Contemporary Engineering Sciences, Vol. 11, 2018, no. 1, 1 - 13
HIKARI Ltd, www.m-hikari.com
https://doi.org/10.12988/ces.2018.711194

A Hybrid Learning Algorithm for Locally Recurrent Neural Networks

Dimitris Varsamis and Evangelos Outsios

Department of Informatics Engineering
Technological Educational Institute of Central Macedonia - Serres
62124, Serres, Greece

Paris Mastorocostas

Department of Computer Systems Engineering,
Piraeus University of Applied Sciences,
12244, Egaleo, Greece

Abstract

In this work a fast and efficient training method for block-diagonal recurrent neural networks is proposed. The method modifies and extends the Simulated Annealing RPROP algorithm, originally developed for static models, by taking into consideration the architectural characteristics and the temporal nature of this category of recurrent neural models. The performance of the proposed algorithm is evaluated through a comparative analysis with a series of algorithms and recurrent models.

Keywords: block-diagonal recurrent neural network, internal feedback, resilient back-propagation, simulated annealing, ordered derivatives

1 Introduction

Recurrent neural networks has become a popular research field of Computational Intelligence during the last. Due to their temporal capabilities, they

have been extensively employed in real-world applications like system identification and pattern recognition. The locally recurrent neural networks with internal feedback connections constitute a special subclass, where the feedback interlinks are limited exclusively between neighbouring neurons. Thus, these neural models have significantly reduced complexity with respect to fully recurrent networks [13].
A special subclass of locally recurrent networks is the Diagonal Recurrent Neural Network (DRNN) [2], where there are no interlinks among neurons in the hidden layer. A modified DRNN is the Block-Diagonal Recurrent Neural Network (BDRNN) [10], where dynamics is introduced between pairs of neurons in the hidden layer. The BDRNNs have been proved to be an efficient modelling tool [4], [5], [11].
Due to the temporal relations of BDRNN, a parameter optimization method should unfold in time. Most of the times these temporal relations are neglected and parameter learning is attempted by considering a few previous time steps. The Back Propagation Through Time algorithm (BPTT) [7] is the most common algorithm for training BDRNNs. However, the BPTT exhibits two major disadvantages: (a) it shows a low speed of convergence, (b) most often it becomes trapped to local minima of the error surface.
In order to overcome the above failings of gradient-based methods like BPTT, the Resilient Propagation algorithm (RPROP, [8]) has been proved to be one of the best performing learning methods for static neural networks [1]. However, in RPROP the problem of poor convergence to local minima is not fully eliminated. Hence, in an attempt to alleviate this drawback, a hybrid scheme combining the global search technique of Simulated Annealing (SA) and RPROP was introduced in [12]. The resulted algorithm, named SARPROP, was proved to be an efficient learning method for static neural networks.
Stem from the above developments in training static neural models, this work proposes an extension of the standard SARPROP method that takes into consideration the temporal relations existing in a locally recurrent neural networks. Since the algorithm is adapted to the special features of BDRNN, it is entitled Hybrid Learning Algorithm for BDRNNs (HLA-BDRNN). The rest of this paper is organized as follows: In Section 2 the structure and characteristics of the BDRNN are illustrated. The learning algorithm is developed in Section 3. In Section 4 a comparative analysis of the proposed method with other learning schemes and recurrent models is conducted. The paper concludes with a brief discussion of the proposed method.

2 The Block-Diagonal Recurrent Neural Network

The BDRNN is a special case of recurrent neural networks. It is not fully connected and it belongs to the class of locally-recurrent-globally-feedforward neural networks [13]. It consists of two layers, where the output layer is static and the hidden layer is dynamic. The hidden layer consists of pairs of neurons (blocks); there are feedback connections between the neurons of each pair, introducing dynamics to the network. For the sake of simplicity, a singleinputsingleoutput BDRNN with four blocks of neurons is shown in Figure 1.

