INFORMATICA, 2003, Vol. 14, No. 2, 223–236

Quick Training Algorithm for Extra Reduced Size Lattice-Ladder Multilayer Perceptrons

Dalius NAVAKAUSKAS *

Department of Radioelectronics, Vilnius Gediminas Technical University
Naugarduko 41, LT-2600 Vilnius, Lithuania
e-mail: dalius@el.vtu.lt

Received: November 2002

Abstract. A quick gradient training algorithm for a specific neural network structure called an extra reduced size lattice-ladder multilayer perceptron is introduced. Presented derivation of the algorithm utilizes recently found by author simplest way of exact computation of gradients for rotation parameters of lattice-ladder filter. Developed neural network training algorithm is optimal in terms of minimal number of constants, multiplication and addition operations, while the regularity of the structure is also preserved.

Key words: lattice-ladder filter, lattice-ladder multilayer perceptron, adaptation, gradient adaptive lattice algorithms.

1. Introduction

During the last decade a lot of new structures for artificial neural networks (ANN) were introduced fulfilling the need to have models for nonlinear processing of time-varying signals (Haykin, 1999; Tsoi and Back, 1997; Juditsky *et al.*, 1995; Sjöberg *et al.*, 1995). One of many and perhaps the most straightforward way to insert temporal behaviour into ANNs is to use digital filters in a place of synapses of multilayer perceptron (MLP). Following that way, a time-delay neural network (Waibel *et al.*, 1989), FIR/IIR MLPs (Back and Tsoi, 1991; Wan, 1993), gamma MLP (Lawrence *et al.*, 1997), a cascade neural network (Back and Tsoi, 1996) to name a few ANN architectures, were developed.

A lattice-ladder realization of IIR filters incorporated as MLP synapses forms a structure of lattice-ladder multilayer perceptron (LLMLP) firstly introduced by (Back and Tsoi, 1992) and followed by several simplified versions proposed by the author (Navakauskas, 1998; Navakauskas, 2001). A LLMLP is an appealing structure for advanced signal processing, however, even moderate implementation of its training hinders the fact that a lot of storage and computation power must be allocated (Navakauskas, 1999).

*For the financial support author would like to thank prof. L. Ljung (Linköping University, Sweden), The Royal Swedish Academy of Sciences and The Swedish Institute – New Visby project Ref. No. 2473/2002 (381/T81).

Well known neural network training algorithms such as backpropagation and its modifications, conjugate-gradients, Quasi–Newton, Levenberg–Marquardt, etc. or their adaptive counterparts like temporal-backpropagation (Wan, 1990), IIR MLP training algorithm (Back and Tsoi, 1993), recursive Levenberg–Marquardt (Ngia and Sjöberg, 2000), etc., essentially are based on the use of gradients (also called sensitivity functions) – partial derivatives of the cost function with respect to current weights. Here we are going to show how the exploration of specific to lattice-ladder filter (LLF) computation of gradients leads to efficient realization of overall family of LLMLP training algorithms. While doing so, we present quickest in terms of number of constants, multiplication and addition operations training algorithm for an extra reduced size LLMLP (XLLMLP).

2. Towards Simplified Computations

In order not to obscure main ideas, we will first work only with one LLF and afterwards in Section 4 generalize and utilize results in a development of training algorithm for a specific – XLLMLP structure. Although, the final LLF training algorithm is fairly simple, many intermediate steps involved in the derivation could be confusing. Thus, here we will present previous results on calculation of LLF gradients, introduce the simplifications we have chosen, and only in next Section 3 actually derive all simplified expressions.

Consider one lattice-ladder filter (see Fig. 1) used in XLLMLP structure, when its computations are expressed by

$$\begin{bmatrix} f_{j-1}(n) \\ b_j(n) \end{bmatrix} = \begin{bmatrix} \cos\Theta_j & -\sin\Theta_j \\ \sin\Theta_j & \cos\Theta_j \end{bmatrix} \begin{bmatrix} f_j(n) \\ z b_{j-1}(n) \end{bmatrix}, \quad j = 1, 2, \ldots, M, \tag{1a}$$

$$s_{\text{out}}(n) = \sum_{j=0}^{M} v_j b_j(n), \tag{1b}$$

with boundary conditions

$$b_0(n) = f_0(n), \qquad f_M(n) = s_{\text{in}}(n). \tag{1c}$$

Here we used the following notations: $s_{\text{in}}(n)$ and $s_{\text{out}}(n)$ are signals at input and output of Mth order LLF, $f_j(n)$ and $b_j(n)$ are forward and backward signals floating in jth section of LLF, Θ_j and v_j are lattice and ladder coefficients correspondingly, while n is time index and z is a delay operator such that $z b_j(n) \equiv b_j(n-1)$.

It could be shown (see, for example, (Haykin, 1996)) that gradient expressions for the calculation of LLF coefficients require M recursions, yielding a training algorithm with total complexity proportional to M^2. One possible way to simplify the LLF gradients calculation was presented in (Rodriguez–Fonollosa and Masgrau, 1991).

