On-Line Learning Algorithms for Locally Recurrent Neural Networks

Paolo Campolucci, *Member, IEEE*, Aurelio Uncini, *Member, IEEE*,
Francesco Piazza, *Member, IEEE*, Bhaskar D. Rao, *Senior Member, IEEE*

***Abstract*—This paper focuses on on-line learning procedures for locally recurrent neural networks with emphasis on multilayer perceptron (MLP) with infinite impulse response (IIR) synapses and its variations which include generalized output and activation feedback multilayer networks (MLN's). We propose a new gradient-based procedure called recursive backpropagation (RBP) whose on-line version, causal recursive backpropagation (CRBP), presents some advantages with respect to the other on-line training methods. The new CRBP algorithm includes as particular cases backpropagation (BP), temporal backpropagation (TBP), backpropagation for sequences (BPS), Back–Tsoi algorithm among others, thereby providing a unifying view on gradient calculation techniques for recurrent networks with local feedback. The only learning method that has been proposed for locally recurrent networks with no architectural restriction is the one by Back and Tsoi. The proposed algorithm has better stability and higher speed of convergence with respect to the Back–Tsoi algorithm, which is supported by the theoretical development and confirmed by simulations. The computational complexity of the CRBP is comparable with that of the Back–Tsoi algorithm, e.g., less that a factor of 1.5 for usual architectures and parameter settings. The superior performance of the new algorithm, however, easily justifies this small increase in computational burden. In addition, the general paradigms of truncated BPTT and RTRL are applied to networks with local feedback and compared with the new CRBP method. The simulations show that CRBP exhibits similar performances and the detailed analysis of complexity reveals that CRBP is much simpler and easier to implement, e.g., CRBP is local in space and in time while RTRL is not local in space.**

***Index Terms*—IIR-MLP, locally recurrent neural networks, on-line learning, recursive backpropagation, system identification, time-delay neural networks.**

I. Introduction

RECENTLY dynamic recurrent neural networks have been attracting much attention from the scientific community (see the special issue of IEEE TRANSACTIONS ON NEURAL NETWORKS, March 1994, and of *Neurocomputing*, June 1997) because they can be very useful for temporal processing, e.g., digital signal processing (DSP), system identification and control, and temporal pattern recognition.

Two main methods exist to provide a static neural network with dynamic behavior: the insertion of a buffer somewhere in the network, i.e., implementing an explicit memory of the past inputs, or the use of feedback. In both approaches, an arbitrary input ($x[t]$) may influence a future output ($y[t+h]$), so that ($\partial y[t+h]/\partial x[t]$) is not equal to zero for some h. In the case of asymptotic stability this derivative goes to zero when h goes to infinity. The value of h for which the derivative becomes negligible is called *temporal depth*, whereas the number of adaptable parameters divided by the temporal depth is called *temporal resolution* [32].

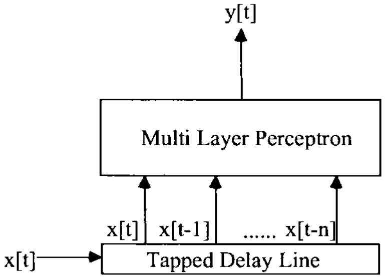


Fig. 1. Buffered multilayer perceptron with input buffer.

The first kind of dynamic network is a buffered multilayer perceptron (MLP) in which tapped delay lines (TDL's) of the inputs are used. The buffer can be applied at the network inputs only, keeping the network internally static as in buffered MLP's [22] (see Fig. 1), or at the input of each neuron as in MLP with finite impulse response (FIR) filter synapses [FIR-MLP, see Fig. 3(a)] [3], [8], [11], [22], [37] often called time-delay neural network (TDNN) [9], [10] and in adaptive time-delay neural networks [34], [35]. It is well known that buffered MLP and FIR-MLP can be shown to be theoretically equivalent [8] since internal buffers can be implemented as an external one. The problem with implementing FIR-MLP's as buffered MLP's is that first layers sub networks must be replicated (with shared weights) [8] and so the complexity is much higher than considering the buffer internal. Therefore buffered MLP and FIR-MLP are different architectures with regard to a real implementation. The main disadvantage of the buffer approach is the limited past history horizon thereby preventing modeling of arbitrary long time dependencies [20], [21], [55] between inputs and desired outputs. It is also difficult to set the length of the buffer given a certain application; moreover to have sufficient temporal depth, a long buffer, i.e., a large number of inputs weights, could be required, usually with a decrease in generalization performance and an increase in the overall computational complexity. In other words the buffer approach with no feedback has the maximum temporal resolution, at the cost of a low temporal depth.

Manuscript received October 28, 1996; revised December 1, 1997 and October 22, 1998.

P. Campolucci, A. Uncini, and F. Piazza are with the Dipartimento di Elettronica ed Automatica, Università di Ancona, Ancona, Italy (e-mail: paolo@eealab.unian.it.

B. D. Rao is with the Department of Electrical and Computer Engineering, University of California, San Diego, CA 92037 USA.

Publisher Item Identifier S 1045-9227(99)01906-2.

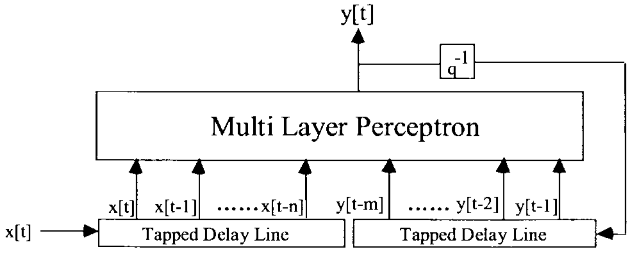
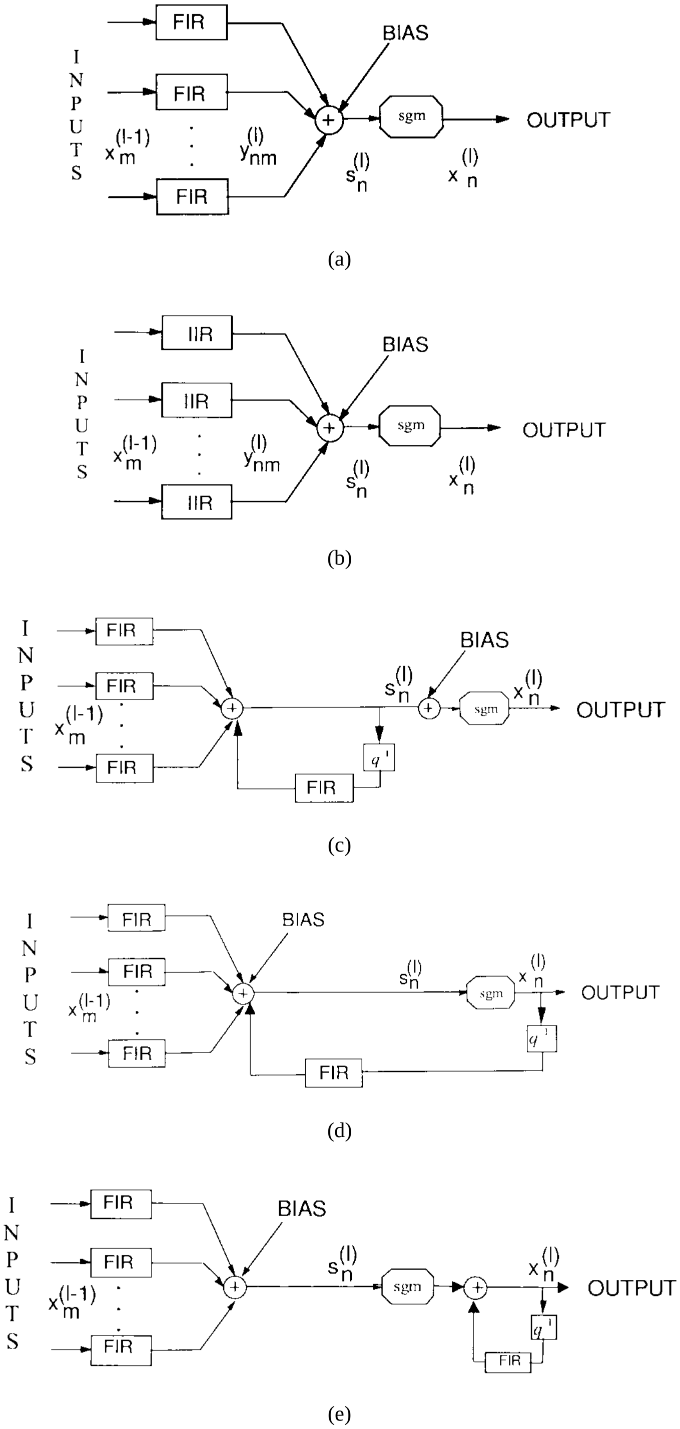


Fig. 2. Buffered multilayer perceptron with input and output buffers, sometimes called Narendra Parthasarathy or NARX neural network.



(a)

(b)

(c)

(d)

(e)

Fig. 3. Model of the neuron for: (a) FIR-MLP, (b) IIR-MLP, (c) locally recurrent activation feedback MLN, (d) output feedback MLN, and (e) auto regressive MLP.

Recently a new buffer type called gamma memory has been proposed by Principe *et al.* [32], [33], for which the delay operator, used in conventional TDL's, is replaced by a single pole discrete time filter. Gamma memory is a dispersive delay line with dispersion regulated by an adaptable parameter, so that it is possible to adaptively trade off temporal depth with temporal resolution. In addition to these advantages of temporal depth and resolution characteristics, it is known that neural networks with feedback have useful dynamic modeling behavior [40], [41].

The main example of implementation of feedback is the classical fully recurrent neural network, i.e., a single layer of neurons fully interconnected with each other [1], [2], [5], [22], [24], [31], [39], [42], [49], [51], [52], or several such layers [29]. Such recurrent networks however exhibit some well known disadvantages: a large structural complexity ($O(n^2)$ weights are necessary for n neurons) [30] and a slow and difficult training, e.g., [4]. In fact they are very general architectures which can model a large class of dynamical systems, but on specific problems simpler dynamic neural networks which make use of available prior knowledge can be better [21], [41].

In the past few years, a growing interest has been devoted to methods which allow introduction of temporal dynamics into the multilayer neural model. In fact the related architectures are less complex and easier to train with respect to the fully recurrent networks. The major difference among these methods lies in how the feedback are included in the network.

Externally: As in the Narendra–Parthasarathy MLP [28] also known as NARX network, where TDL's are used for the outputs that feedback to the input of the network (see Fig. 2), and in the Elman's network [40].

Internally: Inside each neuron.

The latter approach brings us to the so called locally recurrent neural networks (LRNN's) or local feedback multilayer networks (LF-MLN) [4], [18]–[20], [58]. In these structures, classical infinite impulse response (IIR) linear filters [13], here called also autoregressive moving average (ARMA) models, are used either directly or with some modifications. Different architectures arise depending on how the ARMA model is included in the network.

The first architecture is the IIR-MLP proposed by Back and Tsoi [3], [4] where static synapses are substituted by conventional IIR adaptive filters [see Fig. 3(b)]. The second architecture is the activation feedback locally recurrent multilayer network studied by Frasconi *et al.* [18]–[21]. The output of the neuron summing node is filtered by an autoregressive (AR) adaptive filter (all poles transfer function) before feeding the activation function; in the most general case the synapses are FIR adaptive filters [see Fig. 3(c)]. The activation feedback multilayer network is a particular case of the IIR-MLP, when all the synaptic transfer functions of the same neuron have the same denominator.

The third structure is the output feedback locally recurrent network proposed by Gori *et al.* [18]–[21]. In this architecture the IIR filter is not simply placed in the classical neuron model but is modified to make the feedback loop pass through the nonlinearity, i.e., the one time step delayed output of the neuron is filtered by a FIR filter whose output is added to the inputs contributions, providing the activation. Again in the general model the synapses can be FIR filters [see Fig. 3(d)].

The work of Gori *et al.* has its foundation in the work by Mozer [41] in which the main idea was the introduction of context units to include memory in a network, substituting the spatial metaphor of the external buffer (common at that time) with the recurrent context approach, as also suggested by Elman in [40]. Context units are dynamic recurrent neurons placed in the first layer to process the input signals while the following layers are supposed to be static. This architectural constraint also used in the works of Gori *et al.* has been chosen basically to simplify the learning phase.

At last, there is another architecture [see Fig. 3(e)] that has not been studied as the previous ones: it was proposed by Mozer in [41] (with one delay feedback dynamic units in the first layer only) and by Leighton and Conrath in [44] (multiple delays and no restriction on the position of dynamic units). It is again a multilayer network where each neuron has FIR filter synapses and an AR filter after the activation function (AR-MLP). It is easy to see that this network is a particular case of the IIR-MLP, followed by linear all-pole filters.

Recently diagonal recurrent neural networks (DRNN) [30], have been proposed for dynamic systems control, claiming relevant results. This architecture is also a particular case of output feedback MLN since DRNN is a two-layer network with static linear output neurons and dynamic hidden neurons with static synapses but with one delay feedback from the output. Again the position of dynamic units is restricted to the first layer only.

Another version of locally recurrent neural network was presented in [54] with a biological motivation: a multilayer connection of perceptrons with low-pass temporal filtering of the activation.

The major advantages [4], [20], [21], [30], [40], [41], [58] of locally recurrent neural networks with respect to buffered MLP's or fully recurrent networks can be summarized as follows:

1) well-known neuron interconnection topology, i.e., the efficient and hierarchic multilayer;
2) small number of neurons required for a given problem, due to the use of powerful dynamic neuron models;
3) generalization of the popular FIR-MLP (or TDNN) to the infinite memory case;
4) prewired forgetting behavior [20], needed in applications such as DSP, system identification and control;
5) simpler training than fully recurrent networks;
6) many training algorithms could be derived from filter theory and recursive identification.

In the following we will consider only locally recurrent neural networks, particularly IIR-MLP and output feedback MLN which are the most general and interesting architectures. Some algorithms to train such networks exist, although a comprehensive framework is still missing.

In this paper we propose a new gradient-based algorithm for locally recurrent neural networks, called recursive backpropagation (RBP) whose on-line version, causal recursive backpropagation (CRBP) presents some advantages with respect to the already known on-line training methods. The new CRBP algorithm includes as particular cases backpropagation (BP) [5], [48] temporal backpropagation (TBP) [8], [22] backpropagation for sequences (BPS) [18], [19] and Back–Tsoi algorithm [3], [12]. Moreover it allows for the training of generalized output and activation feedback MLN's which have no constraint on the position of the dynamic units, implementing communications among them, as suggested in [21] and [47] for a better modeling.

