On-line Sequential Extreme Learning Machine Based on Recursive Partial Least Squares

Tiago Matias[a,*], Francisco Souza[a], Rui Araújo[a], Nuno Gonçalves[a], João P. Barreto[a]

[a] *Institute of Systems and Robotics (ISR-UC), and*
Department of Electrical and Computer Engineering (DEEC-UC),
University of Coimbra, Pólo II, PT-3030-290 Coimbra, Portugal

Abstract

This paper proposes the online sequential extreme learning machine algorithm based on the recursive partial least-squares method (OS-ELM-RPLS). It is an improvement to the online sequential extreme learning machine based on recursive least-squares (OS-ELM-RLS) introduced in [1]. Like in the batch extreme learning machine (ELM), in OS-ELM-RLS the input weights of a single-hidden layer feedforward neural network (SLFN) are randomly generated, however the output weights are obtained by a recursive least-squares (RLS) solution. However, due to multicollinearities in the columns of the hidden-layer output matrix caused by presence of redundant input variables or by the large number of hidden-layer neurons, the problem of estimation the output weights can become ill-conditioned. In order to circumvent or mitigate such ill-conditioning problem, it is proposed to replace the RLS method by the recursive partial least-squares (RPLS) method. OS-ELM-RPLS was applied and compared with three other methods over three real-world data sets. In all the experiments, the proposed method always exhibits the best prediction performance.

Keywords: Single-hidden layer feedforward neural networks, Least-squares, Partial least-squares, Latent variables.

1. Introduction

Multilayer feedforward neural networks (FFNN) have been used as universal approximators [2, 3] for system identification. However, the training time of FFNN has been the bottleneck of the use of these networks in industrial applications, being the linear models often preferred in comparison to multilayer FFNN [4]. In order to overcome this problem in the construction of FFNN models, a new method called extreme learning machine (ELM) was proposed in [5]. These improvements provided by ELM make these models a valuable tool in industrial processes.

ELM is a batch learning algorithm for single hidden-layer FFNN (SLFN) where the input weights (weights of connections between the input variables and the neurons in the hidden-layer) and the bias of neurons in the hidden-layer are randomly assigned. The output weights (weights of connections between the neurons in the hidden-layer and the output neuron) are obtained using the Moore-Penrose (MP) generalized inverse, considering in the output neuron a linear activation function. However, in some applications a sequential learning should be preferred over the batch learning. One example of sequential learning application is in the online modeling of a process with a time-varying behavior. In this case, collecting a training data set that would be representative of all possible states and conditions of the process can be very difficult. These conditions include different intrinsic states in which the process can be operated, and also different states related to environmental changes, changes of the process input materials, etc. Due to this difficulty, an online adaptation tool should be used in order to construct a SLFN model that has the capability to self-adjust its parameters in order to provide a good estimation in each operation scenario.

In [1, 6] an online sequential extreme learning machine based on the recursive least-squares (RLS) algorithm called OS-ELM-RLS was presented. In both papers, it was shown that OS-ELM-RLS runs much faster than other popular sequential algorithms and provides better generalization performances on many benchmark problems in the regression. However, in the applica-

[*] Corresponding author at: Institute of Systems and Robotics (ISR-UC), University of Coimbra, Pólo II, PT-3030-290 Coimbra, Portugal. Tel.: +351 961569024.

Email addresses: `tmatias@isr.uc.pt` (Tiago Matias), `fasouza@isr.uc.pt` (Francisco Souza), `rui@isr.uc.pt` (Rui Araújo), `nunogon@isr.uc.pt` (Nuno Gonçalves), `jpbar@deec.uc.pt` (João P. Barreto)

tion of OS-ELM-RLS the outputs of hidden neurons can have strong multicollinearities due to a large number of hidden nodes or due to redundancy in input variables. In such situations the output matrix of the hidden-layer, corresponding to a set of data samples, may have not full rank, which can result in an ill-conditioned problem to be solved by the least-squares solution, and in an unstable solution. In order to circumvent or mitigate the ill-condition problem, the RLS method can be replaced by the recursive partial least-squares (RPLS) method.

In this paper, a new SLFN learning method using an online sequential extreme learning machine algorithm based on RPLS (OS-ELM-RPLS) is proposed. In the proposed methodology, the output weights of the SLFN are updated when a new data sample is available using a RPLS method, and the number of latent variables is adapted using a leave-one-out validation: for each new sample, before updating the output weights vector, the preceding data sample is used to select the best number of latent variables. OS-ELM-RPLS was applied and compared with OS-ELM-RLS, RPLS, and OS-ELM-RPLSM over three real-world data sets. The OS-ELM-RPLSM is a modified version of OS-ELM-RPLS method, where the number of RPLS latent variables is not adapted online, but is selected by a 10-fold cross-validation procedure on the training data set. In all the experiments, the proposed OS-ELM-RPLS method always exhibits the best prediction performance.

The paper is organized as follows. The SLFN architecture is overviewed in Section 2. Section 3 gives a brief review of the back-propagation algorithm. A review of the batch and sequential ELM is given in Section 4. The proposed method is presented in Section 5. Section 6 presents experimental results. Finally, concluding remarks are drawn in Section 7.