We assume that the concept of *flowgraph transposition* is already known (if not – consult, for example, (Regalia, 1995; pages 291–293)). Applying the flowgraph transposition rules to the LLF equations (1a) and (1b), we obtain the LLF transpose realization.

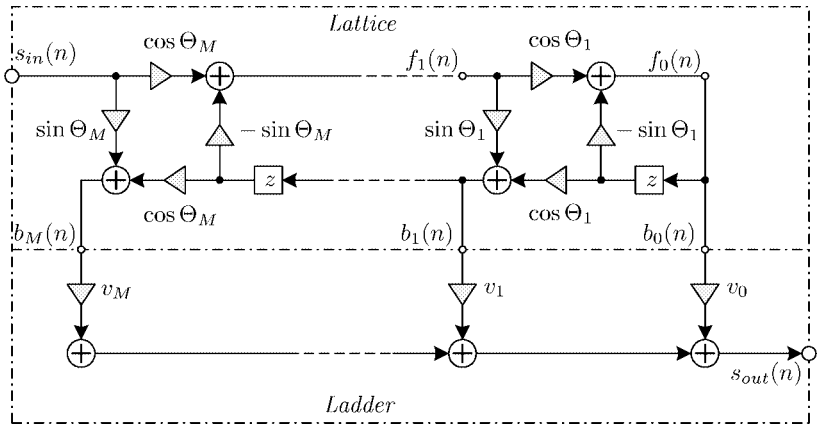

Fig. 1. A lattice-ladder filter of order M. Input signal $s_{\text{in}}(n)$ is processed according to (1) using lattice Θ_j and ladder v_j coefficients and as a result output signal $s_{\text{out}}(n)$ is obtained. Elements of the scheme: z in a square – a time delay, filled triangle – multiplication by a coefficient, plus sign in a circle – addition.

The resulting system gives rise to the following recurrent relation

$$\begin{bmatrix} g_j(n) \\ t_{j-1}(n) \end{bmatrix} = \begin{bmatrix} 1 & \\ & z \end{bmatrix} \begin{bmatrix} \cos\Theta_j & \sin\Theta_j \\ -\sin\Theta_j & \cos\Theta_j \end{bmatrix} \begin{bmatrix} g_{j-1}(n) \\ t_j(n) \end{bmatrix} + \begin{bmatrix} 0 \\ v_{j-1} \end{bmatrix}, \quad j = M, \ldots, 2, 1, \tag{2a}$$

with boundary conditions

$$t_M(n) = v_M, \quad g_0(n) = t_0(n), \quad g_M(n) = s_{\text{out}}(n). \tag{2b}$$

Here by $g_j(n)$ and $t_j(n)$ we indicated forward and backward signals floating in jth section of transposed LLF.

After simple re-arrangements (see, for example, (Navakauskas, 1999; pages 51–54)) it could be shown that filtered regressor components alternatively could be expressed as

$$\nabla\Theta_j(n) = \frac{f_j(n)t_j(n) - b_j(n)g_j(n)}{\cos\Theta_j}\, e(n). \tag{3}$$

Here $e(n)$ is a LLF error.

The main idea given by J. R.–Fonollosa and E. Masgrau towards simplifying the gradient computations for lattice parameters is to find a recurrence relation that realizes the mapping

$$\begin{bmatrix} f_{j-1}(n)t_{j-1}(n) \\ b_{j-1}(n)g_{j-1}(n) \end{bmatrix} \longrightarrow \begin{bmatrix} f_j(n)t_j(n) \\ b_j(n)g_j(n) \end{bmatrix}, \tag{4}$$

in such a way that all the necessary transfer functions may be obtained from one single filter resulting in an algorithm with total complexity proportional to M.

3. Derivation of Simplified Calculation of LLF Gradients

There are in total 14 possible and implementable ways of realization of mapping (4). The optimal way in a sense of minimal number of constants, addition and delay operations involved in the calculations of the gradients for LLF, and also regularity of the regressor lattice structure, was reported in (Navakauskas, 2002). It forms the basis for new training algorithm to be developed in next section. Thus, let us show in the step-by-step fashion the re-arrangements involved in the derivation of optimal realization of mapping (4).

Accordingly, we are seeking a simple realization for the following optimal augmented mapping

$$\begin{bmatrix} f_{j-1}(n)g_{j-1}(n) \\ b_{j-1}(n)g_{j-1}(n) \\ f_j(n)t_j(n) \\ b_j(n)t_j(n) \end{bmatrix} \longrightarrow \begin{bmatrix} f_j(n)g_j(n) \\ b_j(n)g_j(n) \\ f_{j-1}(n)t_{j-1}(n) \\ b_{j-1}(n)t_{j-1}(n) \end{bmatrix}, \tag{5}$$

while the overall system describing single section of regressor lattice is expressed by

$$\begin{bmatrix} f_{j-1}(n)g_{j-1}(n) \\ b_j(n)g_{j-1}(n) \end{bmatrix} = \begin{bmatrix} \cos\Theta_j & -\sin\Theta_j \\ \sin\Theta_j & \cos\Theta_j \end{bmatrix} \begin{bmatrix} f_j(n)g_{j-1}(n) \\ zb_{j-1}(n)g_{j-1}(n) \end{bmatrix}, \tag{6a}$$

