ISCAS 2000 - IEEE International Symposium on Circuits and Systems, May 28-31, 2000, Geneva, Switzerland

On the Composite Squared-Error Algorithm for Neural Networks

Sergio L. Netto and Marcello L. R. de Campos

Programa de Engenharia Elétrica/COPPE, DEL/EE
Universidade Federal do Rio de Janeiro
PO Box 68504, Rio de Janeiro, RJ, 21945-970, Brazil

Abstract

The composite squar ed-err or(CSE) algorithm results from the combination of the backpropagation algorithm with the pseudo-linear algorithm of Scaler o and T epedelenliqlu. It is observe dthat the CSE algorithm is able to avoid suboptimal solutions and associated saddle points, thus achieving lower values of the error function than the pseudo-linear algorithm, in fewer iterations than the backpropagation algorithm. In this paper, we investigate the implementation of the CSE algorithm with the concepts of momentum gain, time-varying learning rate, and time-varying combining factor. It is verifie dthat these features can improve the overall properties of the CSE convergence pro cess.

1 Introduction

Neural networks hav e been used as an efficient tool for solving a wide v ariet y of problems in signal processing. Due to its inherent simplicity and effectiveness, the backpropagation (BP) algorithm has become standard for training feedforward neural netw orks [1]. Despite its relative success, the learning speed of the BP algorithm can often be unsatisfactory. Such characteristic is directly associated to the intrinsic minimization of a nonquadratic error function follo wing a steepest-descent path.

In this article, w e explore the general capabilities of the composite squared-error (CSE) algorithm, namely faster con vergencespeed than the BP algorithm and low er steady-state mean squared-error than the pseudo-linear algorithm [2]. We consider the CSE algorithm with momentum gain, time-varying learning rate, and time-varying combining factor. It is verified ho w these features can affect the o verall con vergence process of the CSE algorithm.

2 Backpropagation Algorithm

The backpropagation (BP) algorithm [1] is based on a gradient-type method to update the weigh ts of a neural net w ork b y minimizing the mean-squared value of the output error, $e_{p,L,k}$, betw een a target (desired) signal, $d_{p,L,k}$, and the signal at the output layer of the net w ork, $x_{p,L,k}$, for the corresponding training-pattern (input) signal, $x_{p,0,k}$ (see Fig. 1). Such error is used to update the weights of the output lay er and it is propagated bac k (thus the name of the algorithm) to the hidden lay ers of the netw ork to update the corresponding weights.

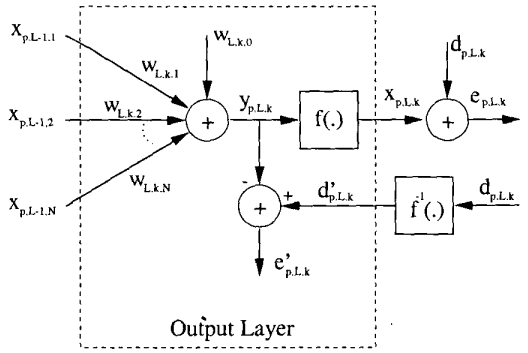

Figure 1: Modified neuron structure for the output lay er.

F or the BP algorithm, the learning process is as follows: at the n-th iteration and for the p-th pattern, the i-th weight of the k-th neuron of the L-th (output) lay er is updated as

$$w(n+1)_{L,k,i} = w(n)_{L,k,i} + \mu e_{p,L,k} x_{p,L-1,i} \qquad (1)$$

where μ is the step-size (learning rate), $x_{p,L,i}$ denotes the output signal of the i-th neuron of the L-th la y er. The output error is then formed as

$$e_{p,L,k} = f'(y_{p,L,k})\,[f(d_{p,L,k}) - x_{p,L,k}] \qquad (2)$$

where $y_{p,L,k}$ denotes the signal at the output of the summation node, and $f'(\cdot)$ is the first derivative of $f(\cdot)$. For the j-th (hidden) layer, the i-th weight is updated as

$$w(n+1)_{j,k,i} = w(n)_{j,k,i} + \mu e_{p,j,k} x_{p,j-1,i} \quad (3)$$

with the respective error calculated as

$$e_{p,j,k} = f'(y_{p,j,k}) \sum_{i=1}^{k} e_{p,j+1,i} w_{j+1,i,k} \quad (4)$$

where $w_{j+1,i,k}$ refers to the k-th weight of i-th neuron of the $(j+1)$-th layer.

