Neurocomputing 116 (2013) 94–101

Contents lists available at SciVerse ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Online sequential extreme learning machine in nonstationary environments

Yibin Ye, Stefano Squartini*, Francesco Piazza

A3LAB, Department of Information Engineering, Università Politecnica delle Marche, Via Brecce Bianche 1, 60131 Ancona, Italy

ARTICLE INFO

Available online 9 October 2012

Keywords:
Nonstationary and nonlinear system identification
Time-varying neural networks
Extreme learning machine
Online sequential learning

ABSTRACT

System identification in nonstationary environments represents a challenging problem to solve and lots of efforts have been put by the scientific community in the last decades to provide adequate solutions on purpose. Most of them are targeted to work under the system linearity assumption, but also some have been proposed to deal with the nonlinear case study. In particular the authors have recently advanced a neural architecture, namely time-varying neural networks (TV-NN), which has shown remarkable identification properties in the presence of nonlinear and nonstationary conditions. TV-NN training is an issue due to the high number of free parameters and the extreme learning machine (ELM) approach has been successfully used on purpose. ELM is a fast learning algorithm that has recently caught much attention within the neural networks (NNs) research community. Many variants of ELM have been appeared in recent literature, specially for the stationary case study. The reference one for TV-NN training is named ELM-TV and is of batch-learning type. In this contribution an online sequential version of ELM-TV is developed, in response to the need of dealing with applications where sequential arrival or large number of training data occurs. This algorithm generalizes the corresponding counterpart working under stationary conditions. Its performances have been evaluated in some nonstationary and nonlinear system identification tasks and related results show that the advanced technique produces comparable generalization performances to ELM-TV, ensuring at the same time all benefits of an online sequential approach.

1. Introduction

Extreme learning machine (ELM) is a fast learning algorithm designed for single hidden layer feedforward neural networks (SLFNs) [1], first introduced by Huang et al. In ELM, the input weights of SLFNs do not need to be tuned and can be randomly generated, whereas the output weights are analytically determined using the least-square method, thus allowing a significant training time reduction. In recent years, ELM has been raising much attention and interest among scientific community, in both theoretic study and applications. A hybrid algorithm called evolutionary extreme learning machine (E-ELM) [2] uses the differential evolutionary algorithm to select the input weights. Incremental-based extreme learning machine algorithms (I-ELM [3], EI-ELM [4] and EM-ELM [5]) can randomly add nodes to the hidden layer. A real-coded genetic algorithm approach called RCGA-ELM has been also proposed to select the optimal number of hidden nodes, input weights and bias values for better performance in multi-category sparse data classification problems [6]. B-ELM is able to optimize the weights of the output layer based on a Bayesian linear regression [7]. The optimally pruned extreme learning machine (OP-ELM) extends the original ELM algorithm and wraps it within a methodology using a pruning of the neurons [8]. In CFWNN-ELM, composite functions are applied at the hidden nodes and learning is accomplished using ELM in a wavelet neural network [9]. Encoding a priori information to obtain better generalization performance for function approximation is another possible way to improve ELM [10]. Radial-basis type neural networks [11,12] also represent a fertile field where ELM paradigm has found application, as confirmed by some recent contributions [13,1]. Among the various extensions, the online sequential learning algorithm based on ELM (OS-ELM) need to be cited, according to which learning is performed by using data one-by-one or chunk-by-chunk with fixed or varying chunk size [14]. The parameters of hidden nodes are randomly selected just as ELM, but the output weights are analytically updated based on the sequentially arriving data rather than the whole data set. Besides theoretical studies, some applications such as system identification and real-time data assessment have been considered to evaluate the effectiveness of ELM [15,16].

Time-varying neural network (TV-NN) represents a relevant example in neural architectures working properly in nonstationary environments. Such networks implement time-varying weights, each

* Corresponding author. Tel.: +39 3288413154.
E-mail addresses: yibin.ye@univpm.it (Y. Ye), s.squartini@univpm.it (S. Squartini), f.piazza@univpm.it (F. Piazza).
URL: http://www.a3lab.dibet.univpm.it (S. Squartini).

being a linear combination of a certain set of basis functions, whose independent variable is time. The candidate orthogonal basis function types employed in time-varying networks include Legendre polynomials, Chebyshev polynomials, Fourier sinusoidal functions, prolate spheroidal functions [17], and more. An extended Back-Propagation algorithm has been first advanced to train these networks (BP-TV) [18]. Later on, an extreme learning machine approach is also developed for TV-NN, accelerating the training procedure significantly (ELM-TV) [19]. Recently, several variants and applications have been proposed based on ELM-TV. A group selection evolutionary approach applies differential evolutionary with group selection to determine the type of basis function and its input weight parameters [20], while two other algorithms referred as EM-ELM-TV [21] and EM-OB [22] are able to automatically determine the number of hidden neurons or output basis functions, respectively. A Volterra system identification application is also studied in [23].

All the algorithms for TV-NN mentioned above belong to batch-learning type. BP-TV is a time consuming algorithm as it involve many epochs (or iterations). Meanwhile, learning parameters such as learning rate, number of learning epochs have to be properly chosen to ensure convergence. As for ELM-based algorithms, whenever a new data arrived, the past data along with the new one have to be used for retraining, thus consuming lots of time or/and memory. With the advantage of no requirement for retraining whenever receiving new data, online learning algorithms are preferred over batch ones in some engineering applications.

Based on [14], in this paper, an online extreme learning machine is extended to deal with time-varying neural networks and referred as OS-ELM-TV, which can learn the training data one-by-one or chunk-by-chunk. The data which have already been used in the training can be discarded, so as to save more memory and computational load to process the new coming data.

The paper is organized as follows. The ELM and ELM-TV algorithms are briefly introduced in Sections 2 and 3, followed by our proposed on-line ELM-TV presented in Section 4 and computational complexity analysis in Section 5. Computer simulations and related results are demonstrated and discussed in Section 6. Finally conclusions are drawn in Section 7.

2. Extreme learning machine

Let us assume that an SLFN with I input neurons, K hidden neurons, L output neurons and activation function $g(\cdot)$ is trained to learn N distinct samples $(\mathbf{X},\mathbf{T})$, where $\mathbf{X}=\{x_i[n]\}\in\mathbb{R}^{N\times I}$ and $\mathbf{T}=\{t_l[n]\}\in\mathbb{R}^{N\times L}$ are the input matrix and target matrix respectively, $x_i[n]$ denotes the input data in i-th input neuron at n-th time instant, and $t_l[n]$ denotes the desired output in l-th output neuron at n-th time instant. In ELM, the input weights $\{w_{ik}\}$ and hidden biases $\{b_k\}$ are randomly generated, where w_{ik} is the weight connecting i input neuron to k-th hidden neuron, and b_k is the bias of k-th hidden neuron. Further let $w_{0k}=b_k$ and $x_0[n]=1$. Hence, the hidden-layer output matrix $\mathbf{H}=\{h_k[n]\}\in\mathbb{R}^{N\times K}$ can be obtained by