2. Single Hidden-Layer Feedforward Network Architecture

The neural network considered in this paper is a single hidden-layer feedforward neural network (Fig. 1) with n input variables, h hidden-layer neurons, and one neuron in the output layer. The output of the SLFN at time-instant k is given by:

$$\hat{y}(k) = g\left(\mathbf{v}^T \mathbf{s}_k\right), \tag{1}$$

where $g(\cdot)$ represents the activation function of the output neuron, $\mathbf{v} = [\beta_O, v_1, \ldots, v_h]^T$ is the vector of output weights and bias, $\mathbf{s}_k = [1, \boldsymbol{\sigma}_k^T]^T$ is the vector of inputs to the output node (the first element is for the output bias), and $\boldsymbol{\sigma}_k = [s_1(k), \ldots, s_h(k)]^T$ is the vector of the outputs of the h hidden-layer neurons given by:

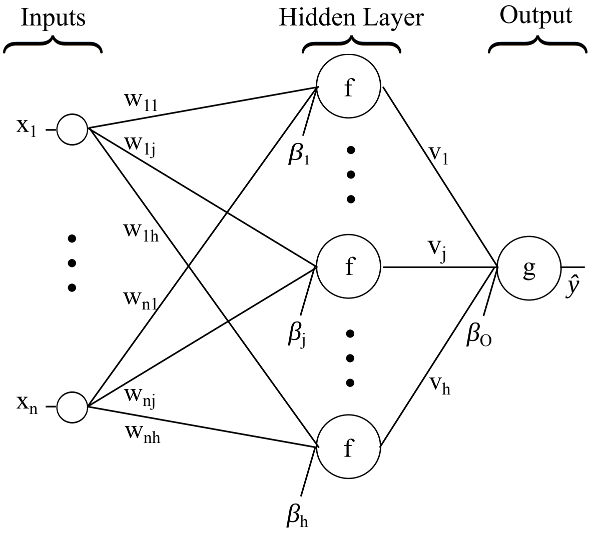

Figure 1: Single hidden-layer feedforward network with adjustable architecture.

$$\boldsymbol{\sigma}_k = \boldsymbol{\Gamma}\left(\mathbf{W}^T \mathbf{x}_k\right), \tag{2}$$

where

$$\mathbf{W} = \begin{bmatrix} \beta_1 & \beta_2 & \ldots & \beta_h \\ w_{11} & w_{12} & \ldots & w_{1h} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n1} & w_{n2} & \ldots & w_{nh} \end{bmatrix} \tag{3}$$

is the matrix of first level weights and biases, b_j is the bias of hidden neuron j, and w_{ij} is the weight between the i-th input variable and the j-th hidden-layer neuron. $\mathbf{x}_k = [1, x_1(k), x_2(k), \ldots, x_n(k)]^T$ is the vector of inputs where the first element is for the bias of each hidden neuron, $\boldsymbol{\Gamma}(\boldsymbol{\xi}) = [f(\xi_1), \ldots, f(\xi_h)]^T$ for $\boldsymbol{\xi} = [\xi_1, \ldots, \xi_h]^T$, where $f(\cdot)$ represents the activation function of the neurons of the hidden layer, and $\xi_1, \ldots, \xi_h$ are general variables used for defining $\boldsymbol{\Gamma}(\boldsymbol{\xi})$.

3. Conventional Gradient-Based Training

Assuming the availability of N input-output data samples, the objective of conventional gradient-based training algorithms is to find the weights $\mathbf{W}$, and $\mathbf{v}$ that minimize the following cost function:

$$e = \sum_{k=1}^{N} \left(g\left(\mathbf{v}^T \mathbf{s}_k\right) - y(k)\right)^2, \tag{4}$$

where $y(k)$ is the desired output at time instant k.

Generally, gradient-based training algorithms, like backpropagation of the error, are used to minimize (4). In the first step, $\mathbf{W}$ and $\mathbf{v}$ are randomly obtained, and in next steps $\mathbf{W}$ and $\mathbf{v}$ are iteratively adjusted as follows:

$$\mathbf{W}_i = \mathbf{W}_{i-1} - \eta \frac{\partial e}{\partial \mathbf{W}}, \quad (5)$$

$$\mathbf{v}_i = \mathbf{v}_{i-1} - \eta \frac{\partial e}{\partial \mathbf{v}}, \quad (6)$$

where η is a learning rate. However, there are some issues in the training of the SLFN using gradient-based training algorithms:

1. The training requires a large amount of data;
2. The convergence to the global minimum can be very slow if the learning rate η is too small. However if η is too large, the algorithm can become unstable and may not be able to reach the global minimum;
3. They are very time-consuming in most applications;
4. The training is prone to local minima and overfitting;

In order to overcome these problems, the batch ELM was proposed in [5]. This method randomly generates the weights and bias of the first level of the SLFN and obtains the output weights using a MP generalized inverse, improving the generalization capability and decreasing both the number of parameters to be adjusted by the user and the computational costs, allowing the implementation in common automation equipment such as programmable logic controllers or microcontrollers. However, in processes with time-varying behaviors, sequential learning algorithms may be preferred over batch learning algorithms as they do an incremental learning, not requiring a complete batch retraining whenever new data is received.

4. Extreme Learning Machine

The batch ELM was proposed in [5]. In [7] it is proved that a SLFN with randomly chosen weights between the input layer and the hidden layer, and adequately chosen output weights are universal approximators for any bounded non-linear piecewise continuous function. In ELM, the input weights and bias matrix $\mathbf{W}$ are randomly assigned and, considering an output neuron with a linear activation function, the SLFN network can be regarded as a linear regression model between the output vector of the hidden layer and the output of the SLFN. Therefore, the output weights vector $\mathbf{v}$ can be estimated as:

$$\hat{\mathbf{v}} = \mathbf{S}_N^{\dagger} \mathbf{y}_N, \quad (7)$$

where $\mathbf{S}_N^{\dagger}$ is the Moore-Penrose generalized inverse of the output node input matrix

$$\mathbf{S}_N = [\mathbf{s}_1, \ldots, \mathbf{s}_N]^T, \quad (8)$$

and $\mathbf{y}_N = [y(1), \ldots, y(N)]^T$ is the vector of the target outputs.

