and rounded to zero if the derivative of the activation function 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 definite, as well as when it is ill–conditioned or singular (see [3] 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 error function E with respect to the ith weight is approximated using for example forward–differences:

$$\partial_i E(w) \simeq [\, E(w + pert \cdot e_i) - E(w) \,]/pert, \quad (1)$$

where *pert* is a small quantity and e_i denotes the ith column of the identity matrix. This approach has been used by several researchers as an alternative to the generation of derivatives using the backpropagation chain rule [12], because only forward operations of the FNN can give the weight updates. As reported in [4], 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 introduced in this case.

The above mentioned problematic situations, as well as those encountered in neural network training for embedded control applications and in the implementation of neural networks with hardware to carry out the training on-chip, 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. The training algorithms introduced in this contribution proceed solely with the minimal information of the gradient of the error function, which is the algebraic sign and allow using a different adaptive learning rate for each weight. Depending on the way they update the weights, they are analyzed as composite nonlinear Successive OverRelaxation (SOR) methods or composite nonlinear Jacobi methods (composite SOR and Jacobi methods are used for the numerical solution of a system of nonlinear equations), applied to the gradient of the error function. The local convergence behavior of the proposed algorithms is also studied. To guarantee the convergence of the algorithms, when the initial weights are away from a minimizer, stabilization techniques are suggested.

The Composite Nonlinear Jacobi and SOR Methods

It is well known that all the local minima w^* of the error function E, when the activation functions used are continuously differentiable, satisfy the necessary conditions:

$$\nabla E(w^*) = \Theta^n = (0, 0, \ldots, 0). \quad (2)$$

Eq. (2) represents a set of n nonlinear equations which must be solved to obtain w^*. Therefore, one approach to the minimization of the error function E is to seek the solutions of the set of Eq. (2) by including a provision to ensure that the solution found does indeed correspond to a local minimum. This is equivalent to solving the following system of equations:

$$\begin{aligned} \partial_1 E(w_1, w_2, \ldots, w_n) &= 0, \\ \partial_2 E(w_1, w_2, \ldots, w_n) &= 0, \\ &\vdots \\ \partial_n E(w_1, w_2, \ldots, w_n) &= 0. \end{aligned} \quad (3)$$

The nonlinear Jacobi scheme and its convergence

The class of *nonlinear Jacobi* methods is widely used for the numerical solution of System (3). These methods are considered as parallel ones since they apply a parallel update of the variables [8]. Starting from an arbitrary initial weight vector $w^0 \in \mathcal{D}$, one can subminimize at the kth epoch the function:

$$E(w_1^k, \ldots, w_{i-1}^k, w_i, w_{i+1}^k, \ldots, w_n^k), \quad (4)$$

along the ith direction and obtain the corresponding subminimizer $\hat{w}_i$. Obviously for the subminimizer $\hat{w}_i$

$$\partial_i E(w_1^k, \ldots, w_{i-1}^k, \hat{w}_i, w_{i+1}^k, \ldots, w_n^k) = 0. \quad (5)$$

This is a one-dimensional subminimization because all other components of the weight vector, except the ith, are kept constant. Then the ith weight is updated according to the equation:

$$w_i^{k+1} = w_i^k + \tau_k(\hat{w}_i - w_i^k), \quad (6)$$

for some relaxation factor τ_k. The error function in (4) is subminimized in parallel for all i.

Depending on the applied one–dimensional minimization method various composite nonlinear Jacobi training algorithms can be obtained. It is worth noticing

that the number of the iterations of the subminimization method is related to the requested accuracy in obtaining the subminimizer approximations. Thus, significant computational effort is needed in order to find very accurate approximations of the subminimizer in each weight direction at each epoch. Moreover, this computational effort is increased for FNNs with several hundred weights. On the other hand, it is not certain that this large computational effort speeds up the minimization process for nonconvex functions when far from a minimizer w^*. Thus, we propose to obtain $\hat{w}_i$ by minimizing the function (4) with one iteration of a minimization method. Note that this practice is also suggested for the iterative solution of nonlinear equations [8].

The convergence analysis is developed under appropriate assumptions and provides useful insight into the new class. The objective is to show that there is a neighborhood of a minimizer of the error function for which convergence to the minimizer can be guaranteed.

