RECURRENT NEURAL NETWORKS FOR SIGNAL PROCESSING TRAINED BY A NEW SECOND ORDER ALGORITHM

Paolo Campolucci[*], *Michele Simonetti, Aurelio Uncini*

Dipartimento di Elettronica ed Automatica
Università di Ancona, 60131 Ancona, Italia.
[*] E-mail: paolo@eealab.unian.it; Internet: http://nnsp.eealab.unian.it

ABSTRACT

A new second order algorithm based on Scaled Conjugate Gradient for training recurrent and locally recurrent neural networks is proposed. The algorithm is able to extract second order information performing two times the corresponding first order method. Therefore the computational complexity is only about two times the corresponding first order method. Simulation results show a faster training with respect to the first order algorithm. This second order algorithm is particularly useful for tracking fast varying systems.

1 INTRODUCTION

Several learning algorithms for neural networks have been proposed in the literature and many of them are based on the well known gradient descend algorithm.

However, second-order algorithms [3,5,9,10] can exhibit better performances than first order ones because they also use the second-order information stored in the Hessian matrix. There are several examples of these algorithms in literature; a sub-class of them, based on the conjugate gradient method, has shown good properties in terms of rate of convergence and computational complexity [9].

Conjugate direction methods are based on choosing the search direction and the step size of a minimization formula by using second order information. It holds

$$E(\mathbf{w}+\mathbf{y}) \approx E(\mathbf{w}) + \nabla_w E(\mathbf{w})^T \mathbf{y} + \frac{1}{2}\mathbf{y}^T \mathbf{H}(\mathbf{w})\mathbf{y} \qquad (1.1)$$

where $E(\mathbf{w})$ is a generic cost function, of weights $\mathbf{w}$, to be minimized, $\mathbf{H}(\mathbf{w})$ is the Hessian matrix, and $\mathbf{y}$ the weight variation.

The Conjugate Gradient (CG) algorithm is based on the following two iterative formulas respectively for updating weights $\mathbf{w}_k$ and conjugate directions $\mathbf{p}_k$ [9]:

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \frac{\mathbf{p}_k^T \nabla E(\mathbf{w}_k)}{\mathbf{p}_k^T \mathbf{H}(\mathbf{w}_k)\mathbf{p}_k}\,\mathbf{p}_k \qquad (1.2)$$

$$\mathbf{p}_{k+1} = \mathbf{r}_{k+1} + \frac{|\mathbf{r}_{k+1}|^2 - \mathbf{r}_{k+1}^T \mathbf{r}_k}{\mathbf{p}_k^T \mathbf{r}_k}\,\mathbf{p}_k \qquad (1.3)$$

where k is the iteration index and $\mathbf{r}_k = -\nabla E(\mathbf{w}_k)$.

This algorithm has two drawbacks. The first is that for each iteration the Hessian matrix $\mathbf{H}(\mathbf{w}_k)$ has to be computed and stored [4]; the second one is that this algorithm works only for functions with positive definite Hessian matrices, and the quadratic approximations can be very poor when the current point is far from a local minimum.

M. Moller [10] has proposed a solution to the second problem based on the Levenberg-Marquardt algorithm combined with the conjugate gradient approach. The problem of the Hessian matrix definition is solved trying to make always positive the quantity in the denominator of (1.2), adding a positive term, which is determined recursively. This algorithm is called Scaled Conjugate Gradient (SCG) [10] and results to be better than CG in terms of convergence properties.

This research was partially supported by the Italian MURST.

With respect to the first problem some methods exist to extract information on the Hessian matrix without calculating or storing it.

From (1.2) we note that the algorithm needs to calculate only the vector $\mathbf{H}(\mathbf{w}_k)\mathbf{p}_k$, not the matrix $\mathbf{H}(\mathbf{w}_k)$ alone. Therefore, having an efficient technique to obtain directly the product $\mathbf{H}(\mathbf{w}_k)\mathbf{p}_k$ there is no need for computing and storing the whole Hessian matrix.

For this reason, in this work a new method for computing such product has been applied to SCG, obtaining a computationally efficient algorithm, with good speed of convergence but without the complexity and memory usage typical of the second order methods already known in literature.

The derived method has been applied to the difficult problem of training recurrent neural networks in on-line mode [2,6,7,8,12,1]. Several simulation results showing faster training in non-linear system identification tests by locally recurrent neural networks are reported for the on-line case, comparing them with those obtained by several first order algorithms.

The complexity of the proposed method is accurately compared with that of the corresponding first order algorithm showing an average increase of about 2 times in terms of number of operations per iteration.