$$h_k[n]=g\left(\sum_{i=0}^{I}x_i[n]\cdot w_{ik}\right) \tag{1}$$

Let $\boldsymbol{\beta}=\{\beta_{kl}\}\in\mathbb{R}^{K\times L}$ be the matrix of output weights, where β_{kl} denotes the weight connection between k-th hidden neuron and l-th output neuron; and $\mathbf{Y}=\{y_l[n]\}\in\mathbb{R}^{N\times L}$ be the matrix of network output data, with $y_l[n]$ the output data in l-th output neuron at n-th time instant. Therefore, this equation can be obtained for the linear output neurons:

$$y_l[n]=\sum_{k=1}^{K}h_k[n]\cdot\beta_{kl} \tag{2}$$

or

$$\mathbf{Y}=\mathbf{H}\cdot\boldsymbol{\beta} \tag{3}$$

Thus, given the hidden-layer output matrix $\mathbf{H}$ and the targets matrix $\mathbf{T}$, to minimize $\|\mathbf{Y}-\mathbf{T}\|_2$, the output weights can be calculated by the minimum norm least-square (LS) solution of the linear system:

$$\hat{\boldsymbol{\beta}}=\mathbf{H}^{\dagger}\cdot\mathbf{T} \tag{4}$$

where $\mathbf{H}^{\dagger}$ is the Moore–Penrose generalized inverse of matrix $\mathbf{H}$. By computing output weights analytically, ELM achieve good generalization performance with speedy training phase.

3. ELM for time-varying neural networks

The time-varying version of extreme learning machine has been studied in [19]. In a time-varying neural network as shown in Fig. 1, the input weights, or output weights, or both are changing with time (both in training and testing phases). The generic weight $w[n]$ can be expressed as a linear combination of a certain set of basis functions [24,18]:

$$w[n]=\sum_{b=1}^{B}f_b[n]w_b \tag{5}$$

in which $f_b[n]$ is the known orthogonal function at n-th time instant of b-th order, w_b is the b-th order coefficient of the basis function to construct time-varying weight w_n, while B (preset by user) is the total number of the bases.

If time-varying input weights are introduced in a SLFN, the hidden neuron output function can be written as

$$h_k[n]=g\left(\sum_{i=0}^{I}x_i[n]\cdot w_{ik}[n]\right) \tag{6}$$

where $w_{ik}[n]=\sum_{b=1}^{B}f_b[n]w_{b,ik}$. Similarly, if time-varying output weights are introduced, the standard output equation can be written as

$$y_l[n]=\sum_{k=1}^{K}h_k[n]\beta_{kl}[n] \tag{7}$$

where $\beta_{kl}[n]=\sum_{b=1}^{B}f_b[n]\beta_{b,kl}$.

To train a TV-NN, input weights $\{w_{b,ik}\}$ are randomly generated and hidden-layer output matrix $\mathbf{H}$ is computed according to Eq. (6). However, the values of a set of output weight parameters $\{\beta_{b,kl}\}$ cannot be so straightforward calculated. Some transformations are needed. Expanding the time-varying output weights $\beta_{kl}[n]$ in (7) and assuming:

$$\mathbf{f}[n]=[f_1[n],f_2[n],\ldots,f_B[n]]^T\in\mathbb{R}^B$$

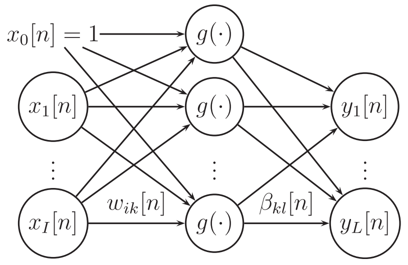


Fig. 1. Architecture of time-varying neural networks.

$$\mathbf{h}[n] = [h_1[n], h_2[n], \ldots, h_K[n]]^T \in \mathbb{R}^K$$
$$\boldsymbol{\beta}_{kl} = [\beta_{1,kl}, \beta_{2,kl}, \ldots, \beta_{B,kl}]^T \in \mathbb{R}^B$$
$$\boldsymbol{\beta}_{(l)} = [\boldsymbol{\beta}_{1l}, \boldsymbol{\beta}_{2l}, \ldots, \boldsymbol{\beta}_{Kl}] \in \mathbb{R}^{B\times K}$$
$$\boldsymbol{\omega}_{(l)} = [\boldsymbol{\beta}_{1l}^T, \boldsymbol{\beta}_{2l}^T, \ldots, \boldsymbol{\beta}_{Kl}^T] \in \mathbb{R}^{B\cdot K\times 1}$$

the following hold:

$$\begin{aligned} y_l[n] &= \sum_{k=1}^{K} h_k[n]\cdot\left(\sum_{b=1}^{B} f_b[n]\cdot\beta_{b,kl}\right) \\ &= \sum_{k=1}^{K} \mathbf{f}[n]^T\cdot\boldsymbol{\beta}_{kl}\cdot h_k[n] \\ &= \mathbf{f}[n]^T\cdot\boldsymbol{\beta}_{(l)}\cdot\mathbf{h}[n] \\ &= (\mathbf{h}[n]^T\otimes\mathbf{f}[n]^T)\cdot\boldsymbol{\omega}_{(l)} \end{aligned} \tag{8}$$

where $\otimes$ denotes the *Kronecker product* of $\mathbf{h}[n]^T$ and $\mathbf{f}[n]^T$. The last step consists in: $vec(\mathbf{AXB}) = (\mathbf{B}^T\otimes\mathbf{A})vec(\mathbf{X})$ [25] (note that $vec(y_l[n]) = y_l[n]$) and $\boldsymbol{\omega}_{(l)} = vec(\boldsymbol{\beta}_{(l)})$ is the vectorization of the matrix $\boldsymbol{\beta}_{(l)}$ formed by stacking the columns of $\boldsymbol{\beta}_{(l)}$ into a single column vector. Moreover, let us define:

$$\mathbf{G} = \mathbf{H}*\mathbf{F} = \begin{bmatrix} \mathbf{h}[1]^T\otimes\mathbf{f}[1]^T \\ \vdots \\ \mathbf{h}[N]^T\otimes\mathbf{f}[N]^T \end{bmatrix}_{N\times B\cdot K} \tag{9}$$

where $\mathbf{H} = [\mathbf{h}[1], \mathbf{h}[2], \ldots, \mathbf{h}[N]]^T \in \mathbb{R}^{N\times K}$, $\mathbf{F} = [\mathbf{f}[1], \mathbf{f}[2], \ldots, \mathbf{f}[N]]^T \in \mathbb{R}^{N\times B}$, $*$ denotes the *Khatri–Rao product* of matrices $\mathbf{H}$ and $\mathbf{F}$, with $\mathbf{h}[n]^T$ and $\mathbf{f}[n]^T$ as their submatrices, respectively. Further assuming that $\mathbf{Y} = \{y_l[n]\} \in \mathbb{R}^{N\times L}$, $\mathbf{T} = \{t_l[n]\} \in \mathbb{R}^{N\times L}$, $\boldsymbol{\Omega} = [\boldsymbol{\omega}_{(1)}, \boldsymbol{\omega}_{(2)}, \ldots, \boldsymbol{\omega}_{(L)}] \in \mathbb{R}^{B\cdot K\times L}$, the following holds:

$$\mathbf{G}\cdot\boldsymbol{\Omega} = \mathbf{Y} \tag{10}$$

Since $\mathbf{F}$ is obtained by the type of the basis function predetermined by the user and $\mathbf{H}$ can be calculated by (6) once input weight parameters are randomly generated, hence $\mathbf{G}$ can be computed. Similarly to the ELM algorithm described in previous section, the time-variant output weight matrix $\boldsymbol{\Omega}$ can be computed by

$$\hat{\boldsymbol{\Omega}} = \mathbf{G}^\dagger\cdot\mathbf{T} \tag{11}$$

where $\mathbf{G}^\dagger$ is the MP inverse of matrix $\mathbf{G}$, and consequently, $\hat{\boldsymbol{\Omega}}$ is a set of optimal output weight parameters minimizing the training error.

