On the Choice of Parameters of the Cost Function in Nested Modular RNN's

Danilo P. Mandic, *Member, IEEE,* and Jonathon A. Chambers, *Senior Member, IEEE*

***Abstract*—We address the choice of the coefficients in the cost function of a modular nested recurrent neural-network (RNN) architecture, known as the pipelined recurrent neural network (PRNN). Such a network can cope with the problem of vanishing gradient, experienced in prediction with RNN's. Constraints on the coefficients of the cost function, in the form of a vector norm, are considered. Unlike the previous cost function for the PRNN, which included a forgetting factor motivated by the recursive least squares (RLS) strategy, the proposed forms of cost function provide "forgetting" of the outputs of adjacent modules based upon the network architecture. Such an approach takes into account the number of modules in the PRNN, through the unit norm constraint on the coefficients of the cost function of the PRNN. This is shown to be particularly suitable, since due to inherent nesting in the PRNN, every module gives its full contribution to the learning process, whereas the unit norm constrained cost function introduces a sense of forgetting in the memory management of the PRNN. The PRNN based upon a modified cost function outperforms existing PRNN schemes in the time series prediction simulations presented.**

***Index Terms*—Cost function, forgetting factor, nesting, pipelined recurrent neural networks, recurrent neural networks.**

I. Introduction

RECURRENT neural networks (RNN's) have been shown to be universal approximators [1], and to have important capabilities not found in feedforward networks, including attractor dynamics and the ability to store information for later use [2]. Computational power of RNN's with the sigmoid activation function, and their dynamical properties have been particularly considered [3].

However, RNN's encounter problems when learning information with long time dependencies, which is a problem in prediction of nonlinear and nonstationary signals, such as speech. In particular, Bengio *et al.* [4] showed that if a system is to latch information robustly, then the fraction of the gradient in a gradient-based training algorithm due to information, k steps in the past, approaches zero as k becomes large. This effect is called the problem of vanishing gradient. Several approaches were suggested to circumvent the problem of vanishing gradient in training RNN's, most of which rest on embedding memory into the neural network [5].

There are several methods for representing temporal information in neural networks [6]. These include: 1) creating a spatial representation of a temporal pattern; 2) putting time delays into the neurons or their connections; 3) employing recurrent connections; 4) using neurons with summing activation inputs over time; and 5) using a combination of 1)–4).

According to the embedding theorem [7], the memory orders need to be large enough in order to provide sufficient embedding. One way to embed memory in a neural network is through filtering synapses. In 1991, Back and Tsoi introduced finite impulse response (FIR) and infinite impulse response (IIR) synapses [8]. In 1993, Wan [9] discussed an architecture which models synapses as FIR linear filters for use in time series prediction. A numerical algorithm for short-term prediction of a time series based on delay coordinate embedding is presented in [10]. One of the most important examples of embedding memory in neural networks is the use of gamma memories, introduced by Principe and DeVries [11], [12]. They proposed a synapse that can be modeled as a convolution operator. Hochreuter and Schmidhuber [13] suggested a specific architectural approach which utilizes high-order gating units. For a comprehensive taxonomic approach based on memory types see Mozer [14], who highlighted the memory aspect of the synapses. A new RNN architecture that generalizes previous architectures by employing alternative discrete-time operations in place of the normally used shift operator, was introduced in [15]. Adjustable delays are an additional mechanism through which networks could achieve a broader range of dynamical trajectories [16].

Recently, there has been an attempt to reduce the complexity of adaptive learning by introducing a relationship between the learning rate and the slope of the nonlinear activation function in RNN's [17], [18]. There is also a number of researchers who have experimented with an architecture that is somewhere in between a feedforward only architecture and a full recurrent architecture (i.e., the Williams–Zipser model [19]). This class is called a locally recurrent globally feedforward (LRGF) architecture [20]. For a most comprehensive overview of the existing architectures see Tsoi and Back's work [21]. In 1998, Frasconi *et al.* [22] have made an attempt to unify adaptive models such as artificial neural nets and belief nets for the problem of processing structural information.

In 1996, Lin *et al.* introduced a so-called NARX RNN which is shown not to be as sensitive to the long time dependencies as networks in Bengio's work [23]. Namely, although embedded memory can be found in all recurrent network models, it is particularly prominent in NARX models [24]. It has been shown that, in theory, one can use NARX networks, rather than con-

Manuscript received April 6, 1998; revised December 1, 1998 and August 23, 1999.

D. P. Mandic is with the School of Information Systems, University of East Anglia, Norwich, NR4 7TJ, U.K. (e-mail: d.mandic@uea.ac.uk).

J. A. Chambers is with the Communications and Signal Processing Research Group of the Department of Electrical and Electronic Engineering, Imperial College of Science, Technology and Medicine, London, U.K. (e-mail: j.chambers@ic.ac.uk).

Publisher Item Identifier S 1045-9227(00)03009-5.

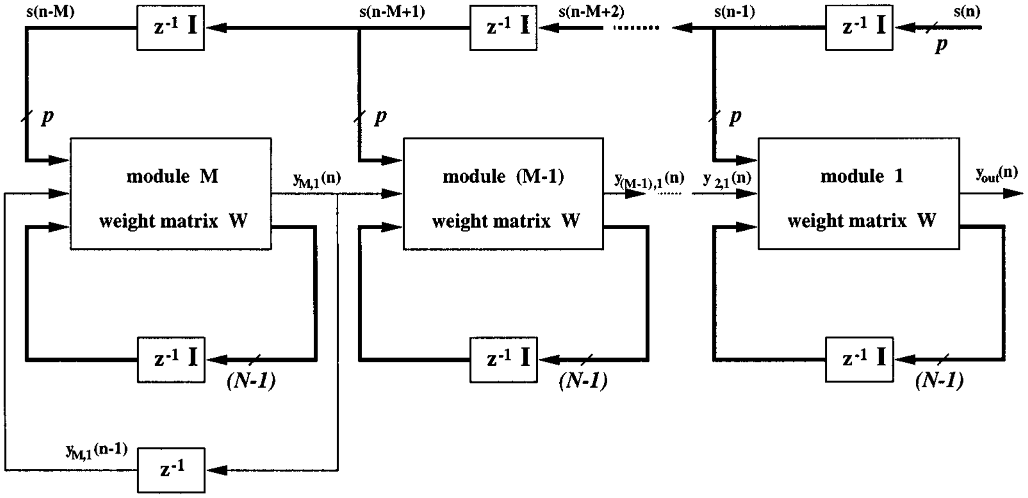


Fig. 1. Pipelined recurrent neural network.

ventional recurrent networks, without any computational loss and that they are at least equivalent to Turing machines [25].