2 EXTRACTION OF SECOND ORDER INFORMATION

A method exists [5] to perform the calculation of **H(w)p,** where **p** is now the conjugate gradient and **H** the Hessian matrix, as a difference between two gradients, computed in two different points.

This technique has four main phases, that must be performed every iteration (of index *k*):

1. compute the gradient $\nabla_w E(\mathbf{w})$ of the cost function with respect to the weights vector **w** ;
2. compute $\mathbf{u} = \mathbf{w} - \nabla_w E(\mathbf{w})$;
3. compute the gradient $\nabla_u E(\mathbf{u})$ of the cost function with respect to the vector **u** ;
4. compute the quantity

$$\mathbf{H}(\mathbf{w})\mathbf{p} = \nabla_w E(\mathbf{w}) - \nabla_u E(\mathbf{u})$$

as a difference between the two gradients calculated before.

This technique is exact only if the cost function $E(\mathbf{w})$ is quadratic but in practice it gives an approximation of the product **H(w)p**.

This technique results to be simple and efficient and can be applied to any neural network (static or dynamic). It allows exploiting the Hessian properties without explicitly calculating it, and using only first order formulas. An important advantage is when implementing it in software, because it uses directly the same formulas of the gradient calculation.

Must also be stressed that the previous technique have never been applied to training neural networks by the SCG algorithm, being implemented only for linear adaptive filtering by CG. The SCG with the technique discussed above gives a new algorithm which is faster than the first order counterpart, as shown in the section on simulation results.

After using this algorithm with some static problems, with good performances, we have applied it to train locally recurrent neural networks, as explained in the next section.

3 SCG-*U* ALGORITHM FOR IIR-MLP NEURAL NETWORKS

With the same notation used in [6,7,8], let consider the generic neuron *k* in the layer *l* of a IIR-MLP neural network with inputs *x*, weights *w* and targets *d*, trained by an epoch of *T* learning patterns. The vector of all the weights of a whole IIR-MLP network is given by

$$\tilde{\mathbf{w}} = \begin{bmatrix} \mathbf{w} \\ \mathbf{v} \end{bmatrix} \qquad \text{where}$$

$$\mathbf{w} = \left[\mathbf{w}^{(1)} \ldots \mathbf{w}^{(l)} \ldots \mathbf{w}^{(M)} \right]^T$$

$$\mathbf{w}^{(l)} = \left[w^{(l)}_{10(0)} \ldots w^{(l)}_{kj(p)} \ldots w^{(l)}_{N_{l+1}N_l(L^{(l)}_{Nl+1,Nl}-1)} \right]$$

$$\mathbf{v} = \left[\mathbf{v}^{(1)} \ldots \mathbf{v}^{(l)} \ldots \mathbf{v}^{(M)} \right]^T$$

$$\mathbf{v}^{(l)} = \left[v^{(l)}_{10(0)} \ldots v^{(l)}_{kj(p)} \ldots v^{(l)}_{N_{l+1}N_l(I^{(l)}_{Nl+1,Nl})} \right] .$$

Therefore the generic element of conjugate direction, because of its definition, will depend on index *k*, *j* and *l* , giving the following vector

$$\tilde{\mathbf{p}} = \begin{bmatrix} \mathbf{pw} \\ \mathbf{pv} \end{bmatrix} \qquad \text{where}$$

$$\mathbf{pw} = \left[\mathbf{pw}^{(1)} \ldots \mathbf{pw}^{(l)} \ldots \mathbf{pw}^{(M)} \right]^T$$

$$\mathbf{pv} = \left[\mathbf{pv}^{(1)} \ldots \mathbf{pv}^{(l)} \ldots \mathbf{pv}^{(M)} \right]^T$$

$$\mathbf{pw}^{(l)} = \left[pw_{10(0)}^{(l)} \cdots pw_{kj(p)}^{(l)} \cdots pw_{N_{l+1}N_l(L_{Nl+1,Nl}^{(l)}-1)}^{(l)} \right]$$

$$\mathbf{pv}^{(l)} = \left[pv_{10(0)}^{(l)} \cdots pv_{kj(p)}^{(l)} \cdots pv_{N_{l+1}N_l(I_{Nl+1,Nl}^{(l)})}^{(l)} \right]$$

As stated above, applying the previous technique to calculate the products $\mathbf{H}(\tilde{\mathbf{w}})\tilde{\mathbf{p}}$, we obtain a new algorithm based on the conjugate gradient method. This algorithm contain only first order formulas, i.e. gradient calculation. A critical point is the use of an efficient gradient calculation for locally recurrent networks; for this purpose we have chosen the Truncated Recursive Back Propagation (TRBP) [8] algorithm.