4. The proposed algorithm: online sequential ELM-TV

All the training data (N samples) have to be available before using the batch algorithm ELM-TV stated in the previous section. However, in some applications, the neural networks may receive the training data sequentially, or large amount of data to process in limited memory at the same time. Therefore, it is therefore convenient to develop an online algorithm for TV-NN.

It is here assumed that the number of samples N is large and the rank of time-varying hidden matrix $R(G) = K\cdot B$, i.e. the number of hidden nodes by the number of output bases; $\mathbf{G}^\dagger$ in (11) can be expressed as

$$\mathbf{G}^\dagger = (\mathbf{G}^T\mathbf{G})^{-1}\mathbf{G}^T \tag{12}$$

Given the initial training set $\{(\mathbf{x}[n], \mathbf{t}[n])\}_{n=1}^{N_0}$ it is assumed that the number of samples is larger than the number of hidden nodes by the number of output bases, e.g. $N_0 > K\cdot B$. Using batch ELM-TV, according to (11) the optimal output weight matrix would be

$$\boldsymbol{\Omega}^{(0)} = \mathbf{G}_0^\dagger\cdot\mathbf{T}_0 = (\mathbf{G}_0^T\mathbf{G}_0)^{-1}\mathbf{G}_0^T\mathbf{T}_0 = \mathbf{C}_0^{-1}\mathbf{A}_0 \tag{13}$$

where $\mathbf{C}_0 = \mathbf{G}_0^T\mathbf{G}_0$; $\mathbf{A}_0 = \mathbf{G}_0^T\mathbf{T}_0$;

$$\mathbf{G}_0 = \begin{bmatrix} \mathbf{h}[1]^T\otimes\mathbf{f}[1]^T \\ \vdots \\ \mathbf{h}[N_0]^T\otimes\mathbf{f}[N_0]^T \end{bmatrix}_{N_0\times K\cdot B} \quad\text{and}\quad \mathbf{T}_0 = \begin{bmatrix} \mathbf{t}[1]^T \\ \vdots \\ \mathbf{t}[N_0]^T \end{bmatrix}_{N_0\times L} \tag{14}$$

Let us suppose that another chunk of data $\{(\mathbf{x}[n], \mathbf{t}[n])\}_{n=N_0+1}^{N_0+N_1}$ arrives, the optimal output weight matrix has to be modified as

$$\boldsymbol{\Omega}^{(1)} = \mathbf{C}_1^{-1}\mathbf{A}_1 \tag{15}$$

where

$$\mathbf{C}_1 = \begin{bmatrix}\mathbf{G}_0\\ \mathbf{G}_1\end{bmatrix}^T\begin{bmatrix}\mathbf{G}_0\\ \mathbf{G}_1\end{bmatrix} = [\mathbf{G}_0^T\ \ \mathbf{G}_1^T]\begin{bmatrix}\mathbf{G}_0\\ \mathbf{G}_1\end{bmatrix} = \mathbf{C}_0 + \mathbf{G}_1^T\mathbf{G}_1 \tag{16}$$

$$\mathbf{A}_1 = \begin{bmatrix}\mathbf{G}_0\\ \mathbf{G}_1\end{bmatrix}^T\begin{bmatrix}\mathbf{T}_0\\ \mathbf{T}_1\end{bmatrix} = [\mathbf{G}_0^T\ \ \mathbf{G}_1^T]\begin{bmatrix}\mathbf{T}_0\\ \mathbf{T}_1\end{bmatrix} = \mathbf{A}_0 + \mathbf{G}_1^T\mathbf{T}_1 \tag{17}$$

$$\mathbf{G}_1 = \begin{bmatrix} \mathbf{h}[N_0+1]^T\otimes\mathbf{f}[N_0+1]^T \\ \vdots \\ \mathbf{h}[N_0+N_1]^T\otimes\mathbf{f}[N_0+N_1]^T \end{bmatrix}_{N_1\times K\cdot B},\quad \mathbf{T}_1 = \begin{bmatrix} \mathbf{t}[N_0+1]^T \\ \vdots \\ \mathbf{t}[N_0+N_1]^T \end{bmatrix}_{N_1\times L} \tag{18}$$

The aim here is to express $\boldsymbol{\Omega}^{(1)}$ as a function of $\boldsymbol{\Omega}^{(0)}$, $\mathbf{C}_1$, $\mathbf{G}_1$ and $\mathbf{T}_1$. According to (13) and (16), $\mathbf{A}_0$ can be written as

$$\mathbf{A}_0 = \mathbf{C}_0\mathbf{C}_0^{-1}\mathbf{A}_0 = (\mathbf{C}_1 - \mathbf{G}_1^T\mathbf{G}_1)\boldsymbol{\Omega}^{(0)} = \mathbf{C}_1\boldsymbol{\Omega}^{(0)} - \mathbf{G}_1^T\mathbf{G}_1\boldsymbol{\Omega}^{(0)} \tag{19}$$

Combining (15)–(17), and (19), the following can be obtained:

$$\boldsymbol{\Omega}^{(1)} = \mathbf{C}_1^{-1}(\mathbf{C}_1\boldsymbol{\Omega}^{(0)} - \mathbf{G}_1^T\mathbf{G}_1\boldsymbol{\Omega}^{(0)} + \mathbf{G}_1^T\mathbf{T}_1) = \boldsymbol{\Omega}^{(0)} + \mathbf{C}_1^{-1}\mathbf{G}_1^T(\mathbf{T}_1 - \mathbf{G}_1\boldsymbol{\Omega}^{(0)}) \tag{20}$$

Since $\mathbf{C}_1^{-1}$ is used for computing $\boldsymbol{\Omega}^{(1)}$, then by setting $\mathbf{P}_0 = \mathbf{C}_0^{-1}$ and $\mathbf{P}_1 = \mathbf{C}_1^{-1}$, and using the Woodbury formula [26] the following can be derived:

$$\mathbf{P}_1 = (\mathbf{C}_0 + \mathbf{G}_1^T\mathbf{G}_1)^{-1} = \mathbf{P}_0 - \mathbf{P}_0\mathbf{G}_1^T(\mathbf{I} + \mathbf{G}_1\mathbf{P}_0\mathbf{G}_1^T)^{-1}\mathbf{G}_1\mathbf{P}_0 \tag{21}$$