The concepts detailed in this paper were developed in [62] and later presented in [63].

The outline of this paper is as follows. Section II reports an overview of gradient-based learning algorithms for locally recurrent networks. Section III gives the complete formulation of the batch mode RBP algorithm for IIR-MLP, while Section IV presents the on-line version, CRBP. Experimental results of the proposed method are reported in Section V. In Section VI we discuss issues related to the computational complexity and implementation of the new algorithm. The extension of the RBP to the other locally recurrent architectures is reported in the Appendix.

II. Gradient-Based Learning Algorithms for Locally Recurrent Neural Networks

Internally static networks can be trained by the simplest algorithms: for the buffered MLP of Fig. 1 with only input buffer (with no recursion) the standard BP should be used, while for the Narendra–Parthasarathy network shown in Fig. 2 the so called "open-loop" approximation [45], [50] of the standard BP is usually employed. It consists in opening the loop during the backward phase, feeding the network with the desired outputs instead of the true network outputs. In IIR adaptive filter theory this is the equation error approximation of the true output error approach [13], in neural-network theory this is the teacher forced technique [2].

Extension of Back propagation to recurrent networks was first proposed in [5], [51], [52]. Pineda and Almeida [51], [52] covered only the case when the recurrent network behavior relaxes to a fixed point. However, if a general temporal processing is needed, two main gradient-based learning approaches exist for recurrent networks [24], [25], [36]: Backpropagation through time (BPTT) [1], [5], [7], [24], [36], [46] and real-time recurrent learning (RTRL) [2], [23], [24], [27], [28], [36], [38], [49], [56]. The algorithm in [23] is an hybrid method.

BPTT is a family of algorithms which extends the BP paradigm to dynamic networks. There are two main points of view to understand the BPTT algorithm. The first is an intuitive one: time unfolding of the recurrent network, i.e., for single layer single feedback delay fully recurrent networks one can think of the network state at time t as if it was obtained from the tth layer output of a multilayer network with T layers [5], where T is the length of the sequence. The other point of view is a mathematical one and it is based on the Werbos theory of ordered derivatives [6]. Werbos provided a mathematical tool to rigorously compute the derivatives of a certain variable with respect to another one in complex structures described by ordered mathematical relations (for example a neural network).

Actually, with ordered derivatives it is possible to derive both BPTT and RTRL algorithms in the same framework [25], [36]. The difference between BPTT and RTRL is in how the chain rule derivative expansion is applied. More specifically, during the learning phase, in BPTT the neural network is computed backward both in the layer and time dimensions, whereas in RTRL it is calculated forward (as in the forward calculation). Reversing the signal flow graph provides the great efficiency of BPTT, but the necessary reversion of time makes it noncausal even when there is only one delay present inside the network (i.e., after an adaptable parameter in the signal flow graph) [26].

Therefore BPTT is local in space but not in time, and is computationally simple but is noncausal; so it can be implemented only in batch mode. For on-line adaptation some approximations are needed, namely causalization and truncation of past history, as explained in [1], [40], and [53] for fully recurrent neural networks.

On the other hand, RTRL is local in time but not in space, computationally complex but intrinsically on-line. RTRL also implements an approximated calculation of the gradient if the parameters are continually adapted since the true derivative would require constant weights [1], [53].

In [53], Williams and Zipser report better performance and convergence rate for truncated BPTT than RTRL and explain this result stating that the history truncation approximation can be better than the approximation implemented in RTRL.

Recently Wan and Beaufays [26] proposed a simple method to derive BPTT for discrete time dynamic neural networks composed of a general interconnection of weights, delays, additive units, differentiable nonlinearities. This derivation can be carried out with simple transformations of the signal flow graph of the network itself; however the algorithm derived in this way is always noncausal and the authors did not address the question of on-line learning for networks with feedback. Recently a new approach was derived based on signal flow graphs that makes this derivation of both RTRL [59] and truncated BPTT [60], [61] feasible but a detailed discussion is beyond the aim of the paper.

Most of the above methods were studied for general fully recurrent networks. On the other side several on-line learning algorithms have been presented for specific dynamic multilayer neural networks, most of the times with no reference to each other and to the general paradigms.

In [8] a learning algorithm, named temporal backpropagation, was proposed by Wan. This is an on-line version of the batch mode BPTT approach [26]. However, it can only be applied to the nonrecurrent FIR-MLP [Fig. 3(a)].

Backpropagation for sequences (BPS) is a learning algorithm proposed by Gori *et al.* [18], [19], [21], [24], [38] both for output and activation local feedback MLN's (LF-MLN's), Fig. 3(c) and (d). It is interesting because it is local both in time and in space, computationally simple and with small memory requirement; in fact it is only slightly more complex than standard BP. However it can be applied only to LF-MLN with no dynamic units in layers other than the first one. BPS basically is the classical backpropagation on the multilayer network with a recursive computation only inside each dynamic neuron. Due to the architectural constraints, this algorithm does not implement backpropagation through a dynamic structure.

The same approach was proposed by Mozer in [41] independently deriving a similar algorithm named focused backpropagation for a particular AR-MLP. BPS was rediscovered in [30] where it was derived for a structure that is a particular case of the output feedback LF-MLN and was applied to control problems with good results.

In [3] and [12], a learning algorithm for IIR-MLP, Fig. 3(b), was proposed by Back and Tsoi. It is similar to BPS, implementing both a backpropagation and a recursive computation, but without any architectural restriction. However, to avoid dynamic backpropagation, they propose using static backpropagation even through a dynamic neuron.

Analogous learning algorithms are also: autoregressive BP proposed by Leighton and Conrath [44] for the AR-MLP, and the algorithm in [54]. They are equivalent to Back–Tsoi algorithm since they also use instantaneous backpropagation without implementing the full backpropagation through a dynamic unit. So in the following of the paper we will call Back–Tsoi algorithm the method with instantaneous backpropagation and we will not refer anymore to the works in [44] and [54].

The on-line algorithm proposed in this paper, i.e., CRBP, whose basic ideas were presented in [16] and [17], implements and combines together BPTT and RTRL paradigms for locally recurrent networks. It works with the most general locally recurrent networks and implements a more accurate computation of the gradient than the Back–Tsoi method. While Back–Tsoi algorithm uses an instantaneous error as cost function, the CRBP algorithm can minimize the global error; this fact results in an improved stability of the algorithm. The name that we use, i.e., causal recursive backpropagation (not to be confused with recurrent backpropagation [51], [52]) was chosen to remember the dual nature of the algorithm: BPTT style formulas are used to backpropagate the error through the neurons and recursive computation of derivatives inside each neuron is implemented to calculate weights variations.

It is well known that RTRL and BPTT approaches are equivalent in batch mode operation [36]: they compute the same weights variations using different chain rule expansions. Since CRBP uses another expansion of the same derivatives, it becomes equivalent to them when used in batch mode (RBP). Since BPTT is computationally simpler than RTRL or RBP it is the algorithm of choice working in batch mode, unless the memory requirement is an issue. In this case RTRL can be preferred since for long sequences it requires less memory (see Section VI).

However the three methods are not equivalent in on-line mode. In this case truncated BPTT must be considered instead of BPTT and CRBP instead of RBP. It must be stressed that in CRBP each local feedback of a certain neuron is taken into account with no history truncation (necessary for truncated BPTT) for the adaptation of the coefficients of the same neuron, using recursive formulas instead of noncausal ones as in the truncated BPTT approach.

In other words, the RBP algorithm computes exact gradient, is not local in time (like BPTT) but has the advantage that it can be efficiently implemented on-line (CRBP) at approximately the same cost, with a parameter that controls the tradeoff between exactness of the gradient and computational time. With respect to RTRL the proposed CRBP algorithm has the advantage of being local in space and in time while RTRL is not local in space.

A reasonable and standard definition of locality is the following. An algorithm is local in space if the update complexity per time step and weight does not depend on network size. A method is local in time if its storage requirements do not depend on input sequence length. In fact the proposed algorithm CRBP is "quasi" local in space, meaning that this property is satisfied asymptotically if the number of neurons increases. This is due to the fact that the complexity is linear with the number of coefficients to be adapted at the network level while is quadratic at the neuron level, since the computation is locally (i.e., inside the neuron) recursive, i.e., of RTRL type.

Of course the gradient calculation techniques developed in this paper can be implemented in second-order methods such as conjugate gradient or Kalman filter based algorithms [29], but this is beyond the aim of the paper.

In the following section the recursive backpropagation batch learning algorithm and its on-line version are derived for the MLP with IIR synapses; the formulas for LF-MLN's and AR-MLP are in the Appendix.

III. The Recursive Backpropagation Algorithm for MLP with IIR Synapses

An IIR-MLP contains in each synapse a linear filter with poles and zeros, which are the AR and moving average (MA) part, respectively. Due to the complexity of the resulting structure, a rigorous notation is needed, where each index is explicitly written. This notation, which is a generalization of that used in [48] for static MLP and in [8] and [11] for FIR-MLP, is appropriate in this case where complex architectures of different kinds are defined and compared.

A. Notation

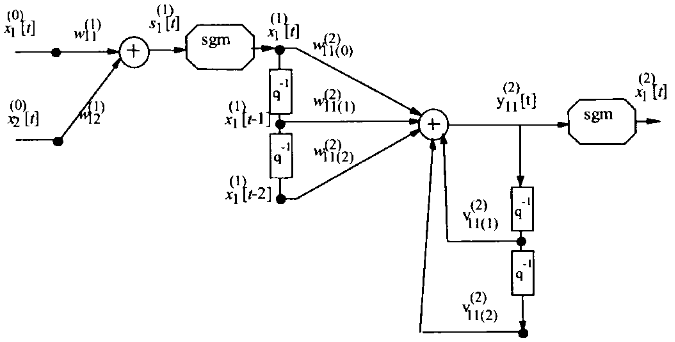


Fig. 4. A simple example of IIR-MLP network (the bias terms are not shown).

M Number of layers in the network.

l Layer index. In particular $l = 0$ and $l = M$ denote the input and output layer, respectively.

N_l Number of neurons of the lth layer. In particular N_0 and N_M denote the number of inputs and outputs, respectively.

n Neuron index.

t Time index, $t = 1, 2, \cdots, T$, where T is the length of the training sequence.

$x_n^{(l)}[t]$ Output of the nth neuron of the lth layer, at time t. In particular $n = 0$ refers to the bias inputs: $x_0^{(l)} = 1$. Note that $x_n^{(0)}[t]$, $n = 1, \cdots, N_0$, are the input signals.

$L_{nm}^{(l)} - 1$ Order of the MA part of the synapse of the nth neuron of the lth layer relative to the mth output of the $(l-1)$th layer. $L_{nm}^{(l)} \geq 1$ and $L_{n0}^{(l)} = 1$.

$I_{nm}^{(l)}$ Order of the AR part of the synapse of the nth neuron of the lth layer relative to the mth output of the $(l-1)$th layer. $I_{nm}^{(l)} \geq 0$ and $I_{n0}^{(l)} = 0$.

$w_{nm(p)}^{(l)}$ $(p = 0, 1, \cdots, L_{nm}^{(l)} - 1)$ coefficients of the MA part of the corresponding synapse. If $L_{nm}^{(l)} = 1$, the synapse has no MA part and the weight notation becomes $w_{nm}^{(l)}$. $w_{n0}^{(l)}$ is the bias.

$v_{nm(p)}^{(l)}$ $(p = 1, \cdots, I_{nm}^{(l)})$ coefficients of the AR part of the synapse. If $I_{nm}^{(l)} = 0$ the synaptic filter is purely MA.

weight Either a w or v coefficient.

$\mathrm{sgm}(z)$ Activation function.

$\mathrm{sgm}'(z)$ Derivative of $\mathrm{sgm}(z)$.

$y_{nm}^{(l)}[t]$ Synaptic filter output at time t relative to the synapse of nth neuron, lth layer, and mth input. $y_{n0}^{(l)} = w_{n0}^{(l)}$ is the bias.

$s_n^{(l)}[t]$ "Net" quantity relative to the nth neuron of the lth layer at time t, i.e., the input to the corresponding activation function.

$d_n[t]$ $(n = 1, \cdots, N_M)$ desired outputs at time t.

To further clarify the notation, a simple two-layer ($M = 2$) IIR-MLP with two inputs ($N_0 = 2$), one hidden neuron ($N_1 = 1$) with no MA and AR parts in each synapse ($L_{1m}^{(1)} = 1$ and $I_{1m}^{(1)} = 0$ for $m = 1, 2$) and one output neuron ($N_2 = 1$) with both MA and AR parts in the synapses ($L_{11}^{(2)} = 3$ and $I_{11}^{(2)} = 2$), is shown in Fig. 4.

B. The Forward Phase

The forward phase at time t can be described by the following equations evaluated for $l = 1, \cdots, M$ and $n = 1, \cdots, N_l$:

$$y_{nm}^{(l)}[t] = \sum_{p=0}^{L_{nm}^{(l)}-1} w_{nm(p)}^{(l)} x_m^{(l-1)}[t-p] + \sum_{p=1}^{I_{nm}^{(l)}} v_{nm(p)}^{(l)} y_{nm}^{(l)}[t-p] \tag{1}$$

$$s_n^{(l)}[t] = \sum_{m=0}^{N_{l-1}} y_{nm}^{(l)}[t]; \qquad x_n^{(l)}[t] = \mathrm{sgm}\Big(s_n^{(l)}[t]\Big). \tag{2}$$

For (1), the direct form I of the IIR filter has been used [13], but other structures are possible. In particular, direct form II structures allow reduction in the storage complexity as well as in the number of operations, both in forward and backward computation (see Section VI). For the sake of clarity the expression corresponding to (1) in the IIR filter usual notation [13] is reported

$$y[t] = \sum_{p=0}^{M-1} w_p[t]x[t-p] + \sum_{p=1}^{N-1} v_p[t]y[t-p] \tag{3}$$

where $y[t]$ is the output, $x[t]$ the input of the IIR filter, w are the coefficients of the MA part, v of the AR part, $M-1$ and $N-1$, respectively, the orders of the MA and AR parts, that can also be written

$$y[t] = \left(\frac{B(t,\, q)}{1 - A(t,\, q)}\right) x[t] \tag{4}$$

where

$$A(t,\, q) = \sum_{p=1}^{N-1} v_p[t]q^{-p} \quad \text{and} \quad B(t,\, q) = \sum_{p=0}^{M-1} w_p[t]q^{-p} \tag{5}$$

where q^{-1} is the delay operator, i.e., $q^{-j}s[t] = s[t-j]$.

In this case the dependence of the coefficients upon time is explicitly stated with the index t, since we are considering adaptive filters that are adapted every time step. However, in the neural network in order to reduce the complexity of the notation we will not use the explicit indication of time.

