A Unifying View of Gradient Calculations and Learning for Locally Recurrent Neural Networks

Paolo Campolucci[*], *Aurelio Uncini, Francesco Piazza*[+]

Dipartimento di Elettronica ed Automatica, Università di Ancona,
via Brecce Bianche, 60131 Ancona, Italy.
[*]e-mail: paolo@eealab.unian.it

Abstract

In this paper a critical review of gradient-based training methods for recurrent neural networks is presented including Back Propagation Through Time (BPTT), Real Time Recurrent Learning (RTRL) and several specific learning algorithms for different locally recurrent architectures. From this survey it comes out the need for a unifying view of all the specific procedures proposed for networks with local feedbacks, that keeps into account the general framework of recurrent networks learning: BPTT and RTRL. Therefore a learning method for local feedback network is proposed which combines together the best feature of BPTT, i.e. the lowest complexity, and of RTRL, i.e. the on-line operation, and includes as special case several specific algorithms already proposed, such as Temporal Back Propagation, Back Propagation for Sequences, Back-Tsoi algorithm and some others. In the general version, this new training method allows on-line efficient and accurate gradient calculation. It compares favourably with the previous algorithms in stability, speed/complexity trade off, accuracy.

1. Introduction

Internally static networks can be trained by the simplest algorithms: for the buffered Multi Layer Perceptron (MLP) with only input buffer (with no recursion) the standard Back Propagation (BP) should be used, while for the Narendra-Parthasarathy network [18], i.e. buffered MLP with input and output buffers, the so called "open loop" approximation 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 Infinite Impulse Response

[+] This research was supported by the Italian MURST.

(IIR) adaptive filter theory this is the equation error approximation of the true output error approach [10], in neural network theory this is the teacher forced technique [2].

Extension of Back Propagation to recurrent networks was firstly proposed by F. Pineda and L. B. Almeida, e.g. [26] only in the case when the recurrent network behaviour relaxes to a fixed point. However, if a general temporal processing is needed, two main gradient-based learning approaches exist for recurrent networks [15,16,20]: Back Propagation Through Time (BPTT) [6,1,27,15] and Real Time Recurrent Learning (RTRL) [2,18,22,27,15].

BPTT is a family of algorithms which extends the BP paradigm to dynamic networks. There are two major points of view to understand what BPTT is. 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 to the network state at time t as if it was obtained from the t-th layer output of a multilayer network with T layers, 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 [5]. 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 [16,20]. The difference between BPTT and RTRL is in how the chain rule derivative expansion is applied. More specifically, during the learning phase, in BPTT signals and time flow in the reverse direction with respect to the forward phase whereas in RTRL they flow as in the forward calculations. Reversing the signal flow graph provides the great efficiency of BPTT but the necessary reversion of signal time scale makes it non-causal even if only one delay is present inside the network.

Therefore BPTT is local in space but not in time, is computationally simple but is non-causal; 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 [1,23,27] for fully recurrent neural networks.

Instead RTRL is local in time but not in space, computationally complex but intrinsically on-line. However even RTRL implements an approximated calculation of the gradient if the parameters are continually adapted since its true derivation would require constant weights [1,27].

In [27] 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 Beaufayas [17] proposed a simple method to derive BPTT for discrete time dynamic neural networks composed of a general interconnection of weights, delays, additive units, differentiable non-linearities. 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 non-causal and the authors did not address the question of on-line learning for networks with feedbacks.

On the contrary, several gradient-based learning algorithms have been presented for specific dynamic multilayer neural networks.

In [7] a learning algorithm, named "Temporal Back Propagation", was proposed by E. Wan. This is an on-line version of the batch mode BPTT approach [17]. However it can only be applied to the non-recurrent MLP with Finite Impulse Response (FIR) filter synapses [3,7,21,9], i.e. FIR-MLP often called Time Delay Neural Network (TDNN) [8].

Back Propagation for Sequences (BPS) is a learning algorithm proposed by Gori et al. [13,14,22,15] for both output and activation Local Feedback Multi Layered Networks (LF-MLNs) [14]. 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 BPTT, but is somewhere between BP and RTRL. The same approch was proposed by Mozer in [24] that indipendently derived a quite similar algorithm named "Focused Back Propagation" for a locally recurrent network analogous to the Auto Regressive MLP (AR-MLP) [25]. BPS was rediscovered in [19] where 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], a learning algorithm for MLP with IIR filter synapses [3,4,10,11,12] was proposed by A.D. Back and A.C. Tsoi. It is similar to BPS, implementing both a backpropagation and a recursive computation, but without any architectural restriction. However to avoid dynamic backpropagation, the authors propose a rough approximation of BPTT, using static backpropagation even through a dynamic neuron.