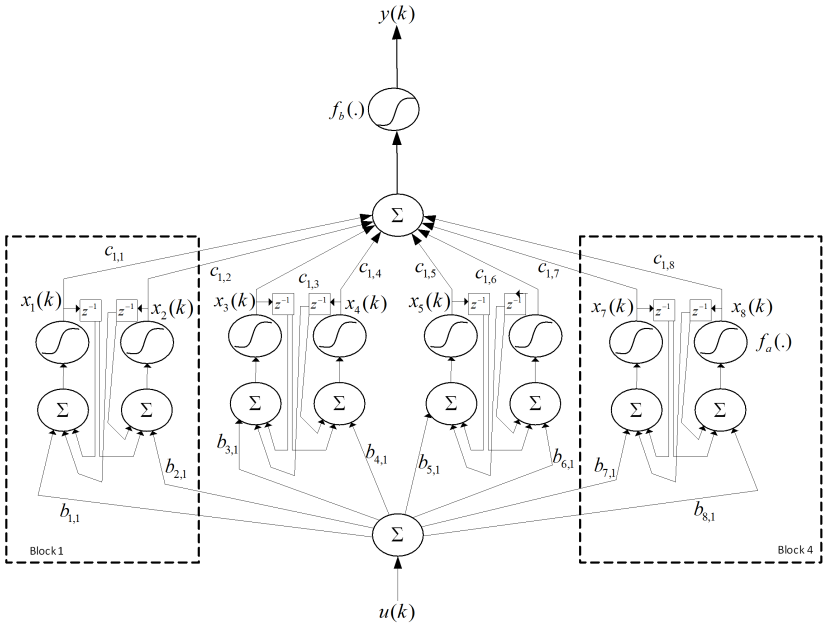

Figure 1: Configuration of BDRNN with four blocks of neurons in the hidden layer

A BDRNN with m inputs, r outputs, and N neurons at the hidden layer operates according to the following state equations:

$$x(k) = f_a\left(W \cdot x(k-1) + B \cdot u(k)\right) \tag{1a}$$

$$y(k) = f_b\left(C \cdot x(k)\right) \tag{1b}$$

where

- f_a, f_b are the neuron activation functions of the hidden and the output layers, respectively. In the following, the activation functions are both chosen to be the sigmoid function $f(z) = \dfrac{1 - e^{-a_n \cdot z}}{1 + e^{-a_n \cdot z}}$.

- $u(k) = [u_i(k)]$ is a m-element input vector, with k being the time variable.
- $x(k) = [x_i(k)]$ is a N-element vector, comprising the outputs of the hidden layer. In particular, $x_i(k)$ is the output of the i-th hidden neuron at time k.
- $y(k) = [y_i(k)]$ is a r-element output vector.
- $B = [b_{i,j}]$ and $C = [c_{i,j}]$ are $N \times m$ and $r \times N$ input and output weight matrices, respectively.
- $W = [w_{i,j}]$ is the $N \times N$ block diagonal feedback matrix. In particular,

$$w_{i,j} = \begin{cases} \neq 0 & \text{if } i = j \\ \neq 0 & \text{if } i \neq j \text{ and } i = j-1 \text{ and } i \text{ is odd} \\ \neq 0 & \text{if } i \neq j \text{ and } i = j+1 \text{ and } i \text{ is even} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

The feedback matrix, W, is block diagonal:

$$W = diag \left[W^{(1)}, \ldots, W^{(\frac{N}{2})} \right] \tag{3}$$

Each diagonal element, corresponding to a block of recurrent neurons, has a block sub-matrix in the form:

$$W^{(i)} = \begin{bmatrix} w_{2i,2i} & w_{2i,2i+1} \\ w_{2i+1,2i} & w_{2i+1,2i+1} \end{bmatrix} \quad i = 1, 2, \ldots \frac{N}{2} \tag{4}$$

Equation (4) describes the general case of BDRNN, which is called BDRNN with free-form sub-matrices. A special case of BDRNN consists of scaled orthogonal sub-matrices in the form

$$W^{(i)} = \begin{bmatrix} w_{2i,2i} & w_{2i,2i+1} \\ -w_{2i,2i+1} & w_{2i,2i} \end{bmatrix} = \begin{bmatrix} w_i^{(1)} & w_i^{(2)} \\ -w_i^{(2)} & w_i^{(1)} \end{bmatrix} \quad i = 1, 2, \ldots \frac{N}{2} \tag{5}$$

From (4) and (5) it is concluded that the Free-Form BDRNN consists of feedback sub-matrices with four distinct elements and provides a greater degree of freedom compared to the Scaled Orthogonal BDRNN, which has two weights at each feedback sub-matrix. Nevertheless, as discussed in [10], the latter network exhibits superior modelling capabilities than the Free-Form BDRNN, and the forthcoming learning method will be developed for this network.