C. The Learning Algorithm (RBP)

The instantaneous global squared error at time t is defined as

$$e^2[t] = \sum_{n=1}^{N_M} e_n^2[t] \quad \text{with} \quad e_n[t] = d_n[t] - x_n^{(M)}[t]. \tag{6}$$

So the global squared error over the whole training sequence is

$$E^2 = \sum_{t=1}^{T} e^2[t] \tag{7}$$

where T is the duration of the sequence.

In the most general case, the training set of a dynamic neural network is composed of a certain number of training sequences (runs), and so the error to be minimized is the statistical average of the error E^2 over all the runs. To simplify the notation we will consider only one run but the extension is straightforward.

Let us define the usual quantities "*backpropagating error*" and "*delta*"

$$e_n^{(l)}[t] = -\frac{1}{2}\frac{\partial E^2}{\partial x_n^{(l)}[t]} \quad \text{and} \quad \delta_n^{(l)}[t] = -\frac{1}{2}\frac{\partial E^2}{\partial s_n^{(l)}[t]}. \tag{8}$$

As in the static case, it holds

$$\delta_n^{(l)}[t] = e_n^{(l)}[t]\mathrm{sgm}'\Big(s_n^{(l)}[t]\Big). \tag{9}$$

Therefore, using gradient descent method and the chain rule expansion

$$\Delta w_{nm(p)}^{(l)} = -\frac{\mu}{2}\frac{\partial E^2}{\partial w_{nm(p)}^{(l)}} = -\frac{\mu}{2}\sum_{t=1}^{T}\frac{\partial E^2}{\partial s_n^{(l)}[t]}\frac{\partial s_n^{(l)}[t]}{\partial w_{nm(p)}^{(l)}} \tag{10}$$

where μ is the learning rate.

The above equation can be rewritten as

$$\Delta w_{nm(p)}^{(l)} = \sum_{t=1}^{T} \Delta w_{nm(p)}^{(l)}[t+1] \tag{11}$$

where

$$\begin{aligned}\Delta w_{nm(p)}^{(l)}[t+1] &= -\frac{\mu}{2}\frac{\partial E^2}{\partial s_n^{(l)}[t]}\frac{\partial s_n^{(l)}[t]}{\partial w_{nm(p)}^{(l)}}\\ &= \mu\delta_n^{(l)}[t]\frac{\partial s_n^{(l)}[t]}{\partial w_{nm(p)}^{(l)}}.\end{aligned} \tag{12}$$

Similarly for the v weights we have

$$\Delta v_{nm(p)}^{(l)} = \sum_{t=1}^{T} \Delta v_{nm(p)}^{(l)}[t+1] \tag{13}$$

where

$$\Delta v_{nm(p)}^{(l)}[t+1] = \mu\delta_n^{(l)}[t]\frac{\partial s_n^{(l)}[t]}{\partial v_{nm(p)}^{(l)}}. \tag{14}$$

Expressions to compute $\delta_n^{(l)}[t]$ and the derivatives in (12) and (14) must be provided.

Differentiating (1) and considering that $(\partial s_n^{(l)}[t]/\partial weight_{nm(p)}^{(l)}) = (\partial y_{nm}^{(l)}[t]/\partial weight_{nm(p)}^{(l)})$, where "$weight$" indicates either w or v, we get

$$\frac{\partial s_n^{(l)}[t]}{\partial w_{nm(p)}^{(l)}} = x_m^{(l-1)}[t-p] + \sum_{r=1}^{I_{nm}^{(l)}} v_{nm(r)}^{(l)}\frac{\partial s_n^{(l)}[t-r]}{\partial w_{nm(p)}^{(l)}} \tag{15}$$

$$\frac{\partial s_n^{(l)}[t]}{\partial v_{nm(p)}^{(l)}} = y_{nm}^{(l)}[t-p] + \sum_{r=1}^{I_{nm}^{(l)}} v_{nm(r)}^{(l)}\frac{\partial s_n^{(l)}[t-r]}{\partial v_{nm(p)}^{(l)}}. \tag{16}$$

Note that such expressions are the same found in the IIR linear adaptive filter theory [13, (16a), (16b)]

$$\frac{\partial y[t]}{\partial w_p} = x[t-p] + \sum_{r=1}^{N-1} v_r\frac{\partial y[t-r]}{\partial w_p}$$

or

$$\frac{\partial y[t]}{\partial w_p} = \left(\frac{1}{1-A(t,\, q)}\right)x[t-p] \tag{17}$$

$$\frac{\partial y[t]}{\partial v_p} = y[t-p] + \sum_{r=1}^{N-1} v_r\frac{\partial y[t-r]}{\partial v_p}$$

or

$$\frac{\partial y[t]}{\partial v_p} = \left(\frac{1}{1-A(t,\, q)}\right)y[t-p] \tag{18}$$

where the weights time index is not explicitly written to avoid notational clutter.

The formulas (15) and (16), as in the IIR linear adaptive filter context, are exactly true only if the weights (w or v) are not time-dependent, because the derivatives evaluation point is fixed, or approximately true if they adapt slowly, i.e., the learning rate is sufficiently small [2], [13]. In batch RBP the weights update is performed only at the end of the learning epoch using the accumulated weight variations computed at every time instant, so that the above expressions are exact and can be computed iteratively starting with null values of the initial derivatives.

Now we want to derive an expression for $\delta_n^{(l)}[t]$: by (9) we need to compute $e_n^{(l)}[t]$; using the chain rule it is possible to obtain

$$\begin{aligned} e_n^{(l)}[t] &= -\frac{1}{2}\frac{\partial E^2}{\partial x_n^{(l)}[t]} \\ &= \sum_{q=1}^{N_{l+1}}\sum_{k=1}^{T} -\frac{1}{2}\frac{\partial E^2}{\partial s_q^{(l+1)}[k]}\,\frac{\partial s_q^{(l+1)}[k]}{\partial x_n^{(l)}[t]}, \qquad \text{for } l<M. \end{aligned} \tag{19}$$

By the last expression, under the hypothesis of IIR synaptic filter causality, the internal summation can start from $k = t$. Then, changing the variables as $k-t \rightarrow p$, using the definition of $\delta_n^{(l)}[t]$ and considering that for $l = M$ the derivative can be directly computed, the backpropagation through the layers can be derived

$$e_n^{(l)}[t] = \begin{cases} e_n[t] \quad \text{for } l = M \\ \displaystyle\sum_{q=1}^{N_{l+1}}\sum_{p=0}^{T-t} \delta_q^{(l+1)}[t+p]\,\frac{\partial y_{qn}^{(l+1)}[t+p]}{\partial x_n^{(l)}[t]}, \\ \quad \text{for } l = (M-1), \cdots, 1 \end{cases} \tag{20}$$

where the partial derivatives are computed using (1)

$$\begin{aligned} \frac{\partial y_{qn}^{(l+1)}[t+p]}{\partial x_n^{(l)}[t]} &= \left(\begin{cases} w_{qn(p)}^{(l+1)}, & \text{if } 0 \le p \le L_{qn}^{(l+1)}-1 \\ 0, & \text{otherwise} \end{cases}\right) \\ &\quad + \sum_{r=1}^{\min(I_{qn}^{(l+1)},\,p)} v_{qn(r)}^{(l+1)}\,\frac{\partial y_{qn}^{(l+1)}[t+p-r]}{\partial x_n^{(l)}[t]}. \end{aligned} \tag{21}$$

These derivatives have a very interesting interpretation. Consider the expression of a generic causal linear filter output as the convolution of the input $x[\tau]$ with an impulse response $h[t, \tau]$ (in general time variant case)

$$y[t] = \sum_{\tau=t_0}^{t} x[\tau]h[t, \tau] \tag{22}$$

where t_0 is the initial time instant. Differentiating we get

$$\frac{\partial y[t+p]}{\partial x[t]} = h[t+p, t]. \tag{23}$$

If the learning algorithm updates the coefficients only at the end of the epoch (batch mode adaptation), then the IIR filter is time invariant and

$$\begin{aligned} &\left(\frac{1}{1-A(t, q)}\right)\left(\begin{cases} w_p, & \text{if } 0 \le p \le M-1 \\ 0, & \text{otherwise} \end{cases}\right) \\ &= \frac{\partial y[t+p]}{\partial x[t]} = h[t+p-t] = h[p] \end{aligned} \tag{24}$$

where the operator q^{-1} now is delaying the p index and not the t index, $h[p]$ is the impulse response of the filter and $A(t, q)$, previously defined, does not depend on t. Obviously, if the filter is time invariant, the derivative does not depend on t. This means that the derivative is obtained through AR filtering of the sequence of the coefficients of the MA part with the AR part of the corresponding IIR synaptic filter. This is true since for the causal filter the derivative inside the summation is zero if $r > p$, allowing the upper limit of the summation in (21) to be written also as $I_{qn}^{(l+1)}$. If the learning rate is small enough, also when on-line adaptation is performed the derivative is slowly changing in time, i.e., with the t index in (21).

Therefore, for MLP with IIR synapses, (20) suggests that each back propagating error at layer l is a summation of all the *delta's* at the following layer filtered by the noncausal version of the respective IIR filter, i.e., filtering by the time reverted impulse response of the synaptic filter.

The expressions (1), (2), (9), (11)–(16), (20), and (21) constitute the RBP algorithm for IIR-MLP. Note that, if all the synapses contain only the MA part ($I_{nm}^{(l)} = 0$ for each n, m, and l), the architecture reduces to FIR-MLP and this algorithm reduces to the temporal backpropagation (TBP; batch mode) as in [8], [11], [22], [37]. Obviously, if all the synaptic filters have no memory ($I_{nm}^{(l)} = 0$ and $L_{nm}^{(l)} = 1$ for each n, m, and l), this algorithm gives standard Back propagation (batch adaptation) for the MLP. Moreover the on-line versions of TBP and BP are obtained as particular cases of CRBP.

Fig. 5 shows the diagram of the RBP applied to the simple IIR-MLP example of Fig. 4, with a simplification of the recursive computation of derivatives, as explained in Section VI. These are the steps of the algorithm for each learning epoch:

- perform forward pass for the entire input sequence saving the states of the network at all times, using (1) and (2);
- start the backward pass computing the error for all the outputs and time instants;
- compute the derivatives in (21) iteratively with null initial conditions;
- for $l = M$ to one
  - —compute $e_n^{(l)}[t]$ by (20) $\forall t \in [1, T]$;
  - —compute $\delta_n^{(l)}[t]$ by (9) $\forall t \in [1, T]$;
  - —compute the weights variations using (11)÷(16);
  - —update weights.

Since the RBP recursive expressions (15), (16), and (21) have the same feedback coefficients as the corresponding IIR filter in the forward expression (1), the learning algorithm calculation will be stable if all the IIR filters are stable.

IV. The On-Line RBP Algorithm Causal Recursive Backpropagation (CRBP)

As previously stated, the RBP algorithm is used only as an intermediate step in the derivation of CRBP. In fact, as shown by (20), the exact RBP algorithm is noncausal, since the $e_n^{(l)}$ at time t depends on the $\delta_q^{(l+1)}$ quantities, taken at future time instants. Therefore the weights update can only be performed in batch mode.

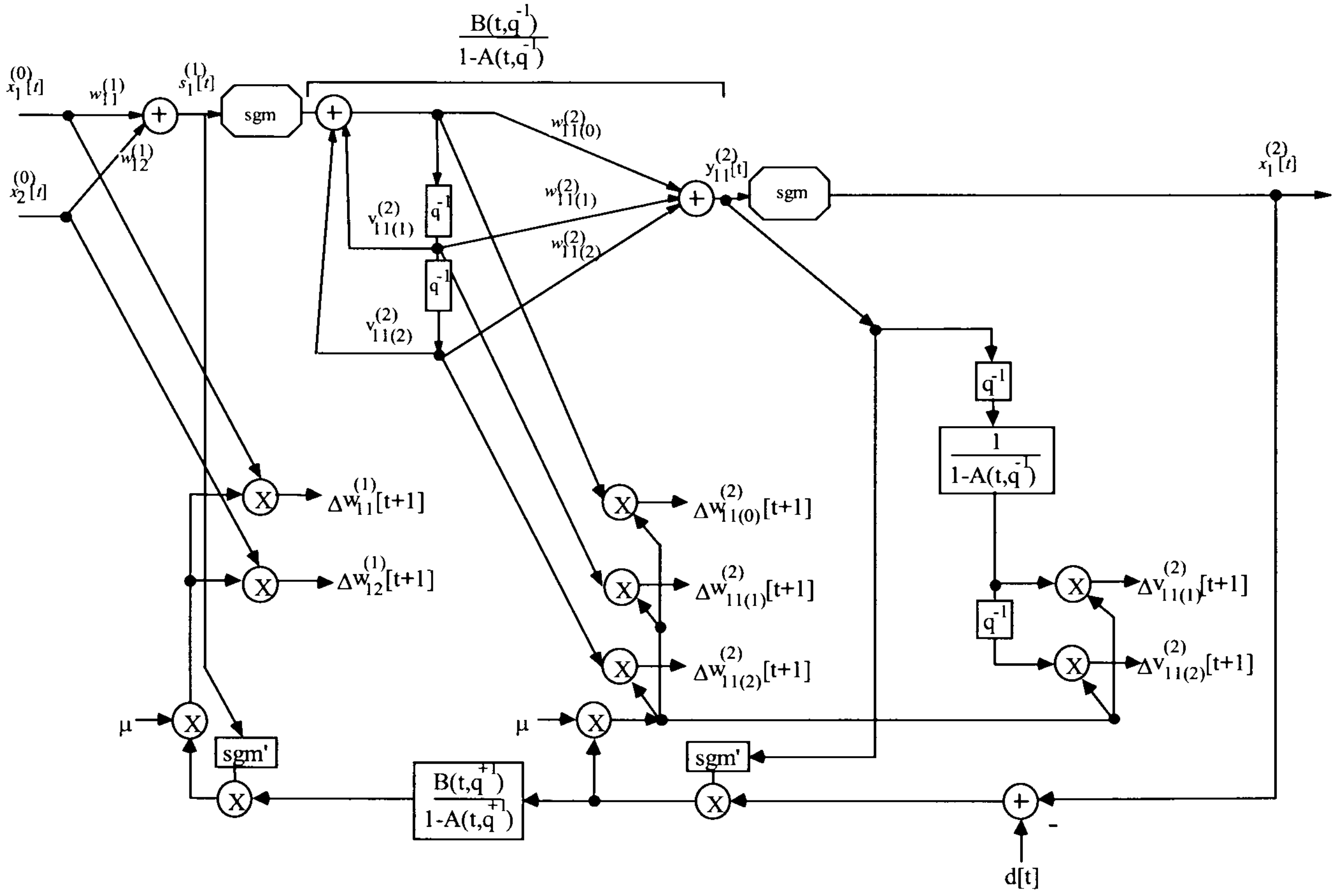

$\frac{B(t,q^{-1})}{1-A(t,q^{-1})}$
$x_1^{(0)}[t]$
$w_{11}^{(1)}$
$s_1^{(1)}[t]$
sgm
$w_{12}^{(1)}$
$x_2^{(0)}[t]$
$w_{11(0)}^{(2)}$
$y_{11}^{(2)}[t]$
sgm
$x_1^{(2)}[t]$
$v_{11(1)}^{(2)}$
q^{-1}
$w_{11(1)}^{(2)}$
$w_{11(2)}^{(2)}$
q^{-1}
$v_{11(2)}^{(2)}$
q^{-1}
$\frac{1}{1-A(t,q^{-1})}$
$\Delta w_{11}^{(1)}[t+1]$
$\Delta w_{12}^{(1)}[t+1]$
$\Delta w_{11(0)}^{(2)}[t+1]$
$\Delta w_{11(1)}^{(2)}[t+1]$
$\Delta w_{11(2)}^{(2)}[t+1]$
q^{-1}
$\Delta v_{11(1)}^{(2)}[t+1]$
$\Delta v_{11(2)}^{(2)}[t+1]$
μ
μ
sgm'
sgm'
$\frac{B(t,q^{+1})}{1-A(t,q^{+1})}$
X
-
d[t]

Fig. 5. The RBP applied to the IIR-MLP example of Fig. 8 (the bias terms are not shown). It is obtained assuming batch mode (no causalization or truncation) and simplified recursive computation of derivatives.