$$\begin{bmatrix} f_{j-1}(n)t_{j-1}(n) \\ b_j(n)t_{j-1}(n) \end{bmatrix} = \begin{bmatrix} \cos\Theta_j & -\sin\Theta_j \\ \sin\Theta_j & \cos\Theta_j \end{bmatrix} \begin{bmatrix} f_j(n)t_{j-1}(n) \\ zb_{j-1}(n)t_{j-1}(n) \end{bmatrix}, \tag{6b}$$

$$\begin{bmatrix} f_j(n)g_j(n) \\ f_j(n)t_{j-1}(n) \end{bmatrix} = \begin{bmatrix} 1 & \\ & z \end{bmatrix} \begin{bmatrix} \cos\Theta_j & \sin\Theta_j \\ -\sin\Theta_j & \cos\Theta_j \end{bmatrix} \begin{bmatrix} f_j(n)g_{j-1}(n) \\ f_j(n)t_j(n) \end{bmatrix} + \begin{bmatrix} 0 \\ v_{j-1} \end{bmatrix} f_j(n), \tag{6c}$$

$$\begin{bmatrix} b_j(n)g_j(n) \\ b_j(n)t_{j-1}(n) \end{bmatrix} = \begin{bmatrix} 1 & \\ & z \end{bmatrix} \begin{bmatrix} \cos\Theta_j & \sin\Theta_j \\ -\sin\Theta_j & \cos\Theta_j \end{bmatrix} \begin{bmatrix} b_j(n)g_{j-1}(n) \\ b_j(n)t_j(n) \end{bmatrix} + \begin{bmatrix} 0 \\ v_{j-1} \end{bmatrix} b_j(n). \tag{6d}$$

In order to achieve mapping (5) two information flow directions in (6a) must be reversed: now $f_j(n)g_{j-1}(n)$ must be computed based on $f_{j-1}(n)g_{j-1}(n)$, and $b_{j-1}(n)t_{j-1}(n)$ must be computed based on $b_j(n)t_{j-1}(n)$.

For the first re-direction to be fulfilled we take (6a) and re-arrange it as follows

$$\begin{bmatrix} f_j(n)g_{j-1}(n) \\ b_j(n)g_{j-1}(n) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & z^{-1} \end{bmatrix} \begin{bmatrix} \dfrac{1}{\cos\Theta_j} & -\dfrac{\sin\Theta_j}{\cos\Theta_j} \\ -\dfrac{\sin\Theta_j}{\cos\Theta_j} & \dfrac{1}{\cos\Theta_j} \end{bmatrix} \begin{bmatrix} f_{j-1}(n)g_{j-1}(n) \\ b_{j-1}(n)g_{j-1}(n) \end{bmatrix}. \tag{7a}$$

Similarly, taking (6b) and re-arranging we get

$$\begin{bmatrix} f_{j-1}(n)t_{j-1}(n) \\ b_{j-1}(n)t_{j-1}(n) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & z^{-1} \end{bmatrix} \begin{bmatrix} \dfrac{1}{\cos\Theta_j} & -\dfrac{\sin\Theta_j}{\cos\Theta_j} \\ -\dfrac{\sin\Theta_j}{\cos\Theta_j} & \dfrac{1}{\cos\Theta_j} \end{bmatrix} \begin{bmatrix} f_j(n)t_{j-1}(n) \\ b_j(n)t_{j-1}(n) \end{bmatrix}. \tag{7b}$$

For the convenience we rewrite unchanged expressions (6c) and (6d) here

$$\begin{bmatrix} f_j(n)g_j(n) \\ f_j(n)t_{j-1}(n) \end{bmatrix} = \begin{bmatrix} 1 & \\ & z \end{bmatrix} \begin{bmatrix} \cos\Theta_j & \sin\Theta_j \\ -\sin\Theta_j & \cos\Theta_j \end{bmatrix} \begin{bmatrix} f_j(n)g_{j-1}(n) \\ f_j(n)t_j(n) \end{bmatrix} + \begin{bmatrix} 0 \\ v_{j-1} \end{bmatrix} f_j(n), \tag{7c}$$

$$\begin{bmatrix} b_j(n)g_j(n) \\ b_j(n)t_{j-1}(n) \end{bmatrix} = \begin{bmatrix} 1 & \\ & z \end{bmatrix} \begin{bmatrix} \cos\Theta_j & \sin\Theta_j \\ -\sin\Theta_j & \cos\Theta_j \end{bmatrix} \begin{bmatrix} b_j(n)g_{j-1}(n) \\ b_j(n)t_j(n) \end{bmatrix} + \begin{bmatrix} 0 \\ v_{j-1} \end{bmatrix} b_j(n). \tag{7d}$$

Now, (7) describes a system that has no conflicting information flow directions. However, there are two advance operations to be performed in (7a) and in (7b), making the system non-causal, hence un-implementable. There is, however, more to this computation than first meets the eye.

