DYNAMICAL SYSTEMS LEARNING BY A CIRCUIT THEORETIC APPROACH

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

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

ABSTRACT

In this paper, we derive a new general method for both on-line and off-line backward gradient computation of a system output, or cost function, with respect to system parameters, using a circuit theoretic approach. The system can be any causal, in general non-linear and time-variant, dynamic system represented by a Signal Flow Graph, in particular any feedforward, time delay or recurrent neural network. The gradient is obtained in a straightforward way by the analysis of two numerical circuits, the original one and its adjoint (obtained from the first by simple transformations) without the complex chain rule expansions of derivatives usually employed.

1. INTRODUCTION

In this paper, making use of the Signal Flow Graph (SFG) representation theory and its known properties [4], we derive *a new general method for both on-line and off-line* backward gradient computation of a system output or cost function with respect to system parameters. The obtained gradient information can be used both for parameter sensitivity computation of systems and for training of adaptive systems. The system can be any causal, in general non-linear and time-variant, dynamical system represented by a SFG, in particular any feedforward (static), time delay or recurrent neural networks [2,3,9]. In this work we use discrete time notation, but the same theory holds for the continuous time case. The gradient is obtained by the analysis of two SFGs, the original one and its adjoint, without the complex chain rule expansions of derivatives usually employed [2,3,9]. This method can be used also for off-line learning. In this case, our approach is equivalent to the method by Wan and Beaufays [8] that is not suited for on-line training of recurrent networks or systems with feedback.

The derivation here proposed (not reported due to space limitation), is more general than the one in [8] and explicitly designed to allow on-line training of adaptive systems. While we use for the proof a theorem similar to Tellegen's theorem, proved in [4], the proof by Wan and Beaufays is based on the interreciprocity property of transposed graphs [8], that is a consequence of the theorem used by us. Moreover our formulation allows to deal both with the sensitivity calculation problem, i.e. computing the derivatives of the system output with respect to parameters, and the learning problem while [8] addressed just the second one.

This work also generalizes a previous one by Osowski [7] that proposed a SFG approach to neural networks learning. That work was basically for feedforward networks and the extension to recurrent networks proposed is valid only for networks that relaxes to a fixed point, as also in the work by Almeida [1]; moreover delay branches were not allowed. General recurrent networks able to process temporal sequences cannot be trained by the proposed adjoint formulas.

A related work is also the one by Martinelli and Perfetti [5] that applies an adjoint transformation to an analog real circuit implementing a Multi-Layer-Perceptron (MLP). Since this work is more related to hardware and the network considered is feedforward and static, it is less general and can be considered a particular case of the method presented here.

On-line training of adaptive systems is very important in real applications such as Digital Signal Processing, system identification and control, channel equalization and predistortion, since it allows the model to follow time-varying features of the real system [2,3]. By on-line learning in contrast to batch learning this is provided with no interruption of the operation of the system (such as a communication channel).

2. SIGNAL FLOW GRAPHS DEFINITION AND NOTATION

In this Section we give a brief presentation of signal flow graphs [4,6,7] following Lee's approach and notation, refer to [4] for more details. The relationship between the initial and final variables for branch j is $v_j = f_j\left[x_j\right]$. By definition, for each n-node, the value of the x variables associated with its outcoming branches is the summation of all v variables associated with its incoming branches. Let the SFG consist of $p+m+r$ nodes (m input nodes, r output nodes, and p n-nodes), and $q+m+r$ branches (m input branches, r output branches, and q f-branches).

A reversed SFG G_r of a SFG G is obtained by reversing the orientation of all the branches in G, this means replacing summing junctions with branching points and viceversa. The functional relationships between the variables v_k and x_k in G and between $\hat{x}_k$ and $\hat{v}_k$ in G_r, is usually considered to be the same (transposed graphs). On the contrary, at this point, we do not specify the branch operators but we are only interested in the connection topology. Let $\hat{u}_k, \hat{y}_k, \hat{x}_k$ and $\hat{v}_k$, be the variables associated with the branches in G_r corresponding to u_k, y_k, x_k and v_k in G. In G_r the input variables are $\hat{y}_k \quad k=1,\ldots,r$, the output variables are $\hat{u}_k \quad k=1,\ldots,m$; $\hat{v}_k$ and $\hat{x}_k$ are the initial and the final variable of the f-branch k. Let the numbers assigned to the branches of G_r be the same as the numbers assigned to the corresponding branches of G. The following theorem, similar to Tellegen's theorem in analog

[+] This research was supported by the Italian MURST.
Please address comments to the first author.

network theory, is derived in [4]. It depends only on the topological properties of reversed graphs and not on the specific functional relationships between the branch variables.

