A Constrained Optimization Algorithm for Training Locally Recurrent Globally Feedforward Neural Networks

P.A. Mastorocostas
Department of Informatics & Communications
Technological Educational Institute of Serres
62124, Serres, Greece
E-mail: mast@teiser.gr

Abstract- This paper presents a novel learning algorithm for training locally recurrent globally feedforward neural networks. The training task is formulated as a constrained optimization problem, whose objective is twofold: (i) minimization of an error measure, leading to successful approximation of the input/output mapping and (ii) optimization of an additional functional, which aims at accelerating the learning process. Simulation results on a benchmark identification problem demonstrate that, compared to other learning schemes, the proposed algorithm has enhanced qualities, including improved speed of convergence, accuracy and robustness.

I. INTRODUCTION

During the last fifteen years there has been considerable research interest for dynamic neural models. Because of their enhanced prediction qualities they have been applied to several problems including adaptive control [1] and speech recognition [2]. A number of researchers have experimented with a special recurrent architecture, called locally recurrent globally feedforward network (LRGF). The models belonging to this class preserve the feedforward structure of neural networks, while local recurrence is introduced in the synapses and in the neurons through finite impulse response (FIR) or infinite impulse response (IIR) filters. In [3] a local synapse feedback is suggested where synapses are implemented using IIR filters.

An interesting class includes those LRGF models that contain dynamic neurons with local output feedback, as suggested by Frasconi, Gori and Soda in [2]. The feedback signal is taken from the output of the activation function and after delayed for several times it is fed as input to the neuron. The rest of neuron inputs are directly taken through synaptic weights from the outputs of the previous layer neurons. The resulting neuron is referred to as Frasconi-Gori-Soda (FGS) neuron. Frasconi et al demonstrated in [2] that their model is capable of approximating the dynamics of a non-linear system and experimented with speech signal modeling. In [4] FGS neurons were incorporated to fuzzy rules, leading to recurrent fuzzy models. As reported in [4], these models were able to effectively identify complex nonlinear dynamic processes.

Since research has been directed mostly towards developing efficient structures, up till now considerable attention has not been paid to the major influence of the learning algorithm on a training procedure. The Back Propagation Through Time (BPTT, [5]) has been used for training this kind of networks ([2], [3]) due to the fact that it is a well established and easily applicable optimization method. However, the BPTT exhibits certain disadvantages, such as: a) it shows a low speed of convergence, b) most often it becomes trapped to local minima of the error surface.

In view of the above, a novel learning algorithm for training locally recurrent globally feedforward neural networks is presented in this work, based on the concept of constrained optimization. The proposed algorithm, entitled Constrained Optimization Algorithm for Locally Recurrent Globally Feedforward Networks (COA-LRGF), is a robust training scheme, which accelerates learning significantly. Its characteristics are investigated via a standard identification example, where a comparative analysis with Resilient Backpropagation (RPROP, [6]) and BPTT is conducted.

II. THE LRGF NEURAL NETWORK

The LRGF neural network (LRGF-NN) employed in this work is a recurrent neural network with one hidden layer. For sake of simplicity to the derivation of the learning algorithm let us consider the multiple-input single-output case, since generalization to the multiple-input multiple-output network is straightforward. The network is in the form \( m - H - 1 \), with \( m \) and \( H \) being the numbers of neurons in the input and hidden layers, respectively. The neurons of the input layer simply pass the input vector to the hidden layer. The neurons of the hidden and output layers are typical FGS neurons, with the exception that the previous layer’s inputs are entered to the neuron through FIR synapses. Thus, dynamics is introduced through the feedback connections as well as through the delayed outputs of the previous layer.

Let \( \mathbf{u}(k) = [u_1(k), u_2(k), \ldots, u_m(k)]^T \) be the input vector at time \( k \) and \( y(k) \) the network output. The operation of
The LRGF-NN is described by the following set of equations:

\[
O_1^i(k) = f_1 \left( \sum_{j=1}^{m} \sum_{q=0}^{Q_1} w_{1jq}^i u_j(k-q) + \sum_{j=1}^{w_1^2} O_1^j(k-j) + w_1^3 \right) \quad i = 1, \ldots, H
\]

\[
y(k) = f_2 \left( \sum_{j=1}^{H} \sum_{q=0}^{Q_2} w_{2jq}^i O_1^j(k-q) + \sum_{j=1}^{w_2^5} y(k-j) + w_2^6 \right)
\]

where the following notation is used:

- \( f_1, f_2 \) are the neuron activation functions of the hidden and the output layers, respectively. They are both chosen to be the hyperbolic tangent \( \tanh(.) \).
- \( O_1^i(k) \) is the output of the \( i \)-th hidden neuron at time \( k \).
- \( y(k) \) is the output of the network.
- \( O_{y2} \) and \( O_{y3} \) are the time lag orders of the inputs and the local output feedback, respectively, for the hidden layer neurons.
- \( O_{y3} \) and \( O_{y2} \) are the time lag orders of the inputs and the local output feedback, respectively, for the output neuron.
- \( w_{1jq}^i \) and \( w_{2jq}^i \) are FIR and IIR synaptic weights at the hidden layer. \( w_{1jq}^i \) and \( w_{2jq}^i \) are FIR and IIR synaptic weights at the output layer. \( w_1^3 \) and \( w_2^6 \) are bias terms for hidden neurons and the output neuron, respectively.

III. THE COA-LRGF ALGORITHM

A. Definitions and statement of the problem

The COA-LRGF method transforms the learning process to a constrained optimization problem, which has been solved by Bryson and Denham in [7], using methods based on optimal control theory and the calculus of variations. The objective of the learning process is to adjust the network parameters so that a prescribed input/output mapping is captured. Learning is carried out in a parallel mode using a data batch \( \{(u(k), \hat{y}(k)), k = 1, \ldots, k_f\} \) comprising \( k_f \) input-output pairs.

The outputs of the hidden and output layers are considered to be the state variables \( s_i(k) \) and equations (1) form the state equations \( f(k)=0 \). The control vector \( w \) includes the FIR, the IIR, and the bias synaptic weights of the recurrent model:

\[
w = [w_1^1, w_2^2, w_3^3, w_4^4, w_5^5, w_6^6]
\]

The suggested algorithm is an iterative procedure which aims at achieving the following objectives, simultaneously:

1) First, it is desired that an error function, \( E \), should be minimized, in order to perform the input-output mapping successfully. The Mean Squared Error (MSE) is selected here, defined by

\[
E = \frac{1}{k_f} \sum_{k=1}^{k_f} [\hat{y}(k)-y(k)]^2
\]

where \( \hat{y}(k) \) is the desired (actual) output of the system at time step \( k \). Minimization of the above functional is attained iteratively, wherever at each iteration \( E \) is decremented by a small amount \( \delta E \). This change is selected in an adaptive way, so that after a succession of training epochs the accumulated changes lead to the establishment of an accurate input-output representation.

2) Apart from the error measure, additional conditions should be fulfilled at each epoch. In particular, the following should be satisfied:

- The architectural constraints of the network, given by (1), describing the state dynamics for different time steps of the training set \( f(u(k),w) = 0 \), \( k = 1, \ldots, k_f \) (4)
- Maximum change of a nonlinear multivariable function, called the pay-off function \( \Phi \), should be performed at each training epoch. At the end of the learning process \( \Phi \) is optimized. Since the primary scope is to facilitate and accelerate learning, the following pay-off function is selected

\[
\Phi = (w - w_{\text{cur}})^T(w_{\text{cur}} - w_{\text{prev}})
\]

where \( w_{\text{cur}} \), \( w_{\text{prev}} \) are the control vectors at the current and the previous epoch, respectively. Maximization of this pay-off function at each epoch implies that the current and previous weight updates are highly aligned, thus avoiding zig-zag trajectories and improving the convergence speed ([8]).

- At each epoch, the control variables are changed by small amounts \( dw \). An additional condition that the weight changes should satisfy is the following positive quadratic form:

\[
\sum_{i=1}^{n} \frac{(dw_i)^2}{A_i^2} - 1 = dw^T \cdot (\Delta^2)^{-1} \cdot dw - 1 = 0
\]