Similar learning algorithms are also Auto Regressive BP proposed by R.R. Leighton and B.C. Conrath [25] for the Auto Regressive MLP [25], and the algorithm in [28]. They are equivalent to Back-Tsoi algorithm since they also use instantaneous backpropagation without implementing the full BPTT 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 [25,28].

The unifying on-line method proposed in this paper, whose basic ideas were presented in [11,12], implements and combines together BPTT and RTRL paradigms for locally recurrent networks. Many algorithms can be derived by this method for locally recurrent networks: they implement an accurate (even if not exact, as usual for on-line training) computation of the gradient. The name that we will use is Causal Recursive Back Propagation (CRBP, not to be confused with Recurrent Back Propagation [26]) thus remembering the dual nature of the method: 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 [20]: 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 working in batch mode (Recursive Back Propagation, RBP).

However the three methods are not equivalent in on-line mode. It must be stressed that in CRBP each local feedback is taken into account during adaptation with no history truncation for the coefficients of the same neuron, using recursive formulas instead of non-causal ones as in the BPTT approach.

In the following the Recursive Back Propagation batch learning method and its on-line version are derived for the MLP with IIR synapses; the formulas for LF-MLNs and AR-MLP can be derived in a similar way.

2. A Unifying Formulation for Learning Methods in Locally Recurrent Neural Networks

The notation that will be used, an extension of Widrow's notation, is summarized in the following:

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 l-th 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, ... T. Where T is the length of the training sequence.
$x_n^{(l)}[t]$	output of the n-th neuron of the l-th 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,...,N_0, are the input signals.
$L_{nm}^{(l)}$-1	order of the MA part of the synapse of the n-th neuron of the l-th layer relative to the m-th output of the (l-1)-th layer. $L_{nm}^{(l)}$>=1 and $L_{n0}^{(l)}$=1.
$I_{nm}^{(l)}$	order of the AR part of the synapse of the n-th neuron of the l-th layer relative to the m-th output of the (l-1)-th layer. $I_{nm}^{(l)}$>=0 and $I_{n0}^{(l)}$=0.
$w_{nm(p)}^{(l)}$	(p=0,1, ..., $L_{nm}^{(l)}$-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, ..., $I_{nm}^{(l)}$) 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.
sgm(z)	activation function.
sgm'(z)	derivative of sgm(z).
$y_{nm}^{(l)}[t]$	synaptic filter output at time t relative to the synapse of n-th neuron, l-th layer and m-th input. $y_{n0}^{(l)}$ =$w_{n0}^{(l)}$ is the bias.

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

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

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

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

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

As in the static case, it holds:

$$\delta_n^{(l)}[t] = e_n^{(l)}[t]\,\mathrm{sgm}'(s_n^{(l)}[t]) \tag{3}$$

Therefore, using gradient descent method and the chain rule expansion:

$$\Delta w_{nm(p)}^{(l)} = \sum_{t=1}^{T} \Delta w_{nm(p)}^{(l)}[t+1] \quad , \quad \Delta w_{nm(p)}^{(l)}[t+1] = \mu\, \delta_n^{(l)}[t]\, \frac{\partial s_n^{(l)}[t]}{\partial w_{nm(p)}^{(l)}} \tag{4}$$

Similarly for the v weights we have:

$$\Delta v_{nm(p)}^{(l)} = \sum_{t=1}^{T} \Delta v_{nm(p)}^{(l)}[t+1] \quad , \quad \Delta v_{nm(p)}^{(l)}[t+1] = \mu\, \delta_n^{(l)}[t]\, \frac{\partial s_n^{(l)}[t]}{\partial v_{nm(p)}^{(l)}} \tag{5}$$

where

$$\frac{\partial s_n^{(l)}[t]}{\partial w_{nm(p)}^{(l)}} = x_m^{(l-1)}[t-p] + \sum_{r=1}^{I_{nm}^{(l)}} v_{nm(r)}^{(l)} \frac{\partial s_n^{(l)}[t-r]}{\partial w_{nm(p)}^{(l)}} \tag{6}$$

$$\frac{\partial s_n^{(l)}[t]}{\partial v_{nm(p)}^{(l)}} = y_{nm}^{(l)}[t-p] + \sum_{r=1}^{I_{nm}^{(l)}} v_{nm(r)}^{(l)} \frac{\partial s_n^{(l)}[t-r]}{\partial v_{nm(p)}^{(l)}} \tag{7}$$

in which the derivatives can be recursively computed.

