Randomized Neural Networks for Recursive System Identification in the Presence of Outliers: A Performance Comparison

César Lincoln C. Mattos[1], Guilherme A. Barreto[1] and Gonzalo Acuña[2]

[1] Federal University of Ceará (UFC)
Department of Teleinformatics Engineering (DETI)
Center of Technology, Campus of Pici, Fortaleza, Ceará, Brazil,
cesarlincoln@terra.com.br, gbarreto@ufc.br,

[2] Universidad de Santiago de Chile (USACH)
Departamento de Ingeniería Informática, Santiago, Chile,
gonzalo.acuna@usach.cl

Abstract. In this paper, randomized single-hidden layer feedforward networks (SLFNs) are extended to handle outliers sequentially in online system identification tasks involving large-scale datasets. Starting from the description of the original batch learning algorithms of the evaluated randomized SLFNs, we discuss how these neural architectures can be easily adapted to cope with sequential data by means of the famed least mean squares (LMS). In addition, a robust variant of this rule, known as the *least mean M-estimate* (LMM) rule, is used to cope with outliers. Comprehensive performance comparison on benchmarking datasets are carried out in order to assess the validity of the proposed methodology.

Keywords: Randomized SLFNs, Online System Identification, NARX Model, Outliers.

1 Introduction

A novel class of supervised single-layer feedforward network (SLFN) architectures, generically called Randomized SLFNs, is attracting a great deal of attention from the computational intelligence community in recent years. A few examples of such architectures are the Random Vector Functional Link (RVFL) [15], the Extreme Learning Machine (ELM) [3], and the No-Prop network [14]. All this interest seems to be primarily motivated by the very fast way they are trained, without resorting to a long learning process across several training epochs, as required by the backpropagation algorithm. Even being a valid argument, many real-world applications present challenging features that demand adaptations in the learning algorithms of the aforementioned randomized SLFNs. One of such applications is online dynamical system identification from large scale datasets in the presence of outliers.

Dynamical system identification is a regression-like problem where the input and output observations come from time series data [1]. In other words, information about the dynamics (i.e. temporal behavior) of the system of interest must be learned from time series data. Despite the rapidly growing number of successful applications of randomized SLFNs in pattern recognition and regression, their use for nonlinear dynamical system identification has not been fully explored yet, with just a few works available [5, 12, 9]. In [5] and [12], for example, the proposed ELM-like models use *batch* learning algorithms based on the ordinary least-squares (OLS) estimation method, which cannot be applied to large scale datasets because it requires a costly matrix inversion and storage of huge data matrices. In [9], a recursive estimation algorithm is proposed aiming at online system identification problems, but despite alleviating the memory storage requirements by using chunks of data samples instead of the whole dataset, the proposed method is still too costly to be applied for large scale datasets.

In what concerns the robustness to outliers, it is widely known that the OLS method is very sensitive to their presence in the estimation data [4]. As a consequence, any randomized SLFN using the OLS method, such as the standard RVFL and ELM networks, will also present a severe degradation in performance when trained with outlier-contaminated data. For online learning in outlier-free scenarios, the No-Prop network [14] becomes a suitable alternative to the standard RVFL and ELM networks because it uses the least mean squares (LMS) rule instead of the OLS method for estimating the output weights. However, like the batch OLS, the adaptive LMS rule is also very sensitive to outliers, an issue that can be resolved by means of an outlier-robust version of it, named the *least-mean M-estimate* (LMM) algorithm [17].

From the exposed, due to the requirements of the applications we are interested in, we pursue randomized nonlinear models capable of fast online learning in large scale datasets *AND* in the presence of outliers. For this purpose, we incorporate into the aforementioned randomized SLFNs, the robust online learning ability of the LMM rule. This strategy is comprehensively evaluated on datasets generated by several benchmarking dynamical systems and shown to be effective. For the sake of completeness, we also introduce the LMM learning rule into the standard backpropagation algorithm in order to carry out a fair performance comparison.