However the RBP algorithm, due to the recursive structure of (15) and (16), can be easily approximated to obtain a very efficient on-line learning algorithm. The on-line approximation consists of three steps: 1) incremental instead of cumulative adaptation; 2) future convolution truncation; and 3) causalization. In this section, the IIR-MLP case is fully considered. The extension to the other LF-MLN architectures is routine, and so we only explain some nonobvious differences.

1) Incremental instead of cumulative adaptation can be implemented using

$$weight\_variation_{nm(p)}^{(l)}[t+1] = \Delta weight_{nm(p)}^{(l)}[t+1] \tag{25}$$

instead of (11) and (13), at each time step, where "$weight$" indicates either w or v. This is the approximation that is usually proposed in literature for dynamic networks. However, a less obvious choice could be done

$$\Delta weight_{nm(p)}^{(l)} = \mu\delta_n^{(l)}[t]\frac{\partial s_n^{(l)}[t]}{\partial weight_{nm(p)}^{(l)}} + \sum_{\tau=1}^{t-1} \Delta weight_{nm(p)}^{(l)}[\tau+1]. \tag{26}$$

This expression can be computed iteratively (with an approximation) that is equivalent to using a momentum term with momentum parameter equal to one. In fact the momentum formula is

$$weight\_variation_{nm(p)}^{(l)}[t+1] = \alpha weight\_variation_{nm(p)}^{(l)}[t] + \mu\delta_n^{(l)}[t]\frac{\partial s_n^{(l)}[t]}{\partial weight_{nm(p)}^{(l)}} \tag{27}$$

where α is the momentum parameter that in general can be chosen in the range: $0 \le \alpha \le 1$. Here we will consider only the incremental adaptation without momentum.

2) If a causalization is desired, a truncation of the future convolution is necessary, due to the infinite memory of the IIR synapses. The truncated formula is therefore

$$e_n^{(l)}[t] = \begin{cases} e_n[t], & \text{for } l = M \\ \displaystyle\sum_{q=1}^{N_{l+1}}\sum_{p=0}^{Q_{l+1}} \delta_q^{(l+1)}[t+p]\frac{\partial y_{qn}^{(l+1)}[t+p]}{\partial x_n^{(l)}[t]}, & \text{for } l = (M-1),\cdots,1 \end{cases} \tag{28}$$

where Q_{l+1} is appropriately chosen. In the particular case when in a given layer $I_{nm}^{(l)} = 0$ for each n, m (i.e., the synapses have finite memory) it is useful to choose

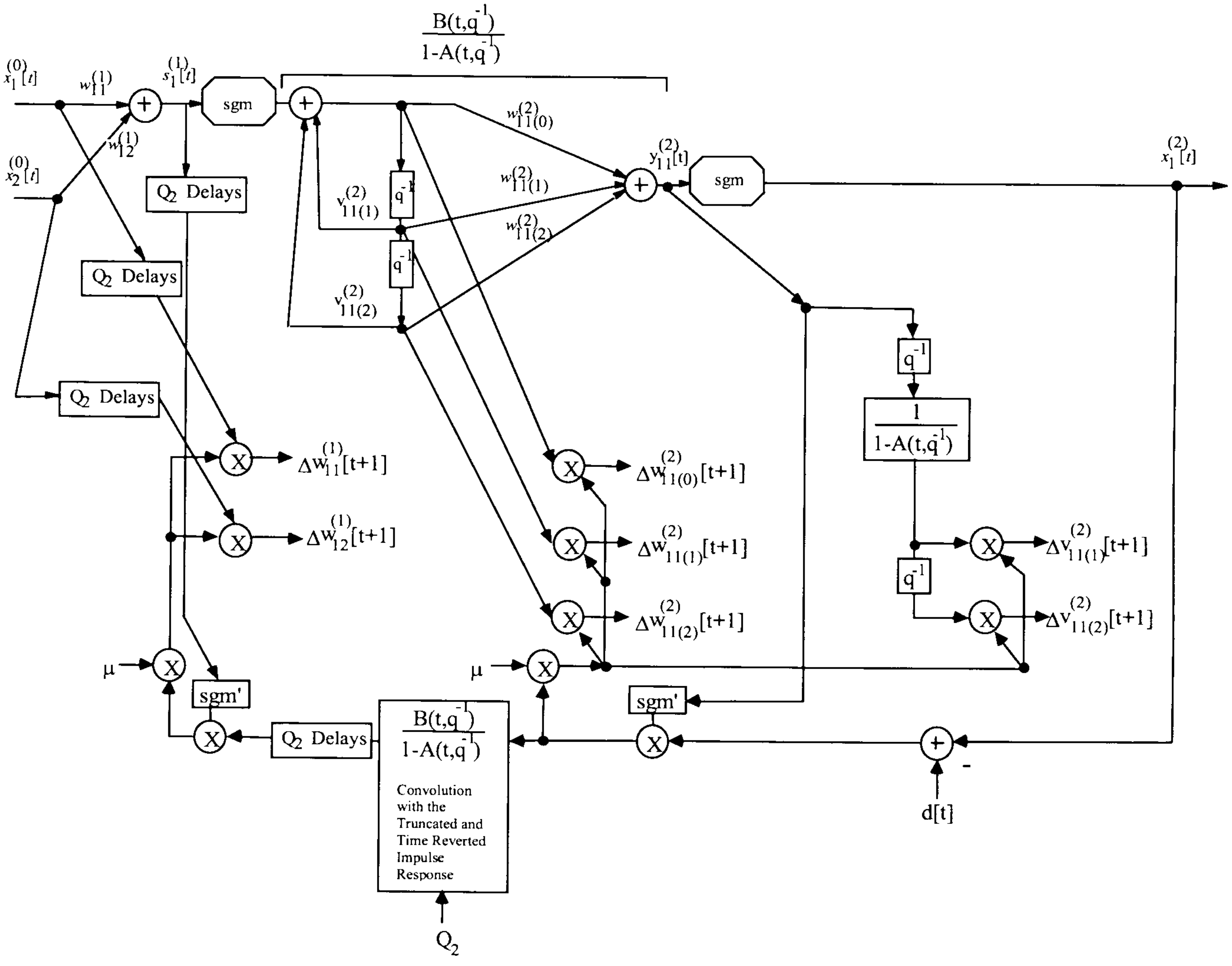

B(t,q^{-1}) / 1-A(t,q^{-1})
$x_1^{(0)}[t]$
$w_{11}^{(1)}$
$s_1^{(1)}[t]$
sgm
$w_{12}^{(1)}$
$x_2^{(0)}[t]$
Q2 Delays
Q2 Delays
Q2 Delays
$w_{11(0)}^{(2)}$
$v_{11(1)}^{(2)}$
q^{-1}
$w_{11(1)}^{(2)}$
$w_{11(2)}^{(2)}$
q^{-1}
$v_{11(2)}^{(2)}$
$y_{11}^{(2)}[t]$
sgm
$x_1^{(2)}[t]$
q^{-1}
1 / 1-A(t,q^{-1})
$\Delta w_{11}^{(1)}[t+1]$
$\Delta w_{12}^{(1)}[t+1]$
$\Delta w_{11(0)}^{(2)}[t+1]$
$\Delta w_{11(1)}^{(2)}[t+1]$
$\Delta w_{11(2)}^{(2)}[t+1]$
$\Delta v_{11(1)}^{(2)}[t+1]$
q^{-1}
$\Delta v_{11(2)}^{(2)}[t+1]$
μ
sgm'
μ
sgm'
Q2 Delays
B(t,q^{-1}) / 1-A(t,q^{-1})
Convolution with the Truncated and Time Reverted Impulse Response
Q2
d[t]
-

Fig. 6. The CRBP applied to the IIR-MLP example of Fig. 8 (the bias terms are not shown). It is obtained assuming on-line mode (with truncation and causalization) and simplified recursive computation of derivatives. The Back–Tsoi approximation uses the same flow diagram but with a multiplication for $w^{(2)}_{11(0)}$ instead of the truncated IIR filtering and $Q_2 = 0$.

$Q_l = \max\limits_{n,m}(L^{(l)}_{nm} - 1)$. In this way, Q_l is set to the maximum memory of the synaptic filters of the layer and no real truncation of the filter response is implemented as in Wan's temporal backpropagation [8], [22].

3) Then we have to introduce a suitable number of delays in the weight adaptation formulas in order to remove the noncausality. In other words

$$weight^{(l)}[t+1] = weight^{(l)}[t] + weight\_variation^{(l)}[t+1-D_l] \quad (29)$$

where D_l is a suitable integer number. It follows that

$$D_l = \begin{cases} 0, & \text{if } l = M \\ \sum\limits_{i=l+1}^{M} Q_i, & \text{if } 1 \le l < M. \end{cases} \quad (30)$$

The causalized formula can be obtained from (28) by reversing the order of the internal summation and issuing the variable change $t+Q_{l+1} \to \tau$, where τ is the current time instant (present)

$$e^{(l)}_n[\tau - Q_{l+1}] = \sum_{q=1}^{N_{l+1}} \sum_{p=0}^{Q_{l+1}} \delta^{(l+1)}_q[\tau - p] \frac{\partial y^{(l+1)}_{qn}[\tau - p]}{\partial x^{(l)}_n[\tau - Q_{l+1}]}, \quad \text{for } l = (M-1), \cdots, 1. \quad (31)$$

In (28) and (31), the trivial hypothesis $\delta[t] = 0$ for $t \notin [1, T]$ has been done. Expression (31) is now causal since it is evaluated at time τ from delta's up to time τ. The impulse response computed by (21) is used by reversing the time scale since $(\partial y^{(l+1)}_{qn}[t-p]/\partial x^{(l)}_n[t - Q_{l+1}]) = h[Q_{l+1} - p]$ in the time invariance hypothesis (or in that approximation), whereas backpropagating errors and delta's are used in the normal time scale, as for standard convolution.

The result is not assigned to the backpropagating error at time τ but at time $\tau - Q_{l+1}$, as dictated by (31). For the sake of clarity, a diagram of CRBP is shown in Fig. 6.

The causalization and the on-line update, compared to the batch mode case, is not a strong approximation if the learning rate is small enough, because in this case the weights variation is small in the time interval of D_l instants. Instead the truncation approximation can be justified by the following property.

- If a linear time invariant IIR filter is asymptotically stable (i.e., all the poles of the transfer function are inside the unit circle) then $(\partial y[t+p]/\partial x[t]) \rightarrow 0$ if $p \rightarrow \infty$ where $y[t]$ is the output of the filter and $x[t]$ the input at time t.

The proof can be done in two ways that are both interesting. The first is just considering that the derivative is the impulse response of the filter that must go to zero in the stable case. The second is considering that in (21) and (24) the recursion coefficients are the same of the corresponding IIR filter therefore their poles must lie inside the unit circle for stability, i.e., the derivative goes to zero as $p \rightarrow \infty$. The second reasoning is more general and it is interesting because it shows a way to verify the validity of the truncation hypothesis for any locally recurrent network architecture, e.g., output feedback MLN. For the derivative to go to zero it is necessary and sufficient that the feedback coefficients in the calculation of (21) or the corresponding expressions in the Appendix, give poles inside the unit circle.

Moreover, it is well known that impulse responses of stable rational transfer functions have an exponentially decaying behavior. This means that the truncation parameter can be chosen quite small. This fact was confirmed by the simulations even if they show that setting the truncation parameter to zero is a too strong approximation that should be avoided.

The previous property can be used to automatically select the desired truncation parameter Q_{l+1} by taking into account the impulse response explicitly computed by the algorithm. In the subsequent discussions this possibility is not further investigated, since for the selected problems a good choice of the truncation parameter was within a very small range.

The condition $(\partial y[t+p]/\partial x[t]) \rightarrow 0$ if $p \rightarrow \infty$ where $y[t]$ is the output (or the net) and $x[t]$ an input of a recurrent neuron [20], [55] holds by definition for each neuron in IIR-MLP, activation-output feedback MLN's and AR-MLP in case each neuron exhibits forgetting behavior. Instead in case of latching behavior (possible only for output feedback MLN), that derivative does not go to zero and the corresponding linear system is unstable [20]. In this case, the truncation of the internal summation in (20′) (see the Appendix) can be too strong an approximation. However, it should be considered that the advantages of networks with local feedback over fully connected ones is especially in modeling a forgetting behavior. Latching behavior is outside normal working conditions for locally recurrent networks.

The algorithm proposed by Back and Tsoi [3], which is the only on-line learning algorithm proposed for locally recurrent networks with no architectural restriction, can be seen as a particular case of our approximation where a strong truncation of the summation is assumed: $Q_{l+1} = 0$ for each l. The simplified formula is

$$e_n^{(l)}[t] = \begin{cases} e_n[t], & \text{for } l = M \\ \displaystyle\sum_{q=1}^{N_{l+1}} \delta_q^{(l+1)}[t] w_{qn(0)}^{(l+1)}, & \text{for } l = (M-1), \cdots, 1. \end{cases} \tag{32}$$

In this way the backpropagation is considering only the instantaneous influence of the IIR filter input on the output (the coefficient $w_{nm(0)}^{(l+1)}$). Hence in the scheme of Fig. 6, the truncated IIR filter should be replaced by a simple multiplication for $w_{11(0)}^{(2)}$. No causalization is needed (because $D_l = 0$ for each l) and the algorithm is very simple. However, we shall show that with the inclusion of only few additional memory terms in the backpropagation ($Q_{l+1} > 0$) it is possible to reach much better stability and speed of convergence.