Considering that $\mathbf{S}_N \in \mathbb{R}^{N \times h}$ with $N \geq h$ and $rank(\mathbf{S}_N) = h$ the Moore-Penrose generalized inverse of $\mathbf{S}_N$ can be given by:

$$\mathbf{S}_N^{\dagger} = \left(\mathbf{S}_N^T \mathbf{S}_N\right)^{-1} \mathbf{S}_N^T. \quad (9)$$

Substituting (9) into (7), the estimate $\hat{\mathbf{v}}$ of $\mathbf{v}$ can be obtained by the following least-squares solution:

$$\hat{\mathbf{v}} = \left(\mathbf{S}_N^T \mathbf{S}_N\right)^{-1} \mathbf{S}_N^T \mathbf{y}_N. \quad (10)$$

The sequential implementation of the ELM results in the application of recursive least-squares (RLS) to estimate the output weights vector [1]. Considering that N_0 ($N_0 \geq (h+1)$) initial data samples are available and that $rank(\mathbf{S}_{N_0}) = h$, the estimate $\hat{\mathbf{v}}$ of $\mathbf{v}$ can be obtained by Algorithm 1.

Algorithm 1 On-line sequential ELM based on RLS.

1. Initialize the covariance matrix $\mathbf{M}_0 = \left(\mathbf{S}_{N_0}^T \mathbf{S}_{N_0}\right)^{-1}$, and the output weights estimate $\hat{\mathbf{v}}_0 = \mathbf{M}_0 \mathbf{S}_{N_0}^T \mathbf{y}_{N_0}$.
2. For each newly available data sample k, the output weights estimate, and the covariance matrix can be recursively obtained by [8]:

$$\hat{\mathbf{v}}_k = \hat{\mathbf{v}}_{k-1} + \mathbf{M}_{k-1}\mathbf{s}_k \frac{y(k) - \mathbf{s}_k^T \hat{\mathbf{v}}_{k-1}}{\lambda + \mathbf{s}_k^T \mathbf{M}_{k-1} \mathbf{s}_k}, \quad (11)$$

$$\mathbf{M}_k = \frac{1}{\lambda}\left(\mathbf{M}_{k-1} - \frac{\mathbf{M}_{k-1}\mathbf{s}_k \mathbf{s}_k^T \mathbf{M}_{k-1}}{\lambda + \mathbf{s}_k^T \mathbf{M}_{k-1} \mathbf{s}_k}\right), \quad (12)$$

where λ is a forgetting factor. Lower values of λ indicate that the recent data will influence more the new model.

Despite of the fast ELM training time, and regardless of using either batch or sequential/recursive ELM implementation, the solution obtained by least-squares may be not the most robust solution. The columns of the output node input matrix $\mathbf{S}_N$ can have strong multicollinearities due to large number of hidden nodes or due to redundancy in the input variables, which can result in an ill-conditioned term $\left(\mathbf{S}_N^T \mathbf{S}_N\right)^{-1}$ and in an unstable least-squares solution [9]. In order to circumvent or mitigate the ill-condition of the output node input matrix, the partial least-squares (PLS) can be used to obtain the output weights.

5. On-line Sequential ELM Based on Recursive Partial Least-Squares

If matrix $\left(\mathbf{S}_N^T\mathbf{S}_N\right)^{-1}$ is well-conditioned, the best estimate of $\mathbf{v}$ in a least squares sense is given by (10). However, if $\left(\mathbf{S}_N^T\mathbf{S}_N\right)^{-1}$ is ill-conditioned, an alternative to the Moore-Penrose generalized inverse should be used. Using a PLS method, the data are projected into a space spanned by a number of latent variables (factors), and $\mathbf{S}_N$ and $\mathbf{y}_N$ are then represented by:

$$\mathbf{S}_N = \mathbf{T}\mathbf{P}^T + \mathbf{E}, \tag{13}$$

$$\mathbf{y}_N = \mathbf{U}\mathbf{q}^T + \mathbf{f}, \tag{14}$$

where $\mathbf{T} = [\mathbf{t}_1, \ldots, \mathbf{t}_l] \in \mathbb{R}^{N\times l}$ and $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_l] \in \mathbb{R}^{N\times l}$ are the latent score matrices, and $\mathbf{P} = [\mathbf{p}_1, \ldots, \mathbf{p}_l] \in \mathbb{R}^{h\times l}$ and $\mathbf{q} = [q_1, \ldots, q_l] \in \mathbb{R}^{1\times l}$ are the loading matrices. $\mathbf{E} \in \mathbb{R}^{N\times h}$ and $\mathbf{f} \in \mathbb{R}^{N\times 1}$ are the input and output data residuals, and l is the number of latent variables used in the model.

The data matrices $\mathbf{S}_N$ and $\mathbf{y}_N$ can also be iteratively decomposed as follows. First, let:

$$\mathbf{S}_N = \mathbf{t}_1\mathbf{p}_1^T + \mathbf{E}_1, \tag{15}$$

$$\mathbf{y}_N = \mathbf{u}_1 q_1 + \mathbf{f}_1, \tag{16}$$

where $\mathbf{t}_1$ and $\mathbf{u}_1$ are the first column of the latent score matrices $\mathbf{T}$ and $\mathbf{U}$, and $\mathbf{p}_1$ and q_1 are the first column of the loading matrices $\mathbf{P}$ and $\mathbf{q}$. $\mathbf{E}_1$ and $\mathbf{f}_1$ are the input and output data residuals in the first iteration. The latent score vectors are related by a linear inner model:

$$\mathbf{u}_1 = b_1\mathbf{t}_1 + \mathbf{r}_1, \tag{17}$$

where b_1 is a coefficient which is determined by minimizing the residual $\mathbf{r}_1$. After going through the first latent score vectors calculation, the second vectors are calculated by decomposing the residuals $\mathbf{E}_1$ and $\mathbf{f}_1$ as follows:

$$\mathbf{E}_2 = \mathbf{E}_1 - \mathbf{t}_1\mathbf{p}_1^T, \tag{18}$$

$$\mathbf{f}_2 = \mathbf{f}_1 - b_1\mathbf{t}_1 q_1, \tag{19}$$

being the $\mathbf{t}_1$ score vector orthogonal to $\mathbf{E}_1$ and $\mathbf{f}_1$. This procedure is repeated until all the $\mathbf{T}$, $\mathbf{P}$, $\mathbf{U}$, $\mathbf{q}$ matrices are calculated. The overall PLS algorithm is summarized in Algorithm 2 [9]. This algorithm is derived with the assumption that the data $\mathbf{S}_N$ and $\mathbf{y}_N$ are scaled to zero mean and unit variance. However, if the levels of the signals at the outputs of the neurons are comparable, the scaling may be unnecessary [10].

Using this algorithm, the matrices $\mathbf{T}$, $\mathbf{P}$, $\mathbf{q}$, $\mathbf{I}$, and $\mathbf{B}$ can be constructed, where:

$$\mathbf{I} = [\mathbf{i}_1, \ldots, \mathbf{i}_l], \tag{20}$$

$$\mathbf{B} = diag\{b_1, \ldots, b_l\}. \tag{21}$$

Algorithm 2 Traditional batch-wise PLS algorithm with N available data samples.

```
Inputs: Output node input matrix, $\mathbf{S}_N$; the vector of target outputs, $\mathbf{y}_N$; and the number of latent variables $l$;
1. Set $\mathbf{E}_0 = \mathbf{S}_N$, $\mathbf{f}_0 = \mathbf{y}_N$, and $k = 0$;
2. Let $k \leftarrow k + 1$ and $\mathbf{u}_k \leftarrow \mathbf{f}_{k-1}$;
3. Compute the latent scores and the loading factors:
   (a) $\mathbf{i}_k = \mathbf{E}_{k-1}^T\mathbf{u}_k/\mathbf{u}_k^T\mathbf{u}_k$;
   (b) $\mathbf{t}_k = \mathbf{E}_{k-1}\mathbf{i}_k/||\mathbf{E}_{k-1}\mathbf{i}_k||$;
   (c) $q_k = \mathbf{f}_{k-1}^T\mathbf{t}_k/||\mathbf{f}_{k-1}^T\mathbf{t}_k||$;
   (d) $\mathbf{u}_k = \mathbf{f}_{k-1}q_k$;
   (e) $\mathbf{p}_k = \mathbf{E}_{k-1}^T\mathbf{t}_k$;
4. Compute the coefficient $b_k$:
   (a) $b_k \leftarrow \mathbf{u}_k^T\mathbf{t}_k$;
5. Compute the residuals $\mathbf{E}_k$ and $\mathbf{f}_k$:
   (a) $\mathbf{E}_k = \mathbf{E}_{k-1} - \mathbf{t}_k\mathbf{p}_k^T$,
   (b) $\mathbf{f}_k = \mathbf{f}_{k-1} - b_k\mathbf{t}_k q_k$;
6. Repeat Steps 2) to 6) until all $l$ principal factors are calculated.
```

Consider the following Lemma [9]:

Lemma 1. *If rank*$(\mathbf{S}_N) = l$ *and* $l \leq (h+1)$, *then*

$$\mathbf{E}_l = \mathbf{E}_{l+1} = \ldots = \mathbf{E}_{h+1} = 0. \tag{22}$$

In matrix form, using Lemma 1 and equations (15)-(17), $\mathbf{S}_N$ and $\mathbf{y}_N$ can be decomposed as:

$$\mathbf{S}_N = \mathbf{T}\mathbf{P}^T + \mathbf{E}_l = \mathbf{T}\mathbf{P}^T, \tag{23}$$

$$\mathbf{y}_N = \mathbf{T}\mathbf{B}\mathbf{q}^T + \mathbf{f}_l. \tag{24}$$

Consider the following Lemma [9], which shows that $\mathbf{f}_l$ is orthogonal to the latent score vector $\mathbf{t}_l$:

Lemma 2. *The output residual* $\mathbf{f}_i$ *is orthogonal to the previous latent score vector* $\mathbf{t}_j$, *i.e.*

$$\mathbf{t}_j^T\mathbf{f}_i = 0, \textit{ for } i \geq j. \tag{25}$$

As the latent score vectors are orthogonal and have unit length (Algorithm 2, Step 3b), all the columns of $\mathbf{T}$ are mutually orthonormal. So, the following relation can be derived using (23), (24) and Lemma 2,

$$\mathbf{S}_N^T\mathbf{S}_N = \mathbf{P}\mathbf{T}^T\mathbf{T}\mathbf{P}^T = \mathbf{P}\mathbf{P}^T, \tag{26}$$

$$\mathbf{S}_N^T\mathbf{y}_N = \mathbf{P}\mathbf{T}^T\mathbf{T}\mathbf{B}\mathbf{q}^T + \mathbf{P}\mathbf{T}^T\mathbf{f}_l = \mathbf{P}\mathbf{B}\mathbf{q}^T. \tag{27}$$

In order to minimize the square residuals $||\mathbf{y}_N - \mathbf{S}_N\mathbf{v}||^2$, using (26)-(27) the LS solution (10) can be transformed into the following PLS solution:

$$\hat{\mathbf{v}} = (\mathbf{P}\mathbf{P}^T)^{-1}\mathbf{P}\mathbf{B}\mathbf{q}^T. \tag{28}$$