The remainder of the paper are organized as follows. In Section 2 we describe all models to be evaluated in this paper, emphasizing the need for adaptive learning rules, such as the LMS rule, for online system identification. In Section 3 we show how to replace the original LMS rule with a robust variant by means of concepts from the M-estimation framework. A comprehensive performance comparison is presented in Section 4. The paper is concluded in Section 5.

2 Evaluated Models

Let us assume that we have already collected N data pairs $\{(\mathbf{x}_n, d_n)\}_{n=1}^{N}$ for building and evaluating the model, where $\mathbf{x}_n \in \mathbb{R}^p$ is the n-th p-dimensional

input pattern and $d_n \in \mathbb{R}$ is the corresponding target value. Then, let us randomly select N_1 ($N_1 < N$) training input-output pairs from the available data pool and arrange the input vectors along the columns of the matrix $\mathbf{X}$ ($p \times N_1$), while the target values are stacked into the column-vector $\mathbf{d}$ ($N_1 \times 1$):

$$\mathbf{X} = [\mathbf{x}_1 \mid \mathbf{x}_2 \mid \cdots \mid \mathbf{x}_{N_1}] \quad \text{and} \quad \mathbf{d} = [d_1 \; d_2 \; \cdots \; d_{N_1}]^T, \tag{1}$$

where the superscript T denotes the transpose of a vector/matrix.

2.1 The Random Vector Functional Link Network (RVFL)

The RVFL [15, 16] is a randomized SLFN with two pathways for processing information from input units to output neurons. These pathways are then added to form the network's output. The first pathway is a linear one, which directly connects the input units to the output neuron. Mathematically, we get

$$y_n^{(1)} = \mathbf{w}_1^T \mathbf{x}_n, \tag{2}$$

where $\mathbf{w}_1 \in \mathbb{R}^p$ is the corresponding weight vector[1]. The second pathway processes the input vectors through a hidden layer of q ($q \geq 1$) nonlinear neurons; that is,

$$y_n^{(2)} = \mathbf{w}_2^T \mathbf{h}_n, \tag{3}$$

where $\mathbf{w}_2 \in \mathbb{R}^q$ is the corresponding weight vector and $\mathbf{h}_n \in \mathbb{R}^q$ is the hidden activation vector, i.e. the vector containing the outputs of the hidden neurons in response to the current input vector $\mathbf{x}_n$. The vector $\mathbf{h}_n$ is computed as

$$\mathbf{h}_n = \phi(\mathbf{M}\mathbf{x}_n) = [\phi(\mathbf{m}_1^T \mathbf{x}_n + b_1), \ldots, \phi(\mathbf{m}_q^T \mathbf{x}_n + b_q)]^T, \tag{4}$$

where $\phi(\cdot)$ is a nonlinear (e.g. sigmoidal) activation function operating at each component of its argument vector, $\mathbf{M}$ is a $q \times p$ weight matrix, and b_j, $j = 1, .., q$, denotes the bias of the j-th hidden neuron. It should be noted that the weight vectors $\mathbf{w}_1$ and $\mathbf{w}_2$ are estimated from data, while the entries of the matrix $\mathbf{M}$ and the biases b_j are randomly sampled either from a uniform or a normal distribution.

If we add the outputs of both pathways, we get

$$y_n = y_n^{(1)} + y_n^{(2)} = \mathbf{w}_1^T \mathbf{x}_n + \mathbf{w}_2^T \mathbf{h}_n = [\mathbf{w}_1^T \mid \mathbf{w}_2^T] \begin{bmatrix} \mathbf{x}_n \\ - \\ \mathbf{h}_n \end{bmatrix} = \mathbf{w}^T \mathbf{z}_n, \tag{5}$$

where $\mathbf{w} = [\mathbf{w}_1^T \mid \mathbf{w}_2^T]^T$ is the $(p+q) \times 1$ vector obtained from the concatenation of the weight vectors $\mathbf{w}_1$ and $\mathbf{w}_2$. By the same token, $\mathbf{z}_n$ is the $(p+q) \times 1$ vector formed from the concatenation of the current input vector $\mathbf{x}_n$ and the current hidden activation vector $\mathbf{h}_n$.

[1] We assume that all vectors are column-vectors, unless stated otherwise.