Notice first, the mapping (5) needs only four expressions to be specified. Thus a simplification of (7) becomes plausible. It could be shown that (7) could be simplified drastically and finally expressed by

$$\begin{aligned} \begin{bmatrix} f_j(n)g_j(n) \\ b_j(n)g_j(n) \\ f_{j-1}(n)t_{j-1}(n) \\ b_{j-1}(n)t_{j-1}(n) \end{bmatrix} &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ &\times \left(I_4 + \begin{bmatrix} \sin\Theta_j & 0 & 0 & 0 \\ 0 & \sin\Theta_j & 0 & 0 \\ 0 & 0 & -\sin\Theta_j & 0 \\ 0 & 0 & 0 & -\sin\Theta_j \end{bmatrix} \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \right) \\ &\times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & z & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_{j-1}(n)g_{j-1}(n) \\ b_{j-1}(n)g_{j-1}(n) \\ f_j(n)t_j(n) \\ b_j(n)t_j(n) \end{bmatrix} + v_{j-1} \begin{bmatrix} 0 \\ 0 \\ f_{j-1}(n) \\ b_{j-1}(n) \end{bmatrix}. \end{aligned} \tag{8a}$$

The simplified system of single section of regressor lattice is already causal. Moreover, its implementation requires only 2 delay operators, 3 constants ($\sin\Theta_j$, $-\sin\Theta_j$ and v_{j-1}) and 8 addition operations (more evident from pictorial representation of (8) that we skiped to save space). In order to finish the derivation, the boundary conditions when

M such systems are cascaded must be considered. Based on (1c) and (2b) we get such new boundary conditions

$$\begin{aligned} &f_0(n)g_0(n) = f_0(n)t_0(n), \qquad && b_0(n)g_0(n) = b_0(n)t_0(n), \\ &f_M(n)t_M(n) = v_M, && b_M(n)t_M(n) = v_M b_M(n). \end{aligned} \tag{8b}$$

4. Simplified Training Algorithm for Extra Reduced Size LLMLP

One way of LLMLP structure reduction could be achieved by restricting each neuron to have only one "output" lattice-ladder filter while connecting layers through conventional synaptic coefficients. It yields an extra reduced size lattice-ladder multilayer perceptron structure (XLLMLP) thats pictorial representation is given on next page in Fig. 2 and definition follows.

DEFINITION 1 (Navakauskas, 2001). A XLLMLP of size L layers, $\{N^0, N^1, \ldots, N^L\}$ nodes and filter orders $\{M^1, M^2, \ldots, M^L\}$ is defined by

$$s_h^l(n) = \Phi^l \underbrace{\left\{ \sum_{i=1}^{N^{l-1}} w_{ih}^l \underbrace{\sum_{j=0}^{M^l} v_{ij}^l \cdot b_{ij}^l(n)}_{=\tilde{s}_i^l(n)} \right\}}_{=\hat{s}_h^l(n)}, \quad h = 1, \ldots, N^l, \;\; l = 1, \ldots, L, \tag{9a}$$

when local flow of information in the lattice part of filters is defined by

$$\begin{bmatrix} f_{i,j-1}^l(n) \\ b_{ij}^l(n) \end{bmatrix} = \begin{bmatrix} \cos\Theta_{ij}^l & -\sin\Theta_{ij}^l \\ \sin\Theta_{ij}^l & \cos\Theta_{ij}^l \end{bmatrix} \begin{bmatrix} f_{ij}^l(n) \\ z b_{i,j-1}^l(n) \end{bmatrix}, \quad j = 1, 2, \ldots, M^l, \tag{9b}$$

with initial and boundary conditions

$$b_{i0}^l(n) = f_{i0}^l(n), \quad f_{i,M^{l+1}}^l(n) = s_i^{l-1}(n). \tag{9c}$$

Here $s_h^l(n)$ is an output signal of the "output" neuron; $\tilde{s}_i^l(n)$ is an output signal of the filter that is connected to i "input" neuron; w_{ih}^l represents single (static) weight connecting two neurons in a layer; Θ_{ij}^l and v_{ij}^l are weights of filter's lattice and ladder parts correspondingly; $b_{ij}^l(n)$ and $f_{ij}^l(n)$ are forward and backward prediction errors of the filter; i, j, h and l index inputs, filter coefficients, outputs and layers respectively.

Let us consider calculation of sensitivity functions using backpropagation algorithm for XLLMLP lth hidden layer neurons. It could be shown (Navakauskas, 2001) that sensitivity functions for XLLMLP could be expressed by