Also the BPS algorithm (on-line mode) [18] can be obtained as particular case of CRBP under the architectural restriction that the dynamic units can be only placed in the first layer. In this case, like BPS, CRBP implements no real truncation of past history. The CRBP applied to AR-MLP can be also viewed as a generalization of the Leighton–Conrath's work [44], although their formulas do not exactly match the expression corresponding to (32) for the AR-MLP.

V. Experimental Results

The simulation results reported here are of two kinds. The first is a comparison of different locally recurrent architectures with more traditional dynamic MLP using a fixed training method for each one. The second and more important simulation is the comparison of runs of CRBP with different values of the truncation parameter and of CRBP with other training methods, keeping the architecture fixed. Each of these two simulations was run on two different system identification tasks.

Many simulations were performed on three locally recurrent architectures [shown in Fig. 3(b)–(d)] while the AR-MLP [Fig. 3(e)] was not implemented. For comparison purposes, also two traditional neural networks were tested, namely the static MLP with input and possibly output buffer, (shown in Figs. 1 and 2), and the FIR-MLP [Fig. 3(a)]. The results reported here refer to two problems of identification of nonlinear dynamical systems.

The number of delays for the five architectures (i.e., buffers lengths for the buffered MLP's, MA orders for the FIR-MLP, MA, and AR orders for the locally recurrent networks) was chosen in order to obtain the best performance (approximately) for each network, while the total number of free parameter was fixed (40 parameters, bias included), as shown in Table I.

All the networks used had two layers, three hidden neurons with hyperbolic tangent activation function, and one linear output neuron. Three different learning algorithms were used: standard static backpropagation for buffered MLP (with open-loop approximation if feedback is present), temporal backpropagation for FIR-MLP [8] and the proposed CRBP algorithm for the locally recurrent networks. Momentum term and adaptive learning rate were not used. The results are given in terms of mean-square-error (MSE), expressed in dB,

TABLE I
NUMBER OF DELAYS FOR THE DIFFERENT BUFFERS USED FOR ALL THE NEURAL NETWORKS IN BOTH THE SYSTEM IDENTIFICATION EXPERIMENTS. FOR THE THREE LOCALLY RECURRENT ARCHITECTURES THE NUMBER OF DELAYS FOR THE MOVING AVERAGE PART (INPUT BUFFER) AND THE AUTO REGRESSIVE PART (FEEDBACK BUFFER) FOR THE HIDDEN AND OUTPUT LAYERS ARE SPECIFIED. ALL THE NEURAL NETWORKS HAVE THE SAME NUMBER OF ADAPTABLE PARAMETERS, i.e., 40 BIAS INCLUDED, WITH THE ONLY EXCEPTION OF THE ACTIVATION FEEDBACK MLN USED FOR THE BACK–TSOI SYSTEM THAT HAS 34 PARAMETERS. INSIDE BRACKETS THE SHORT NAMES USED IN TABLE II AND FIGS. 11 AND 14

Neural architecture used	**Experiment 1: Back and Tsoi system**	**Experiment 2: PAM system**
Buffered MLP (STAT)	5 input delays; 4 feedback delays (tot. 11 inputs)	10 input delays; no feedback (tot. 11 inputs)
FIR-MLP (FIR)	hidden MA-AR: 5-0; output MA-AR: 5-0	hidden MA-AR: 5-0; output MA-AR: 5-0
IIR-MLP (IIR)	hidden MA-AR: 2-3; output MA-AR: 2-3	hidden MA-AR: 3-2; output MA-AR: 3-2
Activation Feedback MLN (ACT)	hidden MA-AR: 2-3; output MA-AR: 2-3	hidden MA-AR: 4-3; output MA-AR: 2-3
Output Feedback MLN (OUT)	hidden MA-AR: 4-3; output MA-AR: 2-3	hidden MA-AR: 4-3; output MA-AR: 2-3

computed on the learning set after each epoch (after all the input–output samples were presented) and averaged over ten runs, each with a different initialization of the weights. Also, after each iteration the network state was reinitialized.

A. First Simulation: Back–Tsoi System Identification

The first set of experiments consisted in identifying the nonlinear system with memory presented by Back and Tsoi in [12]. This system is described by the following input–output relationship:

$$\begin{cases} z(t) = 0.0154x(t) + 0.0462x(t-1) + 0.0462x(t-2) \\ \qquad +0.0154x(t-3) + 1.99z(t-1) - 1.572z(t-2) \\ \qquad +0.4583z(t-3) \\ y(t) = \sin[z(t)] \end{cases} \tag{33}$$

where $x[t]$ and $y[t]$ are the input and output signals at time t, respectively. The input sequence $x[t]$ consists of 1000 points of white random noise, with a uniform probability density function between -1 and $+1$. This sequence and the corresponding output sequence $y[t]$ constituted the training set for the various neural networks.

From Fig. 7, it is evident that the locally recurrent MLP's exhibit much better modeling capabilities than the static MLP or FIR-MLP, and much better accuracy (asymptotic MSE) too.

The learning was stopped at 200 iterations chosen as a reasonable number of iterations for each architecture. This simulation (and the corresponding one in the second set of experiments) is included basically to justify the choice of this test problem for comparing learning algorithms for locally recurrent neural networks. Since locally recurrent networks perform better than traditional ones on this problem it follows that it is a reasonable choice to test the CRBP learning algorithm; the same is true for the second problem. Fig. 8 shows the performance of the CRBP learning algorithm with different values of the truncation parameters Q_2 ($Q_2 = 0$ is the Back–Tsoi algorithm). It is clear that the CRBP performs much better than the previous algorithm even with small Q_2, i.e., with a small number of recursive terms. A very small truncation parameter is required to obtain good performance, while increasing it beyond a certain small range does not change the MSE appreciably. The BPS algorithm was not included in the comparison because it is not applicable to the architecture selected, since the output neuron is dynamic.

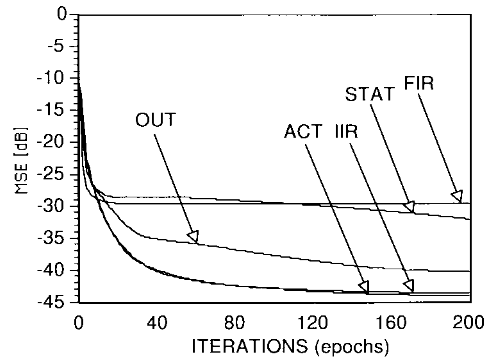
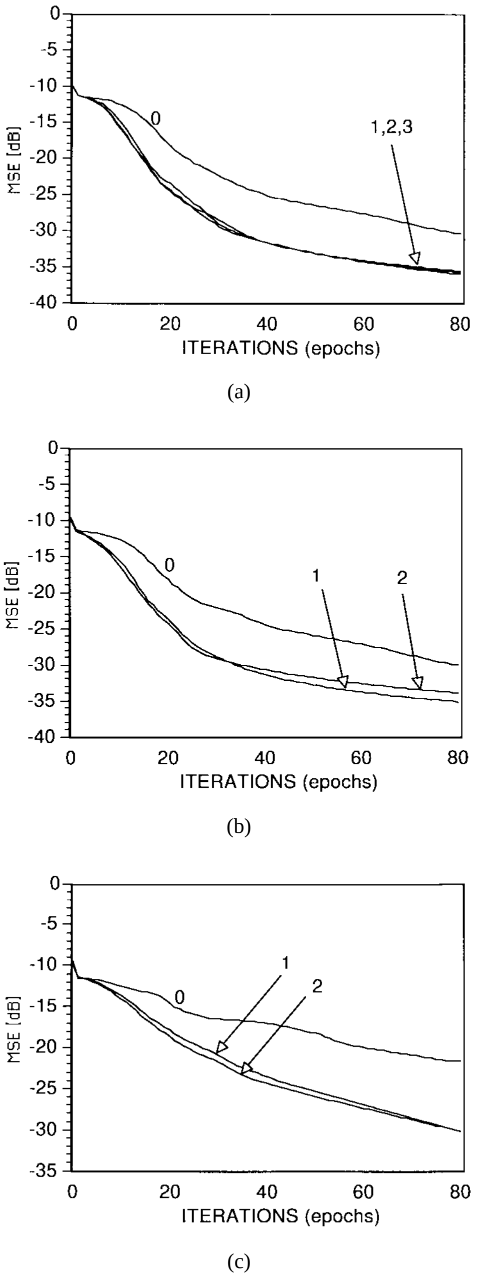


Fig. 7. Convergence performance of various algorithms and networks on identifying the Back–Tsoi test system. STAT is buffered MLP, FIR is FIR-MLP, IIR is IIR-MLP, ACT is activation feedback MLN, OUT is output feedback MLN. Learning rate $\mu = 0.01$. Truncation parameter $Q_2 = 10$. Plots are averaged over ten runs with different weight initializations.

B. Second Simulation: 16-PAM System Identification

The second set of experiments was carried out on the more realistic problem of identifying a baseband equivalent PAM transmission system in the presence of a nonlinearity [14]. The pulse shaping circuit transforms the discrete-time symbols stream $a[n]$ into a continuous-time signal $v(t)$ (PAM) by a filter with a raised-cosine shape and roll-off factor α. The signal $v[t]$ is then processed by the high-power-amplifier (HPA) which is modeled here by the following input–output relationship:

$$w[t] = \frac{2v[t]}{1 + v^2[t]}. \tag{34}$$



(a)

(b)

(c)

Fig. 8. Convergence performance of locally recurrent networks trained by CRBP with various values of the truncation parameter (Q_2) on identifying the Back–Tsoi test system. ($Q_2 = 0$ gives the Back–Tsoi algorithm). Learning rate $\mu = 0.003$. Results for (a) IIR-MLP, (b) activation feedback MLN, and (c) output feedback MLN. The MA and AR orders were chosen, respectively, as 1 and 4, for both the hidden and output layers of all the networks. Plots are averaged over ten runs with different weight initializations.

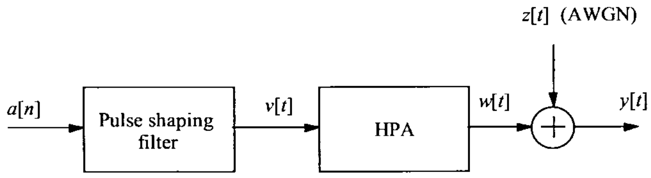


Fig. 9. Block diagram of the PAM transmission channel used in the simulations.

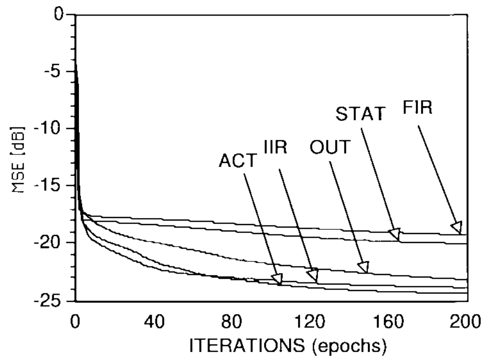


Fig. 10. Convergence performance of various algorithms and networks on identifying the 16-PAM transmission system. STAT is buffered MLP, FIR is FIR-MLP, IIR is IIR-MLP, ACT is activation feedback MLN, OUT is output feedback MLN. Learning rate $\mu = 0.01$. Truncation parameter $Q_2 = 10$. Plots are averaged over ten runs with different weight initializations.

The peak power of the input signal $v[t]$ is set to the value of β dB (back-off factor), with $\beta = 0$ dB being the normalized unit power. The HPA output $w[t]$ is corrupted by an additive white Gaussian noise $z[t]$, producing the final signal $y[t]$ with a given signal-to-noise ratio (SNR) (c.f. Fig. 9). The overall system is clearly dynamic and nonlinear.

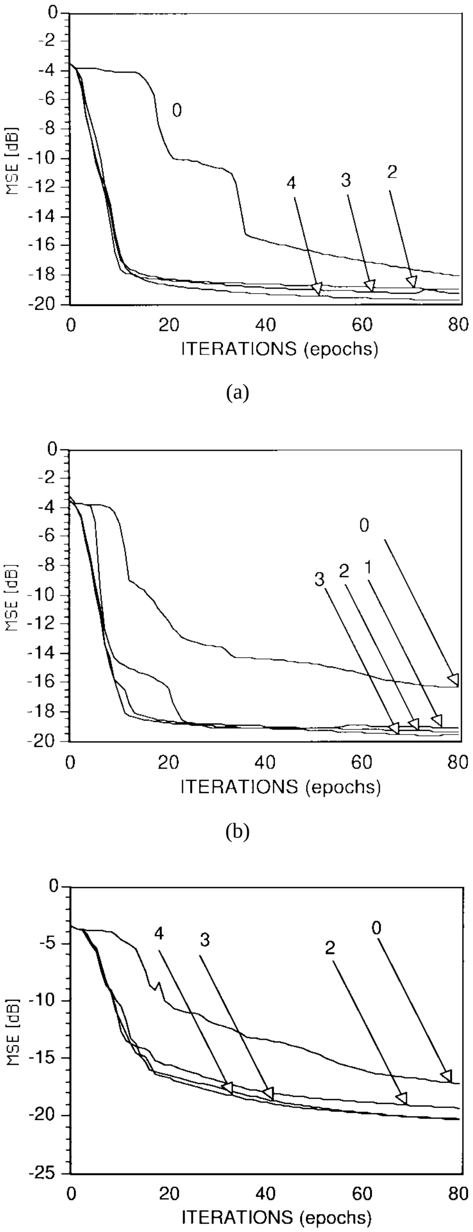
A neural-network approach to equalize this system has already been proposed in the technical literature [15]. In our experiment a neural network was used instead to identify a sampled version of the system. For this purpose, $\{a[n]\}$ was chosen to be a random sequence of 512 symbols drawn from a 16-symbols alphabet. The pulse shaping filter had a roll-off factor $\alpha = 0.3$, and the HPA back-off β was set to -2 dB. The noise level was very low: SNR $= 80$ dB.

By using an over-sampling ratio of four at the output with respect to the symbol rate, the sequences $\{a[n]\}$ of 512 symbols and $\{y[t]\}$ of 2048 samples were used as the learning set and again the MSE was computed after all the 512 input symbols (epoch) were presented.

