Neural Networks for Signal Processing (IEEE, 1992)

ADAPTIVE TRAINING OF FEEDBACK NEURAL NETWORKS FOR NON-LINEAR FILTERING

G. Dreyfus*, O. Macchi**, S. Marcos**, O. Nerrand*, L. Personnaz*,
P. Roussel-Ragot*, D. Urbani*, C. Vignat**

*Ecole Supérieure de Physique et de Chimie Industrielles de la Ville de Paris
10, rue Vauquelin
75005 PARIS - FRANCE

**Laboratoire des Signaux et Systèmes
Ecole Supérieure d'Electricité, Plateau de Moulon
91192 GIF SUR YVETTE - FRANCE

Abstract. The paper proposes a general framework which encompasses the training of neural networks and the adaptation of filters. It is shown that neural networks can be considered as general non-linear filters which can be trained adaptively, i.e. which can undergo continual training. A unified view of gradient-based training algorithms for feedback networks is proposed, which gives rise to new algorithms. The use of some of these algorithms is illustrated by examples of non-linear adaptive filtering and process identification.

INTRODUCTION

In recent papers [1, 2], a general framework, which encompasses algorithms used for the training of neural networks and algorithms used for the adaptation of filters, has been proposed. Specifically, it was shown that neural networks can be used *adaptively*, i.e. can undergo *continual* training with a possibly *infinite* number of *time-ordered* examples - in contradistinction to the traditional training of neural networks with a *finite number* of examples presented in an *arbitrary order;* therefore, neural networks can be regarded as a class of non-linear adaptive filters, either transversal or recursive, which are quite general because of the ability of neural nets to approximate non-linear functions. It was further shown that algorithms which can be used for the adaptive training of feedback neural networks fall into three broad classes; these classes include, as special instances, the methods which have been proposed in the recent past for training neural networks adaptively, as well as algorithms which have been in current use in linear adaptive filtering and control. This framework will be summarized briefly in the first part of the paper.
In addition, this general approach leads to new algorithms. The second part of the paper shows illustrative examples of the application of the latter to problems in adaptive filtering and identification.

ADAPTIVE TRAINING OF FEEDBACK NEURAL NETS FOR NON-LINEAR FILTERING

Network

A neural network architecture of the type shown on Figure 1, featuring M external inputs, N feedback inputs and one output, can implement a fairly large class of non-linear functions; the most general form for the feedforward part is a *fully-connected net.* The basic building block of the network is a "neuron", which performs a weighted sum of its inputs and computes an "activation function" f - usually non linear - of the weighted sum:
$z_i = f_i[v_i]$ with $v_i = \sum_j C_{ij}\, x_j$
where z_i denotes the output of neuron i, and x_j denotes the j-th input of neuron i; x_j may be an external input, a state input, or the output of another neuron.

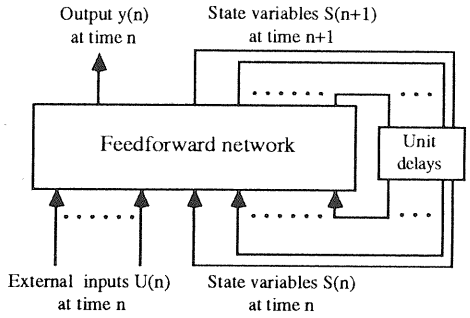

Figure 1.

If the external inputs consist of the values U(n)={u(n),u(n-1),....,u(n-M+1)} of a signal u at successive instants of time, the network may be viewed as a general non-linear recursive filter.
The task of the network is determined by a (possibly infinite) set of inputs and corresponding desired outputs. At each sampling time n, an error e(n) is defined as the difference between the desired output d(n) and the actual output of the network y(n): e(n)=d(n)-y(n). The network adaptation algorithms aim at finding the synaptic coefficients which minimize a given satisfaction criterion involving, usually, the squared error $e(n)^2$ [3].
Thus, it is clear that adaptive filters and neural networks are formally equivalent, and that neural networks, which are potentially capable of realizing *non-linear* input-output relations, are simple generalizations of linear filters. In the next section, we put into perspective the training algorithms developed for discrete-time feedback neural networks and the algorithms used classically in adaptive filtering.