3 Pseudo-Linear Algorithm

The nonquadratic objective function minimized by the BP algorithm arises due to the nonlinear operator, $f(\cdot)$, placed after each summation node. This nonlinearity may generate local minima and associated saddle points that may cause slow convergence or even convergence to a local minimum. As shown in Fig. 1, by means of using the inverse of the nonlinear operator, an alternative pseudo-linear error, $e'_{p,L,k}$, may be constructed yielding a distinct objective function to be minimized. The resulting algorithm, hereby referred to as the pseudo-linear (PL) algorithm, has the output-layer weights updated as [2]

$$w(n+1)_{L,k,i} = w(n)_{L,k,i} + \mu e'_{p,L,k} x_{p,L-1,i} \quad (5)$$

The remaining (hidden-layer) weights are updated in the same fashion as in the BP algorithm, i. e., according to (3) and (4).

Although one may argue that the error minimized is still nonlinear with respect to the weights in the hidden layers, several simulations in [2] have shown significant improvements in the convergence speed of the PL algorithm when compared to the BP algorithm. This suggests that the influence of the hidden layers on the shape of the objective function minimized by the network is not as crucial as the influence of the output layer.

In [2], the authors also claim that once the error at the output of the neural network has been reduced to zero, the modified structure has converged to the global minimum of the corresponding BP error function. However, zero output error may not be achievable due to noise, incorrect modeling, or too-short training periods. In such cases, the PL solution obtained after completing the training period is biased with respect to the BP optimal solution. In fact, an optimal mean-squared pseudo-linearized error, $e'_{p,L,k}$, do not correspond in general to an optimal mean-squared output error, $e_{p,L,k}$.

4 CSE Algorithm

The CSE algorithm [3]–[5], originally developed in the framework of adaptive IIR filtering, was first applied to neural networks in [6]. In the proposed method, a combination of the BP and PL algorithms is used to update the weights of the output layer, that is

$$w(n+1)_{L,k,i} = w(n)_{L,k,i} + \mu [\gamma \Delta_l + (1-\gamma)\Delta_{nl}] \quad (6)$$

where

$$\Delta_l = e'_{p,L,k} x_{p,L-1,i} \quad (7)$$

$$\Delta_{nl} = e_{p,L,k} x_{p,L-1,i} \quad (8)$$

All other layers are updated as in the standard BP algorithm. In this composite algorithm, γ is the combination factor that controls the overall convergence process. A good strategy is to start with $\gamma = 1.0$, thus forcing the CSE algorithm to behave like the PL algorithm at the beginning of the learning process, and then switch the value of γ to 0.0, to achieve convergence to the BP optimal solution, as desired. Switch from one updating scheme to the other may be abrupt, resulting in two distinct phases, or gradual as suggested in [6]. To understand better the convergence process of the CSE algorithm, consider the following example.

Example 1: In this example, a function defined by

$$P = \begin{bmatrix} -6.1 \\ -6.0 \\ -4.1 \\ -4.0 \\ +4.0 \\ +4.1 \\ +6.0 \\ +6.1 \end{bmatrix}; \quad T = \begin{bmatrix} 0.0 \\ 0.0 \\ 0.97 \\ 0.99 \\ 0.01 \\ 0.03 \\ 1.0 \\ 1.0 \end{bmatrix} \quad (9)$$

where P defines eight inputs and T characterizes the associated targets, was approximated by a neural network with 1 layer, 1 neuron, and 1 weight and bias parameters. For this 1-layer case, $f(.)$ was implemented by a sigmoid function with output in the range $[0, 1]$. Convergence speed was optimized for all distinct values of γ, resulting in $\mu = 0.005$ when $\gamma = 1.0$, and $\mu = 0.1$ when $\gamma = 0.0$.

Fig. 2 shows the associated error function for $\gamma = 1.0$, $\gamma = 0.01$, $\gamma = 0.002$, and $\gamma = 0.0$, respectively. Notice how this function starts quadratic when $\gamma = 1.0$ and becomes highly irregular when $\gamma \approx 0.0$.