Theorem: Let's consider a causal dynamical system. Let *G* be the corresponding signal flow graph and G_r a reversed signal flow graph. If $\hat{u}_k, \hat{y}_k, \hat{x}_k, \hat{v}_k$ are the variables in G_r corresponding to u_k, y_k, x_k, v_k in *G*, respectively, then

$$\sum_{i=1}^{r} \hat{y}_i(t) * y_i(t) + \sum_{i=1}^{q} \hat{x}_i(t) * x_i(t) = \sum_{i=1}^{m} \hat{u}_i(t) * u_i(t) + \sum_{i=1}^{q} \hat{v}_i(t) * v_i(t) \tag{2.1}$$

where "*" denote the convolution operator.

3. SENSITIVITY COMPUTATION

A general method to compute the derivative of an output of a dynamical continuous time system represented by a signal flow graph with respect to a *constant* parameter α is derived in [4]. For a discrete time system, an analogous result can be found, in a similar way. However such method has two drawbacks for our purposes; the first is that it is directly applicable only in off-line mode, and the second is that the mathematical meaning of the variables of the adjoint graph is not directly related to derivatives. Here, we derive *a new method* to calculate, both on-line and off-line, the derivatives of a system output with respect to past (or present) parameters of the system. In this way, both the above problems are solved and adaptive systems with time varying parameters can be considered.

In the case of discrete time systems, and according to the notation introduced in Section 2, the functional relationship between the variables x_j and v_j of the *f*-branch *j* can be written in this way

$$v_j(t) = g_j(x_j(t), \alpha_j(t), t) \quad \text{static branch} \tag{3.1}$$

$$v_j(t) = \boldsymbol{q}^{-1} x_j(t) = x_j(t-1) \quad \text{delay branch} \tag{3.2}$$

where $\boldsymbol{q}^{-1}$ is the delay operator and g_j is a general differentiable function, depending on the adaptable parameter (or vector of parameters) α_j.

For a static branch the relationship between x_j and v_j can be particularized as

$$v_j(t) = w_j(t) x_j(t) \quad \text{or} \quad v_j(t) = f_j(x_j(t)) \tag{3.3}$$

where $w_j(t)$ is the *j*-th parameter of the system at time *t*, and f_j is a differentiable function. In neural networks context f_j can be a sigmoidal activation function. More in general (3.1) can represent a non-linear adaptable function controlled by a parameter (or vector) α_j, such as a spline. All these basic components can be interconnected in a general way (with feedback allowed) to provide a complex SFG, describing a complex system, just as electronic devices are connected by a circuit to implement a system.

The derivative of an output (y_k) of the system with respect to a parameter $w_i(t-\tau)$, by the first equation in (3.3) is

$$\frac{\partial y_k(t)}{\partial w_i(t-\tau)} = \frac{\partial y_k(t)}{\partial v_i(t-\tau)} \frac{\partial v_i(t-\tau)}{\partial w_i(t-\tau)} = \frac{\partial y_k(t)}{\partial v_i(t-\tau)} x_i(t-\tau) \tag{3.4}$$

more in general by (3.1):

$$\frac{\partial y_k(t)}{\partial \alpha_i(t-\tau)} = \frac{\partial y_k(t)}{\partial v_i(t-\tau)} \frac{\partial v_i(t-\tau)}{\partial \alpha_i(t-\tau)} =$$

$$= \frac{\partial y_k(t)}{\partial v_i(t-\tau)} \frac{\partial g_i}{\partial \alpha_i}\bigg|_{x_i(t-\tau), \alpha_i(t-\tau), t-\tau} \stackrel{\Delta}{=} \frac{\partial y_k(t)}{\partial v_i(t-\tau)} \frac{\partial g_i}{\partial \alpha_i}(t-\tau)$$

Thus we need to compute $\partial y_k(t) / \partial v_i(t-\tau)$, which is the derivative of an output variable with respect to a signal in the system, precisely the signal getting out from the *f*-branch *i* at time ($t-\tau$). Let *G* be the graph of the system and $\hat{G}$ the adjoint graph that we are going to define to compute these derivatives. *G* and $\hat{G}$ have the same topology but the orientation of each branch in $\hat{G}$ is reversed from that of *G*, i.e. $\hat{G}$ is a reversed graph of *G*. The relations associated with the *f*-branches in $\hat{G}$ are properly chosen to allow a derivation, not reported here, and are shown in **Tab.** 1.