Combining (1)-(5), the state equations for the Scaled Orthogonal BDRNN can take the following form:

$$x_{2i-1}(k) = f_a\left(\sum_{j=1}^{m} b_{2i-1,j}\cdot u_j(k) + w_i^{(1)}\cdot x_{2i-1}(k-1) + w_i^{(2)}\cdot x_{2i}(k-1)\right), i = 1,\ldots,\frac{N}{2} \tag{6a}$$

$$x_{2i}(k) = f_a\left(\sum_{j=1}^{m} b_{2i,j}\cdot u_j(k) - w_i^{(2)}\cdot x_{2i-1}(k-1) + w_i^{(1)}\cdot x_{2i}(k-1)\right), i = 1,\ldots,\frac{N}{2} \tag{6b}$$

$$y_l(k) = f_b\left(\sum_{j=1}^{N} c_{l,j}\cdot x_j(k)\right), l = 1,\ldots,r \tag{6c}$$

where $w_i^{(1)}$, $w_i^{(2)}$ are the feedback weights at the hidden layer.

3 The HLA-BDRNN Algorithm

In gradient-based optimization methods, the weight changes are proportional to the size of the gradient of an error function E:

$$\Delta w_i(t) = -\mu\frac{\partial E(t)}{\partial w_i} \tag{7}$$

where $\frac{\partial E}{\partial w_i}$ is the partial derivative of E with respect to a weight w_i and t represents the epoch index. When it comes to recurrent models, the common partial derivative should be substituted by the *ordered partial derivative*, due to the existence of temporal relations through the feedback connections of BDRNN, as will be discussed in the sequel.

The term μ in (7) is the learning rate, which in BPTT is kept fixed throughout the learning process and is common to all weight updates. Therefore, an appropriate selection of the learning rate is crucial to the evolution of the learning process and constitutes a significant constraint. The RPROP learning scheme attempted to alleviate this disadvantage of BPTT by allowing each fitting parameter to have its individual step size, which is adjusted during the learning process based on the sign of the respective partial derivative at the current and the previous epoch. Therefore, the effect of the adaptation process will not blurred by the influence of the size of the parameter gradient but is only dependent on the temporal behavior of the gradient ([8]).

Particularly, let $\frac{\partial E(t)}{\partial w_i}$ and $\frac{\partial E(t-1)}{\partial w_i}$ denote the derivatives of E with respect to w_i at the present and the preceding epochs, respectively. RPROP is described in pseudo-code as follows:

Some very long text... (a) For all weights w_i initialize the step sizes $\Delta_i^{(1)} = \Delta_0$
Repeat
(b) For all weights w_i compute the error gradient: $\frac{\partial E(t)}{\partial w_i}$
(c) For all weights w_i, update step sizes:
(c.1) If $\frac{\partial E(t)}{\partial w_i} \times \frac{\partial E(t-1)}{\partial w_i} > 0$ Then $\Delta_i^{(t)} = \min\left\{\eta^+ \cdot \Delta_i^{(t-1)}, \Delta_{\max}\right\}$
(c.2) Else if $\frac{\partial E(t)}{\partial w_i} \times \frac{\partial E(t-1)}{\partial w_i} < 0$ Then $\Delta_i^{(t)} = \max\left\{\eta^- \cdot \Delta_i^{(t-1)}, \Delta_{\min}\right\}$
(c.3) Else $\Delta_i^{(t)} = \Delta_i^{(t-1)}$
(d) Update the weights w_i: $\Delta w_i(t) = -sign\left(\frac{\partial E(t)}{\partial w_i}\right) \cdot \Delta_i^{(t)}$
Until convergence