The learning MSE process for the BP, PL, and CSE algorithms are shown in Fig. 3. Notice that when $\gamma = 1.0$ (PL algorithm), convergence is fast but biased, while for $\gamma = 0.0$ (BP algorithm), convergence is slow but unbiased. An excellent compromise is achieved

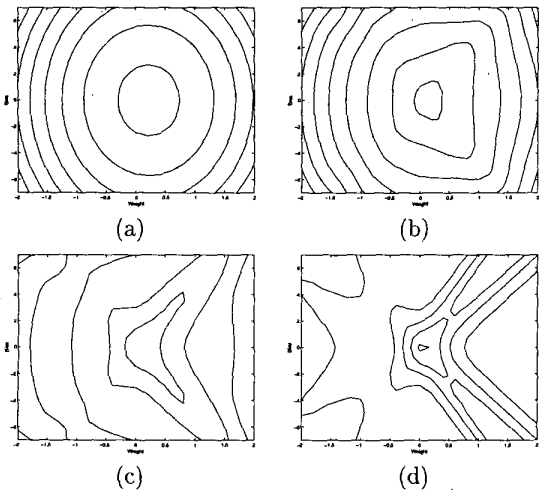

Figure 2: Example 1 - Error surfaces for distinct values of the combination factor: (a) $\gamma = 1.0$; $\gamma = 0.01$; $\gamma = 0.002$; (d) $\gamma = 0.0$.

with the CSE algorithm when the value of γ is switched from $\gamma = 1.0$ to $\gamma = 0.0$ at the 50th iteration (determined by heuristics), resulting in a fast and unbiased learning process.

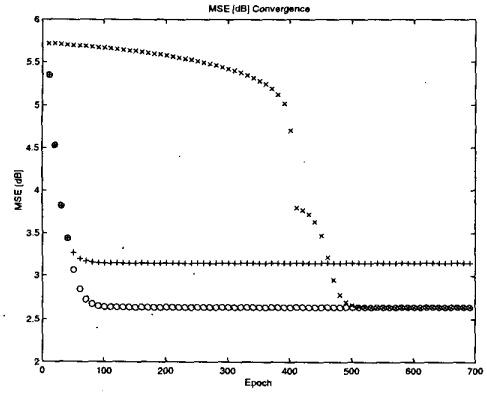

Figure 3: Example 1 - MSE [dB] convergence: BP ('x'), PL ('+'), and CSE ('o') algorithms.

It is interesting to notice that if the function $f(.)$ in the output layer is linear, then both the PL and CSE algorithms become identical to the standard BP algorithm. In most practical classification problems, however, using a linear $f(.)$ is not recommended [1]. Therefore, in such cases, the CSE algorithm becomes a good alternative to the BP and PL algorithms.

5 Extended CSE Algorithm

5.1 Momentum Gain

As demonstrated in [7], using the concept of momentum gain to update the neural network allows its learning process to respond not only to the local gradient, but also to the surroundings of the error surface. Incorporating such concept into the CSE algorithm results into an updating procedure described by

$$w(n+1)_{L,k,i} = w(n)_{L,k,i} + (1-\eta)\mu\left[\gamma\Delta_l + (1-\gamma)\Delta_{nl}\right] + \eta\left[w(n)_{L,k,i} - w(n)_{L,k,i}\right] \quad (10)$$

with Δ_l and Δ_{nl} as in (7) and (8), respectively, and where $0 \leq \eta \leq 1$ is the momentum gain that controls how much of the previous weight change affects the current weight change. Typically such constant is set to $\eta = 0.95$ [7].

5.2 Time-Varying Learning Gain

It has been verified that convergence of adaptation algorithms can be greatly accelerated by using time-varying learning rate. Such feature can be implemented in the CSE algorithm based on heuristics, as suggested in [1], or following some specific rule, as proposed here. Following the approach in [3] for the CSE algorithm applied to adaptive IIR filters, a time-varying learning rate for the j-th layer of the neural network may be of the form

$$\mu_j(n) = \frac{\alpha}{\sum_{p=1}^{size(P)} x_{p,j-1,i}^2} \quad (11)$$

where $0 \leq \alpha \leq 1$. Equation (11) shows that the learning rate μ should be specific for each layer j, and its value is normalized by the total energy of the input vector of such layer. The parameter α is introduced to take into account the stochastic nature of the application at hand. Details of the derivation of Equation (11), although in the adaptive filtering framework, are found in [3].

5.3 Time-Varying Combination Factor

The major capability of the CSE algorithm is its ability to combine the learning processes of the BP and PL algorithms. Such feature is controlled by a single parameter γ. Using a time-varying combination parameter wisely is therefore adamant for a fast and efficient learning process. The difficulty of such scheme, however, lies on determining the precise rate in which the value of γ should change from 1.0 to 0.0.