The backpropagation through the layers can be derived by chain rule:

$$e_n^{(l)}[t] = \begin{cases} e_n[t] & \text{for } l = M \\ \displaystyle\sum_{q=1}^{N_{l+1}} \sum_{p=0}^{T-t} \delta_q^{(l+1)}[t+p]\, \frac{\partial y_{qn}^{(l+1)}[t+p]}{\partial x_n^{(l)}[t]} & \text{for } l = (M-1),\ldots,1 \end{cases} \tag{8}$$

where the partial derivatives are computed deriving expression (1):

$$\frac{\partial y_{qn}^{(l+1)}[t+p]}{\partial x_n^{(l)}[t]} = \left(\begin{cases} w_{qn(p)}^{(l+1)} & \text{if } 0 \le p \le L_{qn}^{(l+1)} - 1 \\ 0 & \text{otherwise} \end{cases} \right) + \sum_{r=1}^{\min(I_{qn}^{(l+1)}, p)} v_{qn(r)}^{(l+1)} \frac{\partial y_{qn}^{(l+1)}[t+p-r]}{\partial x_n^{(l)}[t]} \tag{9}$$

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):

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

where t_0 is the initial time instant. Differentiating we get:

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

Obviously, if the filter is time invariant, the derivative does not depend on *t* but only on *p* and *h*[*t*+*p*,*t*]=*h*[*t*+*p*-*t*]=*h*[*p*]. This means that the derivative in (9) is the impulse response of the IIR filter. It is obtained through auto-regressive filtering of the sequence of the coefficients of the MA part with the AR part of the corresponding IIR synaptic filter. 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 (9).

Therefore, for MLP with IIR synapses, expression (8) means that each back propagating error at layer *l* is a summation of all the *delta's* at the following layer filtered by the non-causal version of the respective IIR filter. The non-causal version of each filter can be obtained practically, convoluting with time reverted impulse response or time reverting the output of the filter giving the input in a reversed time scale.

The expressions from (1) to (9) constitute the Recursive Back Propagation algorithm for IIR-MLP.

As previously stated, the RBP algorithm is used only as an intermediate step in the derivation of CRBP. In fact, as shown by expression (8), the exact RBP algorithm is non-causal, 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.

However the RBP algorithm, due to the recursive structure of expressions (6) and (7), can be easily approximated to obtain a very efficient on-line learning algorithm. The on-line approximation consists in three steps: (a) incremental instead of cumulative adaptation, (b) future convolution truncation, (c) causalization.

(a) Incremental instead of cumulative adaptation can be implemented using:

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

instead of the first expressions in (4) and (5), at each time step, where '*weight*' indicates either *w* or *v* .

(b) If a casualization is desired, a truncation of the future convolution is necessary, due to the infinite memory of the IIR synapses. The truncated formula is therefore:

$$e_n^{(l)}[t] = \begin{cases} e_n[t] & \text{for } l = M \\ \sum_{q=1}^{N_{l+1}} \sum_{p=0}^{Q_{l+1}} \delta_q^{(l+1)}[t+p] \ \frac{\partial y_{qn}^{(l+1)}[t+p]}{\partial x_n^{(l)}[t]} & \text{for } l = (M-1),\ldots,1 \end{cases} \qquad (10)$$

where Q_{l+1} is appropriately chosen.

(c) Then we have to introduce a suitable number of delays in the weight adaptation formulas in order to remove the non-causality. In other words:

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

where D_l is a suitable integer number. It follows that $D_l = \begin{cases} 0 & \text{if } l = M \\ \sum_{i=l+1}^{M} Q_i & \text{if } 1 \le l < M \end{cases}$

The causalised formula can be obtained from (10) 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).

For the sake of clarity, a diagram of CRBP is shown in Fig. 1.