where the step sizes are bounded by $\Delta_{\min}$, $\Delta_{\max}$. The initial values of the step sizes $\Delta_i^{(1)} = \Delta_0$ are chosen rather moderately (e.g. 0.1), since these values directly determine the sizes of the first parameter changes. The increase and attenuation factors are set to $n^+ \in [1.01, 1.3]$ and $n^- \in [0.5, 0.9]$, respectively.
The HLA-BDRNN algorithm performs the following modifications: (i) Substitutes the error gradients with the ordered derivatives $\frac{\partial^+ E(t)}{\partial w_i}$ ([14]), in order to take into consideration the temporal dependencies existing in a dynamic model. (ii) Modifies steps (b) and (c.2) of RPROP as shown:

(b') Compute the HLA-BDRNN error gradient: $\frac{\partial^+ E(t)}{\partial w_i} - 0.01 \cdot SA \cdot \frac{w_i}{1+w_i^2}$
(c.2') Else if $\frac{\partial^+ E(t)}{\partial w_i} \times \frac{\partial^+ E(t-1)}{\partial w_i} < 0$ Then
If $(\Delta_i^{(t)} < 0.4 \cdot SA^2)$ Then $\Delta_i^{(t)} = \max\left\{\eta^- \cdot \Delta_i^{(t-1)} \cdot 0.8 \cdot r \cdot SA^2, \Delta_{\min}\right\}$
Else $\Delta_i^{(t)} = \max\left\{\eta^- \cdot \Delta_i^{(t-1)}, \Delta_{\min}\right\}$

where $SA = 2^{-t \cdot Temp}$ is the simulated annealing term, parameter r takes random values within the interval $[0, 1]$ and $Temp$ is the temperature.
The modified step (c.2') aims at adding noise to the weights, according to the concept of simulated annealing, in order to increase the convergence speed of the learning process. In HLA-BDRNN noise is added to the weight update values when the error gradient changes sign in two successive epochs, and the magnitude of the update value is less than a value that is proportional to the SA term. In this way, the weight update is modified by noise only when it has a relatively small value, thus allowing the weight to move out of local minima, while minimizing the disturbance to the adaptation process.
In the modified step (b'), a weight decay term is added to the error gradient, as proposed in [1]. The effect of this form of weight decay is to modify the error surface such that initially weights with lower values are favoured. As training proceeds, the magnitude of weight decay is reduced, facilitating the increase of bigger weights and allowing the model to explore regions of the error surface that were previously unavailable. Additionally, as mentioned in [1], the use of weight decay has been proved to improve the generalization capability of the

model.
The adaptation mechanism described above has the advantage of correlating the step sizes not to the size of the derivatives but to their signs. Hence, whenever a parameter moves along a direction reducing E (the derivatives at successive epochs have the same sign), its step size is increasing independently of the size of the derivative. In this way, the step sizes can sufficiently increase when needed, even at the final stage of the learning process when the sizes of the derivatives are rather small. Additionally, when changes in the sign of the derivative occur, the step size is diminishing to prevent the error measure from oscillating.
A key issue in the case of locally recurrent networks like BDRNN is the accurate extraction of the error gradients. In the proposed algorithm the time dependencies are fully taken into consideration, and no approximation policy to a few steps back is applied. The error measure used is the Mean Squared Error (MSE), defined by

$$E = \frac{1}{k_f} \sum_{k=1}^{k_f} \sum_{l=1}^{r} [y_l(k) - \hat{y}_l(k)]^2 \tag{8}$$

where $y_l(k)$ is the l-th model output, $\hat{y}_l(k)$ is the l-th desired (actual) output of the system at time step k.
Contrary to static networks, the extraction of the ordered partial derivatives in BDRNN is not straightforward and is accomplished via a set of recursive equations. In order to determine the error gradients of the dynamic part of BDRNN, let us introduce
(a) the state vector $st(t)$, defined as:

$$st(k) = [x_1(k), \ldots, x_N(k), y_1(k), \ldots, y_r(k)]^T$$

comprising the outputs of the hidden and the output layer.
(b) the control vector θ comprising the synaptic and feedback weights ($N \times (m + r + 1)$ weights)

$$\theta = \left[b_{1,1}, \ldots, b_{N,m}, w_1^{(1)}, w_{\frac{N}{2}}^{(1)}, w_1^{(2)}, w_{\frac{N}{2}}^{(2)}, c_{1,1}, \ldots, c_{r,N}\right]^T$$