Generalizing the previous arguments, when m-th chunk of data set arrives:

$$\mathbf{P}_m = \mathbf{P}_{m-1} - \mathbf{P}_{m-1}\mathbf{G}_m^T(\mathbf{I} + \mathbf{G}_m\mathbf{P}_{m-1}\mathbf{G}_m^T)^{-1}\mathbf{G}_m\mathbf{P}_{m-1} \tag{22}$$

$$\boldsymbol{\Omega}^{(m)} = \boldsymbol{\Omega}^{(m-1)} + \mathbf{P}_m\mathbf{G}_m^T(\mathbf{T}_m - \mathbf{G}_m\boldsymbol{\Omega}^{(m-1)}) \tag{23}$$

When the training data is received one-by-one instead of chunk-by-chunk, e.g. $N_m = 1$, the above formula have the following simple format:

$$\mathbf{P}_m = \mathbf{P}_{m-1} - \frac{\mathbf{P}_{m-1}\mathbf{g}_m\mathbf{g}_m^T\mathbf{P}_{m-1}}{1 + \mathbf{g}_m^T\mathbf{P}_{m-1}\mathbf{g}_m} \tag{24}$$

$$\boldsymbol{\Omega}^{(m)} = \boldsymbol{\Omega}^{(m-1)} + \mathbf{P}_m\mathbf{g}_m(\mathbf{t}_m^T - \mathbf{g}_m^T\boldsymbol{\Omega}^{(m-1)}) \tag{25}$$

where $\mathbf{g}_m = (\mathbf{h}_m^T\otimes\mathbf{f}_m^T)^T$

Now, our proposed online algorithm, namely OS-ELM-TV can be summarized as follows. Assume that a single hidden layer time-varying neural network is to be trained, with K hidden nodes and B output basis functions, and receiving the training data set $\{(\mathbf{x}[n], \mathbf{t}[n])\}$ sequentially. Algorithm 1 is thus attained. Note that the time invariant OS-ELM algorithm is just a special case of the proposed OS-ELM-TV (when $B = 1$).

Algorithm 1. OS-ELM-TV.

```
1: Randomly generate the input weights set $\{\omega_{b,ik}\}$ and choose
a set of basis functions.
2: Accumulate $N_0$ samples of training data (make sure
$N_0 > K\cdot B$).
3: Calculate the time-varying hidden layer output matrix $\mathbf{G}_0$
by (14)
```

```
4: Calculate $\mathbf{P}_0=(\mathbf{G}_0^T\mathbf{G}_0)^{-1}$
5: Calculate the initial output weight matrix $\mathbf{\Omega}^{(0)}$ by (13)
6: for $m=1$ to $M$ do
7:When the $m$-th chunk of data arrives, calculate the time-
  varying partial hidden layer output matrix $\mathbf{G}_m$.
8:Update the output weight matrix $\mathbf{\Omega}^{(m)}$ by
  $\mathbf{P}_m=\mathbf{P}_{m-1}-\mathbf{P}_{m-1}\mathbf{G}_m^T(\mathbf{I}+\mathbf{G}_m\mathbf{P}_{m-1}\mathbf{G}_m^T)^{-1}\mathbf{G}_m\mathbf{P}_{m-1}$   (26)
  $\mathbf{\Omega}^{(m)}=\mathbf{\Omega}^{(m-1)}+\mathbf{P}_m\mathbf{G}_m^T(\mathbf{T}_m-\mathbf{G}_m\mathbf{\Omega}^{(m-1)})$   (27)
9:   $m\leftarrow m+1$
10: end for
```

In practice, in order to apply OS-ELM-TV, $N_0>1.2KB$ is usually chosen to make sure $R(G)=KB$ and $\mathbf{G}^T\mathbf{G}$ a full rank matrix so that $\mathbf{P}_0$ exists and $\mathbf{P}_m$ can be computed recursively. In some applications, the selected parameters K and B cannot be too large, otherwise there would occur that $R(G)<KB$ and the online algorithm might likely diverge.

Table 1
Performance comparisons of ELM-TV and OS-ELM-TV with different basis functions and number of input/output bases when updating the 1000th sample in time-varying MLP system, with 100 hidden nodes.

Basis function (#input, #output)	Algorithms	Training RMSE (dB)	Validation RMSE (dB)	Training time (s)
None(1,1)	ELM	−19.26	−18.86	0.0409
	OS-ELM	−19.25	−18.84	0.0046
Legendre(3,1)	ELM-TV	−19.45	−18.96	0.0449
	OS-ELM-TV	−19.13	−18.79	0.0057
Legendre(1,3)	ELM-TV	−26.98	−22.25	0.2350
	OS-ELM-TV	−26.98	−22.19	0.0092
Legendre(3,3)	ELM-TV	−32.24	−25.57	0.2898
	OS-ELM-TV	−33.68	−26.55	0.0071
Chebyshev(3,3)	ELM-TV	−32.77	−26.47	0.3729
	OS-ELM-TV	−33.28	−26.07	0.0095
Prolate(3,3)	ELM-TV	−33.90	−29.79	0.2774
	OS-ELM-TV	−32.61	−29.21	0.0077
Fourier(3,3)	ELM-TV	−20.32	−18.44	0.2883
	OS-ELM-TV	−20.50	−18.48	0.0085

5. Computational complexity analysis

For ELM-TV, when a new training data set arrives, the output weight set can only be updated by (12) and (11). Assuming that arithmetic with individual elements has complexity $O(1)$, the complexity of multiplication of one $m\times n$-matrix and one $n\times p$-matrix is $O(mnp)$. As $\mathbf{G}\in\mathbb{R}^{N\times KB}$, the computational complexities of two matrix multiplications and one matrix inversion in $\mathbf{G}^\dagger=(\mathbf{G}^T\mathbf{G})^{-1}\mathbf{G}^T$ are $O((KB)^2N)$ and $O((KB)^3)$ respectively.

For calculation of output weight matrix $\hat{\mathbf{\Omega}}=\mathbf{G}^\dagger\cdot\mathbf{T}$, with $\mathbf{G}^\dagger\in\mathbb{R}^{KB\times N}$ and $\mathbf{T}\in\mathbb{R}^{N\times L}$, the computational complexity of (11) is $O((KB)NL)$.

Since $KB<N$ and usually $L<KB$, it can be concluded that the computational complexity of updating $\hat{\mathbf{\Omega}}$ is $O((KB)^2N)$.

On the other hand, for online ELM-TV, when m-th chunk of data set arrives, (22) and (23) are used to update the output weight set. As $\mathbf{P}_{m-1}\in\mathbb{R}^{KB\times KB}$ and $\mathbf{G}_m\in\mathbb{R}^{N_m\times KB}$, the computational complexity required to calculate $\mathbf{P}_m$ in (22) is $O((KB)^2N_m)$.