Another way of introducing additional memory into a nonlinear system is by dividing the input space of a network $\mathcal{M} \subseteq \mathbb{R}^n$ into a number of disjoint or partially overlapping subspaces $\mathcal{M}_1 \subseteq \mathcal{M}$, $\mathcal{M}_2 \subseteq \mathcal{M}, \cdots, \mathcal{M}_m \subseteq \mathcal{M}$, which can be achieved by introducing modules which share the entire input space, i.e., $\mathcal{M}_1 \cup \mathcal{M}_2 \cup \cdots \cup \mathcal{M}_m = \mathcal{M}$. Such an architecture is the pipelined recurrent neural network (PRNN), which has been introduced in 1995 [26], and shown to be particularly suitable for nonlinear time series prediction [26]–[29].

The PRNN is a modular network, with every module being a single small-scale fully connected RNN. The PRNN is a realization of a NARMAX process [30], and in terms of the classification of [21], it belongs to a class of globally feedforward locally recurrent neural networks. Such an architecture includes features 1) and 3) from Kim's analysis [6]. Hence, it can cope with the problem of vanishing gradients by including additional memory not only in the structure of a module, but also spatially, along the modules. The PRNN can also offer a solution to some critical points in neural networks for least squares problems addressed by Sontag, such as a loss of dimensionality [31], and sample complexity, that is, the quantification of the amount of information (number of samples) needed to characterize a given mapping (Sontag *et al.* [32], [33]).

The learning algorithm for the PRNN is an extension of the learning algorithm chosen for constitutive RNN's, which is typically the real-time recurrent learning (RTRL) algorithm [19], [34]. The merit of the PRNN is its relatively low computational complexity, which for a network with M modules and N neurons per module, [i.e., $(M \times N)$ neurons], equals $\mathcal{O}(MN^4)$, whereas for the standard RNN with $M \times N$ neurons, it would be $\mathcal{O}\left((MN)^4\right)$. The RTRL algorithm for the PRNN is based upon a cost function which is identical in form to that for recursive least squares (RLS) filtering, and comprises a forgetting factor [26], [27], [35]. Although it has been shown that the PRNN architecture can outperform common RNN's in prediction applications, it has been done either through a choice of its architecture [26], or through an improved learning algorithm [29].

Our aim is to study further the cost function in the PRNN framework, and to impose norm-based constraints on it, in order to obtain improved performance for the PRNN-based prediction. Namely, the previously proposed cost function for the PRNN [26], [29], used the notion of forgetting inherited from the RLS algorithm, which did not take into account the network architecture. For a spatial architecture, such as the PRNN, this is not a suitable strategy. Hence, we introduce a unit norm constrained cost function with weighting along the modules which accounts for the number of modules, and inherently the depth of the embedded memory.

II. The PRNN

The PRNN architecture is shown in Fig. 1. A full mathematical description of the PRNN is given by the following equations [29], [35]:

$$y_{i,k}(n) = \Phi(v_{i,k}(n)) \qquad i = 1, \cdots, M,\ k = 1, \cdots, N \quad (1)$$

$$v_{i,k}(n) = \sum_{l=1}^{p+N+1} w_{k,l}(n) u_{i,l}(n) \quad (2)$$

$$\begin{aligned} \boldsymbol{u}_i^T(n) = {} & [s(n-i), \cdots, s(n-i-p+1), 1, \\ & y_{i+1,1}(n), y_{i,2}(n-1), \cdots, y_{i,N}(n-1)] \\ & \text{for } 1 \le i \le M-1 \end{aligned} \quad (3)$$

$$\begin{aligned} \boldsymbol{u}_M^T(n) = {} & [s(n-M), \cdots, s(n-M-p+1), 1 \\ & y_{M,1}(n-1), y_{M,2}(n-1), \cdots, y_{M,N}(n-1)] \\ & \text{for } i = M. \end{aligned} \quad (4)$$

Indexes i and k denote, respectively, the ith module within the PRNN ($i = 1, \cdots, M$), and the kth neuron within a module ($k = 1, \cdots, N$). Given the input vectors $\boldsymbol{u}_i(n)$ for each module i, $i = 1, \cdots, M$ at the time instant n, the outputs of all the neurons in the network can be calculated using the equations given previously. The two consecutive modules share $(p-1)$

input signals to the PRNN. Function Φ is the logistic sigmoid function.

At the time step n, for each module $i, \; i = 1, \cdots, M$, the one-step forward prediction error $e_i(n)$ associated with a module, is then defined as a difference between the desired response of that module $s(n-i+1)$, which is actually the next incoming sample of the external input signal, and the actual output of the ith module $y_{i,1}(n)$, of the PRNN, i.e.,

$$e_i(n) = s(n-i+1) - y_{i,1}(n). \tag{5}$$

Since the PRNN consists of M modules, a total of M forward prediction error signals are calculated. All of the modules share the same weight matrix $\boldsymbol{W}$. The goal is to minimize some measure of the error in the entire PRNN, termed a *cost function*, which was originally proposed as a weighted sum of all the squared error signals from individual modules [26]. In such a performance criterion, a *forgetting factor* λ, $\lambda \in (0, 1]$, is introduced which determines the weighting of the individual modules. Thus, the overall cost function of the PRNN becomes [26]

$$E(n) = \sum_{i=1}^{M} \lambda^{i-1} e_i^2(n) \tag{6}$$

where $e_i(n)$ is defined in (5).