Theorem 1: Let $E: \mathcal{D} \subset \mathbb{R}^n \rightarrow \mathbb{R}$ be twice continuously differentiable in an open neighborhood $\mathcal{S}_0 \subset \mathcal{D}$ of a point $w^* \in \mathcal{D}$ for which $\nabla E(w^*) = \Theta^n$ and the Hessian, $H(w^*)$ is positive definite with the property A^π. Then there exists an open ball $\mathcal{S} = \mathcal{S}(w^*, r)$ in $\mathcal{S}_0$ (where $\mathcal{S}(w^*, r)$ denotes the open ball centered at w^* with radius r), such that any sequence $\{w^k\}_{k=0}^{\infty}$ generated by the nonlinear Jacobi process converges to w^* which minimizes E.

Proof: Clearly, the necessary and sufficient conditions for the point w^* to be a local minimizer of the function E are satisfied by the hypothesis $\nabla E(w^*) = \Theta^n$ and the assumption of positive definitiveness of the Hessian at w^*. Finding such a point is equivalent to solving iteratively, in parallel, System (3) by applying the nonlinear Jacobi process and employing any one–dimensional method for the subminimization process.

Consider the decomposition of $H(w^*)$ into its diagonal, strictly lower–triangular and strictly upper–triangular parts:

$$H(w^*) = D(w^*) - L(w^*) - L^\top(w^*). \qquad (7)$$

Since, $H(w^*)$ is symmetric and positive definite, then $D(w^*)$ is positive definite [18]. Moreover, since $H(w^*)$ has the property A^π, the eigenvalues of

$$\Phi(w^*) = D(w^*)^{-1}\left[L(w^*) + L^\top(w^*)\right],$$

are real and $\rho(\Phi(w^*)) < 1$ [1] (where $\rho(A)$ indicates the spectral radius of the matrix A); then there exists an open ball $\mathcal{S} = \mathcal{S}(w^*, r)$ in $\mathcal{S}_0$, such that, for any initial weight vector $w^0 \in \mathcal{S}$, there is a sequence $\{w^k\}_{k=0}^{\infty} \subset \mathcal{S}$ which satisfies the nonlinear Jacobi process such that $\lim_{k\rightarrow\infty} w^k = w^*$ [8]. Thus the Theorem is proved.

Remark 1: The matrix A has the property A^π if A can be permuted by $PAP^\top$ into a form that can be partitioned into block–tridiagonal form [1]. For an algorithm which transforms a symmetric matrix to tridiagonal form see [15, p.335].

The nonlinear SOR scheme and its convergence

Starting from an arbitrary initial weight vector $w^0 \in \mathcal{D}$, the nonlinear SOR scheme subminimizes at the kth epoch the function:

$$E(w_1^{k+1}, \ldots, w_{i-1}^{k+1}, w_i, w_{i+1}^k, \ldots, w_n^k), \qquad (8)$$

along the ith direction and obtain the corresponding subminimizer $\hat{w}_i$. Again in this case, the ith weight is updated according to Eq. (6). The main difference from the corresponding Jacobi's approach is that the adaptation of the weight w_i at the kth epoch takes into consideration all the previously updated weights of the same epoch. The corresponding convergence result of the nonlinear SOR scheme is as follows:

Theorem 2: Let $E: \mathcal{D} \subset \mathbb{R}^n \rightarrow \mathbb{R}$ be twice continuously differentiable on an open neighborhood $\mathcal{S}_0 \subset \mathcal{D}$ of a local minimizer $w^* \in \mathcal{D}$. Then there exists an open ball $\mathcal{S} = \mathcal{S}(w^*, r)$ in $\mathcal{S}_0$ such that the sequence $\{w^k\}$ generated by the nonlinear SOR scheme converges to the point w^*.