This solution is designed for the offline case (batch learning). However, when dealing with time-varying environments, and when the samples are delivered sequentially over the time, the solution is achieved by merging the old model, represented by matrices **P**, **B**, and **q**, with the new sample. In the recursive PLS (RPLS), at each time instant k, the data is represented by:

$$\mathbf{S}_k^{PLS} = \begin{bmatrix} \lambda \mathbf{P}^T \\ \mathbf{s}_k^T \end{bmatrix}; \quad \mathbf{y}_k^{PLS} = \begin{bmatrix} \lambda \mathbf{B} \mathbf{q}^T \\ y(k) \end{bmatrix}, \tag{29}$$

where λ is the forgetting factor, which has a role similar to the role it has in the LS estimator. Then, $\mathbf{S}_k^{PLS}$ and $\mathbf{y}_k^{PLS}$ can be applied as inputs $\mathbf{S}_N$, $\mathbf{y}_N$ in Algorithm 2 to find the new PLS parameters.

The method proposed in this paper is based on the use of the recursive partial least-squares method to estimate the output weights of a SLFN (considering an output neuron with a linear activation function) and is called OS-ELM-RPLS.

A set of initializations is performed before the online operation of OS-ELM-RPLS. In the first step of OS-ELM-RPLS initialization, a training data set with size N_0 is collected, where N_0 must be at least equal to the number of neurons in the hidden layer. In the next steps, the input weights and bias are randomly assigned, and the outputs of the hidden neurons are obtained. Using these outputs and considering that the number of latent variables l is equal to the rank of the $\mathbf{S}_{N_0}$, the matrices **T**, **P**, **q**, **I**, and **B** are obtainded using the Algorithm 2. In the last step of the initialization stage, the output weights vector $\hat{\mathbf{v}}_0$ is estimated using (28).

The proposed OS-ELM-RPLS method is summarized in Algorithm 3. In the online operation of the algorithm, first, in each time instant k, the output of the network is estimated. After this, the number of RPLS latent variables is selected using a leave-one-out validation methodology: before updating the output weights vector, the number of latent variables is selected based on the performance of the SLFN using the samples $\mathbf{S}_{k-1}^{PLS}$ and output weights vector $\hat{\mathbf{v}}_{k-1}$ obtained until time instant $(k-1)$. After having selected the number of latent variables, the matrices $\mathbf{S}_k^{PLS}$ and $\mathbf{y}_k^{PLS}$ are recursively updated using (2) and (29), and the matrices **T**, **P**, **q**, **I**, and **B** are obtained using Algorithm 2. At last, the output weights vector of the network is estimated.

6. Results

This section presents experimental results in three real-world data sets. Table 1 describes the data sets, and parameters used in the experiments. For comparison purposes, the proposed OS-ELM-RPLS method is compared with (i) OS-ELM based on RLS proposed in [1] (OS-ELM-RLS), and (ii) RPLS method proposed in [9]. The OS-ELM-RPLSM is a modified version of OS-ELM-RPLS method, where the number of RPLS latent variables l is not adapted online. Thus, OS-ELM-RPLSM does not perform Steps 1b and 1c of Algorithm 3. In the OS-ELM-RPLSM and RPLS methods, the number of latent variables used was determined by a 10-fold cross-validation procedure applied on the training data set.

For each data set, 30 trials were performed using a kind of 30-fold cross-validation. The used method, instead of using 29/30 of the data set as the training set and the remaining data set as the testing set, like in the

Algorithm 3 On-line sequential ELM based on RPLS with adaptive number of latent variables.

```
1. For each newly available data sample $[\mathbf{x}_k, y(k)]$, at instant $k$ do:
    (a) Compute the estimated output $\hat{y}(k)$ using (1).
    (b) To prepare the selection of the number of latent variables, for $j = 1, \ldots, rank(\mathbf{S}_{k-1}^{PLS})$, using the previous matrix $\mathbf{S}_{k-1}^{PLS}$ and vector $\mathbf{y}_{k-1}^{PLS}$, do:
        i. Compute the matrices T, P, q, I, and B, using Algorithm 2 with $l = j$ latent variables;
        ii. Cumpute the output weights estimation $\hat{\mathbf{v}}_{k-1}$ using (28);
        iii. Compute the estimated output $\hat{y}_j(k-1)$ using (1);
        iv. Obtain the error $e_j = \hat{y}_j(k-1) - y(k-1)$ between the estimated and real outputs;
    (c) Select the number of latent variables as $l = \arg\min_{j=1,\ldots,rank(\mathbf{S}_{k-1})} (e_j)$;
    (d) Obtain $\mathbf{S}_k^{PLS}$ and $\mathbf{y}_k^{PLS}$ using (2) and (29);
    (e) Compute the matrices T, P, q, I, and B using Algorithm 2;
    (f) Compute the output weights estimation $\hat{\mathbf{v}}_k$ using (28);
```

Table 1: Data sets description and model parameters: Samples indicates the number of exemplars in the data set; architecture indicates the number of input, hidden, and output nodes used in the SLFN; λ indicates the forgetting factor used in the RPLS and in the RLS.

Data set	Samples	Architecture	λ
Debutanizer	2394	7 – 15 – 1	0.98
Polymerization	647	12 – 20 – 1	0.98
Burning Temp.	1000	7 – 10 – 1	0.99

traditional 30-fold cross-validation, in order to simulate an online learning method, 1/30 of the data is used as initialization part and the remaining is used as testing-set.