$$\nabla w_{ih}^l(n) = \tilde{s}_i^l(n)\, \delta_h^l(n), \tag{10a}$$

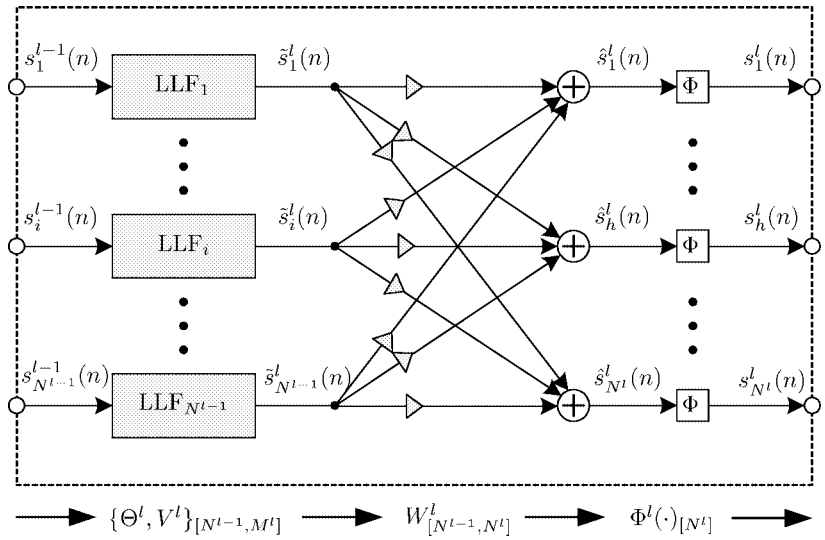

Fig. 2. A layer of XLLMLP. Input signals $s_{\mathrm{x}}^{l-1}(n)$ from a previous layer are processed according to (9) firstly by a group of lattice-ladder filters $\mathrm{LLF}_{\mathrm{x}}$ (see Fig. 1 for more detailed treatment), then by static synapses $w_{\mathrm{x,x}}^l$, finally by neuron activation functions $\Phi^l(\cdot)$ forming current layer output signals $s_{\mathrm{x}}^l(n)$. Emphasized intermediate signals: LLF output signals $\tilde{s}_{\mathrm{x}}^l(n)$ and signals before activation functions $\hat{s}_{\mathrm{x}}^l(n)$. For the purpose of clearness size of main matrices (shown in brackets) are revealed at the bottom.

$$\nabla v_{ij}^l(n) = b_{ij}^l(n) \sum_{h=1}^{N^l} \delta_h^l(n), \tag{10b}$$

$$\nabla \Theta_{ij}^l(n) = \sum_{r=0}^{M^l} v_{ir}^l(n) \frac{\partial}{\partial \Theta_{ij}^l} b_{ir}^l(n) \sum_{h=1}^{N^l} w_{ih}^l \delta_h^l(n). \tag{10c}$$

Note, that these expressions are similar to standard LLF gradient expressions in a way that here we used additional indexes to state LLF position in the whole XLLMLP architecture (being precise, we showed expressions for sensitivity functions for jth coefficients of LLF connecting ith input neuron with hth output neuron in lth layer of LLMLP) and replaced output error $e(n)$ term by the generalized local instantaneous error, i.e., $\delta_h^l(n) = \partial E(n)/\partial \hat{s}_h^l(n)$, that could be explicitly expressed by

$$\delta_h^l(n) = \begin{cases} -e_h^l(n) \Phi'^L\big\{\hat{s}_h^l(n)\big\}, & l = L, \\ \Phi'^l\big\{\hat{s}_h^l(n)\big\} \sum\limits_{j=0}^{M^{l+1}} v_{hj}^{l+1} \gamma_{hj}^{l+1}(n) \sum\limits_{p=1}^{N^{l+1}} w_{hp}^{l+1} \delta_p^{l+1}(n), & l \neq L, \end{cases} \tag{10d}$$

with $\gamma_{hj}^{l+1}(n)$ and its companion $\varphi_{hj}^{l+1}(n)$ by

$$\begin{bmatrix} \varphi_{h,j-1}^{l+1}(n) \\ \gamma_{hj}^{l+1}(n) \end{bmatrix} = \begin{bmatrix} \cos\Theta_{hj}^{l+1} & -\sin\Theta_{hj}^{l+1} \\ \sin\Theta_{hj}^{l+1} & \cos\Theta_{hj}^{l+1} \end{bmatrix} \begin{bmatrix} \varphi_{hj}^{l+1}(n) \\ z\gamma_{h,j-1}^{l+1}(n) \end{bmatrix}. \tag{10e}$$

Aiming to present simplified training of XLLMLP we replace (10c) with

$$\nabla\Theta_{ij}^{l}(n) = \underbrace{\frac{f_{ij}^{l}(n)t_{ij}^{l}(n) - b_{ij}^{l}(n)g_{ij}^{l}(n)}{\cos\Theta_{ij}^{l}}}_{=\,\widehat{\nabla}\Theta_{ij}^{l}(n)} \sum_{h=1}^{N^l} w_{ih}^{l}\delta_{h}^{l}(n), \tag{11}$$