Proof. Since w^* is a local minimizer of E, $\nabla E(w^*) = 0$ and $H(w^*)$ is positive definite. Clearly, finding such a point is equivalent to obtaining the corresponding solution $w^* \in \mathcal{D}$ of System (3) by applying the nonlinear SOR. Suppose now that:

$$\Phi_\tau(w^*) = [D(w^*) - \tau L(w^*)]^{-1}\left[(1-\tau)D(w^*) + \tau L^\top(w^*)\right],$$

for $\tau \in (0, 2)$ where D and L are defined as in Eq. (7). Now, since $H(w^*)$ is symmetric and positive definite then, $D(w^*)$ is nonsingular [18, p.80]. Now, by virtue of Ostrowski Theorem [18, p.77], $\rho\left(\Phi_\tau(w^*)\right) < 1$ for any $\tau \in (0, 2)$ and therefore, by the nonlinear SOR theorem [8, p.326], there exists an open ball $\mathcal{S} = \mathcal{S}(w^*, r)$ in $\mathcal{S}_0$, such that, for any $w^0 \in \mathcal{S}$, there is a unique sequence $\{w^k\} \subset \mathcal{S}$ that satisfies the nonlinear SOR prescription such that $\lim_{k\rightarrow\infty} w^k = w^*$. Thus the Theorem is proved.

Sign-based Training Algorithms

Below we synthesize sign-based adaptive learning rate algorithms. These algorithms employ a different learn-

ing rate for each weight based on one–dimensional methods. We propose the following training methods named *m–step Jacobi bisection* and *m–step SOR bisection* which update the weights utilizing the iterative scheme (6). The first method computes $\hat{w}_i$ of (6) in the interval (a_i, b_i) (see below Relations (12)–(13)) using the sequence $\{\hat{w}_i^p\}_{p=0}^{\infty}$ defined as follows [20, 21]:

$$\hat{w}_i^{p+1} = \hat{w}_i^p + C\,\mathrm{sgn}\partial_i E(z_i^p)\,/2^{p+1}, \quad p = 0, 1, \ldots, \quad (9)$$

where $C = \mathrm{sgn}\partial_i E(z_i^0)h_i$,

$$z_i^p = (w_1^k, \ldots, w_{i-1}^k, \hat{w}_i^p, w_{i+1}^k, \ldots, w_n^k), \quad (10)$$

$$z_i^0 = (w_1^k, \ldots, w_{i-1}^k, a_i, w_{i+1}^k, \ldots, w_n^k), \quad (11)$$

$\hat{w}_i^0 = a_i$, $h_i = b_i - a_i$ and where sgn defines the well known triple valued sign function. The iterations (9) converge to $\hat{w}_i \in (a_i, b_i)$ if for some w_i^p, $p = 1, 2, \ldots$, the following holds:

$$\mathrm{sgn}\partial_i E(w^0)\,\mathrm{sgn}\partial_i E(w^p) = -1.$$

To ensure that $\hat{w}_i$ is a subminimizer along the ith weight direction, we choose the endpoints a_i and b_i in such a way that at the left endpoint a_i, the ith component of the gradient vector has negative value, or, at the right endpoint b_i, the ith component of the gradient vector has positive value. To fulfill this condition the endpoints are chosen by the following relation:

$$a_i = w_i^k - \frac{1}{2}\Big\{1 + \mathrm{sgn}\,\partial_i E(w^k)\Big\}h_i - \mathrm{sgn}\,\partial_i E(w^k)\varepsilon, \quad (12)$$

$$b_i = a_i + h_i, \quad (13)$$

where ε is a small positive number. The maximum number m of the iterations of the sequence (9) which are required to obtain an approximate minimizer w_i^* at each epoch along the ith direction is related to a predefined accuracy $\delta \in (0, 1)$ and it is given by:

$$m = \lceil \log_2(h_i \delta^{-1}) \rceil, \quad (14)$$

where $\lceil \cdot \rceil$, defines the ceiling function.

Thus, in the weight update equation (6) the parameter $\hat{w}_i$ is the approximation of the subminimizer obtained by (9).