Similarly, as $\Omega^{(m-1)}\in\mathbb{R}^{KB\times L}$ and the common case of $L<KB$, the computational complexity to obtain $\mathbf{\Omega}^{(m)}$ from (23) would also be $O((KB)^2N_m)$. Since usually $N_m\ll N$, the computational complexity (in terms of computational load and memory consumption) when applying the proposed OS-ELM-TV rather than the standard ELM-TV, would be reduced dramatically for updating output weight matrix. This is especially true for the case of large total samples ($N>10^3$) with one-by-one data set arrival ($N_m=1$).

As for memory requirements, intuitively one can expect that OS-ELM-TV uses less memory since previous samples can be discarded after the updating process. In fact, in ELM-TV, $\mathbf{G}\in\mathbb{R}^{N\times KB}$ and $\mathbf{T}\in\mathbb{R}^{N\times L}$ have to be always stored; moreover the memory required for computation of $\mathbf{G}^\dagger$ increases when N becomes larger. On the other hand, in the OS-ELM-TV algorithm, we have that only the variable $\mathbf{P}_m\in\mathbb{R}^{KB\times KB}$ rather than $\mathbf{G}$ and $\mathbf{T}$ needs to be stored and that calculations of smaller matrices w.r.t. the ELM-TV case study are involved in the weights updating process.

6. Computer simulations

In this section, OS-ELM-TV (in one-by-one learning mode) is compared with ELM-TV in three identification problems and one estimation task, which can all be categorized as time-varying system identification problems and represent a certain range of nonstationary environment. The polynomial activation function ($g(x)=x^2+x$) is

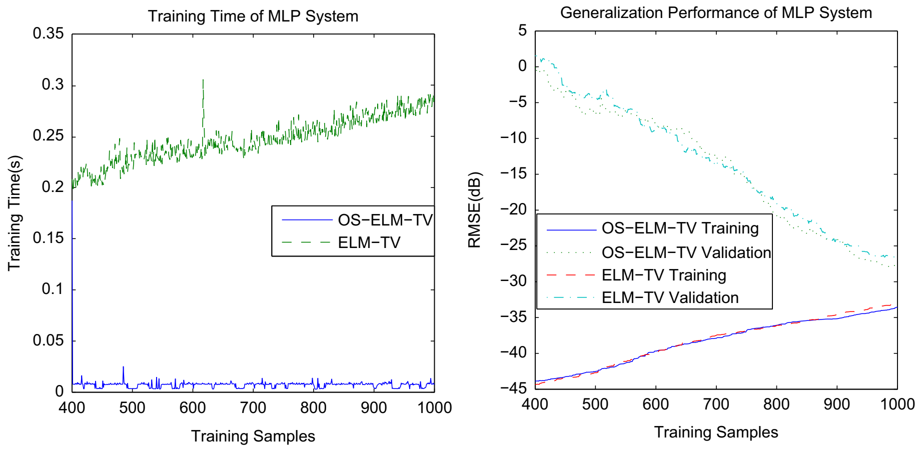


Fig. 2. Performance comparisons of ELM-TV and OS-ELM-TV for MLP system with parameter settings as: Legendre basis functions, three input bases, three output bases and 100 hidden nodes.

used in TV-NN. All simulations have been accomplished by means of the MATLAB 7.8.0 programming environment, running on an Intel Core2 Duo CPU P8400 2.26 GHz, with Windows Vista OS.

6.1. *Time-varying MLP system identification*

The system to be identified here is the same to that in [18,19]: a time-varying IIR-buffered MLP with 11 input lines, one 5 neurons hidden layer, one neuron output layer. The input weights and output weights are combinations of three prolate basis functions; the length of the input and output time delay lines (TDLs) is equal to 6 and 5 respectively. Note that the output neuron of this system is not linear, both the hidden neurons and output neuron use tangent sigmoid activation function.

6.2. *Time-varying Narendra system identification*

The next test identification system is a modified version of the one addressed in [27], by adding the coefficients $a[n]$ and $b[n]$, which are low pass filtered versions of a random sequence, to form a time-varying system, as done in [18,19]:

$$y[n] = a[n] \cdot \frac{y[n-1]\cdot y[n-2]\cdot y[n-3]\cdot x[n-1]\cdot (y[n-3]-1) + x[n]}{1 + b[n]\cdot (y[n-3]^2 + y[n-2]^2)} \tag{28}$$

Table 2
Performance comparisons of ELM-TV and OS-ELM-TV with different basis functions and number of input/output bases when updating the 1000th sample in time-varying Narendra system, with 50 hidden nodes.

Basis function (#input, #output)	Algorithms	Training RMSE (dB)	Validation RMSE (dB)	Training time (s)
None(1,1)	ELM	−10.42	−10.47	0.0093
	OS-ELM	−10.40	−10.44	0.0002
Legendre(3,1)	ELM-TV	−15.10	−14.35	0.0108
	OS-ELM-TV	−15.21	−14.27	0.0003
Legendre(1,3)	ELM-TV	−14.47	−13.75	0.0701
	OS-ELM-TV	−14.43	−13.69	0.0055
Legendre(3,3)	ELM-TV	−16.71	−14.87	0.0850
	OS-ELM-TV	−16.64	−14.55	0.0061
Chebyshev(3,3)	ELM-TV	−16.71	−14.87	0.0773
	OS-ELM-TV	−16.42	−14.45	0.0077
Prolate(3,3)	ELM-TV	−16.91	−14.66	0.0830
	OS-ELM-TV	−16.90	−14.45	0.0081
Fourier(3,3)	ELM-TV	−11.87	−11.83	0.0807
	OS-ELM-TV	−11.61	−11.29	0.0066

The model used for this identification task is a time-varying IIR-buffered neural network with five input lines, where three of them are produced by the input TDL and the remaining two by the output feedback TDL.

6.3. *Time-varying Volterra system identification*

Volterra models are widely used to represent nonlinear systems due to the Weierstrass approximation theorem, which states that every continuous function defined on a finite closed interval can be uniformly approximated as closely as desired by a polynomial function. On the one hand, Volterra system is usually assumed to be time-invariant in most reported Volterra system related literature [28,29]. On the other hand, time variation characteristics has to be considered in many real applications such as communication channels modelling, speech processing, and biological systems. In these cases, the Volterra kernels are no longer fixed but change with time. The input–output relationship of a time-varying Volterra system can be described as [30]

$$y(n) = k_0(n) + \sum_{m_1=0}^{M} k_1(n;m_1)x(n-m_1) + \sum_{m_1=0}^{M}\sum_{m_2=0}^{M} k_2(n;m_1,m_2)x(n-m_1)x(n-m_2) + \cdots \tag{29}$$

where $\{k_i\}$ are the time-varying Volterra kernels and M is the memory length of the system.