Algorithm	$S_{in}^{1}(n)$	$S_{in}^{m}(n)$	$\frac{\partial S_{in}^{1}}{\partial c_{ij}}(n)$	$\frac{\partial S_{in}^{m}}{\partial c_{ij}}(n)$
Directed (D)	Des. val.	Des. val.	zero	zero
D-SD	Des. val.	Des. val.	zero	$\frac{\partial S_{out}^{m-1}}{\partial c_{ij}}(n)$
D-UD	Des. val.	Des. val.	$\frac{\partial S_{out}^{1}}{\partial c_{ij}}(n-1)$	$\frac{\partial S_{out}^{m-1}}{\partial c_{ij}}(n)$
Semi-Directed (SD)	Des. val.	$S_{out}^{m-1}(n)$	zero	$\frac{\partial S_{out}^{m-1}}{\partial c_{ij}}(n)$
SD-D	Des. val.	$S_{out}^{m-1}(n)$	zero	zero
SD-UD	Des. val.	$S_{out}^{m-1}(n)$	$\frac{\partial S_{out}^{1}}{\partial c_{ij}}(n-1)$	$\frac{\partial S_{out}^{m-1}}{\partial c_{ij}}(n)$
Undirected (UD)	$S_{out}^{1}(n-1)$	$S_{out}^{m-1}(n)$	$\frac{\partial S_{out}^{1}}{\partial c_{ij}}(n-1)$	$\frac{\partial S_{out}^{m-1}}{\partial c_{ij}}(n)$
UD-D	$S_{out}^{1}(n-1)$	$S_{out}^{m-1}(n)$	zero	zero
UD-SD	$S_{out}^{1}(n-1)$	$S_{out}^{m-1}(n)$	zero	$\frac{\partial S_{out}^{m-1}}{\partial c_{ij}}(n)$

Table 1.
Summary of algorithms.
Des. val. = desired value

General presentation of the algorithms

The present paper focusses on gradient-based methods using a sliding window of length N_c, whereby the updating of the synaptic coefficients is given, at time n, by

$$\Delta c_{ij}(n) = -\mu \left[\frac{\partial}{\partial c_{ij}} \left(\frac{1}{2} \sum_{k=n-N_c+1}^{n} e(k)^2 \right) \right]_{C(n-1)} \qquad (1)$$

where μ is the gradient step.
The choice of N_c depends on several factors, including the typical time scale of the non-stationarity of the signals.
For the computation of the gradient to be meaningful, the coefficients must be considered as being constant on a window of length $N_t \geq N_c$. Thus, *for the updating at time n*, the N_c errors $\{e(k)\}$ and their partial derivatives, appearing in relation (1), must be computed from N_t *computational blocks*, corresponding to the last N_t sampling times; the values of the coefficients used for all N_t blocks are the coefficients C(n-1) which were updated at time n-1.
We denote by $S_{in}^{m}(n)$ the value of the state input of block m at time n and by $S_{out}^{m}(n)$ the state output.

The choice of the state inputs and of their partial derivatives, as inputs of each block, gives rise to a variety of algorithms. These algorithms fall into three categories depending on the choice of the state inputs:
(i) **directed algorithms**, in which the state inputs are taken equal to their desired values, for all blocks;
(ii) **semi-directed algorithms**, in which the state inputs of the first block at time n are taken equal to their desired values, and in which the state inputs of the other blocks are taken equal to the state outputs of the previous block,
(iii) **undirected algorithms**, in which the state inputs of the first block at time n are taken equal to the corresponding states computed at time n-1, and in which the state inputs of the other blocks are taken equal to the state outputs of the previous block.
Directed and semi-directed algorithms can be used only if all state variables have desired values, as is the case for NARMAX models [4]. If some, but not all, state inputs do not have desired values, **hybrid** versions of the above algorithms can be used: those state inputs for which no desired values are available are taken equal to the corresponding computed state variables (as in an undirected algorithm), whereas the other state inputs may be taken equal to their desired values (as in a directed or semi-directed algorithm).
In each category, three algorithms are defined, depending on the choice of the partial derivatives of the state inputs. This is summarized in Table 1.