For a data set including k_f pairs, the state equations are written $f\left(st(k), \theta(k)\right) = 0$, $k = 1, \ldots, k_f$ with
$f_{2i-1}^{(1)}(k)$ $i = 1, \ldots, \frac{N}{2}$ ($k_f \times \frac{N}{2}$ equations):

$$f_a\left(\sum_{j=1}^{m} b_{2i-1,j} u_j(k) + w_i^{(1)} x_{2i-1}(k-1) + w_i^{(2)} x_{2i}(k-1)\right) - x_{2i-1}(k) = 0 \tag{9a}$$

$f_{2i}^{(1)}(k)$ $i = 1, \ldots, \frac{N}{2}$ ($k_f \times \frac{N}{2}$ equations):

$$f_a\left(\sum_{j=1}^{m} b_{2i,j}u_j(k) - w_i^{(2)}x_{2i-1}(k-1) + w_i^{(1)}x_{2i}(k-1)\right) - x_{2i}(k) = 0 \quad (9b)$$

$f_l^{(2)}(k)$ $l = 1, \ldots, r$ ($k_f \times r$ equations):

$$f_b\left(\sum_{j=1}^{N} c_{l,j}x_j(k)\right) - y_l(k) = 0 \quad (9c)$$

The error gradients are given by $\frac{\partial^+ E}{\partial \theta} = \lambda^T \frac{\partial f}{\partial \theta}$ where the extraction of the Lagrange multipliers λ is based on the formula $\frac{\partial E}{\partial x} + \lambda^T \frac{\partial f}{\partial \theta} = 0$.
After calculations are conducted in (19), the multipliers are determined through the following recursive equations:

$$\lambda_l^{(2)}(k) = \frac{1}{k_f} \cdot [y_l(k) - \hat{y}_l(k)] \quad (10a)$$

$$\lambda_{2i-1}^{(1)}(k) = \frac{1}{k_f} \cdot \sum_{l=1}^{r} \left\{ c_{l,2i-1} \cdot \overset{(l)}{f_b'}(k) \cdot [y_l(k) - \hat{y}_l(k)] \right\} + \sum_{l=1}^{r} \left\{ \lambda_{El}^{(2)}(k) \cdot c_{l,2i-1} \cdot \overset{(l)}{f_b}'(k) \right\} + \lambda_{2i-1}^{(1)}(k+1) \cdot w_i^{(1)} \cdot \overset{(2i-1)}{f_a}'(k+1) - \lambda_{2i}^{(1)}(k+1) \cdot w_i^{(2)} \cdot \overset{(2i)}{f_a}'(k+1) \quad (10b)$$

$$\lambda_{2i}^{(1)}(k) = \frac{1}{k_f} \cdot \sum_{l=1}^{r} \left\{ c_{l,2i} \cdot \overset{(l)}{f_b}'(k) \cdot [y_l(k) - \hat{y}_l(k)] \right\} + \sum_{l=1}^{r} \left\{ \lambda_l^{(2)}(k) \cdot c_{l,2i} \cdot \overset{(l)}{f_b'}(k) \right\} + + \lambda_{2i-1}^{(1)}(k+1) \cdot w_i^{(2)} \cdot \overset{(2i-1)}{f_a'}(k+1) + \lambda_{2i}^{(1)}(k+1) \cdot w_i^{(1)} \cdot \overset{(2i)}{f_a'}(k+1) \quad (10c)$$

where $i = 1, \ldots, \frac{N}{2}$, $l = 1, \ldots, r$ and $\overset{(l)}{f_b'}(k)$, $\overset{(2i-1)}{f_1'}(k+j)$, $\overset{(2i)}{f_1'}(k+j)$ are the derivatives of $y_j(k+l)$ and $x_{2i-1}(k+j)$, $x_{2i}(k+j)$, respectively, with respect to their arguments. Equations (10) are backward difference equations that can be solved for $k = k_f, k_f - 1, \ldots, 1$ using the following boundary conditions:

$$\lambda_l^{(2)}(k_f) = \frac{1}{k_f} \cdot [y_l(k_f) - \hat{y}_l(k_f)] \quad (11a)$$

$$\lambda_{2i-1}^{(1)}(k_f) = \frac{1}{k_f} \cdot \sum_{l=1}^{r} \left\{ c_{l,2i-1} \cdot \overset{(l)}{f_b'}(k_f) \cdot [y_l(k_f) - \hat{y}_l(k_f)] \right\} + \sum_{l=1}^{r} \left\{ \lambda_l^{(2)}(k_f) \cdot c_{l,2i-1} \cdot \overset{(l)}{f_b'}(k_f) \right\} \quad (11b)$$

$$\lambda_{2i}^{(1)}(k_f) = \frac{1}{k_f} \cdot \sum_{l=1}^{r} \left\{ c_{l,2i} \cdot \overset{(l)}{f_b'}(k_f) \cdot [y_l(k_f) - \hat{y}_l(k_f)] \right\} + \sum_{l=1}^{r} \left\{ \lambda_l^{(2)}(k_f) \cdot c_{l,2i} \cdot \overset{(l)}{f_b'}(k_f) \right\} \tag{11c}$$

Substituting (17) and (21) to (18), and taking into consideration (12), the error gradients are given by

$$\frac{\partial^+ E}{\partial c_{li}} = \sum_{k=1}^{k_f} \left\{ \lambda_l^{(2)}(k) \cdot x_i(k) \cdot \overset{(l)}{f_b'}(k) \right\} l = 1, \ldots, r, i = 1, \ldots, N \tag{12a}$$

$$\frac{\partial^+ E}{\partial b_{ij}} = \sum_{k=1}^{k_f} \left\{ \lambda_i^{(1)}(k) \cdot u_j(k) \cdot \overset{(i)}{f_a}'(k) \right\} i = 1, \ldots, N, j = 1, \ldots, m \tag{12b}$$

$$\frac{\partial^+ E}{\partial w_i^{(1)}} = \sum_{k=1}^{k_f} \{ \lambda_{2i-1}^{(1)}(k) \cdot x_{2i-1}(k-1) \cdot \overset{(2i-1)}{f_a}'(k) + \lambda_{2i}^{(1)}(k) \cdot x_{2i}(k-1) \cdot \overset{(2i)}{f_a}'(k) \}, \; i = 1, \ldots, \frac{N}{2} \tag{12c}$$

$$\frac{\partial^+ E}{\partial w_i^{(2)}} = \sum_{k=1}^{k_f} \{ \lambda_{2i-1}^{(1)}(k) \cdot x_{2i}(k-1) \cdot \overset{(2i-1)}{f_a}'(k) - \lambda_{2i}^{(1)}(k) \cdot x_{2i-1}(k-1) \cdot \overset{(2i)}{f_a}'(k) \}, \; i = 1, \ldots, \frac{N}{2} \tag{12d}$$

4 Performance Tests and Results

The stabilizing properties and the performance of the M-SARPROP approach are highlighted by use of a benchmark identification problem of a dynamical system [6]. The actual system is described by the difference equation:

$$y_p(k+1) = \frac{y_p(k) \cdot y_p(k-1) \cdot y_p(k-2) \cdot u(k-1) \cdot [y_p(k-2) - 1] + u(k)}{1 + y_p^2(k-1) + y_p^2(k-2)}$$

A parallel identification system is considered, with the input $u(k)$ being the sole input to the network. The BDRNN comprises four blocks of neurons in the hidden layer and has a total number of 32 weights.
The first objective of the experimentation is to compare the performance of the HLA-BDRNN method to BPTT [5], in training of the BDRNN. A second objective is compare the BDRNN and the learning algorithm with other recurrent models, in terms of approximation accuracy and generalization capabilities. In compliance with previous results reported in the literature, the training data set contains ten batches of 900 patterns. For each data batch, the input $u(k)$ is an independent and identically distributed uniform sequence for the first half of the 900 time steps and a sinusoid given by $1.05 \sin(\pi k/45)$

for the rest of the time instants. The checking data set is composed of 1000 samples with a signal described by

$$u(k) = \begin{cases} \sin(\pi k/25) & k < 250 \\ 1 & 250 \leq k < 500 \\ -1 & 500 \leq k < 750 \\ 0.3\sin(\pi k/25) + 0.1\sin(\pi k/32) + 0.6\sin(\pi k/10) & 750 \leq k < 1000 \end{cases}$$