where \( \Delta_i \) is called maximum parameter change (MPC) for \( w_i \) and \( \Delta = \text{diag}(\Delta_1, \Delta_2, \ldots, \Delta_n) \) is the MPC matrix (\( n \) is the total number of weights). Equation (6) describes a hyper-ellipsoid centered at the point defined by the current control vector, whose axes are the \( \Delta_i \)'s. Thus, at each epoch, search for a new weight value is restricted to a small neighborhood around the current estimate defined in the form of a hyper-ellipsoid. Inclusion of the above functional conditions lead to the establishment of an accurate input-output representation.
is dictated by out intention to facilitate search for optimal solutions within the weight space. Owing to the nonlinearities involved and the dynamic nature of the network, the error function landscape is very complicated, including flat regions and narrow troughs where global minima might be located. In flat regions with small error gradients, large values of the learning rate are required to accelerate learning. On the contrary, in steep regions with large error gradients, the learning rate should take reasonably small values, to avoid missing of the global optimal solutions. The condition imposed on weight changes provides an effective search means where, regardless of the location of the current estimate, the optimal weight changes \( dw \) are decided over the hyper-ellipsoid neighborhood defined by (6).

The perturbations of the control vector induce changes in the state variables \( st \) and the functionals \( E \) and \( \Phi \). Assuming that the MPC’s are small enough, these changes can be approximated by the first differentials \( dE \) and \( d\Phi \). For steepest ascent, we wish to find the optimal parameter changes \( dw \) that maximize \( d\Phi \), for a given value of \( dE \) and \( \Delta \). In that respect, the learning process can be considered as a constrained optimization problem, formulated as: Maximize, with regard to \( w \), the pay-off function, subject to the constraints (1), (6) and \( dE = \delta E \).

B. Derivation of the COA-LRGF algorithm

In order to determine the changes induced in \( \Phi \) and \( E \) for a small perturbation \( dw \) around the current values of the control variables, the quantities

\[
\Phi = \Phi(st,w) + \lambda_{\Phi}^T f(st,w) \quad \text{(7a)}
\]

\[
E = E(st,w) + \lambda_{E}^T f(st,w) \quad \text{(7b)}
\]

are considered, where \( \lambda_{\Phi}^T, \lambda_{E}^T \) are the row matrices of the Lagrange multipliers to be determined. Taking the differentials of (7a) and (7b) we have

\[
\frac{\partial \Phi}{\partial t} + \lambda_{\Phi}^T \frac{\partial f}{\partial t} = \delta \Phi \quad \text{(8a)}
\]

\[
\frac{\partial E}{\partial t} + \lambda_{E}^T \frac{\partial f}{\partial t} = \delta E \quad \text{(8b)}
\]

\( \lambda_{\Phi} \), \( \lambda_{E} \) are chosen such that the dependence of \( \delta t \) on \( \Phi \) is eliminated, and thus the coefficient of \( \delta st \) in (8a) vanishes:

\[
\frac{\partial \Phi}{\partial t} + \lambda_{\Phi}^T \frac{\partial f}{\partial t} \quad \text{(9)}
\]

Since \( \Phi(w) = (w - w_{\text{cur}})^T (w_{\text{cur}} - w_{\text{prev}}) \) does not depend on the state vector, it follows that

\[
\frac{\partial \Phi}{\partial t} = 0
\]

Hence it is derived from (9) that \( \lambda_{\Phi} = 0 \), and (8a) becomes

\[
d\Phi = \frac{\partial \Phi}{\partial w} \cdot dw
\]

Similarly, the Lagrange multipliers \( \lambda_{E} \) are chosen such that the coefficient of \( \delta st \) in (8b) vanishes:

\[
\frac{\partial E}{\partial t} + \lambda_{E}^T \frac{\partial f}{\partial t} = 0
\]

and (8b) is transformed to the following formula:

\[
dE = (\frac{\partial E}{\partial w} + \lambda_{E}^T \frac{\partial f}{\partial w}) \cdot dw = \lambda_{E}^T \frac{\partial f}{\partial w} \cdot dw
\]

The Lagrange multipliers \( \lambda_{E} \) form a \([k_f \cdot (1+H)]\times 1\) vector, defined as follows:

\[
\lambda_{E} = \left[ \lambda_{E}^T(k) \ldots \lambda_{E}^T(k) \right]^T \quad \text{(13)}
\]

where \( \lambda_{E}(k) = \left[ \lambda_{E1}^T(k) \ldots \lambda_{E1}^T(k) \lambda_{E2}^T(k) \right]^T \) \( \text{(14)} \)