The products $\mathbf{H}(\tilde{\mathbf{w}})\tilde{\mathbf{p}}$ is calculated using the following expression

$$\mathbf{H}(\tilde{\mathbf{w}})\tilde{\mathbf{p}} = \nabla_{\tilde{w}} E(\tilde{\mathbf{w}}) - \nabla_{\tilde{u}} E(\tilde{\mathbf{u}})$$

where $\tilde{\mathbf{u}} = \tilde{\mathbf{w}} - \nabla_{\tilde{w}} E(\tilde{\mathbf{w}})$.

The $\nabla_{\tilde{u}} E(\tilde{\mathbf{u}})$ computation is performed by applying TRBP to a second network that has the same structure of the first one, but with weights vector $\tilde{\mathbf{u}}$ instead of $\tilde{\mathbf{w}}$. Therefore we have the same formulas of RBP, or TRBP, but applied to a set of different variables that we call *u*-variables.

Computational complexity and memory storage of the SCG-*u* are about double with respect to TRBP algorithm, because in practice we performed TRBP two times for each iteration: firstly to calculate $\nabla_{\tilde{w}} E(\tilde{\mathbf{w}})$ and then to calculate $\nabla_{\tilde{u}} E(\tilde{\mathbf{u}})$.

The SCG-*u* formulas are not reported, because they are the same as for TRBP.

4 SIMULATION RESULTS

In this section, the results of applying the new SCG-*u* algorithm to two problems of identification of non-linear dynamic systems, reported in literature, are presented.

The locally recurrent architecture [12] chosen for the simulations is the IIR-MLP with two layers: three hidden neurons with hyperbolic tangent activation function and one linear output neuron.

We compare SCG with the TRBP first-order on-line algorithm. The results are given in terms of Mean-Square-Error (MSE) expressed in dB, and its variance, 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.

The first set of experiments consists in identifying the non-linear system with memory presented in [2]. The network used has both MA and AR order equal to three for each neuron. From Fig. 1 is clear that the new SCG algorithm has good performances.

The second set of experiments was carried out on the more realistic problem of identifying a baseband equivalent Pulse Amplitude Modulation (PAM) transmission system in presence of non linearity, see [6,7] for details. The network used in this case has MA and AR order respectly equal to four and two for each neuron. Fig. 2 shows that the new SCG algorithm performs very well.

The second-order algorithm seems to perform better than first-order ones. Moreover the complexity of SCG algorithm, in terms of number of floating point operations (flop) per iteration, is only about 2 times higher with respect to TRBP, as shown in Tab. 1. This is an important result since other second-order algorithms already presented in literature have in general a much higher complexity.

REFERENCES

[1] Campolucci P., Uncini A., Piazza F., "*Fast adaptive IIR-MLP neural networks for signal processing applications*", Proc. of ICASSP-96 IEEE Int. Conference on Acoustic Speech and Signal Processing, Atlanta (USA), May 1996.

[2] Back A.D. and Tsoi A.C., "*A simplified gradient algorithm for IIR-MLP synapse Multilayer Perceptrons*", Neural Computation 5, pp.456-462, 1993.

[3] Battiti R., "*First- and Second-Order Methods for Learning : Between Steepest Descend and Newton's Method*", Neural Computation, 4, 1992.

[4] Bishop C., "*Exact calculation of the Hessian matrix for the multilayer perceptron*", Neural Computation 4(4), 1992.

[5] Boray G.K. and Srinath M.D., "*Conjugate Gradient Tecniques for Adaptive Filtering*", IEEE Transaction on Circuits and Systems-I :Fundamental Theory and applications, Vol. 39, No. 1, Jan. 1992.

[6] Campolucci P., Piazza F., Uncini A., "*On-line learning algorithms for neural networks with IIR synapses*", Proc. of the IEEE International Conference of Neural Networks, Perth, Nov. 1995.

[7] Campolucci P., Uncini A., Piazza F., "*A Unifying View of Gradient Calculations and Learning for Locally Recurrent Neural Networks*", Proc. of WIRN97, Italian Workshop on Neural Networks, Vietri sul Mare (Italy), Spriger-Verlag ed., May 1997.

[8] Campolucci P., Uncini A., Piazza F., "*A new IIR-MLP learning algorithm for on-line signal processing*", International Conference of Acoustic Speech and Signal Processing (ICASSP97), Munich, April 1997.

[9] Fletcher R. and Reeves C.M., "*Function minimization by conjugate gradients*", The computer Journal, 1964.