In order to select the training parameters of HLA-BDRNN and BPTT, several runs are performed with the same initial weights but different parameter combinations. Then, the parameter combination that has exhibited the fastest convergence and low values of the error function is selected. Thus, we are led to $\Delta_0 = 0.01$, $\eta^+ = 1.05$ and $\eta^- = 0.85$, $Temp = 1.15$ for HLA-BDRNN, and a learning rate of $\mu_1 = 0.032$ for BPTT, respectively. Next, a series of 100 independent trials with different weight initializations are attempted. Particularly, the feedback weights W and the weight matrices B and C are randomly selected within the range $[-0.5, 0.5]$. The slope pertaining to the activation functions of the network neurons is set to 2. For each particular trial and for fair comparison, the same weight initializations are used for HLA-BDRNN and BPTT.
The competing rivals are recurrent neural networks and their architectures and learning parameters are determined as follows:

- The IIR-MLP is a multilayered network [13], where the synaptic connections are implemented through IIR filters, including a moving average (MA) and an auto-regressive (AR) part. We selected a 1x8x1 IIR-MLP model with unit delays in the MA and the AR parts, both for the input-to-hidden and the hidden-to-output synaptic filters, respectively.

- The diagonal recurrent neural network (DRNN, [3]) has one hidden layer, containing self-recurrent neurons. A 1x8x1 DRNN model is selected.

- The weights of the IIR-MLP and DRNN are randomly initialized in that range $[-0.5, 0.5]$.

- The learning rate is set to 0.01, selected as best value after several training runs.

- The IIR-MLP and DRNN models are trained by BPTT, while the memory neural network (MNN, [9]) is trained using the real time recurrent learning (RTRL) method [7].

- All network models and learning schemes are trained following a parallel mode approach, with the exception of the MNN where the series-parallel configuration is adopted, as reported in [9].

Table 1 hosts the comparative results attained after five training epochs of the entire data set, with the weight updates taking place at the end of each one of the ten batches. The results for the MNN are taken from [9]. As shown, the BDRNN trained by the HLA-BDRNN method exhibits the best performance among the competing schemes, with regard to both the average and, especially, the standard deviation of the checking data sets error. The former criterion indicates the accuracy of the HLA-BDRNN algorithm while the later one shows its robustness to weight initializations. The BPTT scheme for the BDRNN is considerably inferior to the HLA-BDRNN method regarding the accuracy and the generalization property. Furthermore it has an error standard deviation almost twice as large compared to the one attained by HLA-BDRNN, leading to the conclusion that HLA-BDRNN accelerates the learning process for the BDRNN significantly, while exhibiting insensitivity to initial weight settings.

Table 1: Results of the comparative analysis, averaged over 100 independent trials with different weight initializations

Network type	Training method	Checking MSE Avg	Checking MSE StD	No. of weights
BDRNN	HLA-BDRNN	0.0270	0.0031	32
BDRNN	BPTT	0.0368	0.0060	32
IIR-MLP	BPTT	0.0303	0.0409	32
DRNN	BPTT	0.0287	0.0118	33
MNN	RTRL	0.0752	-	81

It should be pointed out that the proposed algorithm has been developed for Scaled Orthogonal BDRNNs. However, it can be easily modified to take into consideration the architectural differences of the Free-Form BDRNN (i.e. four tunable feedback weights at each block of neurons of the hidden layer).

5 Conclusion

A novel learning algorithm for training the special class of Block-Diagonal Recurrent Neural Networks has been proposed, entitled HLA-BDRNN. The hybrid method combines gradient descent and the random search technique of Simulated Annealing, and is tailored to BDRNN, by taking into account the temporal relations existing in this particular dynamic system. It should be mentioned that the algorithm can be easily adapted with moderate modifications to relevant architectures like the triangular recurrent neural network [6].