The nesting implicit within the PRNN architectures [30] provides its increased nonlinearity [26], [28], [35], which is so attractive in nonlinear prediction and nonlinear system identification applications. Such nesting enables the underlying PRNN architecture to match closer the dynamics of a general nonlinear and nonstationary physical signal. The nesting principle is based upon a contraction mapping [36], [37], which under some mild conditions enables the process which propagates through a nested network to converge [38], [39]. A simple nesting principle for a general nonlinear function F with one variable is given by [37], [40]

$$\begin{aligned} \hat{x} &= F(x_m) = F(F(x_{m-1})) \\ &= \cdots = F(F(F(\cdots(F(x_1))\cdots))) \end{aligned} \tag{7}$$

and represents an implicitly written iterative process

$$\begin{aligned} x_{i+1} &= F(x_i) \Leftrightarrow x_{i+1} = F(F(x_{i-1})) \\ &= F(F(F(\cdots(F(x_1))\cdots))) \end{aligned} \tag{8}$$

which corresponds to the *a posteriori* mode of processing [28], [41], [42].

III. The Cost Function of the PRNN

The overall cost function of the PRNN is [26]

$$E(n) = \sum_{i=1}^{M} \lambda^{i-1} e_i^2(n) \tag{9}$$

where $\{e_i(n) = s(n-i+1) - y_{i,1}\}$, $i = 1, \cdots, M$ are the output errors at the modules in the PRNN. The value of the exponential weighting factor $0 < \lambda \leq 1$ is RLS motivated. The RLS strategy requires minimization of [41], [43], [44]

$$E(n) = \sum_{\tau=0}^{n} \beta^\tau e^2(n-\tau) \longrightarrow \text{minimum}, \qquad 0 < \beta \leq 1 \tag{10}$$

which is carried out recursively as

$$E(n) = \beta E(n-1) + e^2(n). \tag{11}$$

However, the processes $\{y_1(n)\}, \{y_2(n)\}, \cdots, \{y_M(n)\}$ at the outputs of the PRNN (Fig. 1) are not realizations of the same stochastic process. Indeed

$$\begin{aligned} y_{1,1}(n-1) &\neq y_{2,1}(n) \\ y_{2,1}(n-1) &\neq y_{3,1}(n) \\ \cdots \quad & \cdots \\ y_{(M-1),1}(n-1) &\neq y_{M,1}(n). \end{aligned} \tag{12}$$

Hence, the values of the forgetting factor $0 < \lambda \leq 1$, chosen upon the RLS strategy, make little sense in the PRNN framework, and other strategies need to be derived.

IV. Influence of λ on the Learning Process

The updating process of the weight matrix $\boldsymbol{W}$ of the PRNN can be written as [45], [46]

$$\boldsymbol{W}(n+1) = \boldsymbol{W}(n) + \Delta\boldsymbol{W}(n) \tag{13}$$

where the elements of the correction $\Delta\boldsymbol{W}$ to the weight matrix $\boldsymbol{W}$ can be calculated as

$$\begin{aligned} \Delta w_{k,l}(n) &= -\eta \frac{\partial}{\partial w_{k,l}(n)} \left(\sum_{i=1}^{M} \lambda^{i-1} e_i^2(n) \right) \\ &= -2\eta \sum_{i=1}^{M} \lambda^{i-1} e_i(n) \frac{\partial e_i(n)}{\partial w_{k,l}(n)}. \end{aligned} \tag{14}$$

Having in mind the nature of the elements of the sum (14), it can be seen as a weighted sum of the correction factors due to individual modules, namely

$$\begin{aligned} \Delta w_{k,l}(n) &= -\eta \frac{\partial}{\partial w_{k,l}(n)} \left(\sum_{i=1}^{M} \lambda^{i-1} e_i^2(n) \right) \\ &= -2\eta \sum_{i=1}^{M} \lambda^{i-1} \Delta w_{k,l}^i(n) \end{aligned} \tag{15}$$

or, equivalently

$$\Delta\boldsymbol{W}(n) = \sum_{i=1}^{M} \lambda^{i-1} \Delta\boldsymbol{W}_i(n) \tag{16}$$

where $\Delta w_{k,l}^i$ represents the correction factor of a single weight $w_{k,l}$ due to each individual module i, $1 \leq i \leq M$ and $\Delta\boldsymbol{W}_i$ represents the corresponding weight matrix correction due to module i. Therefore, dependent upon the values of the coefficients in the coefficient vector of (16)

$$\boldsymbol{\Lambda} = [\lambda^0, \lambda^1, \cdots, \lambda^{M-1}] \tag{17}$$

the total correction to the weight matrix $\boldsymbol{W}$, can be greater in magnitude than any of the individual corrections $\Delta \boldsymbol{W}_i(n)$, $i = 1, \cdots, M$, in (15). In that case, the value of the forgetting factor λ which provides best performance can even exceed unity [28]. That means that a distant module, i.e., a module whose index i is sufficiently close to M, has more influence in the learning process, than the output module, which is not desirable. It is therefore necessary to impose some constraints not only on the values of the coefficients in (9) and (16), but also on a linear combination of the coefficients λ^i, $i = 1, \cdots, M$, i.e., to introduce a norm which would account for the number of modules, as well as for forgetting along the modules.