where order recursion computation is done in a way of (8) by

$$\begin{bmatrix} f_{ij}^{l}(n)g_{ij}^{l}(n) \\ b_{ij}^{l}(n)g_{ij}^{l}(n) \\ f_{i,j-1}^{l}(n)t_{i,j-1}^{l}(n) \\ b_{i,j-1}^{l}(n)t_{i,j-1}^{l}(n) \end{bmatrix} = \begin{bmatrix} 1 & z\sin\Theta_{ij}^{l} & \sin\Theta_{ij}^{l} & 0 \\ \sin\Theta_{ij}^{l} & z & 0 & \sin\Theta_{ij}^{l} \\ -z\sin\Theta_{ij}^{l} & 0 & z & -z\sin\Theta_{ij}^{l} \\ 0 & -z\sin\Theta_{ij}^{l} & -\sin\Theta_{ij}^{l} & 1 \end{bmatrix} \times \begin{bmatrix} f_{i,j-1}^{l}(n)g_{i,j-1}^{l}(n) \\ b_{i,j-1}^{l}(n)g_{i,j-1}^{l}(n) \\ f_{ij}^{l}(n)t_{ij}^{l}(n) \\ b_{ij}^{l}(n)t_{ij}^{l}(n) \end{bmatrix} + v_{i,j-1}^{l} \begin{bmatrix} 0 \\ 0 \\ f_{i,j-1}^{l}(n) \\ b_{i,j-1}^{l}(n) \end{bmatrix}, \tag{12a}$$

with boundary conditions:

$$f_{i0}^{l}(n)g_{i0}^{l}(n) = f_{i0}^{l}(n)t_{i0}^{l}(n), \quad b_{i0}^{l}(n)g_{i0}^{l}(n) = b_{i0}^{l}(n)t_{i0}^{l}(n), \tag{12b}$$

$$f_{iM^l}^{l}(n)t_{iM^l}^{l}(n) = v_{iM^l}^{l}, \qquad b_{iM^l}^{l}(n)t_{iM^l}^{l}(n) = b_{iM^l}^{l}(n)v_{iM^l}^{l}. \tag{12c}$$

Full XLLMLP training algorithm is presented at the end of the article. Let us here only summarize main steps of it:

1. *XLLMLP recall.* Given input pattern $s_i^0(n)$ is presented to the input layer (see line 1 of the algorithm). Afterwards it is processed through XLLMLP in a layer by layer fashion (lines 2–14): first by corresponding lattice-ladder filters (lines 3–9), then by static synapses (lines 10–13). In that way XLLMLP output pattern $s_h^L(n)$ is obtained.

2. *Node errors.* Using rezults of previous calculations, errors of output layer neurons are determined (line 15). Then, again in a layer by layer fashion – however this time in the backward direction – these errors are processed through XLLMLP (lines 16–23). Following that way errors of hidden layer neurons $\delta_h^l(n)$ are calculated.

3. *Gradient terms.* Simplified calculation of gradient terms for the lattice weight updates is done as in (12). Let us here be more specific. Working in a layer by layer

fashion and taking filter one by one (lines 24–38; note however that processing order here is insignificant), algorithm proceeds as follow: at line 25 initial conditions are assigned as in (12c); in lines 26–29 two lower equations from (12a) are evaluated for all sections of the filter; at line 30 left boundary conditions are fulfilled as in (12b); for all sections of the filter (lines 31–37) by lines 32–35 remaining upper equations from (12a) and in line 36 also $\widehat{\nabla}\Theta_{ij}^{l}(n)$ from (11) are evaluated.

4. *Weight updates.* Dynamic synapses (LLFs) are updated at lines 41 and 42 realizing expressions (10b) and (11), while static ones – at line 44 realizing (10a) and taking into account their previous values, training parameter μ, node errors and gradient terms.

5. *Stability test.* New parameter values of lattice filters are checked if they satisfy stability requirement (see lines 46–48), if some of them do not satisfy – old parameter values are substituted at line 47.

5. Conclusions

In this paper we dealt with the problem of computational efficiency of lattice-ladder multilayer perceptrons training algorithms that are based on the computation of gradients, for example, backpropagation, conjugate-gradient or Levenberg–Marquardt. Here we explored calculations that are most computationally demanding and specific to lattice-ladder filter – computation of gradients for lattice (rotation) parameters.

The optimal way in a sense of minimal number of constants, addition and delay operations involved in aforementioned computations for single lattice-ladder filter, and also regularity of the regressor structure, was found recently by the author. Based on it quick training algorithm for the extra reduced size lattice-ladder multilayer perceptron was here derived.

Not surprisingly, presented algorithm requires approximately M times (where M is the order of filter) less computations while it follows exact gradient path (because of the fact that derivations do not involved any approximations) when coefficients of filters are assumed to be stationary. More importantly, incorporated in the algorithm implementation of a single section of regressor lattice will require only 2 delay elements, 3 constants and 8 addition operations.

Experimental study of the proposed algorithm was not considered because of the fact that computations of gradients are exact and possible comparison inherently will be biased depending on the chosen way of implementation.