Fig. 10 shows the performance of the five neural architectures: again the locally recurrent MLP's perform much better than the two conventional MLP's in modeling the system.

Simulations reported in Fig. 11 show that CRBP, is much faster, stable and also more accurate than the Back–Tsoi algorithm that sometimes does not converge. Again, a very small truncation parameter of CRBP is required to obtain good performance, while increasing it over a certain, small, range does not change the MSE appreciably. Generalization tests for the identification of the 16-PAM channel were made using an input symbol sequence different from the one used for learning. Results are given in Tables II and III; they show a MSE on the test set very close to that obtained using the learning set, for all the dynamic architectures and for various choices of the truncation parameter.

The derivation of RTRL and truncated BPTT for locally recurrent neural networks is quite difficult by traditional approaches (chain rule expansions) and is a not yet published result to the best of our knowledge. Recently, a new approach was derived based on signal flow graphs that makes this derivation feasible [59]–[61] but a detailed discussion is



(a)

(b)

(c)

Fig. 11. Convergence performance of locally recurrent networks trained by CRBP with various values of the truncation parameter (Q_2) on identifying the 16-PAM transmission system. ($Q_2 = 0$ gives the Back–Tsoi algorithm). Learning rate $\mu = 0.003$. Results for (a) IIR-MLP, (b) activation feedback MLN, and (c) output feedback MLN. The MA and AR orders were chosen, respectively, as two and three for both the hidden and output layers of all the networks. Plots are averaged over ten runs with different weight initializations.

TABLE II
Generalization Performance of Various Locally Recurrent Neural Networks on Identifying the 16-PAM Transmission System. STAT is Buffered MLP, FIR Is FIR-MLP, IIR Is IIR-MLP, ACT Is Activation Feedback MLN, OUT Is Output Feedback MLN. The Architectures and Parameter Setting Are the Same as for the Results Reported in Fig. 14. Results Are Averaged over Ten Runs with Different Weights Initializations

MLP type	Asymptotic Learning MSE [dB]	Testing MSE [dB]
STAT	-20.04	-20.09
FIR	-19.27	-19.43
IIR	-23.85	-24.39
ACT	-24.39	-24.5
OUT	-23.18	-23.32

TABLE III
Generalization Performances on the Identification of the 16 PAM System of CRBP with Different Values of the Truncation Term Q_2, for a Specific IIR-MLP, Having the Same Architecture as for the Results Reported in Fig. 15(a). Results Are Averaged over Ten Runs with Different Weights Initializations

Truncation Parameter (Q2)	Asymptotic Learning MSE [dB]	Testing MSE [dB]
0 (Back-Tsoi alg.)	-18.0	-18.0
2	-19.0	-19.5
3	-19.2	-19.1
4	-20.7	-19.4
10	-20.8	-19.7
20	-20.9	-19.6

beyond the aim of the paper. For the sake of completeness, the learning performances and the computational complexities have been studied including these algorithms which are derived in [59] and [60]. Fig. 12 shows that CRBP, RTRL and truncated BPTT are substantially equivalent and the Back–Tsoi method is much less performant. With this regard it should be considered that for implementation reasons the truncated BPTT is implemented with weights values at the proper time step in each time in the past history considered [1], while in CRBP they are set as the ones at the current time step, saving further complexity. Moreover, as explained in the following Section, RTRL and truncated BPTT are more complex than CRBP with regard to arithmetic operations, and also CRBP compared to RTRL has the advantage of being local in space.

VI. Analysis of Complexity and Implementation Issues

For the simple case of a fully recurrent, single layer, single delay neural network composed of n neurons, the computational complexity is $O(n^2)$ operations per time step for epochwise BPTT or $O(n^2h)$ for truncated BPTT [1] (where h is the past temporal depth) compared with $O(n^4)$ for RTRL. The memory requirement is $O(nh)$ for epochwise BPTT (in this case h is the epoch length) or truncated BPTT, and $O(n^3)$ for RTRL. Therefore as far as computational complexity is concerned, in batch mode BPTT is significantly simpler than RTRL, whereas in on-line mode the complexity and also the memory requirement ratios depend on n^2/h. Therefore if h is large enough compared to n^2 then Truncated BPTT is more complex than RTRL, but usually the contrary is true.

The recursion implemented in RBP is less efficient with respect to the calculation in BPTT, so for batch mode, BPTT should be preferred, unless the memory requirement is an issue. In this case RTRL can be preferred since for long sequences it requires less memory.

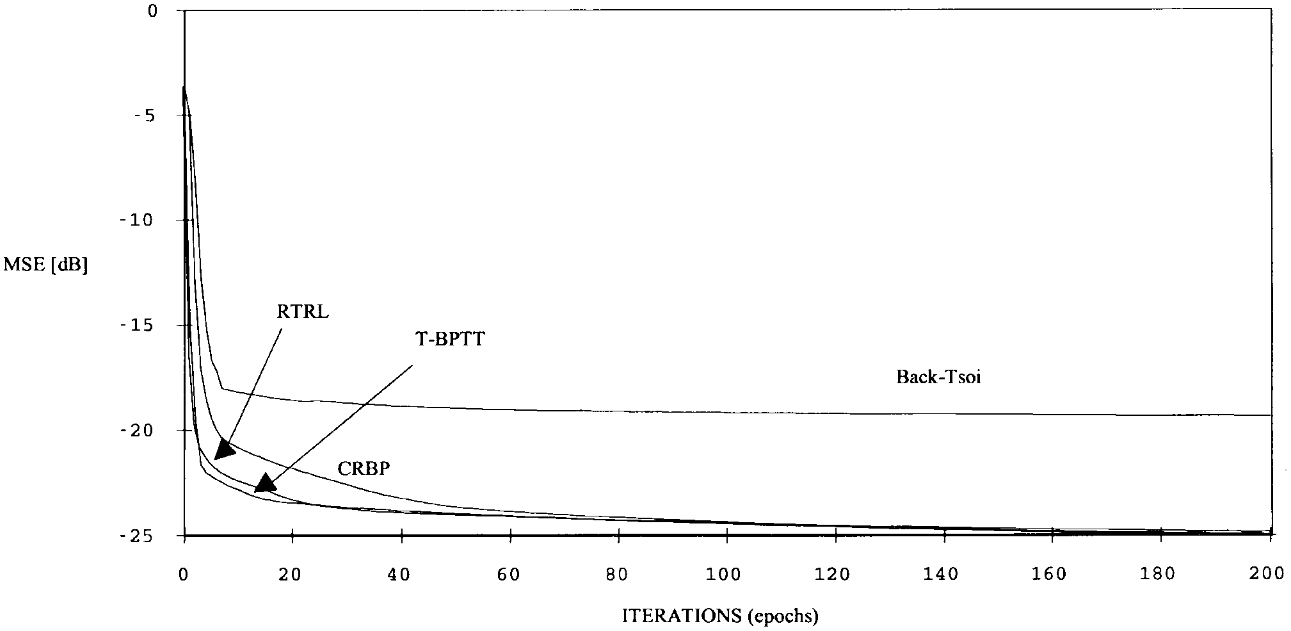


Fig. 12. Convergence performance of an IIR-MLP network trained by CRBP, Back–Tsoi, RTRL, and truncated BPTT on identifying the 16-PAM transmission system. The MA and AR orders were chosen, respectively, as three and two for both the hidden and output layers; $Q_2 = h = 10$. Plots are averaged over ten runs with different weight initializations.

The advantage of RBP is evident only in on-line mode. In this case the proposed CRBP algorithm is more efficient than truncated BPTT, since it has a better scaling of the number of operations increasing the respective truncation parameter, i.e., Q_2 and h.

The explanation of this fact is intuitive and detailed by the analysis of complexity. While in truncated BPTT for each parameter adaptation a summation of the products between delta and the parameter inputs must be computed over the considered h time steps past history, in CRBP the analogous of this summation is computed to calculate the delta (31) that is used to adapt many weights, saving computation. For each parameter adaptation no summation in time is needed but just the product between *delta* and the derivative of the *net* with respect to the parameter [(12) and (14)]. The calculation scheme used is also motivated as a generalization to the recursive case of that implemented in Wan's temporal backpropagation [8], which is the main training method for TDNN, and also of that used in adjoint least mean square algorithm [57], a well known training procedure in the signal processing community.

The increase in computational complexity between the Back–Tsoi algorithm [3] and CRBP is only in formula (28): since, due to the recursion, the truncation parameter (Q_{l+1}) can be usually chosen quite small (verified in our simulations and theoretically motivated by the exponential decaying of impulse responses of stable rational transfer functions) making this increase fairly small. This holds of course in the case of forgetting behavior and not in the case of long-term dependencies but the first is the normal working condition of network with local feedback. In our simulations, we observed that the ratio between the execution times of CRBP and Back–Tsoi algorithm (one iteration) is less than 1.5, for usual architectures and parameter settings.

A mathematical evaluation of complexity can be carried out computing the number of multiplications and additions for one iteration (i.e., one sample here) of the learning phase (on-line learning). In Table IV, results for CRBP, Back–Tsoi, RTRL, and truncated BPTT are reported in the significant special case of two layers IIR-MLP with bias and with MA-AR orders depending only on the layer index: $M = 2$, $L_{nm}^{(l)} = L^{(l)}$, $I_{nm}^{(l)} = I^{(l)}$.

The numbers in Table IV must be added to the number of operations of the forward phase, always done before the backward phase.

The number of multiplications or additions of the forward phase (one iteration) is

$$\begin{aligned} M_{\text{forward}} &= A_{\text{forward}} \\ &= N_1 N_0 \left(L^{(1)} + I^{(1)}\right) + N_2 N_1 \left(L^{(2)} + I^{(2)}\right). \end{aligned} \quad (35)$$

To provide an easy evaluation of complexity, Tables V and VI report the numbers of operations obtained from Table IV for the parameters choice of some simulations reported here. These values should be added to the number of forward operations before computing complexity ratios between different algorithms.

By these formulas, substituting the configuration parameters of each of the three IIR-MLP networks used in the simulations reported, and with $Q_2 = 4$, it is possible to get an averaged ratio $M_{\text{CRBP}}/M_{\text{Back–Tsoi}} = 1.24$ and $A_{\text{CRBP}}/A_{\text{Back–Tsoi}} = 1.18$, proving that the increase in complexity is quite small.

Moreover, for IIR-MLP and activation feedback MLN it is possible to apply to CRBP the same simplification proposed by Back and Tsoi in [12] for their algorithm. The idea

TABLE IV
NUMBER OF OPERATIONS FOR ONE ITERATION OF THE ON-LINE LEARNING PHASE FOR THE DIFFERENT ALGORITHMS: GENERAL EXPRESSIONS

Learning algorithm	# ADDITIONS	# MULTIPLICATIONS
CRBP	$2N_2 + N_1N_0\left[I^{(1)} + L^{(1)}(I^{(1)}+1) + (I^{(1)})^2\right] + N_2N_1\Big[I^{(2)} + L^{(2)}(I^{(2)}+1) +$ $+(I^{(2)})^2 + Q_2 + 1 + \sum_{p=0}^{Q_2}\min(I^{(2)},p) - \begin{cases} Q_2 - L^{(2)} + 1 & \text{if } Q_2 > L^{(2)} - 1 \\ 0 & \text{otherwise}\end{cases}\Big]$	$2(N_1+N_2) + N_1N_0\left[I^{(1)} + L^{(1)}(I^{(1)}+1) + (I^{(1)})^2\right] +$ $+N_2N_1\left[I^{(2)} + L^{(2)}(I^{(2)}+1) + (I^{(2)})^2 + Q_2 + 1 + \sum_{p=0}^{Q_2}\min(I^{(2)},p)\right]$
Back-Tsoi	$2N_2 + N_1N_0\left[I^{(1)} + L^{(1)}(I^{(1)}+1) + (I^{(1)})^2\right] + N_2N_1\Big[I^{(2)} + L^{(2)}(I^{(2)}+1) +$ $+(I^{(2)})^2 + 1\Big]$	$2(N_1+N_2) + N_1N_0\left[I^{(1)} + L^{(1)}(I^{(1)}+1) + (I^{(1)})^2\right] +$ $+N_2N_1\left[I^{(2)} + L^{(2)}(I^{(2)}+1) + (I^{(2)})^2 + 1\right]$
RTRL	$2N_2 + N_1N_0\left(I^{(1)} + L^{(1)}\right)\left[\left(I^{(2)} + L^{(2)}\right)N_2 + I^{(1)}\right] + N_2N_1\left(I^{(2)} + L^{(2)}\right)\left(I^{(2)} + 2\right)$	$3N_2 + N_1N_0\left(I^{(1)} + L^{(1)}\right)\left[\left(I^{(2)} + L^{(2)}\right)N_2 + I^{(1)} + 2N_2 + 1\right] +$ $+N_2N_1\left(I^{(2)} + L^{(2)}\right)\left(I^{(2)} + 3\right) + 2N_2N_1 + N_1$
T-BPTT	$N_2 + h\left[N_1N_0\left(2I^{(1)} + L^{(1)}\right) + 2N_2N_1\left(I^{(2)} + L^{(2)}\right) - N_2\left(N_1 - 1\right) + N_1\right]$	$N_2 + h\left[N_1N_0\left(2I^{(1)} + L^{(1)} + 1\right) + 2N_2N_1\left(I^{(2)} + L^{(2)}\right) + N_1\left(N_2 + 1\right)\right]$

TABLE V
NUMBER OF OPERATIONS FOR ONE ITERATION OF THE ON-LINE LEARNING PHASE FOR THE DIFFERENT ALGORITHMS WHEN:
$N_0 = 1, N_1 = 3, N_2 = 1, L^{(1)} = L^{(2)} = 3,$
$I^{(1)} = I^{(2)} = 2, Q_2 = 10, h = 10$

Learning Algorithm	# ADDITIONS	# MULTIPLICATIONS
CRBP	140	188
Back-Tsoi	83	101
RTRL	167	237
T-BPTT	521	601

TABLE VI
NUMBER OF OPERATIONS FOR ONE ITERATION OF THE ON-LINE LEARNING PHASE FOR THE DIFFERENT ALGORITHMS WHEN:
$N_0 = 1, N_1 = 3, N_2 = 1, L^{(1)} = L^{(2)} = 3,$
$I^{(1)} = I^{(2)} = 2, Q_2 = 4, h = 4$

Learning Algorithm	# ADDITIONS	# MULTIPLICATIONS
CRBP	104	134
Back-Tsoi	83	101
RTRL	167	237
T-BPTT	209	241

is exactly the same used in the adaptive IIR filter context [13], [43] and is explained in the following, considering only IIR-MLP for the sake of simplicity. To compute the $y_{nm}^{(l)}$ quantity we have to pass the input of the synapse through the IIR synaptic filter; while to compute $(\partial s_n^{(l)}[t]/\partial w_{nm(0)}^{(l)})$ we need to pass the same input through the AR part only of the same filter. So, implementing the IIR filter in direct form II (i.e., filtering the synaptic input during the forward phase separately with the AR and MA parts) and keeping the intermediate value, the $(\partial s_n^{(l)}[t]/\partial w_{nm(0)}^{(l)})$ quantity is immediately available during the backward phase without further computations.