The second proposed m–step SOR bisection method is similar to m–step Jacobi bisection method and it is formulated by replacing Relations (10) and (11) with the following ones:

$$z_i^p = (w_1^{k+1}, \ldots, w_{i-1}^{k+1}, \hat{w}_i^p, w_{i+1}^k, \ldots, w_n^k), \quad (15)$$

$$z_i^0 = (w_1^{k+1}, \ldots, w_{i-1}^{k+1}, a_i, w_{i+1}^k, \ldots, w_n^k), \quad (16)$$

To alleviate the calculations of the signs of the gradient values in (9) we propose the alternative sequence $\{\hat{w}_i^p\}_{p=1}^{\infty}$:

$$\hat{w}_i^{p+1} = \hat{w}_i^p + C\,\mathrm{sgn}\Big(E(z_i^p) - E(z_i^0)\Big)\,/2^{p+1}, \quad (17)$$

where z_i^p, z_0^p are computed by means of (10)–(11) or (15)–(16), $C = \mathrm{sgn}\big(E(a_i) - E(w_i^k)\big)h_i$, $h_i = b_i - a_i$, and $z_i^1 = a_i + h_i/2$.

Suppose that the sequence (17) converges to a $\hat{w}_i^\alpha$ then the final approximation to $\hat{w}_i$ is given by:

$$\hat{w}_i = \hat{w}_i^0 + \gamma_k\left(\hat{w}_i^\alpha - \hat{w}_i^0\right), \quad (18)$$

for some relaxation factor γ_k.

Also, the sign of the gradient in Relation (12) can be computed as follows (cf. (1)):

$$\mathrm{sgn}\partial_i E(w^k) = \mathrm{sgn}\Big(E(w^k + \varepsilon e_i) - E(w^k)\Big).$$

The signs of the error function values or its gradient values in the iterative schemes (9), (17) can be achieved by solely comparing the relative sizes of the error function values. Thus, the corresponding methods based on these frameworks are able to cope with imprecisions or noisy error function values.

Obviously the above procedures handle n–dimensional problems using reduction to simpler one–dimensional equations. For convergence properties of the above methods see [22]. It is evident from the sequence (17) that the only computable information required by these methods is the algebraic signs. So, rounding and quantitative 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 (17) is a global convergence method, it always converges within the given interval, it is optimal [13], 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 δ (cf. Relation (14)).

Simulation results

To evaluate the performance of the algorithms we have tested the sign-based m-step SOR-modified bisection method (Alg-1) against several popular batch training methods. We have also tested the sign-based one-step SOR-Rprop method (Alg-2). This method combines the SOR with the update formula of the Rprop method [10], instead of the bisection. Additionally,

for the (Alg-1) and (Alg-2) methods an Armijo line search [2, 9] has been performed along the direction determined by the weights w^k and w^{k+1} of two consecutive epochs.

Parameter γ in (18) had fixed value $\gamma = 0.5$ for all simulation experiments. In the tables summarizing the simulation results, the reported parameters are: μ_{EFE} the mean number of the error function evaluations; σ_{EFE} the standard deviation of the error function evaluations; μ_{ASE} the mean number of the algebraic signs evaluations; σ_{ASE} the standard deviation of the algebraic sign evaluations; and % the percentage of success.

The XOR problem

The four XOR [12] 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) [6], and the adaptive BP (ABP) [19] 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.

Table 1: *Results for the XOR problem.*

Method	μ_{EFE}	σ_{EFE}	μ_{ASE}	σ_{ASE}	%
BP	561	550	—	—	24
MBP	511	525	—	—	21
ABP	233	332	—	—	28
Alg-1	117	72	171.075	207	40

Next, the performance of the Alg-1 in the presence of measurement noise will be demonstrated. The classification of the XOR patterns 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 2. The results for Alg-1 here are worse than the results of Table 1. This is due to the fact that many times the signs required in Eqs. (9) and (12) are not correct.

Table 2: *Training with imprecise error values.*

Method	μ_{EFE}	σ_{EFE}	μ_{ASE}	σ_{ASE}	%
Alg-1	396	234	4151	1321	30

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 Alg-1 for the XOR problem are shown in Table 3. BP, MBP and ABP have not converged with such large weight variations.

Table 3: *Training with weight variations.*

Method	μ_{EFE}	σ_{EFE}	μ_{ASE}	σ_{ASE}	%
Alg-1	395	252	3967	1181	44

Modeling a system

Next, we report results of the Alg-2, on modeling a system through a set of 20 samples, which are uniformly chosen from the systems step response. A 1-10-1 FNN is chosen in order to use the minimal architecture and reduce memory cost. The goal is an average error $E_{av} < 0.025$ and 1000 different initial weight vectors have been tested.