For each delay operator branch in the original SFG the corresponding branch of the adjoint SFG is to be considered a delay branch (with zero initial condition). The non-linearity without parameters f_j in the original SFG corresponds to a gain equal to the derivative of f_j computed at the initial variable of the branch at time *t*-τ in the adjoint SFG, where *t* is the time of the original SFG and τ of the adjoint SFG. The weight at time *t* is substituted by the weight at time t-τ in the adjoint SFG. A similar transformation should be performed for the general non-linearity with control parameters, as in **Tab.** 1.

The output of the original SFG is the input of the adjoint SFG. To get the proper sensitivities the adjoint SFG must be feeded by an impulse signal in correspondence with the output of which computing the sensitivity and by a constant null signal in all other outputs, see **Tab.** 1.

For an example of adjoint SFG construction for a simple MLP with Infinite Impulse Response (IIR) filter synapses [2,3] see **Fig.** 1.

In order to compute the gradient using the adjoint graph let $\hat{u}_j, \hat{y}_j, \hat{x}_j$ and $\hat{v}_j$ be the variables associated with the branches in $\hat{G}$ corresponding to u_j, y_j, x_j and v_j in *G*; it follows by the theorem (the proof is not reported here, due to space limitation):

$$\frac{\partial y_k(t)}{\partial v_i(t-\tau)} = \hat{v}_i(\tau) \tag{3.5}$$

Using this result in (3.4) we get

$$\frac{\partial y_k(t)}{\partial w_i(t-\tau)} = \hat{v}_i(\tau) x_i(t-\tau) \tag{3.6}$$

This result state that the derivative of an output variable of a SFG with respect to one of its parameters $w_i(t-\tau)$ is the product of the initial variable of the *i*-th branch in the original SFG at time $(t-\tau)$ and the initial variable of the *i*-th branch in the adjoint graph, at τ time units. This result can be easily generalized for the non-linear parametric function:

$$\frac{\partial y_k(t)}{\partial \alpha_i(t-\tau)} = \hat{v}_i(\tau)\frac{\partial g_i}{\partial \alpha_i}(t-\tau) \tag{3.7}$$

4. NON-LINEAR DYNAMICAL SYSTEMS LEARNING

Let's consider a discrete time non-linear dynamical system with inputs $u_i \quad i=1...m$, outputs $y_i \quad i=1...r$, and parameters w_i and α_i, which have to be adapted with respect to an output error.

The instantaneous global squared error at time *t* is defined as

$$e^2(t) = \sum_{k=1}^{r} e_k^2(t) \quad , \quad e_k(t) = d_k(t) - y_k(t) \tag{4.1}$$

where $d_k(t) \quad k=1,\ldots,r$ are the desired outputs. So the global squared error over a time interval $[t_0, t_1]$ is

$$E(t_0, t_1) = \sum_{t=t_0}^{t_1} e^2(t) \tag{4.2}$$

In the case of non-adaptive (or batch) training, we can choose as cost function, for example the global squared error over the all learning epoch $E(0,T)$, where *T* is the final time of the epoch. Whereas if we want on-line training, we can choose to consider only the most recent errors $e^2(t)$, for example using $E(t-N_c+1,t)$ as cost function, where *t* is the current time step.

The choice of the MSE as cost function is arbitrary for the SFG method and any cost function that can be represented by a SFG with y_k and d_k as inputs and the cost as output can be considered, as explained in the following.

Considering gradient based learning algorithms with the variation of the weights calculated by the steepest-descent method, the parameters updating rule is

$$\Delta w_i = -\mu \frac{\partial J}{\partial w_i} \qquad , \quad \mu > 0 \tag{4.3}$$

Where *J* is a cost function and μ is the learning rate. The problem is the calculation of the derivative $\partial J / \partial w_i$. Since the parameters can change at each time instant:

$$\frac{\partial J}{\partial w_i} = \sum_{\tau=0}^{t} \frac{\partial J}{\partial w_i(\tau)} = \sum_{\tau=0}^{t} \frac{\partial J}{\partial w_i(t-\tau)} \tag{4.4}$$

By the Backward Computation (BC) SFG technique we may calculate $\partial J / \partial w_i(t-\tau)$.