Relations with known algorithms for neural nets and for adaptive filtering

Some of the above algorithms have been proposed independently in the field of neural nets and in the field of signal processing, under different names.
Two approaches have been used in order to adapt linear recursive filters: the *equation-error* formulation and the *output-error* formulation. In the equation-error formulation (also termed series-parallel in the control literature), the recursive nature of the filter is not taken into account: thus, directed algorithms generalize the equation-error approach; they generate stable adaptation behaviours. The *"Teacher Forcing"* algorithm [5] is based on the same idea. On the other hand, the output-error formulation takes into account the recursive form of the filter during adaptation: thus, undirected algorithms generalize the output-error approach. The stability of these algorithms is not easy to predict.
For instance, the *"Recursive Prediction Error (RPE)"* algorithm [6], used in linear adaptive filtering, is a UD algorithm with $N_t=N_c=1$. The *"Real-Time Recurrent Learning Algorithm"* [7] is the generalization of RPE to non-linear filters. The *"Truncated Backpropagation Through Time"* algorithm [8] is a UD-SD algorithm with $N_c=1$ and $N_t>1$. The *extended-LMS* algorithm [9] is identical to the UD-D algorithm with $N_t=N_c=1$ and is used in linear adaptive filtering for its autostabilization property. The *"A Posteriori Error Algorithm"* is a UD-D algorithm with $N_t=2$, $N_c=1$ [10]. The choice of $N_t=N_c=1$ is economical in terms of computation time; it is justified if the coefficients change slowly, i.e. if the gradient step μ is small enough.

APPLICATION TO AN ADAPTIVE FILTERING PROBLEM

The use of the new algorithms introduced above is illustrated in the case of the Adaptive Differential Pulse Code Modulation (ADPCM) system for bit rate reduction in speech transmission [11] (Figure 2). We show the influence of the training algorithm on the behaviour of the system, in the simple case of a predictor with a single adaptive coefficient b, and a two-level quantizer implemented as a neuron with transfer function f(x) = a tanh (px/a). The input signal is constant.

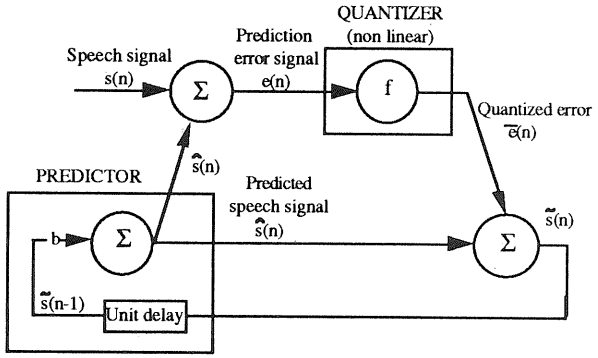

FIGURE 2

We first analyze the behaviour of the fixed, i.e. non adaptive, encoder. Fig.3 shows the prediction error e(n) versus b: for b<0.55, the error is a fixed point whose value decreases with b. For higher values of b, successive bifurcations generate limit cycles of lengths 2, 4 and 8.
The dynamical behaviour of the adaptive system depends on the choice of the adaptation algorithm, as illustrated on Figure 4. For example, the cycle P1 (of length 2) which is an attractive cycle for the non-adaptive system, remains attractive when the system is adapted with the UD or UD-SD algorithms. However, this cycle becomes a repeller when the system is adapted with the UD-D algorithm. Conversely, point P2 on Fig. 2 was found to be a repeller for the UD and UD-SD algorithms, while it is an attractor when the network is trained with a UD-D algorithm. The reported results were obtained with N_c=1 and N_t=5. The parameter N_t was found to have no influence on the results in this case; this is a specific feature of the system under consideration [12].
The mean square error for point P1 is smaller than for point P2.