V. Alternative Forms of the Cost Function

Notice that functions (9) and (16), provide filtering of their arguments, which can be written as

$$\Delta \boldsymbol{W} = \sum_{i=1}^{M} \xi_i \Delta \boldsymbol{W}_i, \qquad \xi_i = \lambda^{i-1} \tag{18}$$

which takes the form of a common finite impulse response (FIR) filter equation [43], [47]. It is therefore desirable to impose some constraints on the coefficient vector $\boldsymbol{\Lambda}$ (17) of the cost function (9) in order to provide a measure of the size of the final correction factor $\Delta \boldsymbol{W}$ to the weight matrix $\boldsymbol{W}$ as compared to the contributions from particular modules $\Delta \boldsymbol{W}_i$, $i = 1, \cdots, M$. A natural measure of a vector is its norm [36], [37], [48]. We will therefore consider cost functions with no amplification of the individual weight contribution, i.e., with the constraint

$$\|\boldsymbol{\Lambda}\| = 1. \tag{19}$$

This constraint is well founded in signal processing [41], [43], [47]. However, the norm in (19) can be either the $\|\cdot\|_1 = \mathcal{L}_1$, $\|\cdot\|_2 = \mathcal{L}_2$, or $\|\cdot\|_\infty = \mathcal{L}_\infty$ norm defined by

$$\|\boldsymbol{\Lambda}\|_l = \left(\sum_{i=1}^{M} |\lambda_i|^l \right)^{1/l}, \qquad l = 1, 2, \text{ or} \infty. \tag{20}$$

We consider therefore various forms of the cost function, through imposing constraints on the coefficients of the cost function (18) of the PRNN via $\|\cdot\|_1 = 1$, $\|\cdot\|_2 = 1$, or $\|\cdot\|_\infty = 1$.

A. Teaching Modules Separately

A method to improve the network may be to have a separate weight matrix $\boldsymbol{W}_i$ associated with each module i, $i = 1, \cdots, M$. Since in the common weight updating algorithm, all elements of the correction factor to each particular module have to be calculated (15), for separate weights there is no further expense in terms of the order of computational complexity, whereas there is a need for storing additional $(M-1) \times (N \times (N+p+1))$ elements of corresponding matrices. The adaptation is made in order to minimize each squared error signal $e_i^2(n)$, $i = 1, \cdots, M$ at the output of adjacent modules. A question emerges: Does training of the PRNN in each module separately, i.e., having a minimum mean square error (MSE) nonlinear predictor for each output signal of every module, mean that the overall performance achieved for the PRNN will be the best possible in the MSE sense? That is not the case, since

$$\min \left(\sum_{i=1}^{M} e_i^2(n) \right) \neq \sum_{i=1}^{M} \min(e_i^2(n)). \tag{21}$$

That is, although we can optimize each output of a PRNN module in the MSE sense, that does not mean that the overall performance of the PRNN achieves its optimum.

B. Averaging the Contributions of the Modules to the Weight Matrix Update

Since it is desirable to have the cost function of the PRNN robust to random disturbances [49], an averaging, i.e., smoothing filter, implicitly within the cost function of the PRNN could be a natural choice. In order to satisfy that paradigm, a correction to the weight matrix at time n can be calculated as

$$\Delta \boldsymbol{W}(n) = \frac{1}{M} \sum_{i=1}^{M} \Delta W_i(n) \tag{22}$$

i.e., as an average of the corrections due to particular modules, rather than the weighted sum (15), while still maintaining the common overall weight matrix $\boldsymbol{W}$. Notice that the sum of coefficients in (22) equals unity, which satisfies the unit norm constraint (19) in the $\mathcal{L}_1$ sense. Moreover, using this algorithm, means that the coefficients $\lambda_i = 1/M$, $i = 1, \cdots, M$, maintain the equality of modules in the information theoretic sense [50]. That means, that every module gives equal contribution in the sense of learning of the overall network, since there is no forgetting along the modules.

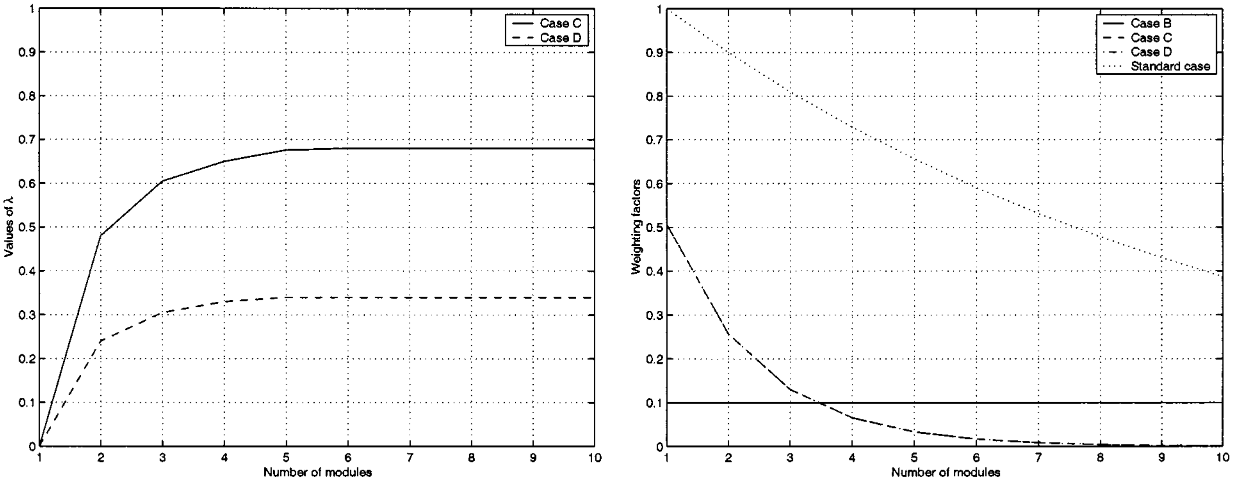
C. Cost Function with Exponentially Distributed Coefficients and the $\mathcal{L}_1$ Constraint