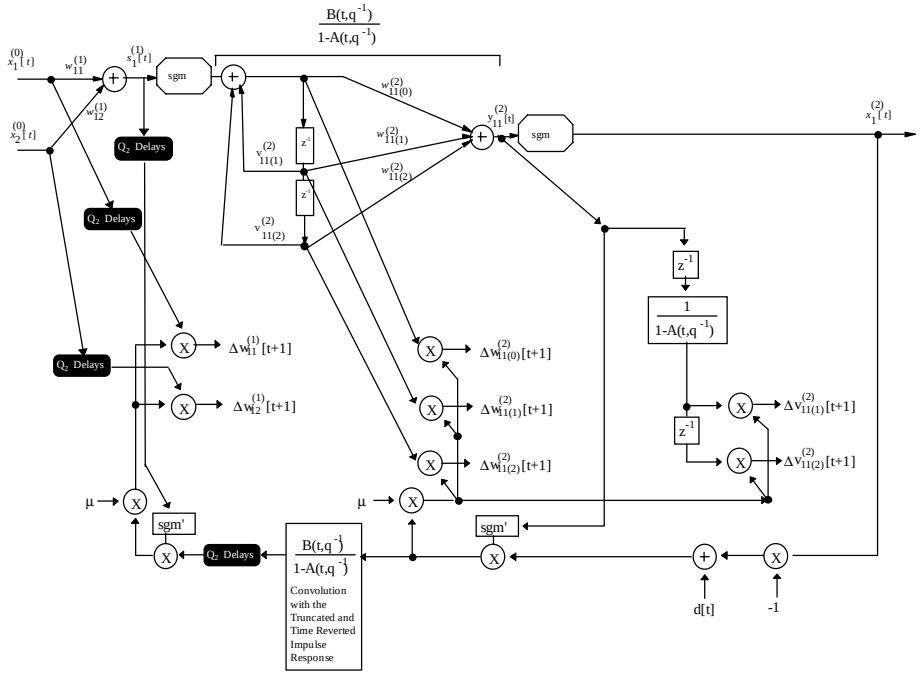

Fig. 1. The CRBP applied to an IIR-MLP example network (the bias terms are not shown). It is obtained assuming simplified recursive computation of derivatives. The Back-Tsoi approximation uses the same flow diagram but with a multiplication for $w_{11(0)}^{(2)}$ instead of the truncated IIR filtering and Q_2=0.

The causalisation 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 are inside the unit circle) then $\frac{\partial y[t+p]}{\partial x[t]} \to 0$ if $p \to \infty$ *where* $y[t]$ *is the output of the filter and* $x[t]$ *the input at time t.*

The proof can be done just considering that the derivative is the impulse response of the filter that must go to zero in the stable case. For the derivative to go to zero it is necessary and sufficient that the feedback coefficients in the calculation of (9) or the corresponding expressions for a locally recurrent network other than IIR-MLP, give poles inside the unit circle.

The condition $\frac{\partial y[t+p]}{\partial x[t]} \to 0$ if $p \to \infty$ where $y[t]$ is the output (or the *net*) and $x[t]$ an input of a recurrent neuron holds by definition for each neuron in IIR-MLP, activation-output feedback MLNs and AR-MLP in case each neuron exhibits forgetting behaviour. Instead in case of latching behaviour (possible only for output feedback MLN), that derivative does not go to zero and the corresponding linear system is unstable [14]. In this case, the truncation can be too strong an approximation.

It must be stressed that in CRBP each local feedback is taken into account during adaptation with no history truncation for the coefficients of the same neuron, using recursive formulas (6) and (7), instead of non-causal ones as in the BPTT approach.

The algorithm proposed by Back and Tsoi [3] that is the only on-line learning algorithm proposed specifically for the IIR-MLP, can be seen as a particular case of this approximation if a strong truncation of the summation is did: $Q_{l+1} = 0$ for each l . In this way the backpropagation is considering only the instantaneous influence of the IIR filter input to the output (the coefficient $w_{nm(0)}^{(l+1)}$). Hence in the scheme of Fig. 1, the truncated IIR filter should be replaced by a simple multiplication for $w_{11(0)}^{(2)}$. No causalization is needed (being 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 [13] can be obtained as particular case of CRBP when the architecture restriction that the dynamic units can only be placed in the first layer is given. The CRBP applied to the AR-MLP can be also viewed as a generalization of Mozer's and Leighton-Conrath's work [24,25]. Moreover 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 Back Propagation as in [7,9,21]. 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 becomes to standard on-line Back Propagation for the MLP.