After calculations in (11) are conducted, it is derived:

\[
\lambda_{E2}^T(k) = \frac{2}{k_f} \cdot (y(k) - y_d(k) + \sum_{i=1}^{n} \lambda_{E2}^T(k+l) f_i^2(k+l) w_i^2)
\]

\[
\lambda_{E1}^T(k) = \sum_{i=1}^{n} \lambda_{E1}^T(k+l) f_i^2(k+l) w_i^2 + \sum_{q=0}^{n} \lambda_{E2}^T(k+q) f_i^2(k+q) w_i^2 \quad \text{(16a)}
\]

where \( f_i^2(k+q) \) and \( f_i^2(k+l) \) are the derivatives of \( y(k+q) \) and \( O_i^2(k+l) \), with respect to their arguments. Equations (15a) and (16a) are backward difference equations that can be solved for \( k = k_f -1 \ldots 1 \) using the boundary conditions in (15b) and (16b), respectively, for \( k = k_f \).

Defining the intermediate variables

\[
A_{\Phi} = \frac{\partial \Phi}{\partial w} + \lambda_{\Phi}^T \frac{\partial f}{\partial w} , \quad \Lambda_{\Phi} = A_{\Phi} + \lambda_{\Phi}^T \frac{\partial f}{\partial w}
\]

and performing the necessary calculations, it is derived:

\[
d\Phi = A_{\Phi} \cdot dw \quad \text{(17)}
\]

\[
dE = A_{E} \cdot dw \quad \text{(18)}
\]

\[
A_{\Phi} = \lambda_{\Phi} \frac{\partial f}{\partial w} \quad \text{(19)}
\]

\[
\Lambda_{E1} = \sum_{k=1}^{k_f} [\lambda_{E1}^T(k)f_i^2(k)\mu_j(k-q)] \quad \text{(20a)}
\]
\[ A_{E_j}^2 = \sum_{k=1}^{k_L} [A_{E_j}(k)f_j(k)O_j(k-j)], \quad (20b) \]
\[ A_{E_i}^2 = \sum_{k=1}^{k_L} [A_{E_i}(k)f_i(k)] \quad (20c) \]
\[ A_{E_j}^4 = \sum_{k=1}^{k_L} [A_{E_j}(k)f_j^2(k)(k-q)] \quad (20d) \]
\[ A_{E_j}^6 = \sum_{k=1}^{k_L} [A_{E_j}(k)f_j^2(k)] \quad (20f) \]

Considering a linear combination of (17), (18) and (6), it follows that
\[
d \Phi = A_{\Phi} \cdot dw + \nu \cdot [dE - A_{E} \cdot dw] + \mu \cdot [1 - dw^T(\Delta^2)^{-1} dw] \quad (21)\]

where the quantities multiplying \( \nu \) and \( \mu \) are zero, as shown in (18) and (6), respectively.

Taking the second differential of (21), it is derived:
\[
d^2 \Phi = \left( A_{\Phi}^{-1} - 2 \mu (\Delta^2)^{-1}\right) \cdot d^2 w \quad (22)\]

Thus, the maximum of \( d \Phi \) occurs when the coefficient of \( d^2 w \) in (22) vanishes. This will be the case if
\[
A_{\Phi}^{-1} - 2 \mu (\Delta^2)^{-1} = 0 \quad (23)\]

After calculations are conducted, according to [7], the changes in the control variables are given by
\[
dw_i = \pm \Delta_i \cdot \sum_{j=1}^{n} \frac{I_{EE} \cdot \sum_{j=1}^{n} (A_{E_i}(j) \cdot (\Delta_i)_j)^2}{I_{EE} \cdot \sum_{j=1}^{n} (A_{E_i}(j) \cdot (\Delta_i)_j)^2} \quad (24)\]

where
\[
I_{EE} = \sum_{j=1}^{n} (A_{E_i}(j))^2 (\Delta_i)_j^2 \quad (25a) \]
\[
I_{\Phi \Phi} = \sum_{j=1}^{n} (A_{E_i}(j))^2 (\Delta_i)_j^2 \quad (25b) \]
\[
I_{E \Phi} = \sum_{j=1}^{n} A_{E_i} A_{E_j} (\Delta_i)_j^2 \quad (25c) \]

and
\[
\delta E = -\xi \sqrt{I_{EE}} \quad (26)\]

with \( 0 \leq \xi \leq 1 \) (it is chosen 0.9).

The network weights are updated according to the following formula:
\[
dw = w(t+1) - w(t) \quad (27)\]

where \( t \) denotes the training epoch.