Although the algorithm given in (22) is a simple solution, it may suffer from the same problems as the cost function (9). Namely, it is natural that the uppermost module gives the biggest contribution toward learning, whereas the lowermost module plays a less significant role. In those cases, some alternative strategy needs to be found. A choice is to choose the coefficients in (17) as an exponentially decaying series. Having in mind the constraint (19), and also some recent results in gradient learning theory [51], where an exponential gradient is analyzed versus gradient descent for linear filters, let us choose the $\mathcal{L}_1$ constrained coefficient set as

$$\xi_i = e^{-\lambda * i}, \qquad i = 1, \cdots, M. \tag{23}$$

In this case, it is straightforward to find λ for any number of modules, by solving

$$\sum_{i=1}^{M} e^{-\lambda * i} = 1. \tag{24}$$



(a) λ versus M for $\mathcal{L}_1$ and $\mathcal{L}_2$ unit norm constraints

(b) Coefficients ξ_i for $M = 10$ modules

Fig. 2. Values of λ and weighting coefficients for various cases.

The vector $\boldsymbol{M}_1 = [m_1, m_2, \cdots, m_M]$, whose elements are the values of λ from (23) for the correspondent number of modules, is given by

$$\begin{aligned}\boldsymbol{M}_1 = &[0,\, 0.48,\, 0.605,\, 0.65,\, 0.676\\ &0.68,\, 0.68,\, \cdots,\, 0.68,\, \cdots] \end{aligned} \tag{25}$$

which preserves the constraint $\mathcal{L}_1(\boldsymbol{\Lambda}) = \sum_i |\xi_i| = 1$, and introduces forgetting in the spatial sense, i.e., along the PRNN.

D. Cost Function with Exponentially Distributed Coefficients and the $\mathcal{L}_2$ Constraint

From FIR filter theory, the power amplification at the output of an FIR filter can be expressed as [41], [43], [47]

$$P_{\text{out}} = \sum_{i=1}^{M} (\xi_i)^2 \tag{26}$$

which is actually the $\mathcal{L}_2$ norm of the coefficient vector $\boldsymbol{\Lambda}$. Let us therefore look for a set of coefficients $\boldsymbol{\Lambda} = \{\xi_i | P_{\text{out}} = 1\}$. Although there is continuum of such sets of coefficients, a choice could be a series of exponentially decreasing coefficients, such as

$$\sum_{i=1}^{M} (e^{-\lambda i})^2 = 1 \Leftrightarrow \xi_i = e^{-2\lambda * i}. \tag{27}$$

The problem of finding M coefficients ξ_i, $i = 1, \cdots, M$ becomes one of finding only one coefficient λ, which determines the series (27). Clearly, λ will be a function of the number of modules M. As the set of coefficients $\{\xi_i\}$ is monotone decreasing, a saturation in the values of the coefficients ξ, or equivalently λ, is expected. That is to emphasize a penalty introduced in the selection of the order of pipelining for large M. In the following equation, vector $\boldsymbol{M}_2$ comprises the values of parameter λ for any number of modules

$$\boldsymbol{M}_2 = [0,\, 0.24,\, 0.305,\, 0.33,\, 0.34,\, 0.34,\, \cdots,\, 0.34,\, \cdots]. \tag{28}$$

Thus, e.g., for $M = 3$ corresponding to $\lambda = \boldsymbol{M}(3) = 0.305$, which gives

$$\boldsymbol{\Lambda} = [0.7371,\, 0.5434,\, 0.4005], \qquad ||\boldsymbol{\Lambda}||_2 = 1. \tag{29}$$

Hence, there is again the forgetting effect in the cost function.

E. Training Considering Only the Output Module

If the computational complexity is a problem, one can opt to train the PRNN only upon the error obtained from its output module, i.e., the first module in the PRNN. This kind of training offers even more reduction in computational complexity, although it is not expected that the performance would be as good as with full training of the PRNN. Namely, for heavily correlated signals, it is expected that the statistical characteristics of a signal shall not change considerably over as few samples as $(p + M - 1)$, which is the total number of external input samples entering the PRNN. In this case, the coefficient vector $\boldsymbol{\Lambda}$ can be expressed as

$$\boldsymbol{\Lambda} = [1,\, 0,\, 0,\, 0,\, 0,\, \cdots,\, 0,\, \cdots] \tag{30}$$

which satisfies the constraint $||\boldsymbol{\Lambda}|| = 1$ in the $\mathcal{L}_1$, $\mathcal{L}_2$, and $\mathcal{L}_\infty$ sense.

F. Some Observations

In Section V-E, we have analyzed various forms of the cost function, with the constraint of a unit norm on the weighting factors within the cost function. The sequence of weighting factors, which are actually coefficients of an MA filter, is a function of the number modules within the PRNN. Due to the choice of λ from Section V-D, a saturation in λ occurs, which causes the weighting coefficients ξ to be small for a large number of modules. To further depict this situation, Fig. 2(a) shows the values of λ versus the number of modules for the cases in Sections V-C and V-D, whereas Fig. 2(b) shows the weighting coefficients $\xi_1, \cdots, \xi_{10}$, for the case of the PRNN with $M = 10$

modules. The cases B, C, and D in the legends of Fig. 2 refer to the cases from Sections V-B–V-D, respectively. As desired, for both the cases $||\mathbf{\Lambda}||_2$ and$||\mathbf{\Lambda}||_2 = 1$, the weighting coefficients die out for a relatively large number of modules M. Moreover, for $M > 5$, the values of λ converge toward their stationary values [Fig. 2(a)]. This complies with the Rissanen principle of parsimony, i.e., it penalizes the cost function of the PRNN for a relatively large number of modules. The values of weighting coefficients from Sections V-C and V-D also converge much faster than for the standard cost function (9) [dotted line in Fig. 2(b)].