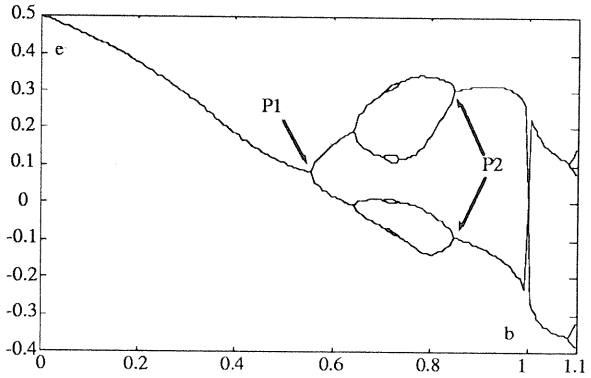

FIGURE 3

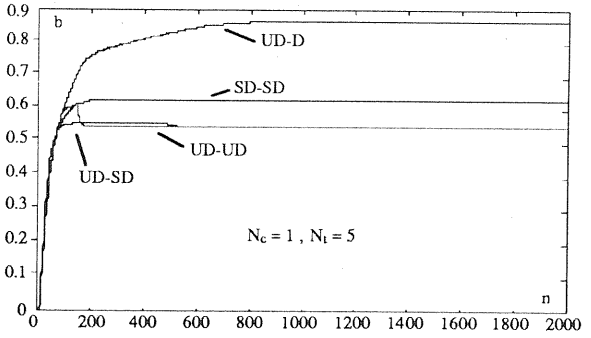

FIGURE 4

We focus now on the SD or D algorithms. Since the system to be adapted is trained by these algorithms as if it were a feedforward net consisting of N_t identical blocks and initialized with the desired outputs, the curves of Fig. 3 are irrelevant. Fig. 4 shows that the system adapted with the SD algorithm converges towards b≈0.63, which corresponds to a very small mean square error: this indicates that the feedforward structure consisting of 5 blocks, adapted with the SD algorithm, is appropriate for the problem under consideration. Conversely, when the feedback nature of the system must be preserved, SD- or D-type algorithms are inappropriate.

APPLICATION TO IDENTIFICATION PROBLEMS

We show on the following example that semi-directed algorithms bridge the gap between the *output-error* approach (UD algorithms) and the *equation-error* approach (D algorithms).
We first consider the process identification example described in [13], which illustrates the fact that, in the presence of additive noise, the *equation-error* formulation may lead to biased estimates of the coefficients, in contrast to the *output-error* formulation.
The process to be modelled is simulated by the linear recursive equation
$y^*(n) = \alpha\, y^*(n-1) + \beta\, x(n)$
and $d(n) = y^*(n) + v(n)$ with $\alpha=\beta=0.5$,
where x is the input, d the measured output and v an additive noise.
The model used in the adaptive filter is described by $y(n) = a\, y(n-1) + b\, x(n)$, and the desired value is d(n).
If the input x(n) and the noise v(n) are uncorrelated, white sequences with zero mean value and a signal-to-noise ratio $S = \sigma_x^2/\sigma_v^2$, the equation-error (D algorithm) estimate a of the coefficient α is biased:
$(a-\alpha) / \alpha = (\alpha^2-1) / (1-\alpha^2+\beta^2 S)$.
The equation-error estimate of b is unbiased ($b = \beta$).
Conversely, both output-error estimates (UD algorithms) are unbiased.
We computed analytically the expectation value of the squared error ($N_c=1$) in the case of a semi-directed algorithm, and determined the values of a and b which minimize it. Figure 5 shows the biases with respect to N_t for S = 10; the bias of a decreases from the above value (for $N_t=1$) to zero ($N_t \rightarrow \infty$), which is consistent with the fact that a SD algorithm with $N_t=1$ is a D algorithm, and that it is a UD algorithm if $N_t \rightarrow \infty$. Furthermore, it is shown that the bias of b is zero in the two limiting cases (D and UD), and that it is small, but non-zero for $N_t>1$.
To summarize, the use of SD algorithms provides, in this example, a tradeoff between the stability of D algorithms and the unbiased estimates which result from the use of a UD algorithm.

Similarly, we consider the second process identification example described in [13] which illustrates the fact that, if the order of the model is smaller than the order of the process, the *output-error* formulation generates an error surface (MSOE) which may have local minima, whereas the *equation-error* formulation generates an error surface (MSEE) which has only a global minimum.
The process to be modelled is simulated by:
$d(n) = \alpha_1\, d(n-1) + \alpha_2\, d(n-2) + \beta_0\, x(n) + \beta_1\, x(n-1)$,
where x is the input and d the output of the process.
The model used in the adaptive filter is described by:
$y(n) = a\, y(n-1) + b\, x(n)$, and the desired value is d(n).
The MSE surface is a paraboloid when using the *equation-error* formulation (MSEE). In the case of the *output-error* formulation, the MSE surface (MSOE) exhibits one local minimum (which corresponds to a damped oscillatory behaviour, $-1<a<0$), and one global minimum ($0<a<1$) . We have computed analytically the MSE in the case of a semi-directed algorithm with $N_c=1$. As for the first example, the MSE surface changes from the MSEE surface to the MSOE surface when N_t increases from 1 to ∞: for $N_t=2$, a second minimum appears, with $1<a<0$, which shifts to the location of the local minimum of the MSOE surface when N_t grows; meanwhile, the other minimum shifts from the location of the minimum of the MSEE surface to the location of the global minimum of the MSOE surface.