XLLMLP Training Algorithm

1. *XLLMLP Recall*

Let $f^1_{i,M^1}(n) = s^0_i(n)$.

for $l = 1, 2, \ldots, L$ **do**

  for $i = 1, 2, \ldots, N^{l-1}$ **do**

    for $j = M^l, \ldots, 2, 1$ **do**

$$\begin{bmatrix} f^l_{i,j-1}(n) \\ b^l_{ij}(n) \end{bmatrix} = \begin{bmatrix} \cos\Theta^l_{ij}(n) & -\sin\Theta^l_{ij}(n) \\ \sin\Theta^l_{ij}(n) & \cos\Theta^l_{ij}(n) \end{bmatrix} \begin{bmatrix} f^l_{ij}(n) \\ b^l_{i,j-1}(n-1) \end{bmatrix}.$$

    end for j

$$b^l_{i0}(n) = f^l_{i0}(n),$$

$$\tilde{s}^l_i(n) = \sum_{j=0}^{M^{l+1}} v^l_{ij}(n) b^l_{ij}(n).$$

  end for i

  for $h = 1, 2, \ldots, N^l$ **do**

$$\hat{s}^l_h(n) = \sum_{i=1}^{N^{l+1}} w^l_{ih} \tilde{s}^l_i(n),$$

$$s^l_h(n) = \Phi^l\left\{\hat{s}^l_h(n)\right\}.$$

  end for h

end for l

2. *Node Errors*

$$e^L_h(n) = d_h(n) - s^L_h(n), \quad h = 1, 2, \ldots, N^L.$$

for $l = L-1, L-2, \ldots, 1$ **and** $h = 1, 2, \ldots, N^l$ **do**

  Let $\gamma^l_{h,M^l}(n) = 0$.

  for $j = M^l, \ldots, 2, 1$ **do**

$$\begin{bmatrix} \varphi^l_{h,j-1}(n) \\ \gamma^l_{hj}(n) \end{bmatrix} = \begin{bmatrix} \cos\Theta^l_{hj}(n) & -\sin\Theta^l_{hj}(n) \\ \sin\Theta^l_{hj}(n) & \cos\Theta^l_{hj}(n) \end{bmatrix} \begin{bmatrix} \varphi^l_{hj}(n) \\ \gamma^l_{h,j-1}(n-1) \end{bmatrix}.$$

  end for j

$$\varphi^l_{h0}(n) = \gamma^l_{h0}(n).$$

$$\delta^l_h(n) = \begin{cases} -e^l_h(n)\Phi'^L\left\{\hat{s}^l_h(n)\right\}, & l = L, \\ \Phi'^l\left\{\hat{s}^l_h(n)\right\} \sum_{j=0}^{M^{l+1}} v^{l+1}_{hj}\gamma^{l+1}_{hj}(n) \sum_{p=1}^{N^{l+1}} w^{l+1}_{hp}\delta^{l+1}_p(n), & l \neq L. \end{cases}$$

end for h, l

3. *Gradient Terms*

for $l = 1, 2, \ldots, L$ **and** $i = 1, 2, \ldots, N^{l-1}$ **do**

Let $bt^l_{i,M^l} = v^l_{i,M^l}(n) b^l_{ij}(n)$, $ft^l_{i,M^l} = s^{l-1}_i(n) v^l_{i,M^l}(n)$.

for $j = M^l, \ldots, 2, 1$ **do**

$$bt^l_{i,j-1} = bt^l_{ij} - \left[ft^l_{ij}(n) + bg^l_{i,j-1}(n-1)\right] \sin\Theta^l_{ij}(n) + v^l_{i,j-1}(n) b^l_{i,j-1}(n),$$

$$ft^l_{i,j-1}(n) = ft^l_{i,j}(n-1) + v^l_{i,j-1}(n) f^l_{i,j-1}(n).$$

end for j

Let $temp = bt^l_{i0}$, $fg^l_{i0} = ft^l_{i0}(n-1)$.

for $j = 1, 2, \ldots, M^l$ **do**

$$ft^l_{i,j-1}(n) = ft^l_{ij}(n-1) - \left[fg^l_{i,j-1} + bt^l_{ij}\right] \sin\Theta^l_{ij}(n),$$

$$fg^l_{ij} = fg^l_{i,j-1} + \left[bg^l_{i,j-1}(n-1) + ft^l_{ij}(n)\right] \sin\Theta^l_{ij}(n),$$

$$bg^l_{i,j-1}(n) = temp,$$

$$temp = bg^l_{i,j-1}(n-1) + \left[fg^l_{i,j-1} + bt^l_{ij}\right] \sin\Theta^l_{ij}(n),$$

$$\widehat{\nabla}\Theta^l_{ij}(n) = \left[ft^l_{ij}(n) - temp\right] / \cos\Theta^l_{ij}(n).$$

end for j

end for i, l

4. *Weight Updates*

for $l = 1, 2, \ldots, L$ **and** $i = 1, 2, \ldots, N^{l-1}$ **do**

for $j = 1, 2, \ldots, M^l$ **do**

$$v^l_{ij}(n+1) = v^l_{ij}(n) + \mu b^l_{ij}(n) \sum_{h=1}^{N^l} \delta^l_h(n),$$

$$\Theta^l_{ij}(n+1) = \Theta^l_{ij}(n) + \mu \widehat{\nabla}\Theta^l_{ij}(n) \sum_{h=1}^{N^l} w^l_{ih} \delta^l_h(n).$$

end for j

for $h = 1, 2, \ldots, N^l$ **do** $w^l_{ih}(n+1) = w^l_{ih}(n) + \mu \delta^l_h(n) \tilde{s}^l_i(n)$.

end for i, l

5. *Stability Test*

for $l = 1, 2, \ldots, L$ **and** $i = 1, 2, \ldots, N^{l-1}$ **and** $j = 1, 2, \ldots, M^l$ **do**

if $|\Theta^l_{ij}(n+1)| > \pi/2$ **then** $\Theta^l_{ij}(n+1) = \Theta^l_{ij}(n)$.

end for j, i, l

References

Back, A.D., and A.C. Tsoi (1991). FIR and IIR synapses, a new neural network architecture for time series modeling. *Neural Computation*, **3**, 375–385.