Notice also that the graphs for the cases in Sections V-C and V-D show little difference, since the optimization for both the results is similar. It can also be seen if we compare the vectors $\boldsymbol{M}_1$ with the vector $2\boldsymbol{M}_2$ [this $2\boldsymbol{M}_2$ term is because of the square term in optimization of the cost function (26)]. The resulting difference is

$$\boldsymbol{M}_2 - \boldsymbol{M}_1 = [0,\, 0,\, 0.05,\, 0.05,\, 0.1,\, 0.04,\, 0,\, 0,\, \cdots].$$

VI. Experimental Results

The simulations were undertaken on speech, which is a typical example of nonstationary signals. The speech signals considered were denoted by $s1$ and $s2$, whose contents were, respectively, *"Oak is strong and* $\cdots$*"* and *"When recording audio data* $\ldots$*"*, with length 10 000 samples, sampled at 8 kHz. The measure that was used to assess the performance of the predictors was the forward prediction gain R_p given by

$$R_p \triangleq 10\log_{10}\left(\frac{\hat{\sigma}_s^2}{\hat{\sigma}_e^2}\right) \text{ dB} \tag{31}$$

where $\hat{\sigma}_s^2$ denotes the estimated variance of the speech signal $\{s(n)\}$, whereas $\hat{\sigma}_e^2$ denotes the estimated variance of the forward prediction error signal $\{e(n)\}$. The choice $p = 4$, where p is the number of external input signals per module of the PRNN, was taken from the theory of linear adaptive predictors, where for $p > 4$, the prediction gain was a very slowly increasing function [43]. The results of the simulations are shown in Table I. The parameters of the PRNN were chosen as in [26] and [35], in order to compare the performances achieved with modified cost functions to the performances achieved by other authors. The number of modules was $M = 5$, the number of neurons per module $N = 2$, the number of external input signals per module was $p = 4$, and initialization was undertaken with 300 epochs with 200 samples per epoch. The results achieved were also compared to the results achieved with classical least mean square (LMS) and RLS linear adaptive filters whose length were ten. With the modified algorithms, the best results were achieved with $p = 1$, which additionally decreases the computational complexity. Namely, as an RNN remembers the previous history presented to it, with $p > 1$, there is an overlapping between the input signals to consecutive modules. Indeed, for a PRNN with $p = 4$ and $M = 3$, all the modules would have two common input signals. This is in collision with the Rissanen principle of parsimony, and results in sample complexity for learning RNN mappings [33]. For an analysis of prediction gain versus p for the PRNN, refer to [52].

TABLE I
Comparison Between Prediction Gains Rp for Different Schemes

speech signal	$s1$	$s2$
$Rp[dB]$ LMS only	9.24	8.06
$Rp[dB]$ RLS only	12.70	11.55
$Rp[dB]$ SG+LMS	10.25	9.49
$Rp[dB]$ ERLS + RLS	14.77	13.40
Modified PRNN		
$Rp[dB]$ for $p = 4, M = 5, \lambda > 1$	10.04	13.54
$Rp[dB]$ for teaching only the uppermost module	11.1	12.94
$Rp[dB]$, for $\mathcal{L}_2$, exponential $\mathbf{\Lambda}$, $p = 4$, $M = 5$	12.80	13.16
Modified PRNN with $p = 1$		
$Rp[dB]$ for $\lambda > 1, p = 1, M = 5$	11.74	17.4
$Rp[dB]$ average $\Delta\mathbf{W}$ (22), $p = 1, M = 5$	16.10	17.64
$Rp[dB]$ for $\lambda > 1, p = 1, M = 5$ Every module thought separately	13.66	15.76
$Rp[dB]$ for $\mathcal{L}_1$ exponential $\mathbf{\Lambda}$, $p = 1$, $M = 5$	16.45	17.6

The SG + LMS algorithm (the third row in Table I) represents the result achieved in [26], where the output of the PRNN was fed into a linear LMS filter to improve the prediction gain. The results achieved by our algorithms were taken from only the output of the PRNN. The results achieved by the ERLS + RLS algorithm [35], used the extended recursive least squares (ERLS) algorithm for training the PRNN and the RLS linear filter afterwards. The modified PRNN with $\lambda > 1$ [28] achieved better prediction gain than the LMS, RLS, SG + LMS, and ERLS + RLS algorithms. Afterwards, two more experiments with $p = 4$ were undertaken. With teaching only the uppermost module in the PRNN, somewhat worse results were obtained, as expected, but having in mind that the LMS and RLS filters contribute to the total prediction gain of the PRNN with approximately 2 dB [35], the results for teaching only the last module still outperform even the ERLS + RLS algorithm. The $\mathcal{L}_2$ exponentially decreasing cost function was then applied to the PRNN, and the results, which were close to the best achieved were obtained. Furthermore, $p = 1$ appears to be the optimal choice of the number of external input signal to the PRNN, under new cost functions [53]. For that case both the $\mathcal{L}_1$ exponentially decreasing and the average weights cost function achieved slightly better results then the results in [28], with $\lambda > 1, p = 1, M = 5$. As expected, teaching every module separately performs worse than $\mathcal{L}_1$, $\mathcal{L}_2$, and weight average schemes.

VII. Conclusions

Choice of the parameters of the cost function for the modular, nested RNN architecture called the PRNN was addressed. The selection of the cost function was shown to be crucial for modular architectures, since it has a direct influence on teaching, and hence on the performance of a modular PRNN. A unit value constraint on the norm of the coefficients in the cost function was introduced, and variants of the cost function for the $\mathcal{L}_1$, $\mathcal{L}_2$, and $\mathcal{L}_\infty$ norms were derived, as well as some practically desirable simplified cost function based upon the same criteria. Simulations on speech signals support the choice of newly introduced cost functions for nonlinear and nonstationary signal prediction.