The second-order time-varying Volterra system of Fig. 4 is considered as the system to be identified [30,23], where L1 and L2 are linear filters, $g(\cdot)$ is a second-degree polynomial nonlinearity. Specifically, they are

$$L1 : h_1(n) = 0.7950e^{-n/2} \tag{30}$$

$$L2 : h_2(n) = 1.1517n^2e^{-n} \tag{31}$$

$$g(x) = x^2 + x \tag{32}$$

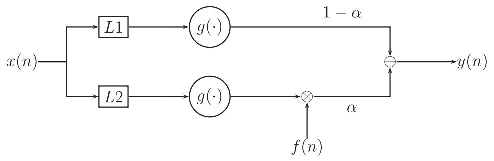
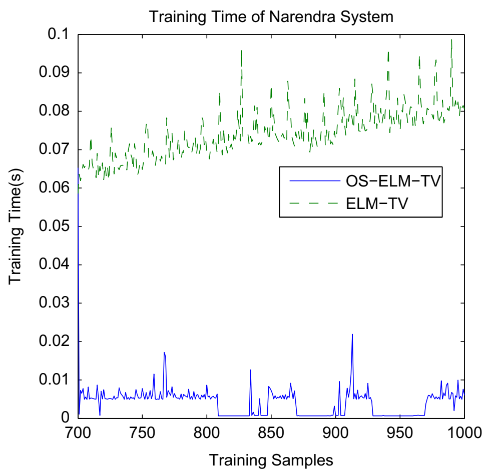


Fig. 4. Nonlinear nonstationary system with one stationary and one nonstationary branch.



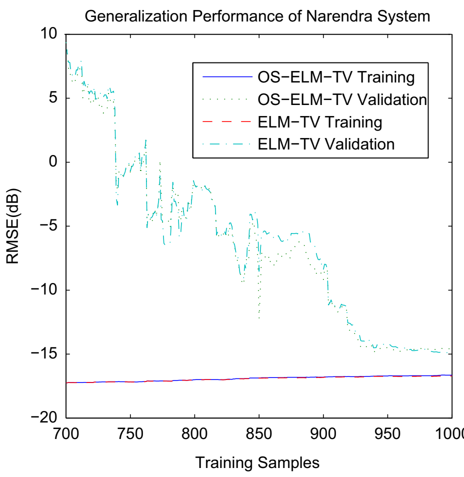


Fig. 3. Performance comparisons of ELM-TV and OS-ELM-TV for Narendra system with parameter settings as: Legendre basis functions, three input bases, three output bases and 50 hidden nodes.

The modulator $f(n)$ is applied only to the second path of the scheme in Fig. 4 and set as a combination of prolate basis functions, with the weight parameter α (equal to 0.5 in our simulation), controlling the degree of nonstationarity introduced in the system.

6.4. Electricity load estimation

Our OS-ELM-TV is also applied in one system identification problem related to a real-world data series. The objective here is building and validating a short term electricity load estimation model, which takes into account multiple sources of information including temperatures, holiday and historical loads [31]. The data set used here is a table of historical hourly loads and temperature observations from the NEPOOL region (courtesy ISO New England) for the years 2004–2008. The weather information includes the dry bulb temperature and the dew point. In order to track the consistency of time-varying characteristic in training and testing phase, training and testing data were generated by decimating the original data set with a downsampling factor equal to 2 (as shown in Fig. 6), thus resulting in two distinct sequences of 20 000 samples each, a subset of which is selected for simulation purposes.

The results of the identification tasks are depicted in Table 1 and Fig. 2, Table 2 and Fig. 3, Table 3 and Fig. 5, as well as Table 4 and Fig. 7 respectively. The numbers in parentheses in first column of the tables represent the numbers of basis functions in input and output layers. Such results consistently show that OS-ELM-TV has comparable generalization performance to ELM-TV. Note that as number of training samples increases, the training RMSE value becomes worse, because more training samples mean more rows in **G** and **T**, hence more equations to

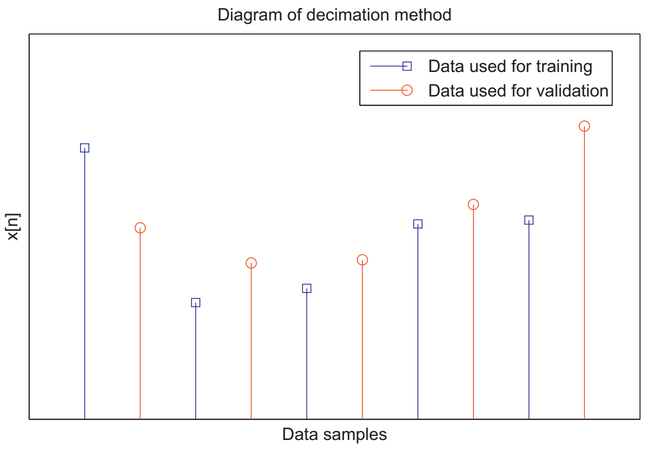


Fig. 6. Sampling of the electricity load dataset used for training and validation.

Table 3
Performance comparisons of ELM-TV and OS-ELM-TV with different basis functions and number of input/output bases when updating the 1500th sample in time-varying Volterra system, with 100 hidden nodes.

Basis function (#input, #output)	Algorithms	Training RMSE (dB)	Validation RMSE (dB)	Training time (s)
None(1,1)	ELM	−6.667	0.5344	0.0564
	OS-ELM	−6.682	0.6433	0.0065
Legendre(3,1)	ELM-TV	−9.071	0.1897	0.0579
	OS-ELM-TV	−9.151	−0.3696	0.0053
Legendre(1,4)	ELM-TV	−11.43	−4.141	0.9128
	OS-ELM-TV	−11.20	−3.699	0.0102
Legendre(3,4)	ELM-TV	−12.97	−2.508	1.0048
	OS-ELM-TV	−12.66	−2.235	0.0111
Chebyshev(1,4)	ELM-TV	−11.45	−4.089	0.9929
	OS-ELM-TV	−11.45	−3.887	0.0104
Prolate(1,4)	ELM-TV	−14.27	−4.723	0.8397
	OS-ELM-TV	−14.44	−4.450	0.0106
Fourier(1,4)	ELM-TV	−11.22	−0.634	0.8800
	OS-ELM-TV	−11.22	−0.861	0.0109

Table 4
Performance comparisons of ELM-TV and OS-ELM-TV with different basis functions and number of input/output bases when updating the 2000th sample in electricity load estimation task, with 30 hidden nodes.

Basis function (#input, #output)	Algorithms	Training RMSE (dB)	Validation RMSE (dB)	Training time (s)
None(1,1)	ELM	−14.84	−14.83	0.0123
	OS-ELM	−14.66	−14.64	0.0002
Legendre(3,1)	ELM-TV	−12.90	−12.90	0.0173
	OS-ELM-TV	−12.90	−12.91	0.0003
Legendre(1,3)	ELM-TV	−15.74	−15.69	0.0616
	OS-ELM-TV	−15.45	−15.41	0.0004
Legendre(3,3)	ELM-TV	−13.98	−13.92	0.0781
	OS-ELM-TV	−14.07	−14.03	0.0004
Chebyshev(1,3)	ELM-TV	−15.62	−15.60	0.0656
	OS-ELM-TV	−15.60	−15.59	0.0004
Prolate(1,3)	ELM-TV	−15.74	−15.71	0.0679
	OS-ELM-TV	−15.69	−15.67	0.0004
Fourier(1,3)	ELM-TV	−15.41	−15.40	0.0607
	OS-ELM-TV	−15.32	−15.30	0.0004