All the input and output variables have been normalized to zero mean and unit variance. The training data set was used to initialize/train the estimators and the approximation performance was evaluated on the testing data set. SLFNs with sigmoidal activation functions in the hidden-layer neurons, and a linear activation function in the output neuron were used. The optimal number of hidden neurons h and the forgetting factor λ used in all data sets were determined by means of experimentation. At each time instant k, the estimation of the output is performed before the estimator adaptation. The approximation performances of the estimators were evaluated using the average of the mean square error (MSE) between the predicted and desired outputs in the 30 trials. The computational time results refer to the mean time taken by all methods to predict y and perform the SLFN model adaptation for all the samples of the testing data set, over the 30 trials

A statistical paired Student t-test using MSE was also conducted for all data sets (for further details see [11], [12]). Specifically, paired t-test between OS-ELM-RPLS and each one of the other methods was conducted using the 30 trials realized in each data set. In this test it is considered that the null hypothesis is that the mean MSE of the two tested methods in the 30 trials is the same, and that the significance level is 0.05 for all experiments. So, if a p-value is under the significance level, it means that the observed difference is "very significant". The symbols "(+)" and "(−)" are used to indicate better or worse performances of OS-ELM-RPLS over the other tested method, respectively.

All the simulation experiments have been made in the Matlab environment running on a PC with 2.20 [GHz] CPU with 4 cores and 4GB RAM.

6.1. Experiment I - Debutanizer Process

The first case study consists of the prediction of the butane (C4) concentration at the bottom flow of a debutanizer column. This case study was introduced in [13] and an associated data set is available for download in the book website. The data set of plant variables that is available for learning consists of 7 input variables, $\mathbf{x}_k = [x_1(k), \ldots, x_7(k)]^T$, and one target output variable to be estimated, $y(k)$. The variables correspond to temperatures, pressures, flows, and the output concentration. See Table 2 for further details.

The results of application of all methods are presented in Table 3 and Figure 2. Figure 3 shows the number of latent variables in the testing data set chosen by the proposed method for all the samples in the first trial.

Table 2: Variables of the debutanizer data set.

Variables	Description
x_1	Top temperature
x_2	Top pressure
x_3	Reflux flow
x_4	Flow to next process
x_5	6^{th} tray temperature
x_6	Bottom temperature
x_7	Bottom temperature
y	Butane (C4) concentration

Table 3: Performance results of the four methods in the testing data set of the debutanizer process.

Method	Mean Testing MSE	Mean Time [s]	p-value
OS-ELM-RPLS	**0.240**	11.58	-
OS-ELM-RPLSM	0.647	11.36	**0.00**(+)
OS-ELM-RLS	0.384	0.27	**0.00**(+)
RPLS	0.618	6.13	**0.00**(+)

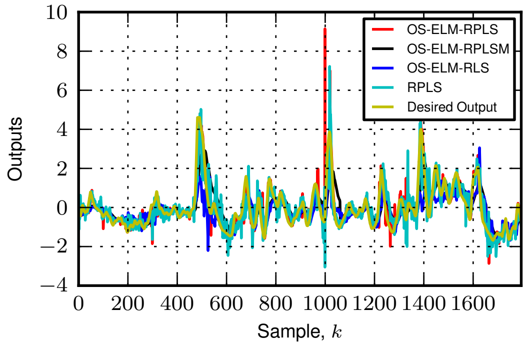

Figure 2: Predicted and desired outputs using the four methods in the debutanizer data set in the first trial.

Analyzing the results, it can be verified that the performance of the proposed OS-ELM-RPLS method is statistically better that the performance of the other three methods, followed by the OS-ELM-RLS. It can also be seen that the performance of the RPLS, which generates a linear model, is better than the performance of the OS-ELM-RPLSM. Comparing the results of the OS-ELM-RPLSM and the OS-ELM-RPLS, it can be verified the importance of the adaptation of the number of latent variables performed by the OS-ELM-RPLS during the estimation. With respect to the computational time, the OS-ELM-RLS is the fastest method. However,

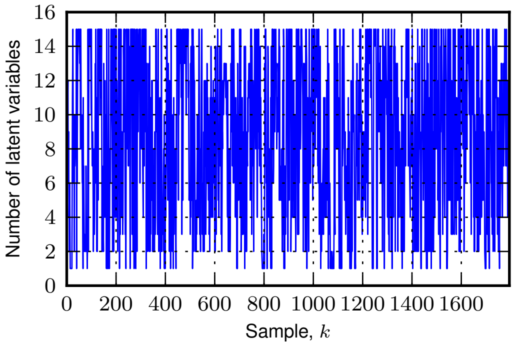

Figure 3: Number of latent variables used by the OS-ELM-RPLS in the debutanizer testing data set.

Table 4: Performance results of the four methods in the testing data set of the polymerization process.

Method	Mean Testing MSE	Mean Time [s]	p-value
OS-ELM-RPLS	**0.085**	1.36	-
OS-ELM-RPLSM	0.108	1.33	**0.00**(+)
OS-ELM-RLS	0.109	0.06	**0.00**(+)
RPLS	0.1442	2.53	**0.00**(+)

the time taken by OS-ELM-RPLS in each iteration is approximately $11.58/(2394 * 29/30) \approx 10$ [milliseconds], which is a good time for real applications.

6.2. Experiment II - Polymerization Process

The polymerization data set is a benchmark for adaptive soft sensors introduced in [14, 15]. This data set describes a polymerization reactor and the objective is the prediction of the catalyst activity in the multitube. The data set covers 1 year of acquisition with 8687 available samples and is composed by 15 input variables.

This paper follows the same pre-processing procedure as was done in [15]: downsampling of the first 5800 samples by a factor of 10 to restrict the available information in the training set, the removal of variables 3, 4, and 15, and removing all samples which have missing values. The preprocessing results on a data set with 647 data samples.