The weight vector $\mathbf{w}$ can be readily estimated via the ordinary least squares (OLS) method by means of the following expression:

$$\mathbf{w} = (\mathbf{Z}\mathbf{Z}^T)^{-1}\mathbf{Z}\mathbf{d}, \tag{6}$$

where $\mathbf{Z} = [\mathbf{z}_1 \,|\, \mathbf{z}_2 \,|\, \cdots \,|\, \mathbf{z}_{N_1}]$ is a $(p+q) \times N_1$ matrix whose N_1 columns are the augmented vectors $\mathbf{z}_n = [\mathbf{x}_n^T \mid \mathbf{h}_n^T]^T \in \mathbb{R}^{p+q}$, $n = 1, \ldots, N_1$, where N_1 is the number of available training input patterns. The vector $\mathbf{d}$ is defined in Eq. (1). To avoid numerical problems, a regularized version of Eq. (6) is commonly used, which is given by

$$\mathbf{w} = (\mathbf{Z}\mathbf{Z}^T + \lambda\mathbf{I})^{-1}\mathbf{Z}\mathbf{d}, \tag{7}$$

where the constant $\lambda > 0$ is the regularization parameter.

2.2 The Extreme Learning Machine (ELM)

The ELM network is a recent randomized SLFN introduced by Huang *et al.* [3], whose weights from the inputs to the hidden neurons are randomly chosen, while only the weights from the hidden neurons to the output are analytically determined. Consequently, ELM offers significant advantages such as fast learning speed, ease of implementation, and less human intervention when compared to more traditional SLFNs, such as the Multilayer Perceptrons (MLP) and RBF networks.

From an architectural point of view, the ELM network can be understood as a simplified version of the RVFL in which the direct linear path is removed. Thus, the equations of the ELM are easily obtained as follows:

Output computation: From Eq. (5), once we remove the direct linear pathway, we get

$$y_n = y_n^{(2)} = \mathbf{w}_2^T\mathbf{h}_n, \tag{8}$$

where $\mathbf{h}_n$ is defined as in Eq. (4).

Estimation of $\mathbf{w}_2$: In this case, the expression of the OLS estimate in Eq. (6) reduces to

$$\mathbf{w}_2 = (\mathbf{H}\mathbf{H}^T)^{-1}\mathbf{H}\mathbf{d}, \tag{9}$$

where $\mathbf{H} = [\mathbf{h}_1 \,|\, \mathbf{h}_2 \,|\, \cdots \,|\, \mathbf{h}_{N_1}]$ be a $q \times N_1$ matrix whose N_1 columns are the hidden activation vectors $\mathbf{h}_n \in \mathbb{R}^q$, $n = 1, \ldots, N_1$, where N_1 is the number of available training input patterns.

2.3 Sequential Learning Rules for RVFL and ELM

In some applications, such as adaptive channel equalization and online system identification, adaptive learning rules are a better option, where the weight vector $\mathbf{w}$ is updated following the arrival of each input pattern. The input pattern is then discarded after being used for updating the parameters. One of such sequential learning rules is the well-known *least mean squares* (LMS) algorithm, also known as the Widrow-Hoff or Delta rule, which was used recently by Widrow

at al. [14] to introduce a randomized SLFN architecture, named *No-Propagation* (No-Prop) network. Basically, the No-Prop network is like an ELM network with output weights computed by means of a sequential learning rule.

In order to allow the RVFL network to process sequential data, we replace the standard OLS equation with the LMS rule. For this purpose, let us consider first the instantaneous cost function associated with the output neuron at the presentation of the n-th input vector:

$$J\left(\mathbf{w}_n\right) = \frac{1}{2}e_n^2 = \frac{1}{2}(d_n - y_n)^2 = \frac{1}{2}\left(d_n - \mathbf{w}_n^T\mathbf{h}_n\right)^2, \tag{10}$$

where $\mathbf{w}_n \in \mathbb{R}^{p+q}$ is the weight vector of the output neuron at iteration n, and $e_n = d_n - y_n$ is the instantaneous error of that neuron at iteration n. Then, in order to derive the LMS learning rule, we resort to a stochastic gradient descent recursive formula given by