Moreover it is possible to introduce an approximation [12] to get an additional decrease on complexity: computing the different values of $(\partial s_n^{(l)}[t]/\partial weight_{nm(p)}^{(l)})$ ($weight$ is w or v) for the different p as the same quantity obtained for the first p delayed by a suitable number of time steps, as shown in Fig. 6. This approximation is already known in the adaptive IIR filter theory [13], [43] and is reasonable if the coefficients change slowly in time (a small learning rate is used) since it assumes the time invariance of the filter. The simulations do not show an appreciable loss of performance in the learning process [12], [13], [43], while the number of AR filtering operation per synapse is reduced from $L_{nm}^{(l)} + I_{nm}^{(l)}$ to one. Even if this approximation works well in our simulations we used the exact formulas.

We tested another minor simplification of the CRBP formulas, neglecting the recursive part of (15), (16), and (21) (i.e., assuming null past values) as it is sometimes done for IIR linear adaptive filters. However, the obtained performance of just this minor simplification were so unsatisfactory that we did not use it anymore.

About the stability of the CRBP algorithm, in all the simulations we observed that, if the IIR network was initialized with stable synapses, the final result is a stable network as long as a small enough learning rate is used. A similar behavior was exhibited by all the other locally recurrent architectures.

VII. CONCLUSIONS

In conclusion, the results presented in this work show that the locally recurrent MLP's have superior modeling capabilities with respect to more traditional networks, namely MLP with external memory and FIR-MLP (or TDNN) for identification of the systems tested; see [39] for a comparison on other test problems. We described a general approach for deriving on-line algorithms for locally recurrent networks that are local both in time and in space and proposed a new learning method, that, includes as special cases, several algorithms already known in the literature. The proposed algorithm has better stability and higher speed of convergence compared to the Back–Tsoi method, as expected by the theoretical development and confirmed by simulations. Stability and speed of convergence are very important in real on-line applications, e.g., where time varying systems have to be tracked. The only drawback of the algorithm is a slight increase in complexity with respect to Back–Tsoi method, which however can be easily reduced.

With respect to an application of RTRL and truncated BPTT, the proposed algorithm is computationally simpler and easier to implement.

APPENDIX
RBP FOR LOCAL FEEDBACK MLN'S AND AUTOREGRESSIVE MLP

For uniformity of presentation, the RBP algorithm is covered in this Appendix instead of CRBP. Of course the same modifications explained in Section IV can be implemented to derive the CRBP from the RBP method.

In all of the following cases if no feedback is present ($I_n^{(l)} = 0$ for each n and l), each architecture becomes a FIR-MLP and the RBP algorithm gives Wan's temporal backpropagation (batch mode). If no memory is present at all ($I_n^{(l)} = 0$ and $L_{nm}^{(l)} = 1$ for each $n, m,$ and l) then each architecture becomes a standard MLP and RBP gives BP (batch mode). The on-line mode versions of TBP and BP are obtained as particular case of CRBP.

The forward and backward formulas for LF-MLN's and AR-MLP differ from those already derived for the IIR-MLP because the AR filtering is performed once for every neuron and not for every synapse, lowering the number of free parameters. Therefore the input index of the AR coefficients (v), and the AR order (I) is now meaningless and will be dropped in the following.

A. Activation Feedback MLN

Only the formulas which are different from those corresponding to the IIR-MLP are reported here. They can be easily derived in a similar way. The forward formulas are now replaced by

$$s_n^{(l)}[t] = \sum_{m=1}^{N_{l-1}} \sum_{p=0}^{L_{nm}^{(l)}-1} w_{nm(p)}^{(l)} x_m^{(l-1)}[t-p] + \sum_{p=1}^{I_n^{(l)}} v_{n(p)}^{(l)} s_n^{(l)}[t-p] \tag{1'}$$

$$x_n^{(l)}[t] = \mathrm{sgm}\Big(s_n^{(l)}[t] + w_{n0}^{(l)}\Big). \tag{2'}$$

It should be noted that now $s_n^{(l)}[t]$ does not represent the net quantity anymore since it differs from that for the additive bias. Due to this bias term the relation between delta and backpropagating error is

$$\delta_n^{(l)}[t] = e_n^{(l)}[t]\,\mathrm{sgm}'\Big(s_n^{(l)}[t] + w_{n0}^{(l)}\Big). \tag{9'}$$

The coefficient variations are now computed using the derivatives expressed by

$$\frac{\partial s_n^{(l)}[t]}{\partial w_{nm(p)}^{(l)}} = x_m^{(l-1)}[t-p] + \sum_{r=1}^{I_n^{(l)}} v_{n(r)}^{(l)} \frac{\partial s_n^{(l)}[t-r]}{\partial w_{nm(p)}^{(l)}} \tag{15'}$$

$$\frac{\partial s_n^{(l)}[t]}{\partial v_{n(p)}^{(l)}} = s_n^{(l)}[t-p] + \sum_{r=1}^{I_n^{(l)}} v_{n(r)}^{(l)} \frac{\partial s_n^{(l)}[t-r]}{\partial v_{n(p)}^{(l)}}. \tag{16'}$$

The backpropagation is now

$$e_n^{(l)}[t] = \begin{cases} e_n[t], & \text{for } l = M \\ \displaystyle\sum_{q=1}^{N_{l+1}} \sum_{p=0}^{T-t} \delta_q^{(l+1)}[t+p] \frac{\partial s_q^{(l+1)}[t+p]}{\partial x_n^{(l)}[t]}, & \\ \qquad \text{for } l = (M-1), \cdots, 1 \end{cases} \tag{20'}$$

where the derivatives are computed as

$$\frac{\partial s_q^{(l+1)}[t+p]}{\partial x_n^{(l)}[t]} = \left(\begin{cases} w_{qn(p)}^{(l+1)}, & \text{if } 0 \le p \le L_{qn}^{(l+1)} - 1 \\ 0, & \text{otherwise} \end{cases} \right) + \sum_{r=1}^{\min(I_q^{(l+1)},\, p)} v_{q(r)}^{(l+1)} \frac{\partial s_q^{(l+1)}[t+p-r]}{\partial x_n^{(l)}[t]}. \tag{21'}$$

Backpropagation for sequences [18] (batch mode) for activation feedback MLN is obtained as a particular case if the architecture is constrained to have dynamic units only in the first layer. In our notation this is written: $I_n^{(l)} = 0$ and $L_{nm}^{(l)} = 1$ for $l > 1$ and for each $n,\ m$. In this case standard BP is applied to compute delta since (20′) and (21′) become

$$e_n^{(l)}[t] = \begin{cases} e_n[t], & \text{for } l = M \\ \displaystyle\sum_{q=1}^{N_{l+1}} \delta_q^{(l+1)}[t] w_{qn}^{(l+1)}, & \text{for } l = (M-1), \cdots, 1. \end{cases}$$

Since RBP is a batch mode algorithm (cumulative adaptation of weights) the particular case is still a batch mode algorithm. The on-line original version of BPS is obtained from the CRBP or simply substituting cumulative with incremental adaptation in RBP.

B. Output Feedback MLN

The important difference of output feedback MLN [Fig. 3(d)] with respect to IIR-MLP and activation feedback MLN is that in the former the dynamic part and the nonlinearity are not separated anymore.

Now it is not possible to look at the internal summation of (20) as future convolution, because the derivative is not an impulse response anymore, due to the nonlinearity. The differences in the learning formulas with respect to the activation feedback MLN is basically that the AR coefficients are now multiplied with the derivative of the activation function computed on the net in a certain time instant, as it is easy to prove by chain rule.

Only the formulas which are different from those corresponding to the activation feedback MLN are reported here. The forward formulas are now replaced by

$$s_n^{(l)}[t] = \sum_{m=0}^{N_{l-1}} \sum_{p=0}^{L_{nm}^{(l)}-1} w_{nm(p)}^{(l)} x_m^{(l-1)}[t-p] + \sum_{p=1}^{I_n^{(l)}} v_{n(p)}^{(l)} x_n^{(l)}[t-p] \tag{1''}$$

$$x_n^{(l)}[t] = \mathrm{sgm}\Big(s_n^{(l)}[t]\Big). \tag{2''}$$

As for the IIR-MLP it holds

$$\delta_n^{(l)}[t] = e_n^{(l)}[t]\,\mathrm{sgm}'\Big(s_n^{(l)}[t]\Big). \qquad (9'')$$

The coefficient variations are now computed using the derivatives expressed by

$$\frac{\partial s_n^{(l)}[t]}{\partial w_{nm(p)}^{(l)}} = x_m^{(l-1)}[t-p] + \sum_{r=1}^{I_n^{(l)}} v_{n(r)}^{(l)}\,\mathrm{sgm}'\Big(s_n^{(l)}[t-r]\Big) \cdot \frac{\partial s_n^{(l)}[t-r]}{\partial w_{nm(p)}^{(l)}} \qquad (15'')$$

$$\frac{\partial s_n^{(l)}[t]}{\partial v_{n(p)}^{(l)}} = x_n^{(l)}[t-p] + \sum_{r=1}^{I_n^{(l)}} v_{n(r)}^{(l)}\,\mathrm{sgm}'\Big(s_n^{(l)}[t-r]\Big) \cdot \frac{\partial s_n^{(l)}[t-r]}{\partial v_{n(p)}^{(l)}}. \qquad (16'').$$

The derivatives used in the backpropagation are now computed as

$$\frac{\partial s_q^{(l+1)}[t+p]}{\partial x_n^{(l)}[t]} = \left(\begin{cases} w_{qn(p)}^{(l+1)}, & \text{if } 0 \le p \le L_{qn}^{(l+1)} - 1 \\ 0, & \text{otherwise} \end{cases}\right) + \sum_{r=1}^{\min(I_q^{(l+1)},p)} v_{q(r)}^{(l+1)}\,\mathrm{sgm}'\Big(s_q^{(l+1)}[t+p-r]\Big) \cdot \frac{\partial s_q^{(l+1)}[t+p-r]}{\partial x_n^{(l)}[t]}. \qquad (21'').$$

As for the previous case, also for output feedback MLN, BPS learning algorithm is obtained, as particular case, if the architecture is constrained to have dynamic units only in the first layer.

C. Autoregressive MLP

The main difference with all the previous cases is that now the chain rule expansions are more easily written with respect to the neuron outputs instead of the net quantities. This is because the AR memory is not included in the *net*, see Fig. 3(e). So delta's are not useful anymore and they will be replaced by the backpropagating errors. The forward formulas are now replaced by

$$s_n^{(l)}[t] = \sum_{m=0}^{N_{l-1}} \sum_{p=0}^{L_{nm}^{(l)}-1} w_{nm(p)}^{(l)} x_m^{(l-1)}[t-p] \qquad (1''')$$

$$x_n^{(l)}[t] = \mathrm{sgm}\Big(s_n^{(l)}[t]\Big) + \sum_{p=1}^{I_n^{(l)}} v_{n(p)}^{(l)} x_n^{(l)}[t-p]. \qquad (2''').$$

The w coefficient variations are expressed by

$$\Delta w_{nm(p)}^{(l)}[t+1] = -\frac{\mu}{2}\frac{\partial E^2}{\partial x_n^{(l)}[t]}\frac{\partial x_n^{(l)}[t]}{\partial w_{nm(p)}^{(l)}} = \mu e_n^{(l)}[t]\,\frac{\partial x_n^{(l)}[t]}{\partial w_{nm(p)}^{(l)}}. \qquad (12''')$$

Similarly for the v weights we have

$$\Delta v_{nm(p)}^{(l)}[t+1] = \mu e_n^{(l)}[t]\,\frac{\partial x_n^{(l)}[t]}{\partial v_{nm(p)}^{(l)}}. \qquad (14''')$$

The coefficient variations are now computed using the derivatives expressed by

$$\frac{\partial x_n^{(l)}[t]}{\partial w_{nm(p)}^{(l)}} = \mathrm{sgm}'\Big(s_n^{(l)}[t]\Big)x_m^{(l-1)}[t-p] + \sum_{r=1}^{I_n^{(l)}} v_{n(r)}^{(l)}\frac{\partial x_n^{(l)}[t-r]}{\partial w_{nm(p)}^{(l)}} \qquad (15''')$$

$$\frac{\partial x_n^{(l)}[t]}{\partial v_{n(p)}^{(l)}} = x_n^{(l)}[t-p] + \sum_{r=1}^{I_n^{(l)}} v_{n(r)}^{(l)}\frac{\partial x_n^{(l)}[t-r]}{\partial v_{n(p)}^{(l)}}. \qquad (16''')$$

The backpropagation is now

$$e_n^{(l)}[t] = \begin{cases} e_n[t], & \text{for } l = M \\ \displaystyle\sum_{q=1}^{N_{l+1}} \sum_{p=0}^{T-t} e_q^{(l+1)}[t+p]\,\frac{\partial x_q^{(l+1)}[t+p]}{\partial x_n^{(l)}[t]}, & \text{for } l = (M-1), \cdots, 1 \end{cases} \qquad (20''')$$

where the derivatives are computed as

$$\frac{\partial x_q^{(l+1)}[t+p]}{\partial x_n^{(l)}[t]} = \mathrm{sgm}'\Big(s_q^{(l+1)}[t+p]\Big)\left(\begin{cases} w_{qn(p)}^{(l+1)}, & \text{if } 0 \le p \le L_{qn}^{(l+1)} - 1 \\ 0, & \text{otherwise} \end{cases}\right) + \sum_{r=1}^{\min(I_q^{(l+1)},p)} v_{q(r)}^{(l+1)}\frac{\partial x_q^{(l+1)}[t+p-r]}{\partial x_n^{(l)}[t]}. \qquad (21''')$$

ACKNOWLEDGMENT

P. Campolucci wishes to thank L. A. Feldkamp, C. L. Giles, M. Gori, B. Pearlmutter, A. C. Tsoi, and E. Wan for interesting discussions on this research. The authors wish to thank the anonymous reviewers for providing useful comments and suggestions that contributed to improve the quality of this paper.