As mentioned above, at each epoch the weights vary independently from each other, under the constraints \([dw_i] \leq \Delta_i\). Since the search space is defined in terms of \( \Delta_i \), the MPC’s play an essential role in the learning process and is desired to be adaptable throughout training. Therefore, an adaptation mechanism is employed in this paper, to determine \( \Delta_i \)’s at each epoch. The MPC’s take an initial value \( \Delta_i = \Delta_{0i} \) that is chosen rather moderately. At each epoch the MPC’s are adapted according to the following conditions:

- If \( \frac{\partial E}{\partial w_i} \cdot \frac{\partial E}{\partial w_{i\_prev}} > 0 \) then \( \Delta_i = n^+ \cdot \Delta_i \)
- If \( \frac{\partial E}{\partial w_i} \cdot \frac{\partial E}{\partial w_{i\_prev}} < 0 \) then \( \Delta_i = \max\{n^- \cdot \Delta_i, \Delta_{\min}\} \)

where the increase and attenuation factors are usually set \( n^+ \in [1, 1.3] \) and \( n^- \in [0.5, 0.8] \), and the quantity \( \Delta_{\min} \), a lower bound of the MPC’s, is taken in the range \( 10^{-3} - 10^{-5} \).

At each epoch \( t = 1, 2, \ldots \) the COA-LRGF algorithm proceeds as follows: First, the current values of \( \Delta_i \)’s are derived according to the adaptation schedule described above. As a next step, parameter learning is performed, by calculating the optimal parameter changes through (15)-(27). If the resulting error measure is smaller than a predefined threshold learning is completed, otherwise the whole process is repeated.

IV. SIMULATION RESULTS

In this section, extensive experimentation is carried out in an attempt to demonstrate the learning properties of the COA-LRGF approach and compare its performance to other learning schemes. To this end, a benchmark problem is employed, the identification of a dynamical system, most frequently used in literature for comparing different learning algorithms. The example is taken from [1], where the plant to be identified is governed by the following difference equation
\[
y_p(k+1) = \frac{y_p(k) \cdot y_p(k-1) \cdot y_p(k-2) \cdot u(k-1) \cdot [y_p(k-2) - 1]}{1 + y_p^2(k-1) + y_p^2(k-2)} + \frac{u(k)}{1 + y_p^2(k-1) + y_p^2(k-2)} \quad (28)\]

As it can be seen, the current output of the plant depends on three previous outputs and two previous inputs. A parallel identification scheme is considered, with the input \( u(k) \) being the sole input to the network. The LRGF-NN used comprises eight neurons in the hidden layer, while the rest of the structural characteristics are presented in Table I. The resulting model has 50 weights. As a measure to test the identification performance we use the mean squared error (MSE) criterion.

720
In the training phase, the input \( u(k) \) consists of 400 samples from an independent and identically distributed uniform sequence over \([-2,2]\), and 400 samples from the sinusoidal signal \( 1.05 \sin(\pi k/45) \). In the testing phase, the input \( u(k) \) comprises 1000 samples from the following signal:

\[
\begin{align*}
\sin(\pi k/25) & \quad k < 250 \\
1 & \quad 250 \leq k < 500 \\
-1 & \quad 500 \leq k < 750 \quad (29) \\
0.3 \sin(\pi k/25) + 0.1 \sin(\pi k/32) + \\
0.6 \sin(\pi k/10) & \quad 750 \leq k < 1000
\end{align*}
\]

In order to compare the performance of COA-LRGF to other methods, the BPTT and the RPROP are selected to be the competing rivals. BPTT is selected since it is the most commonly used learning method for recurrent models, while RPROP constitutes one of the best performing first order learning methods for static neural systems and has been successfully applied to recurrent neural networks as well ([9]).

As a first stage in the experimentation, the training parameters of the three competing algorithms are chosen as described in the sequel. For each method, several runs are performed with different parameter combinations. In the above runs the network starts from the same initial weights. Then, the parameter combination that exhibits the fastest convergence and low values of the error function is selected. Following this approach, we are led to \( \Delta_0 = 0.1 \), \( \Delta_{\text{min}} = 10^{-4} \), \( \Delta_{\text{max}} = 0.5 \), \( \eta^+ = 1.1 \) and \( \eta^- = 0.8 \) for RPROP, and a learning rate of \( \mu = 0.002 \) for the BPTT algorithm. The selected learning parameters for COA-LRGF are hosted in Table I.