The results of application of all methods are presented in Table 4 and Figure 4. Figure 5 shows the number of latent variables in the testing data set chosen by the proposed OS-ELM-RPLS method in the first trial. As in the previous experiment, the prediction performance of the proposed method in the testing data set is statistically the best, however now followed by the OS-ELM-RPLSM. Once again, it is shown the importance of the online adaptation of the number of the latent variables. In this data set, the RPLS is the method with worst performance.

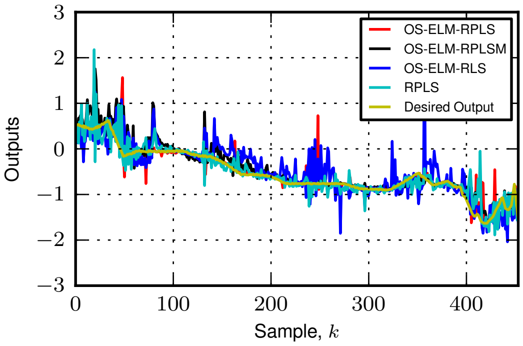

Figure 4: Predicted and desired outputs using the four methods in the polymerization data set in the first trial.

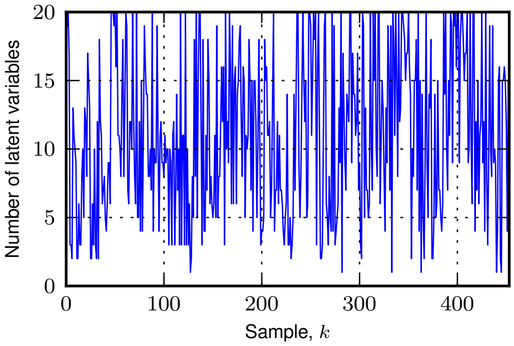

Figure 5: Number of latent variables used by the OS-ELM-RPLS in the polymerization testing data set.

6.3. Experiment III - Estimation of the Burning Zone Temperature on a Cement Kiln

Inside a rotary cement kiln, temperatures in the range of 1200-1700°C heat a mixture of limestone, shale, clay, sand, and smaller quantities of other substances, resulting in small black nodules called clinkers. Outside the kiln these clinkers are cooled and grounded to produce cement [16]. The control of the temperature inside the kiln is crucial: insufficiently high maximum temperatures in the kiln result in incompletely reacted products and poor-quality cement, while excessive maximum temperatures waste energy and propitiate the formation of NO_x pollutant compounds that have several negative environmental impacts [17].

As temperature measurement is impossible using contact, the measurement is made using a pyrometer.

However, due to the flying dust inside the kiln system that blocks the sensor after some time in operation, it has to be removed and cleaned by an operator, which can take a long time. It is therefore desirable to develop a model that is able to replace the pyrometer in the measurement of the burning zone temperature.

In this work, the experiments are made in a simulation environment using a real-world data set[1] obtained in a cement kiln plant. This data set is composed by 194 monitored variables, recorded with a sampling interval of $T = 1\,[\text{min}]$. The monitored variables refer to several system variables from the preheater (cyclone) tower until the chimney and cement mill, and include, for example, temperatures, and pressures. Most variables are online measured, but there are also some manual entries and laboratory entries. The used data has a total of 10000 samples which represent approximately one week.

Due to the large number of input variables, the selection of the most relevant variables for the estimation of the burning zone temperature was performed. As a first step the selection of the initial set of input variables was based on knowledge about the process. From the 193 available input variables, 17 variables were selected. Some of the variables represent certain temperatures and pressures in the input and output of the kiln, fuel flows (coal and alternative fuels), temperatures in the coal mill and in the cooler, etc. In a second step, the set of input variables was refined using the sequential backward search (SBS) approach proposed in [18]. After this procedure, the following set of input variables was obtained:

- Temperature of the clinker at the output of the kiln;
- Pressure of the air inside of the kiln in the burner area;
- Speed of the fan responsible for the return from the kiln to cyclone (tertiary air);
- Flow of alternative fuels;
- Flow of the fuel in the central burner;
- Flow of the fuel in the radial burner;
- Temperature of the air at the input of the kiln.

The results of application of all the prediction methods are presented in Table 5 and Figure 6. Figure 7 shows the number of latent variables in the testing data set chosen by the proposed OS-ELM-RPLS method in the first trial.

Table 5: Performance results of the four methods in the testing set for the estimation of the burning zone temperature.

Method	Mean Testing MSE	Mean Time [s]	p-value
OS-ELM-RPLS	**0.568**	2.14	-
OS-ELM-RPLSM	0.587	2.07	**0.00**(+)
OS-ELM-RLS	0.617	0.09	**0.00**(+)
RPLS	0.606	2.58	**0.00**(+)

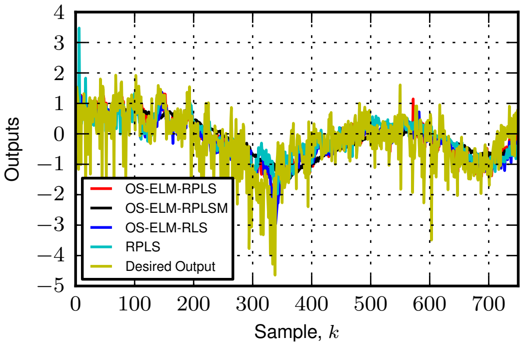

Figure 6: Predicted and desired outputs using the four methods in the estimation of the burning zone temperature in the first trial.

From the analyses of the results it can be verified again that the proposed method is, statistically, the method with best estimation performance, followed again by the OS-ELM-RPLSM. In this data set, the OS-ELM-RLS method is the method with worst performance.