Here, we propose to monitor the learning process for $\gamma = 1.0$. Thus, when some slowing down is detected, that is, when the value of the error does not change much in a given number of iterations (say 10) that should indicate that the PL is close to its solution, and then γ should be made equal zero. Hence,

$$\gamma(n) = \begin{cases} 1.0, & \text{for } n < \tau \\ 0.0, & \text{for } n \geq \tau \end{cases} \quad (12)$$

where τ is the smallest n such that

$$\frac{\sum_{k=1}^{K} [e_{p,L,k}(n) - e_{p,L,k}(n-10)]}{\sum_{k=1}^{K} e_{p,L,k}(n-10)} \leq \epsilon \quad (13)$$

and ϵ is a small positiv enumber. Experiments have shown that $\epsilon = 0.01$, for instance, yields very good results in terms of the overall CSE convergence speed.

6 Simulation Results

Example 2: In this example a neural network consisting of 2 neurons in the first layer and 1 neuron in the output layer was used to implement the classical problem where the output signal must be an XOR operation onto the inputs. The sigmoid function was used as $f(.)$ in both layers. Adaptation parameters were set as $\mu = 0.3$. Fig. 4 shows the MSE for 50000 epochs for the BP, PL, and extended CSE algorithms. From this figure, we can clearly verify the superior performance of the proposed method. Notice how the BP algorithm remained trapped for a long time (about 26000 epochs) in a flat portion of the error surface, while the extended CSE algorithm was able to avoid such behavior with the help of the PL algorithm in the beginning of the convergence process.

7 Conclusions

In this article, an extended version of the composite squared-error (CSE) algorithm for training neural networks was introduced. The CSE algorithm combines the good properties of the backpropagation algorithm (namely, global optimal solution), and of the pseudo-linear algorithm of [2] (faster convergence, in general). In several simulations performed, the CSE algorithm regularly converged to the global minimum with a speed superior to the one of the backpropagational-gorithm. Extensions of the CSE algorithm to incorporate features such as momentum gain, and time-varying learning and combination factors were proposed. The extended CSE algorithm has the additional advantage of requiring less heuristics to be implemented in practice than the standard version of the CSE algorithm given in [6].

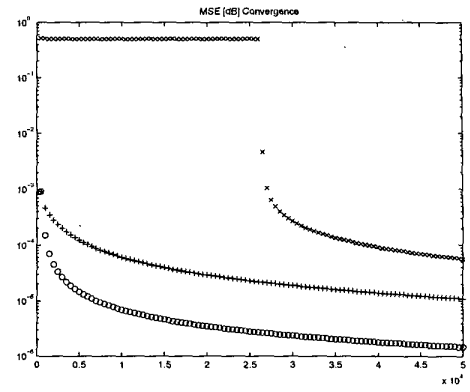

Figure 4: Example 2 - MSE [dB] convergence: BP ('x'), PL ('+'), and Extended-CSE ('o') algorithms.

References

[1] S. Haykin, *Neural Networks: A Comprehensive Foundation*, Englewood Cliffs:NJ, Prentice-Hall, 3rd ed., 1997.

[2] R. S. Scalero and N. Tepedelenlioglu, "A fast new algorithm for training feedforward neural networks," *IEEE Trans. Signal Processing*, vol. 40, pp. 202–210, Jan. 1992.

[3] S. L. Netto, *On Algorithms, Structures, and Implementations of Adaptive IIR Filters*, PhD Dissertation, Univ. of Victoria, Canada, 1996.

[4] S. L. Netto and P. Agathoklis, "A new composite adaptive IIR algorithm," *Proc. 28th. Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, CA, pp. 1506–1510, Oct./Nov. 1994.

[5] S. L. Netto and P. Agathoklis, "On the composite squared error algorithm for adaptive IIR filters," *Proc. 31st. Asilomar Conference on Signals, Systems, and Computers*, PacificGrove, CA, vol. 2, pp. 1683–1687, Nov. 1997.

[6] D. Gonzaga, M. L.R. de Campos, and S. L. Netto, "Composite squared-error algorithm for training feedforward neural networks," *Proc. IEEE Symposium on Advances in Digital Filtering and Signal Processing* Victoria, Canada, pp. 116–120, June 1998.

[7] H. Demuth and M. Beale, *Neural Network Toolbox User's Guide*, MathWorks Inc., 1994.