$$\mathbf{w}_{n+1} = \mathbf{w}_n - \eta\frac{\partial J\left(\mathbf{w}_n\right)}{\partial \mathbf{w}_n} = \mathbf{w}_n + \eta e_n\mathbf{h}_n = \mathbf{w}_n + \eta(d_n - y_n)\mathbf{h}_n, \tag{11}$$

where $0 < \eta \ll 1$ is the learning rate. A widely used variant of the LMS rule, known as the normalized LMS (NLMS) algorithm [2], is given by

$$\mathbf{w}_{n+1} = \mathbf{w}_n + \frac{\eta}{\epsilon + \|\mathbf{h}_n\|^2}e_n\mathbf{h}_n = \mathbf{w}_n + \frac{\eta}{\epsilon + \mathbf{h}_n^T\mathbf{h}_n}e_n\mathbf{h}_n, \tag{12}$$

where ϵ is a very small positive constant needed to avoid division by zero. The strong points of the LMS and NLMS algorithms are ease of implementation and optimal performance under important practical conditions [13]. For these reasons, the LMS algorithm has enjoyed very widespread application in adaptive filtering and signal processing applications. For instance, it is used in almost every modem for channel equalization and echo canceling.

3 A Robust Learning Rule for RVFL and ELM

An important feature of both OLS and LMS rules is that they are derived from cost functions that assign the same importance to all error samples, i.e. all errors contribute the same way to the final solution. Hence, outliers tend to produce large errors and then degrade the parameter estimation process.

Bearing this in mind, a robust variant of the LMS rule, named the Least Mean M-Estimate (LMM) algorithm [17], has been introduced for the purpose of better dealing with outliers. The theory behind the LMM rule is provided by an elegant and principled estimation framework, known as M-estimation, introduced by Huber [4]. The letter M stands for "maximum likelihood" type, where robustness is achieved by minimizing a function distinct from the sum of the squared errors.

Based on Huber's theory, the instantaneous cost function to be minimized by the output neuron is now given by

$$J\left(\mathbf{w}_n\right) = \rho(e_n) = \rho(d_n - y_n) = \rho\left(d_n - \mathbf{w}_n^T\mathbf{h}_n\right), \tag{13}$$

where $\mathbf{w}_n \in \mathbb{R}^{p+q}$ is the weight vector of the output neuron at iteration n, and $e_n = d_n - y_n$ is the instantaneous error of that neuron at iteration n. The function $\rho(\cdot)$ should possess the following properties: (i) $\rho(e_n) \geq 0$; (ii) $\rho(0) = 0$; (iii) $\rho(e_n) = \rho(-e_n)$; and, (iv) $\rho(e_n) \geq \rho(e_{n'})$, for $|e_n| > |e_{n'}|$. For $\rho(e_n) = e_n^2/2$, we get the instantaneous cost function of the standard LMS rule shown in Eq. (10).

Thus, we develop a robust learning rule for the RVFL network as follows:

$$\begin{aligned}\mathbf{w}_{n+1} &= \mathbf{w}_n - \eta\frac{\partial J(\mathbf{w}_n)}{\partial \mathbf{w}_n} = \mathbf{w}_n - \eta\frac{\partial \rho(e_n)}{\partial \mathbf{w}_n} = \mathbf{w}_n - \eta\frac{\partial \rho(e_n)}{\partial e_n}\frac{\partial e_n}{\partial \mathbf{w}_n} \\ &= \mathbf{w}_n - \eta\frac{\partial \rho(e_n)}{\partial e_n}(-\mathbf{h}_n) = \mathbf{w}_n + \eta q(e_n) e_n \mathbf{h}_n, \end{aligned} \tag{14}$$

where $q(e_n) = \frac{1}{e_n}\frac{\partial \rho(e_n)}{\partial e_n}$ is called the weighting function. The normalized version of the LMM rule is then written as

$$\mathbf{w}_{n+1} = \mathbf{w}_n + \frac{\eta}{\epsilon + \mathbf{h}_n^T \mathbf{h}_n} q(e_n) e_n \mathbf{h}_n, \tag{15}$$

where ϵ has the same meaning as in Eq. (12). Note that if $\rho(e_n) = e_n^2/2$, then $q(e_n) = 1$, and Eq. (14) reduces to Eq. (11)