3. Simulations

Many simulations were performed on IIR-MLP and activation-output feedback MLN while the AR-MLP was not implemented. For comparison purposes, also two traditional neural networks were tested, namely the static MLP with input and possibly output buffer, and the FIR-MLP. The results reported here refer to a problem of identification of non-linear dynamical system: a 16-PAM transmission channel [9].

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 modelled here by the following input-output relationship:

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

The peak power of the input signal $v[t]$ is set to the value of β dB (back-off factor), with β=0 dB being the normalised 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), see Fig 2. The overall system is clearly dynamic and non-linear.

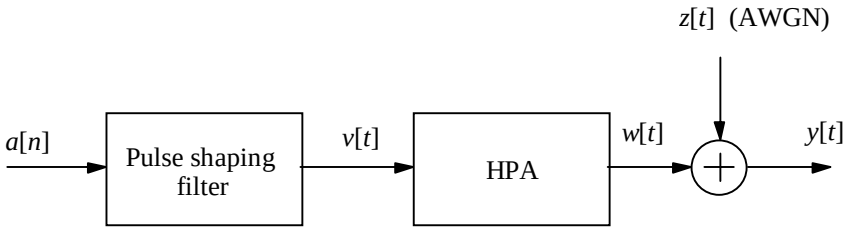

Fig. 2. Block diagram of the PAM transmission channel used in the simulations.

A neural network approach to equalise this system has already been proposed in the technical literature. 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 α=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.

The number of delays for the five architectures (i.e. buffers lengths for the buffered MLPs, 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).

All the used networks 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, temporal backpropagation for FIR-MLP [7] and the proposed CRBP algorithm for the locally recurrent networks. The results are given in terms of Mean-Square-Error (MSE), expressed in dB, computed on the learning set after each epoch (after all the input-output samples were presented) and averaged over 20 runs, each with a different weights initialization.

First we verified that the locally recurrent MLPs perform much better than the two conventional MLPs in modeling the system and then we compared the new and the previous algorithms.

Fig. 3 shows that CRBP, is much faster, stable and also more accurate than the Back-Tsoi algorithm. 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.

From Fig. 4 it is evident that the averaged impulse response of the IIR filter in the IIR-MLP network goes to zero so the truncation approximation hypothesis is correct.

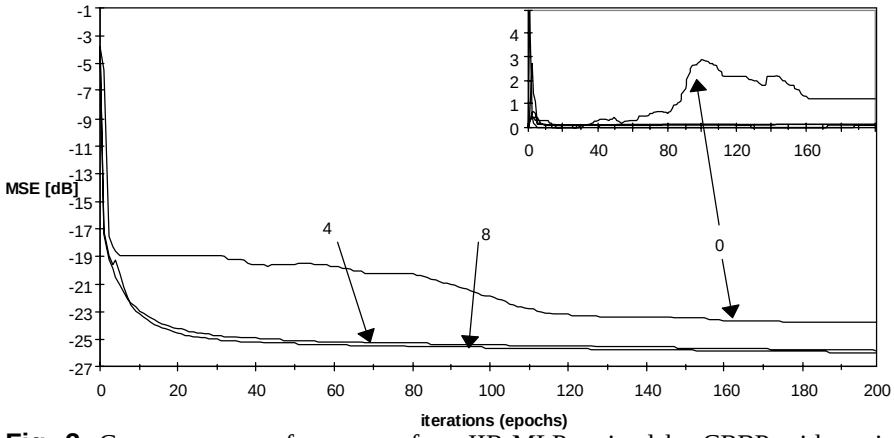

Fig. 3. Convergence performance of an IIR-MLP trained by CRBP with various values of the truncation parameter (Q_2; Q_2=0 gives the Back-Tsoi algorithm) on identifying the 16-PAM transmission system. The MA and AR orders were chosen respectively as 4 and 2 for both the hidden and output layers. Plots are averaged over 20 runs with different weight initializations. Momentum and Delta-Bar-Delta adaptive learning rate have been used. The variance on the 20 runs is shown on the top right.