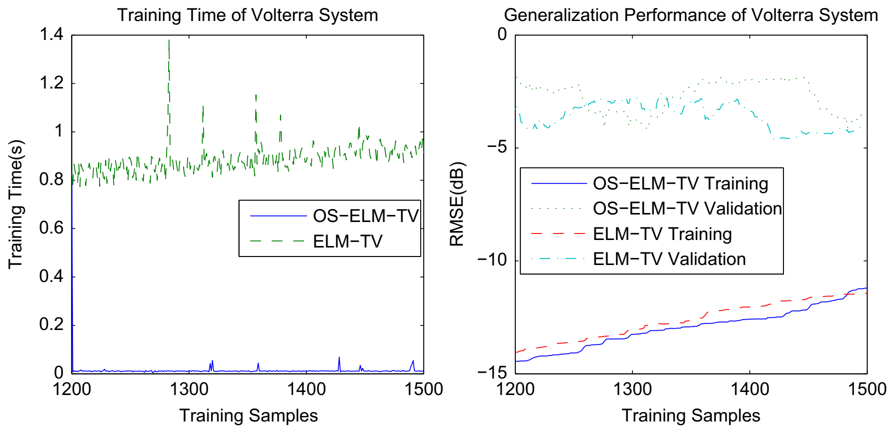


Fig. 5. Performance comparisons of ELM-TV and OS-ELM-TV for Volterra system with parameter settings as: Legendre basis functions, one input basis, four output bases and 100 hidden nodes.

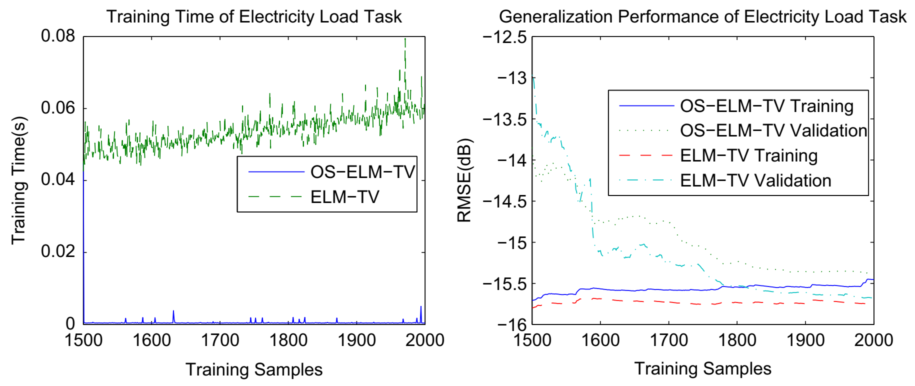


Fig. 7. Performance comparisons of ELM-TV and OS-ELM-TV for electricity load estimation task with parameter settings as: Legendre basis functions, one input basis, three output bases and 30 hidden nodes.

be satisfied in (10) and thus an increasing value of the term $\|\mathbf{G}\boldsymbol{\Omega}-\mathbf{T}\|$. However, the validation RMSE turns out to be better, as expected: the more the network is trained (if no overfitting occurs), the better it can identify the target systems.

On the other hand, OS-ELM-TV consumes much less training time when updating the output weights in the applications in which the training data set arrived sequentially. Especially in the case of large number of hidden nodes and large number of output bases, as shown in Table 3 and Fig. 5 for Volterra system identification task: once the TV-NN receives a new training sample, OS-ELM-TV takes only about 0.01 s to update its output weights, while ELM-TV takes more than 0.8 s to retrain its network; as well as in Table 4 and Fig. 7 for electricity load estimation task, 0.4 ms in OS-ELM-TV versus 60 ms in ELM-TV. Moreover, it can be easily observed from Figs. 2, 3, 5, 7 that the training time at the starting point N_0 of OS-ELM-TV is similar to that of ELM-TV, since their computation load is equivalent at N_0.

Although there are no memory requirement results explicitly shown in the simulations, we can briefly notice that ELM-TV simply runs out of memory when N comes to 10^5, while OS-ELM-TV is still able to work.

Last but not least, it is noteworthy to underline that in all tasks the nonstationary ELM approaches lead to significant system identification performance improvements w.r.t. the stationary counterpart (denoted by the "None" designation in Tables), both in the batch and in the sequential online case studies. This confirms the trend already observed in previous publications.

7. Conclusions

In this contribution, the authors have extended the online sequential extreme learning machine to train time-varying neural networks, giving origin to the new algorithm namely OS-ELM-TV. The proposed technique is oriented to applications with sequential arrival or large number of training data set. The main advantage of the online approach consists in updating the output weights with much less time when new training data arrived, or consuming less memory if large number of samples have to be trained. It has to be noted that this issue is surely much more relevant in the TV-NN case study, where the presence of time-varying synapses increases the overall number of free parameters w.r.t. standard time-invariant NN. Simulation results showed that with the benefits discussed above, OS-ELM-TV achieved comparable generalization performances to ELM-TV. It has also to be observed that the OS-ELM-TV behavior perfectly matches with that one of the original OS-ELM [14], working in stationary environments, allowing us to positively conclude about the effectiveness of the proposed generalization to the time-varying neural network case study.

Future works are first oriented to enlarge the applicability of the ELM paradigm in real-world nonstationary environments. Moreover some efforts are actually on-going in combining the sequential approach with the incremental one, already investigated by the authors [21,22] and allowing to automatize the selection of number of hidden nodes and basis function within the training algorithm.