In this work, Hampel's three-part function [8] will be used, being defined as

$$\rho(e_n) = \begin{cases} e_n^2/2, & 0 \leq |e_n| < \xi \\ \xi|e_n| - \xi^2/2, & \xi \leq |e_n| < \Delta_1 \\ \frac{\xi}{2}(\Delta_1 + \Delta_2) - \frac{\xi^2}{2} + \frac{\xi}{2}\frac{(|e_n| - \Delta_2)^2}{\Delta_1 - \Delta_2}, & \Delta_1 \leq |e_n| < \Delta_2 \\ \frac{\xi}{2}(\Delta_1 + \Delta_2) - \frac{\xi^2}{2}, & \Delta_2 \leq |e_n| \end{cases}, \tag{16}$$

$$q(e_n) = \begin{cases} 1, & 0 \leq |e_n| < \xi \\ \frac{\xi}{e_n}\text{sign}(e_n), & \xi \leq |e_n| < \Delta_1 \\ \frac{\xi}{e_n}\text{sign}(e_n)\frac{|e_n| - \Delta_2}{\Delta_1 - \Delta_2}, & \Delta_1 \leq |e_n| < \Delta_2 \\ 0, & \Delta_2 \leq |e_n| \end{cases}, \tag{17}$$

where ξ, Δ_1, Δ_2 are user-defined thresholds which avoid the influence of inputs with large errors. As in [17], we use $\xi = 1.96\hat{\sigma}_n$, $\Delta_1 = 2.24\hat{\sigma}_n$ and $\Delta_2 = 2.576\hat{\sigma}_n$, where $\hat{\sigma}_n$ is the standard deviation of the output, estimated recursively.

4 Simulation and Results

In this section we report the results of the evaluation of four randomized SLFNs, namely: two variants of the RVFL network, named the RVFL-NLMS and the RVFL-NLMM, and two variants of the ELM network, named the ELM-NLMS and ELM-NLMM[2]. We also evaluate the performances of two variants of a single-hidden-layered MLP network trained with the backprop algorithm, using the tanh activation function for the hidden neurons and a linear activation function

[2] The second term in the name of the evaluated randomized SLFN denotes the online learning rule used to estimate the output weights.

for the output neuron. The variants of the MLP network differ in the way the weights of the output neuron are adjusted, with one using the LMS rule (MLP-LMS) and the other using the LMM rule (MLP-LMM).

The dynamics of the systems of interest are assumed to be described by a nonlinear autoregressive model with exogenous inputs (NARX) [1]:

$$d_n = f(d_{n-1},\ \cdots\ d_{n-L_y},\ u_{n-1},\ \cdots, u_{n-L_u}), \tag{18}$$

where L_u and L_y denote the input and output memory orders, respectively. The target function $f(\cdot) : \mathbb{R}^{L_y+L_u} \to \mathbb{R}$ is unknown and assumed to be nonlinear. Observed data, in the form of an input times series $\{u_n\}$ and an output time series $\{d_n\}_{n=1}^{N}$, are used to build an approximating model $\hat{f}(\cdot)$ for $f(\cdot)$.

Experiments were performed with an artificial and a real-world dataset. The *Artificial* dataset is generated by simulating the following dynamical system [7]:

$$d_n = \frac{d_{n-1}}{1 + d_{n-1}^2} + u_{n-1}^3, \tag{19}$$

where the training input time series is generated by sampling from an uniform distribution between -2 and 2 (i.e. $u_n \sim U(-2, 2)$), $n = 1, \ldots, 10000$, and the test input time series is given by $u_n = \sin(2\pi i/25) + \sin(2\pi i/10)$, $n = 1, \ldots, 100$. To the resulting output time series $\{d_n\}$, we add zero-mean Gaussian noise with variance equal to 0.65. We use $L_y = 1$ and $L_u = 1$.