REFERENCES

[1] R. J. Williams and J. Peng, "An efficient gradient-based algorithm for on line training of recurrent network trajectories," *Neural Comput.,* vol. 2, pp. 490–501, 1990.

[2] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Comput.,* vol. 1, pp. 270–280, 1989.

[3] A. D. Back and A. C. Tsoi, "FIR and IIR synapses, a new neural network architecture for time series modeling," *Neural Comput.,* vol. 3, pp. 375–385, 1991.

[4] A. C. Tsoi and A. D. Back, "Locally recurrent globally feedforward networks: A critical review of architectures," *IEEE Trans. Neural Networks,* vol. 5, pp. 229–239, Mar. 1994.

[5] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," *Parallel Distributed Processing: Explorations in the Microstructure of Cognition,* vol. 1, D. E. Rumelhart, J. L. McClelland, and the PDP Research Group, Eds. Cambridge, MA: MIT Press, 1986.

[6] P. J. Werbos, "Beyond regression: New tools for prediction and analysis in the behavioral sciences," Ph.D. dissertation, Committee on Appl. Math., Harvard Univ., Cambridge, MA, Nov. 1974.

[7] ——, "Backpropagation through time: What it does and how to do it," in *Proc. IEEE, Special Issue on Neural Networks,* Oct. 1990, vol. 78, pp. 1550–1560.

[8] E. A. Wan, "Temporal backpropagation for FIR neural networks," in *Proc. Int. Joint Conf. Neural Networks,* 1990, vol. 1, pp. 575–580.

[9] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang, "Phoneme recognition using time-delay neural networks," *IEEE Trans. Acoust., Speech, Signal Processing,* vol. 37, Mar. 1989.

[10] K. J. Lang, A. H. Waibel, and G. E. Hinton, "A time-delay neural network architecture for isolated word recognition," *Neural Networks,* vol. 3, pp. 23–43, 1990.

[11] N. Benvenuto, F. Piazza, and A. Uncini, "Comparison of four learning algorithms for multilayer perceptron with FIR synapses," in *Proc. IEEE Int. Conf. Neural Networks,* 1994.

[12] A. D. Back and A. C. Tsoi, "A simplified gradient algorithm for IIR synapse multilayer perceptron," *Neural Comput.,* vol. 5, pp. 456–462, 1993.

[13] J. J. Shynk, "Adaptive IIR filtering," *IEEE ASSP Mag.,* Apr. 1989.

[14] J. G. Proakis, *Digital Communications.* New York: McGraw-Hill, 1989.

[15] S. Chen, G. J. Gibson, C. F. N. Cowan, and P. M. Grant, "Adaptive equalization of finite nonlinear channels using multilayer perceptrons," *Signal Processing,* vol. 20, pp. 107–119, June 1990.

[16] P. Campolucci, F. Piazza, and A. Uncini, "On-line learning algorithms for neural networks with IIR synapses," in *Proc. IEEE Int. Conf. Neural Networks*, Perth, Australia, Nov. 1995.

[17] P. Campolucci, A. Uncini, and F. Piazza, "Causal Backpropagation through time for locally recurrent neural networks," in *Proc. IEEE Int. Symp. Circuits Syst.,* Atlanta, GA, May 1996.

[18] M. Gori, Y. Bengio, and R. De Mori, "BPS: A learning algorithm for capturing the dynamic nature of speech," in *Proc. Int. Joint Conf. Neural Networks,* 1989.

[19] Y. Bengio, R. De Mori, and M. Gori, "Learning the dynamic of speech with backpropagation for sequences," *Pattern Recognition Lett.,* vol. 13, pp. 375–385, 1992.

[20] P. Frasconi, M. Gori, and G. Soda, "Local feedback multilayered networks," *Neural Comput.,* vol. 4, pp. 120–130, 1992.

[21] P. Frasconi, "Reti ricorrenti ed elaborazione adattativa di sequenze," Ph.D. dissertation, Dip. di Sistemi e Informatica, Università di Firenze, Italy, Feb. 1994 (in Italian).

[22] S. Haykin, *Neural Networks: A Comprehensive Foundation.* New York: IEEE Press, 1994.

[23] J. Schmidhuber, "A fixed size storage $O(n^3)$ time complexity learning algorithm for fully recurrent continually running networks," *Neural Comput.,* vol. 4, pp. 243–248, 1992.

[24] B. A. Pearlmutter, "Gradient calculations for dynamic recurrent neural networks: A survey," *IEEE Trans. Neural Networks,* vol. 6, Sept. 1995.

[25] B. Srinivasan, U. R. Prasad, and N. J. Rao, "Backpropagation through adjoints for the identification of non linear dynamic systems using recurrent neural models," *IEEE Trans. Neural Networks,* pp. 213–228, Mar. 1994.

[26] E. A. Wan, and F. Beaufays, "Diagrammatic derivation of gradient algorithms for neural networks," *Neural Comput.,* vol. 8, pp. 182–201, 1996.

[27] K. S. Narendra and K. Parthasarathy, "Gradient methods for the optimization of dynamical systems containing neural networks," *IEEE Trans. Neural Networks,* vol. 2, Mar. 1991.

[28] ———, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neural Networks,* vol. 1, pp. 4–27, Mar. 1990.

[29] G. V. Puskorius and L. A. Feldkamp, "Neurocontrol of nonlinear dynamical systems with Kalman filter-trained recurrent networks," *IEEE Trans. Neural Networks,* vol. 5, pp. 279–297, Mar. 1994.

[30] C.-C. Ku and K. Y. Lee, "Diagonal recurrent neural networks for dynamic systems control," *IEEE Trans. Neural Networks,* vol. 6, Jan. 1995.

[31] O. Nerrand, P. Roussel-Ragot, L. Personnaz, G. Dreyfus, and S. Marcos, "Neural networks and nonlinear adaptive filtering: Unifying concepts and new algorithms," *Neural Comput.,* vol. 5, pp. 165–199, 1993.

[32] B. De Vries and J. C. Principe, "The gamma model—A new neural model for temporal processing," *Neural Networks,* vol. 5, pp. 565–576, 1992.

[33] M. A. Motter and J. C. Principe, "A gamma memory neural network for system identification," in *Int. Conf. Neural Networks,* pp. 3232–3237, 1994.

[34] S. P. Day and M. R. Davenport, "Continuous-time temporal backpropagation with adaptable time delays," *IEEE Trans. Neural Networks,* vol. 4, Mar. 1993.

[35] D.-T. Lin, J. E. Dayhoff, and P. A. Ligomenides, "Trajectory production with the adaptive time-delay neural network," *Neural Networks,* vol. 8, no. 3, pp. 447–461, 1995.

[36] F. Beaufays and E. Wan, "Relating real-time backpropagation and backpropagation-through-time: An application of flow graph interreciprocity," *Neural Comput.,* vol. 6, pp. 296–306, 1994.

[37] A. D. Back, E. Wan, S. Lawrence, and A. C. Tsoi, "A unifying view of some training algorithms for multilayer perceptrons with FIR filter synapses," in *Proc. IEEE Workshop Neural Networks Signal Processing,* pp. 146–154, 1994.

[38] B. A. Pearlmutter, "Two new learning procedures for recurrent networks," *Neural Networks Rev.,* vol. 3, no. 3, pp. 99–101, 1990.

[39] B. G. Horne and C. L. Giles, "An experimental comparison of recurrent neural networks," in *Advances in Neural Information Processing Systems*, vol. 7. Cambridge, MA: MIT Press, 1995, p. 697.

[40] J. L. Elman, "Finding structure in time," *Cognitive Sci.,* vol. 14, pp. 179–211, 1990.

[41] M. C. Mozer, "A focused backpropagation algorithm for temporal pattern recognition," Univ. Toronto, Canada, Tech. Rep. CRG-TR-88-3, 1988; Complex Syst., vol. 3, pp. 349–381, 1989.

[42] S. Santini, A. Del Bimbo, and R. Jain, "Block-structured recurrent neural networks," *Neural Networks,* vol. 8, no. 1, pp. 135–147, 1995.

[43] T. C. Hsia, "A simplified adaptive recursive filter design," *Proc. IEEE,* vol. 69, Sept. 1981.

[44] R. R. Leighton and B. C. Conrath, "The autoregressive backpropagation algorithm," in *Proc. Int. Joint Conf. Neural Networks,* pp. 369–377, 1991.

[45] H.-U. Bauer and T. Geisel, "Dynamics of signal processing in feedback multilayer perceptrons," in *Proc. Int. Joint Conf. Neural Networks,* 1990, pp. 131–-136.

[46] D. H. Nguyen and B. Widrow, "Neural networks for self-learning control systems," *IEEE Contr. Syst. Mag.,* pp. 18–23, Apr. 1990.

[47] J. Schmidhuber, "Learning complex, extended sequences using the principle of history compression," *Neural Comput.,* vol. 4, no. 2, pp. 234–242, 1992.

[48] B. Widrow and M. A. Lehr, "30 years of adaptive neural networks: Perceptron, madaline, and backpropagation," *Proc. IEEE,* vol. 78, pp. 1415–1442, Sept. 1990.

[49] R. J. Williams and D. Zipser, "Experimental analysis of the real-time recurrent learning algorithm," *Connection Sci.,* vol. 1, no. 1, pp. 87–111, 1989.

[50] M. I. Jordan, "Attractor dynamics and parallelism in a connectionist sequential machine," in *Proc. 8th Annu. Conf. Cognitive Sci. Soc.,* 1986, pp. 531–546.

[51] F. Pineda, "Generalization of Backpropagation to recurrent neural networks," *Phys. Rev. Lett.,* vol. 59, no. 19, pp. 2229–2232, Nov. 9, 1987.

[52] L. B. Almeida, "A learning rule for asynchronous perceptrons with feedback in combinatorial environment," in *Proc. Int. Conf. Neural Networks,* 1987, vol. 2, pp. 609–618.

[53] R. J. Williams and D. Zipser, "Gradient-based learning algorithms for recurrent networks and their computational complexity," *Backpropagation: Theory, Architectures and Applications*, Y. Chauvin and D. E. Rumelhart, Eds. Hillsdale, NJ: Lawrence Erlbaum, 1994.

[54] T. Uchiyama, K. Shimohara, and Y. Tokunaga, "A modified leaky integrator network for temporal pattern recognition," in *Proc. Int. Joint Conf. Neural Networks,* vol. 1, pp. 469–475, 1989.

[55] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.,* vol. 9, no. 8, pp. 1735–1780, 1997.

[56] A. J. Robinson and F. Fallside, "The utility driven dynamic error propagation network," Cambridge Univ. Eng. Dept., Tech. Rep. CUED/F-INFENG/TR.1, 1987.

[57] E. A. Wan, "Adjoint LMS: An efficient alternative to the filtered-XLMS and multiple error LMS algorithms," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Processing,* Atlanta, GA, May 1996.

[58] M. C. Mozer, "Neural net architectures for temporal sequence processing," *Predicting the Future Understanding the Past,* A. Weigend and N. Gershenfeld, Eds. San Mateo, CA: Addison-Wesley, 1993.

[59] P. Campolucci, "Signal flow graph approach to gradient calculation and learning: Forward computation," Univ. Ancona, Italy, Dip. Elettronica ed Automatica, Tech. Rep., 1996.

[60] ———, "Signal flow graph approach to gradient calculation and learning: Backward computation," Univ. Ancona, Dip. Elettronica ed Automatica, Tech. Rep., 1996.

[61] P. Campolucci, A. Uncini, and F. Piazza, "Dynamical systems learning by a circuit theoretic approach," in *Proc. ISCAS-98, IEEE Int. Symp. Circuit Syst.,* Monterey, CA, June 1998.

[62] P. Campolucci, "Learning algorithms for recurrent neural networks," Università di Ancona, Italy, Dip. Elettronica ed Automatica, Tech. Rep., 1994; "Reti neurali artificiali con dinamica temporale," Tesi di Laurea, Univ. degli studi di Ancona, 1994 (in Italian).

[63] ———, "A circuit theory approach to recurrent neural network architectures and learning methods," Ph.D. dissertation, Univ. Bologna, Italy, 1998.

Paolo Campolucci (M'94) was born in Forli, Italy, in 1969. He received the Laurea degree in electronic engineering summa cum laude in 1994 from the University of Ancona, Ancona, Italy, and the Ph.D. degree in circuit theory in 1998 from the University of Bologna, Bologna, Italy.

He is currently with the Digital Signal Processing and Neural Networks Research Group, Electronics and Automatics Department, University of Ancoma. From 1995 to 1996, he was a Visiting Scholar with the Electrical and Computer Engineering Department, University of California, San Diego, and in 1998, he was with the Engineering Department, Brown University, Providence, RI. His current research interests include dynamic and recurrent neural networks, learning theory, digital signal processing, and circuit theory.

Aurelio Uncini (M'88) received the Laurea degree in electronic engineering from the University of Ancona, Italy, and the Ph.D. degree in electrical engineering from the University of Bologna, Italy, in 1983 and 1993, respectively.

During 1984 to 1996, he was with the "Ugo Bordoni" Foundation, Rome, Italy, engaged in research on digital processing of speech signal. From 1986 to 1987, he was at the Italian Ministry of Communication. From 1994 to 1998, he was a Researcher at the Electronics and Automatics Department at the University of Ancona. Currently, he is Associate Professor of Electrical Engineering at the INFOCOM Department, University of Rome "La Sapienza." His research interests include digital signal processing, neural networks, and circuit theory. His present research interests include adaptive filters, audio and speech processing, optimization algorithms for circuits design, and neural networks for signal processing.

Francesco Piazza (M'87) was born in Jesi, Italy, in February 1957. He received the Laurea degree with honors in electronic engineering from the University of Ancona, Italy, in 1981.

From 1981 to 1983, he worked on CCD-image processing at the Physics Department of the University of Ancona. In 1983, he worked at the Olivetti OSAI Software Development Center. In 1985, he joined the Department of Electronics and Automatics of the University of Ancona, first as Researcher in Electrical Engineering and then as Associate Professor. His current research interests are in the areas of circuit theory and digital signal processing and include adaptive DSP algorithms and circuits, neural networks, speech and audio processing.

Bhaskar D. Rao (S'80–M'83–SM'91) received the B.Tech. degree in electronics and electrical communication engineering from the Indian Institute of Technology, Kharagpur, in 1979 and the M.S. and Ph.D. degrees from the University of Southern California, Los Angeles, in 1981 and 1983, respectively.

Since 1983, he has been with the University of California, San Diego, where he is currently a Professor in the Electrical and Computer Engineering Department. His interests include digital signal processing, estimation theory, and optimization theory, with applications to speech, communications, and human–computer interactions.

Dr. Rao has been a member of the Statistical Signal and Array Processing Technical Committee.