References

[1] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: theory and applications, Neurocomputing 70 (1–3) (2006) 489–501. (Neural Networks-Selected Papers from the 7th Brazilian Symposium on Neural Networks (SBRN'04)).
[2] Q.-Y. Zhu, A. Qin, P. Suganthan, G.-B. Huang, Evolutionary extreme learning machine, Pattern Recognition 38 (10) (2005) 1759–1763.
[3] G.-B. Huang, L. Chen, C.-K. Siew, Universal approximation using incremental constructive feedforward networks with random hidden nodes, IEEE Trans. Neural Networks 17 (4) (2006) 879–892.
[4] G. Huang, L. Chen, Enhanced random search based incremental extreme learning machine, Neurocomputing 71 (16–18) (2008) 3460–3468.
[5] G. Feng, G.-B. Huang, Q. Lin, R. Gay, Error minimized extreme learning machine with growth of hidden nodes and incremental learning, IEEE Trans. Neural Networks 20 (8) (2009) 1352–1357.
[6] S. Suresh, S. Saraswathi, N. Sundararajan, Performance enhancement of extreme learning machine for multi-category sparse data classification problems, Eng. Appl. Artif. Intell. 23 (7) (2010) 1149–1157.
[7] E. Soria-Olivas, J. Gomez-Sanchis, J. Martin, J. Vila-Frances, M. Martinez, J. Magdalena, A. Serrano, BELM: Bayesian extreme learning machine, IEEE Trans. Neural Networks 22 (3) (2011) 505–509.
[8] Y. Miche, A. Sorjamaa, P. Bas, O. Simula, C. Jutten, A. Lendasse, OP-ELM: optimally pruned extreme learning machine, IEEE Trans. Neural Networks 21 (1) (2010) 158–162.
[9] J. Cao, Z. Lin, G. Huang, Composite function wavelet neural networks with extreme learning machine, Neurocomputing 73 (7–9) (2010) 1405–1416.
[10] F. Han, D.-S. Huang, Improved extreme learning machine for function approximation by encoding a priori information, Neurocomputing 69 (16–18) (2006) 2369–2373.
[11] D. Huang, Radial basis probabilistic neural networks: model and application, Int. J. Pattern Recognition Artif. Intell. 13 (1999) 1083–1101.
[12] D.-S. Huang, J.-X. Du, A constructive hybrid structure optimization methodology for radial basis probabilistic neural networks, IEEE Trans. Neural Networks 19 (12) (2008) 2099–2115.
[13] G. Huang, C. Siew, Extreme learning machine: RBF network case, in: Control, Automation, Robotics and Vision Conference, 2004. ICARCV 2004 8th, vol. 2, IEEE, 2004, pp. 1029–1036.
[14] N.-Y. Liang, G.-B. Huang, P. Saratchandran, N. Sundararajan, A fast and accurate online sequential learning algorithm for feedforward networks, IEEE Trans. Neural Networks 17 (6) (2006) 1411–1423.
[15] M.-B. Li, M.J. Er, Nonlinear system identification using extreme learning machine, in: Proceedings of the 9th International Conference on Control, Automation, Robotics and Vision ICARCV '06, 2006, pp. 1–4.
[16] Y. Xu, Z. Dong, K. Meng, R. Zhang, K. Wong, Real-time transient stability assessment model using extreme learning machine, Gener. Transm. Distrib. IET 5 (3) (2011) 314–322.
[17] D. Percival, A. Walden, Spectral Analysis for Physical Applications: Multitaper and Conventional Univariate Techniques, Cambridge University Press, 1993.

[18] A. Titti, S. Squartini, F. Piazza, A new time-variant neural based approach for nonstationary and non-linear system identification, in: Proceedings of the IEEE International Symposium on Circuits and Systems ISCAS 2005, 2005, pp. 5134–5137.
[19] C. Cingolani, S. Squartini, F. Piazza, An extreme learning machine approach for training time variant neural networks, in: Proc. IEEE Asia Pacific Conference on Circuits and Systems APCCAS 2008, 2008, pp. 384–387.
[20] Y. Ye, S. Squartini, F. Piazza, A group selection evolutionary extreme learning machine approach for time-variant neural networks, in: Neural Nets WIRN10: Proceedings of the 20th Italian Workshop on Neural Nets, IOS Press, 2011, pp. 22–33.
[21] Y. Ye, S. Squartini, F. Piazza, ELM-based Algorithms for Nonstationary Volterra System Identification, Frontiers in Artificial Intelligence and Applications, Volume 234, 2011 Neural Nets WIRN11 - Proceedings of the 21st Italian Workshop on Neural Nets Edited by Bruno Apolloni, Simone Bassis, Anna Esposito, Carlo Francesco Morabito, ISBN 978-1-60750-971-4.
[22] Y. Ye, S. Squartini, F. Piazza, ELM-based time-variant neural networks with incremental number of output basis functions, in: Advances in Neural Networks—ISNN 2011, 2011, pp. 403–410.
[23] Y. Ye, S. Squartini, F. Piazza, ELM-based algorithms for nonstationary volterra system identification. Frontiers in artificial intelligence and applications, 234, (2011) Neural Nets WIRN11. In: Proceedings of the 21st Italian Workshop on Neural Nets Edited by Bruno Apolloni, Simone Bassis, Anna Esposito, Carlo Francesco Morabito, ISBN 978-1-60750-971-4.
[24] Y. Grenier, Time-dependent ARMA modeling of nonstationary signals, IEEE Trans. Acoust. Speech Signal Process. 31 (4) (1983) 899–911.
[25] R. Horn, C. Johnson, Topics in Matrix Analysis, Cambridge University Press, 1994.
[26] G. Golub, C. Loan, Matrix Computations, Johns Hopkins Studies in the Mathematical Sciences, 3rd ed., Johns Hopkins University Press, 1996.
[27] K. Narendra, K. Parthasarathy, Identification and control of dynamical systems using neural networks, IEEE Trans. Neural Networks 1 (1) (1990) 4–27.
[28] L. Azpicueta-Ruiz, M. Zeller, A. Figueiras-Vidal, J. Arenas-Garcia, W. Kellermann, Adaptive combination of Volterra kernels and its application to nonlinear acoustic echo cancellation, IEEE Trans. Audio Speech Lang. Process. 19 (1) (2011) 97–110.
[29] G. Glentis, P. Koukoulas, N. Kalouptsidis, Efficient algorithms for Volterra system identification, IEEE Trans. Signal Process. 47 (11) (1999) 3042–3057.
[30] M. Iatrou, T. Berger, V. Marmarelis, Modeling of nonlinear nonstationary dynamic systems with a novel class of artificial neural networks, IEEE Trans. Neural Networks 10 (2) (1999) 327–339.
[31] A. Deoras, Electricity Load and Price Forecasting Webinar Case Study, 2010. URL 〈http://www.mathworks.com/matlabcentral/fileexchange/28684-electricity-load-and-price-forecasting-webinar-case-study〉.

Francesco Piazza (Member IEEE) was born in Jesi, Italy, on February 1957. He got the Italian Laurea with honors in electronic engineering from the University of Ancona, Italy, in 1981. From 1981 to 1983 he worked on image processing at the Physics Department. In 1983 he worked at the Olivetti OSAI software development center (Ivrea, Italy). In 1985 he joined the Department of Electronics and Automatics of the University of Ancona, first as Researcher in Electrical Engineering, then as Associate Professor and lastly as Full Professor. He is author or coauthor of more than 200 international papers. His current research interests are in the areas of circuit theory and digital signal processing including adaptive DSP algorithms and circuits, artificial neural networks, speech and audio processing.

Stefano Squartini (Member IEEE/ISCA) was born in Ancona, Italy, on March 1976. He got the Italian Laurea with honors in electronic engineering from University of Ancona (now Polytechnic University of Marche, UnivPM), Italy, in 2002. He got his Ph.D. degree (2005) and also a PostDoc fellowship (June 2006–October 2007) at UnivPM. He currently works as Assistant Professor in Circuit Theory at Department of Information Engineering (UnivPM). He is one of the founding members of the research group 3MediaLabs and he actively participated to various (funded) regional, national and European projects on multimedia Digital Signal Processing (DSP). He is author and coauthor of several international scientific peer-reviewed articles. He is currently Associate Editor for IEEE Transactions on Neural Networks (2010–2011), member of the Cognitive Computation Editorial Board and member of the Executive Board of SIREN (the Italian Society on Neural Networks). His research interests are in the area of digital signal processing and machine learning, with special attention to speech and audio processing related problems.

Yibin Ye was born in Guangdong, China, on January 10, 1981. He received the B.S. degree in electronic and information engineering in 2003 at South China University of Technology. He is currently working toward the Ph.D. degree in the field of neural networks at Università Politecnica delle Marche (Italy). His research interests include wireless networks and machine learning.