The real-world dataset, named *Silver box* [10, 6], is an electronic nonlinear feedback laboratory experiment, which simulates a second order mechanical system with a nonlinear spring constant, acting as mass-spring-damper structure. The control input in the mechanical system is the force applied to the mass and its displacement is the output. The electrical circuit is excited with ten different realizations of odd random phase multisine, resulting in 91072 training samples. The test set contains 40000 samples generated with a filtered Gaussian sequence with increasing variance. The regressors' lags are fixed as $L_y = 10$ and $L_u = 10$.

For all the following experiments, time series data are normalized to zero mean and unit variance. All neural models were implemented from scratch using the *R* software, version 3.3.2, running on Ubuntu 16.04, installed in an Acer notebook, Core i7, 2.40GHz, 16GB RAM. We perform experiments with scenarios containing 0%, 5%, 10%, 15%, 20%, 25% and 30% of outliers. The outliers were sampled from $\sigma(\boldsymbol{d}) \times \mathcal{T}(0, 2)$, where $\sigma(\boldsymbol{d})$ is the standard deviation of the original training set and $\mathcal{T}(0, 2)$ is a Student-t distribution with zero mean and 2 degrees of freedom.

The models are trained in an online way, where the training samples are presented as they are made available, one after another, within a single full pass of the training data. The figure of merit of the evaluation is the root mean square error (RMSE) values for both one-step ahead (OSA) prediction, where predictions are made using the actual output samples in the regressors:

$$y_n = \hat{d}_n = \hat{f}(d_{n-1},\ \cdots\ d_{n-L_y},\ u_{n-1},\ \cdots, u_{n-L_u}), \tag{20}$$

and free simulation, where predicted output values are fed back in order to build the output regressor:

$$\begin{aligned} y_n = \hat{d}_n &= \hat{f}(y_{n-1}, \cdots y_{n-L_y}, u_{n-1}, \cdots, u_{n-L_u}), \\ &= \hat{f}(\hat{d}_{n-1}, \cdots \hat{d}_{n-L_y}, u_{n-1}, \cdots, u_{n-L_u}), \end{aligned} \tag{21}$$

where the "hat" symbol ^ denotes the predicted values.

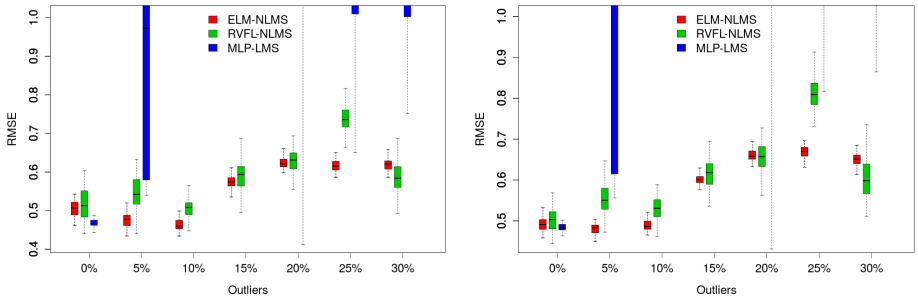
For each model the number of hidden units and the learning rate were optimized via Bayesian optimization [11] using the mean prediction RMSE within the 10-fold cross-validation performed in the outlier-free training data. We apply a linearly decaying learning rate, i.e., the rate for the n-th iteration is given by $\alpha_n = \alpha_1 + (\alpha_{N_1} - \alpha_1)\frac{n-1}{N_1-1}$, where N_1 is the number of training samples. Only the final learning rate α_{N_1} is optimized, while its initial value α_1 was fixed after preliminary experiments. In Table 1 we summarize the hyperparameters selected for each model. We emphasize that the hyperparameters selection step was executed only once per dataset using outlier-free data, but the adjustment of the models' parameters was performed separately in each contaminated scenario.

Table 1. Hyperparameters selected for each evaluated model: the number of hidden units q, the initial learning rate α_1 and the final learning rate α_{N_1}. α_1 was fixed after preliminary experiments. The other hyperparameters were determined via Bayesian optimization of the 10-fold cross-validation error in the outlier-free data.