Since the length of the summation increases linearly with *t*, for on-line learning, in order to apply the algorithm, the summation in (4.4) must be truncated, that is

$$\Delta w_i = -\mu \sum_{\tau=t-h+1}^{t} \frac{\partial J}{\partial w_i(\tau)} = -\mu \sum_{\tau=0}^{h-1} \frac{\partial J}{\partial w_i(t-\tau)} \tag{4.5}$$

where *h* is a fixed positive integer. In this way not the all history of the system is considered, but only the most recent part in the interval [*t*-*h*+1,*t*]. In fact it is easy to show that a real truncation is necessary only for circuits with feedback, e.g. recurrent neural networks whereas for feedforward networks with delays (e.g. Time Delay Neural Networks or TDNNs) a finite *h* can be chosen so that all the memory of the system is taken into account. For a perfect selection of *h*, it should be dependent on which parameter is considered; for layered TDNN *h* should depend on the layer and chosen higher for layers closer to the input one, since more memory is involved in computing the gradient with respect to first layers parameters. The above updating law (4.5) holds theoretically in the hypothesis of constant weights but practically it is a good approximation. Equations (3.6) and (3.7) instead does not require that hypothesis, and can be used in a more general context.

The method derived in Section 3 can be used to on-line calculate the gradient of a cost function with respect to the parameters

$$\frac{\partial J}{\partial w_i} = \sum_{\tau=0}^{h-1} \hat{v}_i(\tau) x_i(t-\tau) \tag{4.6}$$

that can be easily generalized

$$\frac{\partial J}{\partial \alpha_i} = \sum_{\tau=0}^{h-1} \hat{v}_i(\tau) \frac{\partial g_i}{\partial \alpha_i}(t-\tau) \tag{4.7}$$

To apply this formulas we need to consider the SFG which yields the cost function as its output and that is formed by the connection of the non-linear adaptive system and a second SFG (the error calculation SFG) which has the outputs y_k and the targets d_k of the first SFG as inputs and *J* as output, see **Fig.** 2. Obviously the SFG of the cost can contain the same operators as the system SFG, i.e. delays, non-linearities, parameters to be adapted (e.g. for regularization purpose), feedback. In some applications the cost function needed can be very complex but still, by the SFG representation its use is feasible.

In order to calculate the gradient, the adjoint of the overall system should be considered, but it can be shown that it can be calculated considering the adjoint SFG only, for simple MSE based cost functions.

Applying this method in a non adaptive context, the parameter variations computed by the summation of all gradient terms in [0,*T*] are the same of those reported in [8] (BPTT method).

This work extends previous results in different fields and in particular provides a nice analogy with the sensitivity analysis for analog networks by the "adjoint network" obtained applying Tellegen's theorem that relates voltages and currents of an analog network. In that case the "adjoint network" method allows to compute sensitivity of a circuit output with respect to all the component values solving only one circuit, i.e. the adjoint circuit, [10].

5. REFERENCES

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

[2] 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.

[3] P. Campolucci, A. Uncini, F. Piazza, "*A Unifying View of Gradient Calculations and Learning for Locally Recurrent Neural Networks*", Proc. of the Italian Workshop on Neural Networks (WIRN97), May 1997.

[4] A.Y. Lee, "*Signal Flow Graphs--Computer-Aided System Analysis and Sensitivity Calculations*", IEEE Transactions on Circuits and Systems, vol. cas-21, no. 2, pp. 209-216, March 1974.

[5] G. Martinelli and R. Perfetti, "*Circuit theoretic approach to the backpropagation learning algorithm*", Proc. of the IEEE Int. Symposium on Circuits and Systems, 1991.

[6] A.V. Oppenheim and R.W. Schafer, "*Digital Signal Processing*", Prentice-Hall, 1975.

[7] S. Osowski, "*Signal flow graphs and neural networks*", Biological Cybernetics, 70, pp. 387-395, 1994.

[8] E.A. Wan and F. Beaufays "*Diagrammatic Derivation of Gradient Algorithms for Neural Networks*", Neural Computation 8: 182-201, 1996.

[9] P.J. Werbos, "*Backpropagation through time: what it does and how to do it*", Proc. of IEEE, Special issue on neural networks, vol. 78, no. 10, pp. 1550-1560, 1990.