Learning is significantly time consuming with the minimal architecture. BP with line search instead of a fixed learning rate, needs, on the average, more than 4×10^6 error function evaluations. BP with a fixed learning rate never found a global minimum due to oscillations; when BP is approaching a global minimum a smaller learning rate is necessary for the algorithm to continue decreasing the error. BPM exhibits the same problematic performance. Results for ABP and Alg-2 are exhibited in Table 4.

Table 4: *Neural model of a system.*

Method	μ_{EFE}	σ_{EFE}	μ_{ASE}	σ_{ASE}	%
ABP	3630102	2979036	—	—	61
Alg-2	1217	980	46995	29715	100

Controlling a lathe cutting process

The cutting tool, driven by a servo–motor is augmented with a force sensor which returns a signal contaminated by tool chatter and unwanted noise to the controller [16, 17] whose object is to maintain a constant force on the tool by varying the material feed rate. The controller generates a signal to the actuator to effect the necessary optimum feed rate in order to assure the desired product quality. Feed rate demand is the input to the device and the cutting force, as measured by the force sensor on the workpiece, constitutes the device output. A 2-2-1 FNN is used for controlling the process. The neural controller is trained using fuzzified

values for the control system error and the error change. The controller provides a correction signal which passes through an integrator to give the control input to the device. In Table 5 we exhibit results regarding the training performance of the neuro-controller starting form 1000 different initial weight vectors.

Table 5: *Lathe cutting process neuro-controller training.*

Method	μ_{EFE}	σ_{EFE}	μ_{ASE}	σ_{ASE}	%
ABP	120	41	—	—	90
Alg-2	16	7	60	10	100

Concluding Remarks

New training methods suitable to work under imprecise conditions are presented. The proposed training algorithms proceed solely with the minimal information of the error function or its gradient, namely the algebraic sign. They take minimization steps in each weight direction. 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. Their convergence has been proved under appropriate assumptions.

The main feature of the proposed methods 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 the error function E. They require only the algebraic signs of the error function and gradient values to be correct.

Preliminary results suggest that our methods cope successfully with on–line training. They may also 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] O. Axelsson, Iterative Solution Methods, Cambridge University Press, New York, (1996).

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

[3] R. Battiti, First– and second–order methods for learning: between steepest descent and Newton's method, Neural Computation, 4, 141–166, (1992).

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

[5] J. L. Holt and J. Hwang, Finite precision error analysis of neural network hardware implementations, IEEE Trans. Comp., 42, 281–290, (1993).

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

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

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

[9] E. Polak, Optimization: Algorithms and Consistent Approximations, Springer-Verlag, New York, (1997).

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

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

[12] 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., 318–362, MIT Press, (1986).

[13] K. Sikorski, Bisection is optimal, Numer. Math., 40, 111–117, (1982).

[14] P.P. Van der Smagt, Minimisation methods for training feedforward neural networks, Neural Networks, 7, 1–11, (1994).

[15] G.W. Stewart, Introduction to Matrix Computations, Academic Press, New York, (1973).

[16] M. Tomizuka and S. Zhang, Modeling and conventional/adaptive PI control of a lathe cutting process, Transactions ASME, vol. 110, pp. 305–354, (1988).

[17] G. Tsitouras and R. E. King, Rule-based neural control of mechatronic systems, Int. J. of Intelligent Mechatronics, vol. 2, pp. 1–11, (1997).

[18] R. Varga, Matrix Iterative Analysis, Prentice–Hall, Inc., Englewood Cliffs, New Jersey, (1962).

[19] 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., 59, 257–263, (1988).

[20] M.N. Vrahatis, Solving systems of nonlinear equations using the nonzero value of the topological degree, ACM Trans. Math. Software, 14, 312–329, (1988).

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

[22] M.N. Vrahatis, G.S. Androulakis, and G.E. Manoussakis, A new unconstrained optimization method for imprecise function values, J. Math. Anal. Appl., 197, 586–607, (1996).

[23] J. Wray and G.G.R. Green, Neural networks, approximation theory and finite precision computation, Neural Networks, 8, 31–37, (1995).