	Artificial			**Silver box**		
	q	α_1	α_{N_1}	q	α_1	α_{N_1}
ELM-NLMS	967	1	2.52e-02	981	1	3.14e-05
ELM-NLMM	946	1	3.30e-02	997	1	7.20e-03
RVFL-NLMS	101	1	2.19e-03	708	1	3.91e-04
RVFL-NLMM	319	1	1.60e-03	672	1	2.84e-02
MLP-LMS	35	0.1	3.09e-03	31	0.01	9.98e-03
MLP-LMM	41	0.1	2.11e-03	43	0.01	9.88e-03

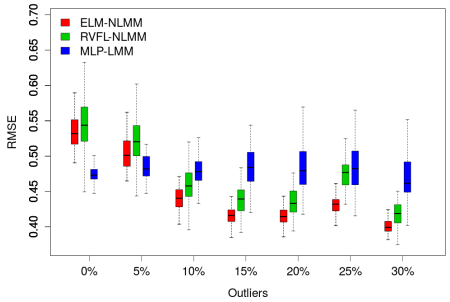
From this table, one can easily note that the RVFL variants require much smaller numbers of hidden neurons than the ELM variants. A possible explanation could rely on the direct linear pathway ($y_n^{(1)}$), which by capturing the linear part of the system dynamics let only the nonlinear (and more difficult) part of the dynamics to the nonlinear pathway ($y_n^{(2)}$).

The obtained RMSE values and the corresponding variances for the models using the *Artificial* dataset are shown in Fig. 1. Fig. 1.a stems for the standard algorithms in OSA predictions while Fig. 1.b shows mutiple-step-ahead predictions (free simulation). The performances of the robust versions of the algorithms are shown in Figs. 1.c (OSA) and 1.d (free simulations). For both kinds of prediction, results are rather consistent and show that no matter the % of contamination by outliers robust algorithms achieve better performances

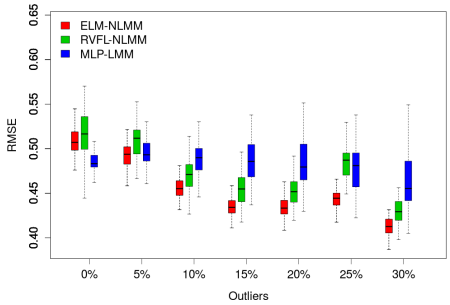


a) Standard models - OSA prediction.

b) Standard models - free simulation.



c) Robust models - OSA prediction.

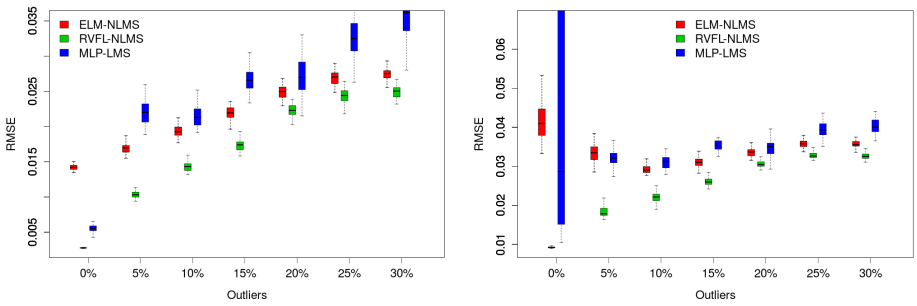


d) Robust models - free simulation.

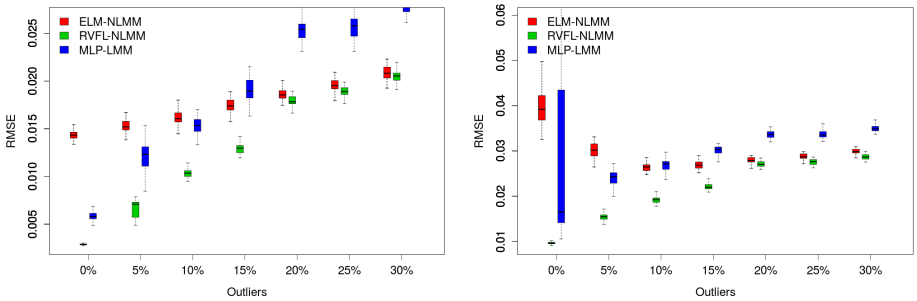
Fig. 1. Results for the *Artificial* dataset.