REFERENCES

[1] L. K. Li, "Approximation theory and recurrent networks," in *Proc. Int. Joint Conf. Neural Networks*, vol. II, 1992, pp. 266–271.

[2] J.-P. S. Draye, D. A. Pavisic, G. A. Cheron, and G. A. Libert, "Dynamic recurrent neural networks: A dynamical analysis," *IEEE Trans. Syst., Man, Cybern. B*, vol. 26, pp. 692–706, 1996.

[3] J. Kilian and H. T. Siegelmann, "The dynamic universality of sigmoidal neural networks," *Inform. Comput.*, vol. 128, no. 1, pp. 48–56, 1996.

[4] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Networks*, vol. 5, pp. 157–166, 1994.

[5] T. Lin, B. G. Horne, and C. L. Giles, "How embedded memory in recurrent neural-network architectures helps learning long-term temporal dependencies," *Neural Networks*, vol. 11, pp. 861–868, 1998.

[6] S.-S. Kim, "Time-delay recurrent neural network for temporal correlations and prediction," *Neurocomput.*, vol. 20, pp. 253–263, 1998.

[7] T. Sauer, J. A. Yorke, and M. Casdagly, "Embedology," *J. Statist. Phys.*, vol. 65, pp. 579–616, 1991.

[8] A. D. Back and A. C. Tsoi, "FIR and IIR synapses, a new neural-network architecture for time series modeling," *Neural Comput.*, vol. 3, pp. 375–385, 1991.

[9] E. A. Wan, "Time series prediction by using a connectionist network with internal delay lines," in *Time Series Prediction: Forecasting the Future and Understanding the Past*, A. S. Weigend and N. A. Gershenfeld, Eds. Reading, MA: Addison-Wesley, 1993.

[10] T. Sauer, "Time series prediction by using delay coordinate embedding," in *Time Series Prediction: Forecasting the Future and Understanding the Past*, A. S. Weigend and N. A. Gershenfeld, Eds. Reading, MA: Addison-Wesley, 1993.

[11] J. C. Principe, B. deVries, and P. G. deOliveira, "The gamma filter—A new class of adaptive IIR filters with restricted feedback," *IEEE Trans. Signal Processing*, vol. 41, pp. 649–656, 1993.

[12] B. deVries and J. C. Principe, "The gamma model—A new neural model for temporal processing," *Neural Networks*, vol. 5, pp. 565–576, 1992.

[13] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, pp. 1735–1780, 1997.

[14] M. C. Mozer, "Neural net architectures for temporal sequence processing," in *Time Series Prediction: Forecasting the Future and Understanding the Past*, A. S. Weigend and N. A. Gershenfeld, Eds. Reading, MA: Addison-Wesley, 1993.

[15] A. D. Back and A. C. Tsoi, "A low-sensitivity recurrent neural network," *Neural Comput.*, vol. 10, pp. 165–188, 1998.

[16] P. Baldi and A. F. Atiya, "How delays affect neural dynamics and learning," *IEEE Trans. Neural Networks*, vol. 5, pp. 612–621, 1994.

[17] D. P. Mandic and J. A. Chambers, "Relationship between the slope of the activation function and the learning rate for the RNN," *Neural Comput.*, vol. 11, no. 5, pp. 1069–1077, 1999.

[18] ——, "Exploiting inherent relationships in RNN architectures," *Neural Networks*, vol. 12, no. 10, pp. 1341–1345, 1999.

[19] R. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Comput.*, vol. 1, pp. 270–280, 1989.

[20] A. C. Tsoi and A. D. Back, "Locally recurrent globally feedforward networks: A critical review of architectures," *IEEE Trans. Neural Networks*, vol. 5, pp. 229–239, 1994.

[21] ——, "Discrete-time neural-network architectures: A unifying review," *Neurocomput.*, vol. 15, no. 3, pp. 183–223, 1997.

[22] P. Frasconi, M. Gori, and A. Sperduti, "A general framework for adaptive processing of data structures," *IEEE Trans. Neural Networks*, vol. 9, pp. 768–786, 1998.

[23] T. Lin, B. G. Horne, P. Tino, and C. L. Giles, "Learning long-term dependencies in NARX recurrent neural networks," *IEEE Trans. Neural Networks*, pp. 1329–1338, 1996.

[24] T.-N. Lin, C. L. Giles, B. G. Horne, and S.-Y. Kung, "A delay damage model selection algorithm for NARX neural networks," *IEEE Trans. Signal Processing*, vol. 45, pp. 2719–2730, 1997.

[25] H. T. Siegelmann, B. G. Horne, and C. L. Giles, "Computational capabilities of recurrent NARX neural networks," *IEEE Trans. Syst., Man, Cybern. B*, vol. 27, pp. 208–215, 1997.

[26] S. Haykin and L. Li, "Nonlinear adaptive prediction of nonstationary signals," *IEEE Trans. Signal Processing*, vol. 43, pp. 526–535, 1995.

[27] P.-R. Chang and J.-T. Hu, "Optimal nonlinear adaptive prediction and modeling of MPEG video in ATM networks using pipelined recurrent neural networks," *IEEE J. Select. Areas Commun.*, vol. 15, no. 6, pp. 1087–1100, 1997.

[28] D. P. Mandic and J. A. Chambers, "Advanced PRNN based nonlinear prediction/system identification," *Dig. IEE Colloquium Nonlinear Signal Image Processing*, pp. 11/1–11/6, 1998.

[29] J. Baltersee and J. A. Chambers, "Nonlinear adaptive prediction of speech signals using a pipelined recurrent neural network," *IEEE Trans. Signal Processing*, vol. 46, pp. 2207–2216, 1998.

[30] D. P. Mandic and J. A. Chambers, "From an *a priori* RNN to an *a posteriori* PRNN nonlinear predictor," in *Proc. VIII IEEE Wkshp. Neural Networks Signal Processing (NNSP98)*, 1998, pp. 174–183.