Back, A.D., and A.C. Tsoi (1992). An adaptive lattice architecture for dynamic multilayer perceptrons. *Neural Computation*, **4**, 922–931.

Back, A.D., and A.C. Tsoi (1993). A simplified gradient algorithm for IIR synapse multilayer perceptrons. *Neural Computation*, **5**, 456–462.

Back, A.D., and A.C. Tsoi (1996). A cascade neural network model with nonlinear poles and zeros. In *Proc. of 1996 Int. Conf. on Neural Information Processing*, Vol. 1. pp. 486–491.

Haykin, S. (1996). *Adaptive Filter Theory*. 3rd ed. Prentice–Hall International, Inc.

Haykin, S. (1999). *Neural Networks: A Comprehensive Foundation*. 2nd ed. Prentice–Hall, Upper Saddle River, N.J.

Juditsky, A., H. Hjalmarsson, A. Benveniste, B. Delyon, L. Ljung, J. Sjöberg and Q. Zhang (1995). Nonlinear black-box modeling in system identification: Mathematical foundations. *Automatica*, **31**, 1725–1750.

Lawrence, S., A.D. Back, A.C. Tsoi and C.L. Giles (1997). The Gamma MLP – using multiple temporal resolutions for improved classification. *Neural Networks for Signal Processing*, **7**, 256–265.

Navakauskas, D. (1998). A reduced size lattice-ladder neural network. In *Signal Processing Society Workshop on Neural Networks for Signal Processing*, Cambridge, England. pp. 313–322.

Navakauskas, D. (1999). *Artificial Neural Network for the Restoration of Noise Distorted Songs Audio Records*. Doctoral dissertation, Vilnius Gediminas Technical University, No. 434.

Navakauskas, D. (2001). Reducing implementation input of lattice-ladder multilayer perceptrons. In *Proceedings of the 15th European Conference on Circuit Theory and Design*, Espoo, Finland, Vol. 3. pp. 297–300.

Navakauskas, D. (2002). Speeding up the training of lattice-ladder multilayer perceptrons. *Technical Report LiTH-ISY-R-2417*, Department of Electrical Engineering, Linköping University, SE-581 83 Linköping, Sweden.

Ngia, L.S.H., and J. Sjöberg (2000). Efficient training of neural nets for nonlinear adaptive filtering using a recursive Levenberg–Marquardt algorithm. *IEEE Trans. on Signal Processing*, **48**(7), 1915–1927.

Regalia, P. A. (1995). *Adaptive IIR Filtering in Signal Processing and Control*. Marcel Dekker, Inc.

Rodriguez–Fonollosa, J.A. and E. Masgrau (1991). Simplified gradient calculation in adaptive IIR lattice filters. *IEEE Trans. on Signal Processing*, **39**(7), 1702–1705.

Sjöberg, J., Q. Zhang, L. Ljung, A. Benveniste, B. Delyon, P.-Y. Glorennec, H. Hjalmarsson and A. Juditsky (1995). Nonlinear black-box modeling in system identification: A unified overview. *Automatica*, **31**, 1691–1724.

Tsoi, A.C. and A.D. Back (1997). Discrete time recurrent neural network architectures: A unifying review. *Neurocomputing*, **15**, 183–223.

Waibel, A., T. Hanazawa, G. Hinton *et al.* (1989). Phoneme recognition using time-delay neural networks. *IEEE Trans. on ASSP*, **37**(3), 328–339.

Wan, E. A. (1990). Temporal backpropagation for FIR neural networks. In *Proc. of International Joint Conf. on Neural Networks*. pp. 575–580.

Wan, E. A. (1993). *Finite Impulse Response Neural Networks with Applications in the Time Series Prediction*. Doctoral dissertation, Stanford University, USA.

D. Navakauskas is an associate professor at Radioelectronics Department of Vilnius Gediminas Technical University. He received honor diploma of radioelectronics engineer in 1992, M.Sc. in electronics degree in 1994, Doctor of electrical and electronical engineering degree in 1999, all at Vilnius Gediminas Technical University. His main research interests include artificial neural networks, speech signal processing and nonlinear signal processing.

Greitas ypatingai mažo dydžio pynučių-kopetėlių daugiasluoksnių perceptronų mokymo algoritmas

Dalius NAVAKAUSKAS

Straisnyje pateiktas greičiausias pagal naudojamų koeficientų bei sudėties ir daugybos operacijų skaičių ypatingai mažo dydžio pynučių-kopetėlių daugiasluoksnių perceptronų mokymo algoritmas. Jis yra išvestas nagrinėjant esminę visų gradientinių mokymo algoritmų dalį – specifinius pynučių-kopetėlių daugiasluoksniams perceptronams gradientų pynutėms skaičiavimus.