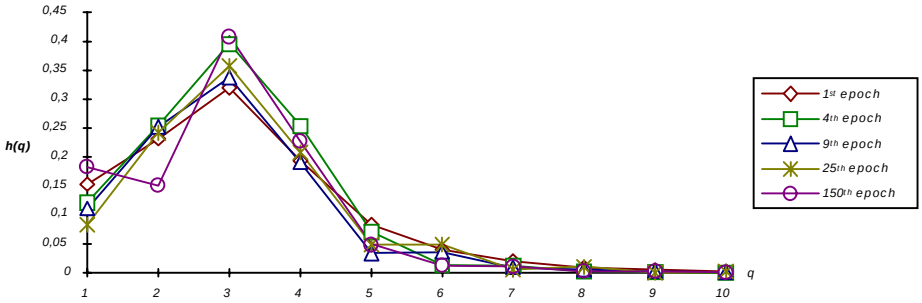

Fig. 4. Averaged Impulse Response of the IIR filters in the Multi Layer network identifying the P.A.M. system by the CRBP algorithm.

4. Conclusions

The increase in computational complexity between the Back-Tsoi algorithm [3] and CRBP is fairly low since, due to the recursion, the truncation parameter (Q_{l+1}) can be chosen quite small. 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 all the architectures and parameter settings considered in the simulations.

A mathematical evaluation of complexity to compare CRBP with previous methods, in particular Back-Tsoi algorithm, can be carried out computing the number of multiplications and additions for one iteration of the learning phase. In the following, results for CRBP 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 number of multiplications ($M_{backward}$) of the backward phase (one iteration) is:

$$M_{backward} = 2(N_1+N_2)+N_1N_0\left(I^{(1)}+L^{(1)}(I^{(1)}+1)+(I^{(1)})^2\right)+$$

$$+N_2N_1\left(I^{(2)}+L^{(2)}(I^{(2)}+1)+(I^{(2)})^2+Q_2+1+\sum_{p=0}^{Q_2}\min(I^{(2)},p)\right)$$

The number of additions of the backward phase (one iteration) is:

$$A_{backward} = 2N_2+N_1N_0\left(I^{(1)}+L^{(1)}(I^{(1)}+1)+(I^{(1)})^2\right)+$$

$$+N_2N_1\left(I^{(2)}+L^{(2)}(I^{(2)}+1)+(I^{(2)})^2+Q_2+1+\sum_{p=0}^{Q_2}\min(I^{(2)},p)+\right.$$

$$\left.-\begin{cases}Q_2-L^{(2)}+1 & \text{if } Q_2>L^{(2)}-1\\ 0 & \text{otherwise}\end{cases}\right)$$

The complexity of the Back-Tsoi algorithm is obtained choosing Q_2=0.

They 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:
$M_{forward} = A_{forward} = N_1 N_0 \left(L^{(1)} + I^{(1)} \right) + N_2 N_1 \left(L^{(2)} + I^{(2)} \right)$.
By these formulas, substituting the configuration parameters of each of the IIR-MLP networks used in the simulations (only some reported here), and with Q_2=4, it is possible to get an averaged ratio $M_{CRBP}/M_{BackTsoi}$=1.20 and $A_{CRBP}/A_{BackTsoi}$=1.17, proving that the increase in complexity is quite small.

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

In conclusion, this work show that the locally recurrent MLPs can have superior modelling capabilities with respect to more traditional networks, namely MLP with external memory and FIR-MLP (or TDNN) for system identification. We described a general approach to derive on-line algorithms for locally recurrent networks and shown a new learning method, that, includes as special cases, the ones already known in literature. The algorithm derived by us has better stability and higher speed of convergence compared to the previous ones, 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, which however can be easily reduced.