[10] S.W. Director and R.A. Rohrer, "*The generalized adjoint network and network sensitivities*", IEEE Trans. on Circuit Theory, vol. CT-16, pp. 318-323, Aug. 1969.

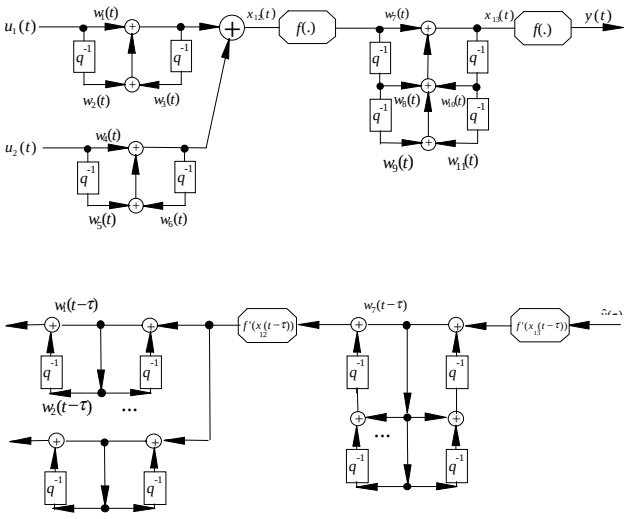

Fig. 1. (a) SFG of a simple IIR-MLP ; (b) its adjoint.

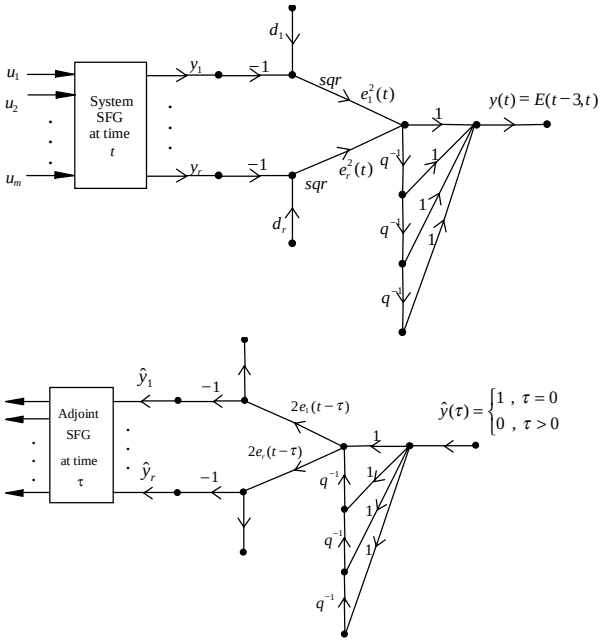

Fig. 2. (a) System SFG and error calculation SFG, with $J=E(t\text{-}3,t)$ and $sqr(x)=x^2$; (b) its adjoint.

Branch Classification	Original Signal-Flow-Graph	Adjoint Signal-Flow-Graph for Parameter Sensitivity on y_k
Delay	$v_j(t)=q^{-1}x_j(t)\ ,\ v_j(0)=0$	$\hat{x}_j(\tau)=q^{-1}\hat{v}_j(\tau)\ ,\ \hat{x}_j(0)=0$
Static	$v_j(t)=f_j(x_j(t))$	$\hat{x}_j(\tau)=\hat{h}_j(\tau)\hat{v}_j(\tau)$ $\hat{h}_j(\tau)=f_j'(x_j(t-\tau))$
	$v_j(t)=w_j(t)x_j(t)$	$\hat{x}_j(\tau)=\hat{w}_j(\tau)\hat{v}_j(\tau)$ $\hat{w}_j(\tau)=w_j(t-\tau)$
	$v_j(t)=g_j(x_j(t),\alpha_j(t),t)$	$\hat{x}_j(\tau)=\hat{h}_j(\tau)\hat{v}_j(\tau)$ $\hat{h}_j(\tau)=\left.\frac{\partial g_j}{\partial x_j}\right\|_{x_j(t-\tau),\alpha_j(t-\tau),t-\tau}$
System Output	y_j	$\hat{y}_j$ $\hat{y}_j(\tau)=\begin{cases}0 & \forall\tau \quad \text{if } j\neq k\\ \begin{cases}1\ ,\ \tau=0\\ 0\ ,\ \tau>0\end{cases} & \text{if } j=k\end{cases}$

Tab. 1. Adjoint SFG transformations**.**