7. Conclusion

A novel learning algorithm for SLFNs called on-line sequential extreme learning machine based on recursive partial least-squares with adaptation of the number of the latent variables (OL-ELM-RPLS) is presented. The proposed method is an improvement of the OS-ELM-RLS method proposed in [1], where RLS is used to update the estimation of the output weights. Due to the multicollinearities that can exist in the hidden-layer output matrix columns caused by the excessive number of hidden neurons or redundant input variables, the estimation of the output weights by LS can result in an ill-conditioned problem and therefore in an unstable solution. Therefore, in the proposed methodology the RLS method was replaced by a RPLS method. Furthermore,

[1] Provided by "Acontrol - Automação e Controle Industrial, Lda", Coimbra, Portugal.

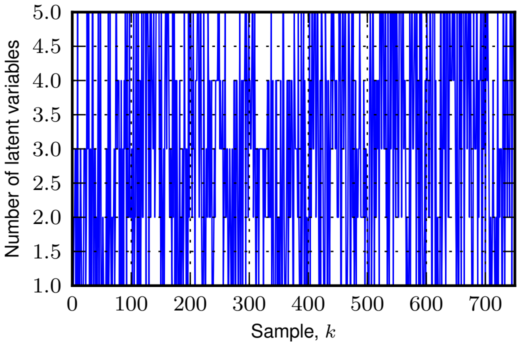

Figure 7: Number of latent variables used by the OS-ELM-RPLS in the testing set for the estimation of the burning zone temperature.

an online adaptation of the number of RPLS latent variables was introduced in the proposed algorithm.

To validate and demonstrate the performance and effectiveness of the proposed method, it was applied on three real-world data sets. The performance of the proposed method was better than the performance of OS-ELM-RPLSM, OS-ELM-RLS, and RPLS in all data sets. The results also show that the adaptation of the number of latent variables leads to an improvement of the performance of the proposed method and that the time taken in each iteration of the method allows its online implementation in real-world applications.

Acknowledgment

This work was supported by Project SCIAD "Self-Learning Industrial Control Systems Through Process Data" (reference: SCIAD/2011/21531) co-financed by QREN, in the framework of the "Mais Centro - Regional Operational Program of the Centro", and by the European Union through the European Regional Development Fund (ERDF).

Tiago Matias and Rui Araújo acknowledge the support of FCT project PEst-C/EEI/UI0048/2011.

Francisco Souza has been supported by FCT under grant SFRH/BD/63454/ 2009.

References

[1] G.-B. Huang, N.-Y. Liang, H.-J. Rong, P. Saratchandran, N. Sundararajan, On-line sequential extreme learning machine, in: M. H. Hamza (Ed.), IASTED International Conference on Computational Intelligence, IASTED/ACTA Press, 2005, pp. 232–237.

[2] R. Hecht-Nielsen, Theory of the back propagation neural network, in: International Joint Conference on Neural Networks, 1989, pp. 593–605.

[3] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Networks 2 (5) (1989) 359–366.

[4] Y. Miche, A. Sorjamaa, P. Bas, O. Simula, C. Jutten, A. Lendasse, Op-elm: Optimally pruned extreme learning machine, IEEE Transactions on Neural Networks 21 (1) (2010) 158–162.

[5] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: Theory and applications, Neurocomputing 70 (1-3) (2006) 489–501.

[6] N.-Y. Liang, G.-B. Huang, P. Saratchandran, N. Sundararajan, A fast and accurate online sequential learning algorithm for feedforward networks, IEEE Transactions on Neural Networks 17 (6) (2006) 1411–1423.

[7] G.-B. Huang, L. Chen, C.-K. Siew, Universal approximation using incremental constructive feedforward networks with random hidden nodes, IEEE Transactions on Neural Networks 17 (4) (2006) 879–892.

[8] L. Ljung (Ed.), System Identification: Theory for the User, 2nd Edition, Prentice Hall PTR, Upper Saddle River, NJ, USA, 1999.

[9] S. J. Qin, Recursive pls algorithms for adaptive data modeling, Computers & Chemical Engineering 22 (4-5) (1998) 503–514.

[10] K. Helland, H. E. Berntsen, O. S. Borgen, H. Martens, Recursive algorithm for partial least squares regression, Chemometrics and Intelligent Laboratory Systems 14 (1-3) (1992) 129–137.

[11] D. C. Montgomery, G. C. Runger, Applied Statistics and Probability for Engineers, 3rd Edition, John Wiley & Sons, New York, NY, USA, 2003.

[12] J.-B. Yang, C.-J. Ong, Feature selection using probabilistic prediction of support vector regression, IEEE Transactions on Neural Networks 22 (6) (2011) 954–962.

[13] L. Fortuna, S. Graziani, A. Rizzo, M. G. Xibilia, Soft Sensors for Monitoring and Control of Industrial Processes, Springer, 2007.

[14] P. Kadlec, R. Grbić, B. Gabrys, Review of adaptation mechanisms for data-driven soft sensors, Computers & Chemical Engineering 35 (1) (2011) 1–24.

[15] P. Kadlec, B. Gabrys, Local learning-based adaptive soft sensor for catalyst activation prediction, AIChE Journal 57 (5) (2011) 1288–1301.

[16] M. Shoaib, M. Balahab, A. Abdel-Rahman, Influence of cement kiln dust substitution on the mechanical properties of concrete, Cement and Concrete Research 30 (2000) 371–377.

[17] M. Sadeghian, A. Fatehi, Identification of nonlinear predictor and simulator models of a cement rotary kiln by locally linear neuro-fuzzy technique, World Academy of Science, Engineering and Technology 58 (2009) 1121–1127.

[18] E. Romero, J. M. Sopena, Performing feature selection with multilayer perceptrons, IEEE Transations on Neural Networks 19 (3) (2008) 431–441.