The learning scheme has been compared with a series of algorithms and recurrent networks in the context of a nonlinear system identification benchmark problem, where its learning characteristics have been highlighted.

Acknowledgements. The authors wish to acknowledge financial support provided by the Research Committee of the Technological Education Institute of Central Macedonia, under grant SAT/IC/15112017-199/12.

References

[1] C. Igel, H. Husken, Empirical Evaluation of the Improved RPROP Learning Algorithms, *Neurocomputing,* **50** (2003), 105–123. https://doi.org/10.1016/s0925-2312(01)00700-7

[2] C.-C. Ku, K.Y. Lee, Diagonal Recurrent Neural Networks for Dynamic Systems Control, *IEEE Transactions on Neural Networks,* **6** (1995), no. 1, 144–156. https://doi.org/10.1109/72.363441

[3] R. Kumar, S. Srivastava, J.R.P. Gupta, Diagonal Recurrent Neural Network Based Adaptive Control of Nonlinear Dynamical Systems Using Lyapunov Stability Criteria, *ISA Transactions,* **67** (2017), 407–427. https://doi.org/10.1016/j.isatra.2017.01.022

[4] P. Mastorocostas, C. Hilas, D. Varsamis, S. Dova, A Recurrent Neural Network-based Forecasting System for Telecommunications Call Volume, *Applied Mathematics & Information Sciences,* **7** (2013), no. 5, 1643–1650. https://doi.org/10.12785/amis/070501

[5] P. Mastorocostas, J.B. Theocharis, A Stable Learning Algorithm for Block-Diagonal Recurrent Neural Networks: Application to the Analysis of Lung Sounds, *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics,* **36** (2006), no. 2, 242–254. https://doi.org/10.1109/tsmcb.2005.856722

[6] K.S. Narendra, K. Parthasarathy, Identification and Control of Dynamical Systems using Neural Networks, *IEEE Transactions on Neural Networks,* **1** (1990), no. 1, 4–27. https://doi.org/10.1109/72.80202

[7] S. Piche, Steepest Descent Algorithms for Neural Network Controllers and Filters, *IEEE Transactions on Neural Networks,* **5** (1994), no. 2, 198–212. https://doi.org/10.1109/72.279185

[8] M. Riedmiller, H. Braun, A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm, *IEEE International Conference on Neural Networks,* (1993), 586–591. https://doi.org/10.1109/icnn.1993.298623

[9] P.S. Sastry, G. Santharam, K.P. Unnikrishnan, Memory Neuron Networks for Identification and Control of Dynamical Systems, *IEEE Transactions on Neural Networks,* **5** (1994), no. 2, 306–319. https://doi.org/10.1109/72.279193

[10] S. Sivakumar, W. Robertson, W.J. Phillips, Online Stabilization of Block-Diagonal Recurrent Neural Networks, *IEEE Transactions on Neural Networks,* **10** (1999), no. 1, 167–175. https://doi.org/10.1109/72.737503

[11] S. Sivakumar, Sh. Sivakumar, Marginally Stable Triangular Recurrent Neural Network Architecture for Time Series Prediction, *IEEE Transactions on Cybernetics,* (2017), 1–15. https://doi.org/10.1109/tcyb.2017.2751005

[12] N.K. Treadgold, T.D. Gedeon, Simulated Annealing and Weight Decay in Adaptive Learning: The SARPROP Algorithm, *IEEE Transactions on Neural Networks,* **9** (1998), no. 4, 662–668. https://doi.org/10.1109/72.701179

[13] A.C. Tsoi, A.D. Back, Locally Recurrent Globally Feedforward Networks: A Critical Review of Architectures, *IEEE Transactions on Neural Networks,* **5** (1994), no. 2, 229–239. https://doi.org/10.1109/72.279187

[14] P. Werbos, *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*, Ph.D. Thesis, Harvard Univ., 1974.

Received: December 9, 2017; Published: January 5, 2018