[31] E. D. Sontag, "Critical points for neural net least-squares problems," in *Proc. Int. Conf. Neural Networks*, vol. 6, 1995, pp. 2949–2954.

[32] B. DasGupta, H. T. Siegelmann, and E. Sontag, "On the complexity of training neural networks with continuous activation functions," *IEEE Trans. Neural Networks*, vol. 6, pp. 1490–1504, 1995.

[33] B. DasGupta and E. D. Sontag, "Sample complexity for learning recurrent perceptron mappings," *IEEE Trans. Inform. Theory*, vol. 42, pp. 1479–1487, 1996.

[34] S. Haykin, *Neural Networks—A Comprehensive Foundation*. Englewood Cliffs, NJ: Prentice-Hall, 1994.

[35] D. P. Mandic, J. Baltersee, and J. A. Chambers, "Nonlinear prediction of speech with a pipelined recurrent neural network and advanced learning algorithms," in *Signal Analysis and Prediction*, A. Prochazka, J. Uhlir, P. J. W. Rayner, and N. G. Kingsbury, Eds. Boston, MA: Birkhauser, 1998, pp. 291–309.

[36] E. Zeidler, "Nonlinear functional analysis and its applications, vol. 1," in *Fixed-Point Theorems*. New York: Springer-Verlag, 1986.

[37] J. E. Dennis, Jr. and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, ser. Computational Mathematics. Englewood Cliffs, NJ: Prentice-Hall, 1983.

[38] J. M. Zurada and W. Shen, "Sufficient condition for convergence of a relaxation algorithm in actual single-layer neural networks," *IEEE Trans. Neural Networks*, vol. 1, pp. 300–303, 1990.

[39] J. J. Shynk and S. Roy, "Convergence properties and stationary points of a perceptron learning algorithm," *Proc. IEEE*, vol. 78, pp. 1599–1604, 1990.

[40] S. Lipschutz, *General Topology*, ser. Schaum's Outline Series: McGraw-Hill, 1981.

[41] J. R. Treichler, C. R. Johnson, Jr., and M. G. Larimore, *Theory and Design of Adaptive Filters*. New York: Wiley, 1987.

[42] D. P. Mandic and J. A. Chambers, "*A posteriori* real-time recurrent learning schemes for a recurrent neural-network-based nonlinear predictor," *Proc. Inst. Elect. Eng. Vision, Image, Signal Processing*, vol. 145, no. 6, pp. 365–370, 1998.

[43] S. Haykin, *Adaptive Filter Theory*, 3rd ed. Englewood Cliffs, NJ: Prentice-Hall, 1996.

[44] L. Ljung and T. Soderstrom, *Theory and Practice of Recursive Identification*. Cambridge, MA: MIT Press, 1983.

[45] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neural Networks*, vol. 1, pp. 4–27, 1990.

[46] ——, "Gradient methods for the optimization of dynamical systems containing neural networks," *IEEE Trans. Neural Networks*, vol. 2, pp. 252–262, 1991.

[47] R. W. Hamming, *Digital Filters*, 3rd ed. Englewood Cliffs, NJ: Prentice-Hall, 1989.

[48] D. G. Luenberger, *Optimization by Vector Space Methods*. New York: Wiley, 1969.

[49] A. Assoum, N. E. Radi, R. Velazco, F. Elie, and R. Ecoeffet, "Robustness against S.E.U. of an artificial neural-network space application," *IEEE Trans. Nucl. Sci.*, vol. 43, no. 3, pp. 973–978, 1996.

[50] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York: Wiley, 1991.

[51] J. Kivinen, "Exponentiated gradient versus gradient descent for linear predictors," *Inform. Comput.*, vol. 132, pp. 1–63, 1997.

[52] D. P. Mandic and J. A. Chambers, "Toward an optimal PRNN-based nolinear predictor," *IEEE Trans. Neural Networks*, vol. 10, pp. 1435–1442, 1999.

[53] J. Rissanen, *Stochastic Complexity in Statistical Inquiry*, Singapore: World, 1989.

Danilo P. Mandic (M'99) received the B.Sc. (Hons.) degree in electrical and electronic engineering, and the M.Sc. degree in signal processing, from the University of Banja Luka, Bosnia-Herzegovina. He received the Ph.D. degree in nonlinear adaptive signal processing from the Imperial College, London, U.K.

He is currently a Lecturer in computer science in the School of Information Systems, University of East Anglia, Norwich, U.K. His areas of interest include linear and nonlinear adaptive signal processing, neural networks, biomedical signal and image processing, system identification, stability theory, and computer vision.

Dr. Mandic has received awards for his collaboration with industry, and was also awarded the Nikola Tesla Medal for his innovative work.

Jonathon A. Chambers (M'93–SM'99) was born in Peterborough, U.K., in 1960. After an electronics artificer apprenticeship in the Royal Navy, he received the first class B.Sc. (Hons.) degree in electrical and electronic engineering from the Polytechnic of Central London, U.K., receiving the Robert Mitchell Medal as the top graduate in 1985. He received the Ph.D. degree in adaptive signal processing in 1990 after studying at Imperial College, London, U.K., and at Cambridge University, Cambridge, U.K.

Dr. Chambers spent three years as a Research Scientist at Schlumberger Cambridge Research, applying adaptive signal processing techniques to oilfield-related applications. He returned to a lectureship in signal processing in the Department of Electrical and Electronic Engineering, Imperial College, in 1994 and was promoted to a readership in signal processing in 1998. He has authored and coauthored many technical publications on adaptive signal processing and its applications in mobile communication systems.

Dr. Chambers is a member of the IEE Professional Group Committee E5 on Signal Processing, a Guest Editor for the International Journal of Adaptive Control and Signal Processing, and has served as an Associate Editor for IEEE TRANSACTIONS ON SIGNAL PROCESSING.