[10] Möller M., "*A scaled conjugate gradient algorithm for fast supervised learning*", Neural Networks 6(4) , 1993.

[11] Haykin S., "Ne*ural Networks Expand Sp's Horizons*" IEEE Signal Processing Magazine, vol. 13 no. 2, March 1996.

[12] Tsoi A.C. and Back A.D., "*Locally recurrent globally feedforward networks: a critical review of architectures*", IEEE Transactions on Neural Networks, vol. 5, no. 2, 229-239, March 1994.

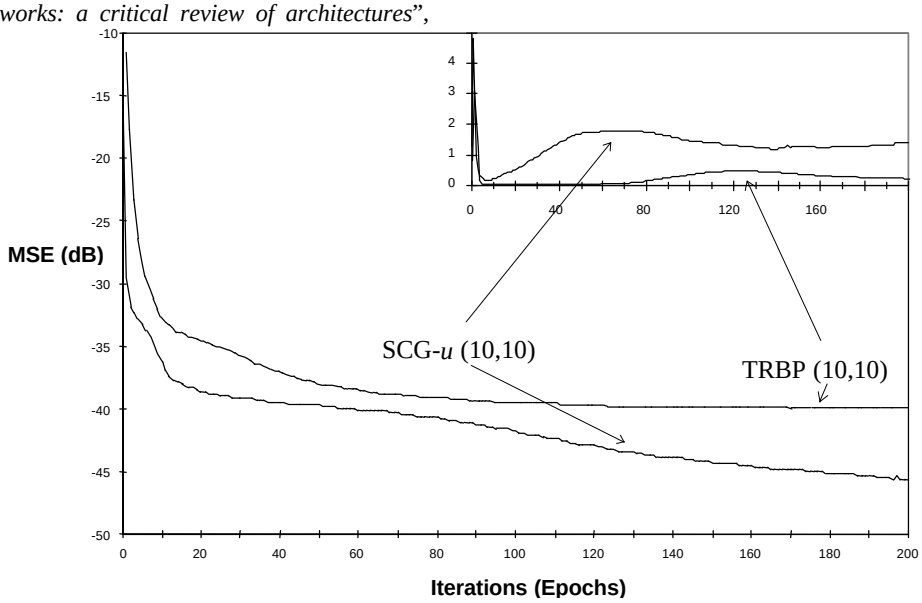

Fig. 1 Back-Tsoi test results in on-line mode, using SCG-*u* (*h*,*h*') and TRBP(*h*,*h*'). *h* is the past history length considered and *h*' is every how many samples parameters are adapted.

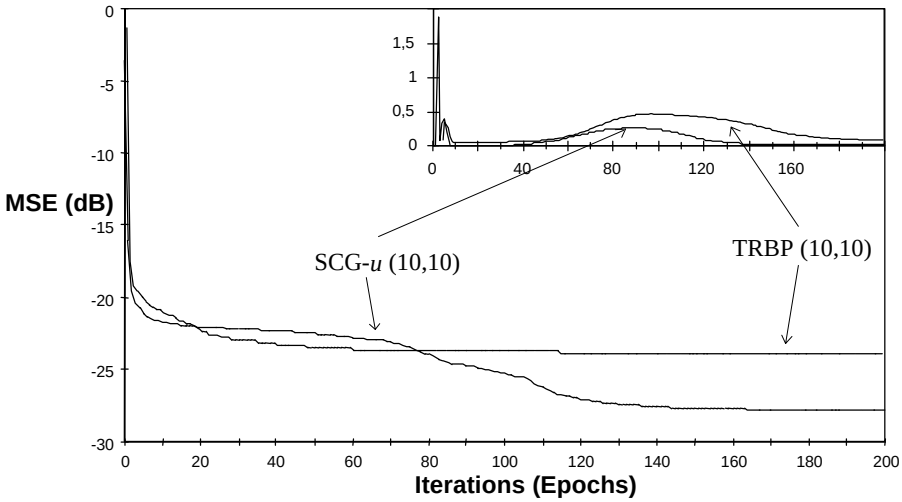

Fig. 2 PAM test results in on-line mode, using SCG-*u* (*h*,*h*') and TRBP(*h*,*h*').

	Back-Tsoi (1000 samples)	P.A.M. (2048 samples)
TRBP(10,10)	1.852e+006	2.692e+006
SCG- *u* (10,10)	3.534e+006	5.065e+006

Tab. 1 Computational complexity on the Back-Tsoi and P.A.M. (on-line mode) tests - number of flops per iteration.