References

1. R.J. Williams, J. Peng. An efficient gradient-based algorithm for on line training of recurrent network trajectories. Neural Computation 2: 490-501, 1990.
2. R.J. Williams, D. Zipser. A Learning Algorithm for Continually Running Fully Recurrent Neural Networks. Neural Computation 1: 270-280, 1989.
3. A.D. Back, A.C. Tsoi. FIR and IIR synapses, a new neural network architecture for time series modelling. Neural Computation 3: 375-385, 1991.
4. A.C. Tsoi, A.D. Back. Locally recurrent globally feedforward networks: a critical review of architectures. IEEE Transactions on Neural Networks, vol. 5, no. 2, 229-239, March 1994.
5. P.J. Werbos. Beyond regression: New tools for prediction and analysis in the behavioural sciences. Ph.D. dissertation, Committee on Appl. Math., Harvard Univ., Cambridge, MA, Nov. 1974.
6. P.J. Werbos. Backpropagation through time: what it does and how to do it. Proceedings of IEEE, Special issue on neural networks, vol. 78, No. 10, pp.1550-1560, October 1990.
7. E.A. Wan. Temporal backpropagation for FIR neural networks. Proceedings of the International Joint Conference on Neural Networks, 1:575-580, 1990.
8. A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, K.J. Lang. Phoneme recognition using time-delay neural networks. IEEE Trans. on Acoustic, Speech, and Signal Processing, Vol. 37, No.3, March 1989.
9. N. Benvenuto, F. Piazza, A. Uncini. Comparison of four learning algorithms for multilayer perceptron with FIR synapses. Proceeding of the IEEE International Conference of Neural Networks, 1994.
10. J.J. Shynk. Adaptive IIR filtering. IEEE ASSP Magazine, April 1989.
11. P. Campolucci, F. Piazza, A. Uncini. On-line learning algorithms for neural networks with IIR synapses. Proc. of the IEEE International Conference of Neural Networks, Perth, Nov. 1995.
12. P. Campolucci, A. Uncini, F. Piazza. Causal Back Propagation Through Time for Locally Recurrent Neural Networks. Proc. of the IEEE International Symposium on Circuits and Systems, Atlanta, May 1996.
13. Y.Bengio, R. De Mori, M.Gori. Learning the dynamic of speech with back-propagation for sequences. Pattern Recognition Letters, 13: 375-385, 1992.
14. P.Frasconi, M.Gori, G.Soda. Local Feedback Multilayered Networks. Neural Computation 4: 120-130, 1992.
15. B.A.Pearlmutter. Gradient Calculations for Dynamic Recurrent Neural Networks: A Survey. IEEE Trans. on Neural Networks vol. 6, no. 5, September 1995.
16. 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. on Neural Networks pp. 213-228, March 1994.

17. E.A.Wan, F.Beaufays. Diagrammatic Derivation of Gradient Algorithms for Neural Networks. Neural Computation 8: 182-201, 1996.
18. K.S. Narendra, K. Parthasarathy. Identification and control of dynamical systems using neural networks. IEEE Trans. on Neural Networks, vol. 1, pp.4-27, March 1990.
19. C.-C. Ku, K.Y.Lee. Diagonal Recurrent Neural Networks for Dynamic Systems Control. IEEE Trans. on Neural Networks vol. 6, no. 1, January 1995.
20. F.Beaufays, E.Wan. Relating Real-Time Backpropagation and Backpropagation-Through-Time: An Application of Flow Graph Interreciprocity. Neural Computation 6: 296-306, 1994.
21. A.D.Back, E.Wan, S.Lawrence, A.C.Tsoi. A unifying view of some training algorithms for multilayer perceptrons with FIR filter synapses. Proc. IEEE workshop on Neural Networks for Signal Processing, pp. 146-154, 1994.
22. B.A. Pearlmutter. Two New Learning Procedures for Recurrent Networks. Neural Networks Rev. vol. 3, no. 3, pp. 99-101, 1990.
23. J.L. Elman. Finding Structure in Time. Cognitive Science 14: 179-211, 1990.
24. M.C. Mozer. A Focused Back-propagation Algorithm for Temporal Pattern Recognition. Tech Rep. CRG-TR-88-3, University of Toronto, 1988 and Complex Systems 3: 349-381, 1989.
25. R.R. Leighton and B.C. Conrath. The Autoregressive Backpropagation Algorithm. Proc. International Joint Conference on Neural Networks, pp. 369-377, 1991.
26. L.B. Almeida. A learning rule for asynchronous perceptrons with feedback in combinatorial environment. Proc. International Conference on Neural Networks, vol. 2, pp. 609-618, 1987.
27. R.J. Williams, D. Zipser. Gradient-based learning algorithms for recurrent networks and their computational complexity. In Backpropagation: Theory, Architectures and Applications Y. Chauvin and D.E. Rumelhart, Eds. Hillsdale, NJ: Lawrence Erlbaum Associates, 1994.
28. T. Uchiyama, K. Shimohara, Y. Tokunaga. A modified leaky integrator network for temporal pattern recognition. Proc. of the International Joint Conference on Neural Networks, vol. 1, pp. 469-475, 1989.