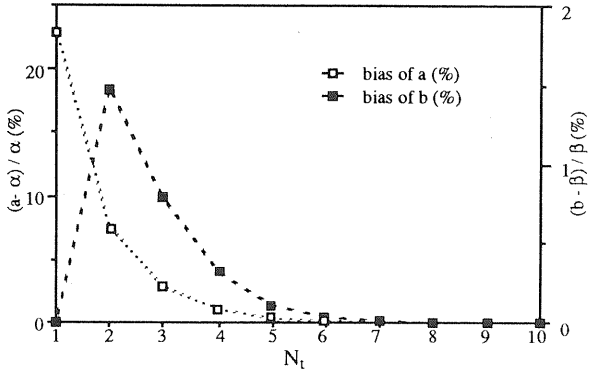

FIGURE 5

CONCLUSION

We have shown that a large variety of algorithms are available for training recurrent neural networks to perform adaptive filtering, and that the algorithms used thus far are but a small fraction of the available possibilities. We have illustrated some features of the new algorithms on three examples. Neural networks, viewed as adaptive non-linear filters, have a considerable potential which needs be explored, and basic issues, such as the stability of the algorithms, are still open.

Acknowledgements
The authors wish to thank L. CAPELY and D. MARSAN for computer simulations.
This work has been supported in part by EEC contract ST2J 0312 C.

References

[1] O. Nerrand , P. Roussel-Ragot, L. Personnaz, G. Dreyfus, S. Marcos, "Neural Networks and Non-linear Adaptive Filtering: Unifying Concepts and New Algorithms", Neural Computation, to be published.

[2] S. Marcos, P. Roussel-Ragot, L. Personnaz, O. Nerrand, G. Dreyfus, C. Vignat, "Réseaux de Neurones pour le Filtrage Non Linéaire Adaptatif", Traitement du Signal, in press (1992).

[3] B. Widrow, S.D. Stearns, Adaptive Signal Processing (Prentice-Hall, 1985).

[4] S. Chen, S.A. Billings, "Representations of Non-Linear Systems: the NARMAX Model", Int. J. Control, vol. 49, pp. 1013-1032, 1989.

[5] M.I. Jordan, "Attractor Dynamics and Parallelism in a Connectionist Sequential Machine", in Proceedings of the Eighth Annual Conference of the Cognitive Science Society, 1986, pp. 531-546.

[6] L. Ljung, T. Söderström, Theory and Practice of Recursive Identification, M.I.T. Press, 1983.

[7] R.J. Williams, D. Zipser, "A Learning Algorithm for Continually Running Fully Recurrent Neural Networks", Neural Computation, vol. 1, pp. 270-280, 1989.

[8] R.J. Williams, J. Peng, "An Efficient Gradient-based Algorithm for On-Line Training of Recurrent Network Trajectories", Neural Computation, vol. 2, pp. 490-501, 1990.

[9] P.L. Feintuch, "An Adaptive Recursive LMS Filter", Proc. IEEE, pp. 1622-1624, 1976

[10] C.R. Johnson, I.D. Landau, "On Adaptive IIR Filters and Parallel Adaptive Identifiers with Adaptive Error Filtering", Proc. ICASSP, pp. 5387, 1981.

[11] N.S. Jayant, P. Noll, Digital Coding of Waveforms. Principles and Applications to Speech and Video, Signal Processing Series, A. Oppenheim, ed., Prentice-Hall, 1984.

[12] C. Vignat, C. Uhl, S. Marcos, "Analysis of gradient-based adaptation algorithms for linear and nonlinear recursive filters", Proceedings of ICASSP-92, Vol. IV, pp. IV 189-IV 192, March 23-26, 1992, San Francisco.

[13] J. J. Shynk, "Adaptive IIR Filtering", IEEE ASSP Magazine, pp. 4-21, 1989.