In the following, the above learning parameters are fixed, and a series of 100 independent trials are attempted, each with different weight initializations. Particularly, the weights are randomly selected within the range \([-0.5,0.5]\). For each particular trial and for fair comparison, the same weight initializations are used for all competing methods.

We store the number of iterations required to reach some specific error levels and calculate the average results for each method. A maximum number of 1000 training epochs is considered for all methods. A training run is regarded as successful, provided that the method succeeds to reach the given error level within 1000 epochs; otherwise, it is declared that it failed to converge.

In order to arrive to accurate conclusions, the simulation results for COA-LRGF, RPROP and BPTT methods are presented in Table II, derived on a statistical basis. As a means for measuring the speed of convergence and the robustness of the algorithms, the average and the standard deviation of the iterations required to attain a MSE level for the testing set are introduced. The MSE level is set to 0.005. The number of trials succeeded to reach the specified error level is also included. Additionally, the attained MSE after 200 epochs is presented in Table II as well.

<table>
<thead>
<tr>
<th>MSE after 2000 epochs</th>
</tr>
</thead>
<tbody>
<tr>
<td>COA-LRGF</td>
</tr>
<tr>
<td>Average</td>
</tr>
<tr>
<td>Std. dev.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Number of epochs for a MSE value of 0.005 to be attained</th>
</tr>
</thead>
<tbody>
<tr>
<td>COA-LRGF</td>
</tr>
<tr>
<td>Converg. rate</td>
</tr>
<tr>
<td>Average</td>
</tr>
<tr>
<td>Std. dev.</td>
</tr>
</tbody>
</table>

From the results in Table II it can be concluded that the COA-LRGF algorithm outperforms RPROP significantly, while BPTT exhibits the worst performance by far. In particular:

- The proposed method needs 60% of the epochs required by RPROP to attain a MSE value of 0.005, while it converges to a considerably lower MSE after 200 epochs compared both to RPROP (33% lower) and BPTT (3.5 times lower). This implies that COA-LRGF provides considerably faster convergence as contrasted to RPROP and BPTT.

- The standard deviation of iterations required by RPROP is at least two times larger that the one exhibited by COA-LRGF. This observation suggests that COA-LRGF is a much more robust algorithm, with its performance having limited dependence on initial conditions and weight initializations.

- It should be noted that, for the COA-LRGF method 91% of the trials converged successfully to an error level of 0.005. On the contrary, only 73% of the trials conducted with RPROP converged successfully. Finally, a remark should be made for the inefficiency of BPTT, since it failed to converge to the demanding prescribed MSE in all trials.

V. CONCLUSIONS

A new learning algorithm for training locally recurrent globally feedforward neural networks has been proposed, entitled Constrained Optimization Algorithm for Locally

### Table I

<table>
<thead>
<tr>
<th>LRGF-NN structure</th>
<th>( \theta_1 )</th>
<th>( \theta_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( n^+ )</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>( n^- )</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>( \Delta_{\text{min}} )</td>
<td>1E-4</td>
<td>0.5</td>
</tr>
<tr>
<td>( \Delta_0 )</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

### Table II

<table>
<thead>
<tr>
<th>COMPARATIVE ANALYSIS</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSE after 2000 epochs</td>
</tr>
<tr>
<td>-----------------------</td>
</tr>
<tr>
<td>COA-LRGF</td>
</tr>
<tr>
<td>Average</td>
</tr>
<tr>
<td>Std. dev.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Number of epochs for a MSE value of 0.005 to be attained</th>
</tr>
</thead>
<tbody>
<tr>
<td>COA-LRGF</td>
</tr>
<tr>
<td>Converg. rate</td>
</tr>
<tr>
<td>Average</td>
</tr>
<tr>
<td>Std. dev.</td>
</tr>
</tbody>
</table>
Recurrent Globally Feedforward networks (COA-LRGF). The method has transformed the learning process to a constrained optimization problem, where minimization of the error function is performed under the architectural constraints of the network. Moreover, an additional functional, which aims at accelerating learning, is simultaneously optimized. The proposed learning scheme has been applied to an identification problem and a comparative analysis with Resilient Backpropagation and Backpropagation Through Time has been conducted, underlining the effectiveness of COA-LRGF.

ACKNOWLEDGMENT

This work was supported in part by the European research project Archimedes II.

REFERENCES