and are rather insensitive to the % of contamination. This is not the case for the standard algorithms that exhibit a certain degree of sensitivity to the % of outliers. In both kinds of prediction MLP-LMS show great variance for some of the % of contamination by outliers and in both cases the introduction of the corresponding robust algorithm greatly reduces these poor performances. In the case of robust algorithms, ELM-NLMM exhibits the best performance followed by RVFL-NLMM and MLP-LMM. There are no great differences in the corresponding performances of standard and robust algorithms depending on the kind of prediction (OSA or free simulation), except for the case of MLP-LMS.

Things are different for the *Silver box* dataset results shown in Fig. 2. In this case robust algorithms (Figs. 2.c and 2.d) do not exhibit clear better performances than the standard algorithms as it is the case for the *Artificial* dataset (Fig. 1). A slight reduction of the obtained RMSE and a more consistent and important reduction of the variances are the advantages shown by the robust algorithms. This is more evident in the case of MLP. The RMSE obtained for all models increased significantly when the contamination also increased, specially in the case of OSA predictions. Even robust models are not as insensitive to the % of outliers as in the case shown in Fig. 1.



a) Standard models - OSA prediction. b) Standard models - free simulation.



c) Robust models - OSA prediction. d) Robust models - free simulation.

Fig. 2. Results for the *Silver box* dataset.

Table 2. *p*-values computed via Wilcoxon signed-rank test for the residuals obtained in the best free simulation on the *Artificial* dataset with 30% of outliers. Red indicates statistically similar results.

	ELM-NLMM	RVFL-NLMS	RVFL-NLMM	MLP-LMS	MLP-LMM
ELM-NLMS	4.608945e-04	1.423818e-11	2.789759e-06	0.3039999415	1.054542e-06
ELM-NLMM		3.127407e-09	1.738496e-16	0.2769369016	1.007799e-10
RVFL-NLMS			1.314366e-02	0.0001062009	6.663213e-02
RVFL-NLMM				0.0086282427	7.283775e-01
MLP-LMS					3.664618e-03

An additional Wilcoxon signed-rank test for the residuals obtained in the best free simulation with 30% of outliers was performed. Results are reported in Tab. 2 for the *Artificial* dataset and in Tab. 3 for the *Silver box* dataset. It can be seen that for the *Silver box* dataset all models perform significantly different from each other and this is due to the small variance they exhibit. This is not the case for the *Artificial* dataset, where some models perform not significantly different from others. This is the case for ELM models which exhibit no significant difference from the MLP-LMS model. RVFL models also exhibit no significative difference with MLP-LMM model.

Table 3. p-values computed via Wilcoxon signed-rank test for the residuals obtained in the best free simulation on the *Silver box* dataset with 30% of outliers. All the results were statistically different.

	ELM-NLMM	RVFL-NLMS	RVFL-NLMM	MLP-LMS	MLP-LMM
ELM-NLMS	5.370802e-67	7.093561e-75	8.371801e-09	1.122239e-159	8.294698e-86
ELM-NLMM		1.036182e-03	9.285395e-100	1.466633e-41	3.493612e-11
RVFL-NLMS			2.397354e-68	1.904553e-30	9.663732e-02
RVFL-NLMM				9.958828e-160	3.048651e-115
MLP-LMS					4.956787e-63

As a final remark it can be said that even though results are not equally clearly interpretable for both datasets, the robust algorithms achieved consistently better performances. From the point of view of a reduced RMSE and variance for the *Artificial* dataset and from the point of view of a reduced variance in the case of the experiments performed with the *Silver box* dataset. In which concerns the % of outliers, the behavior of the robust algorithms was clearly insensitive to the increment of the contamination rate for the *Artificial* dataset, although that was not the case for the *Silver box* dataset. Overall, robust versions of ELM and RVFL consistently achieved the best performances.

5 Conclusions and Further Work

In this paper we tackled the task of online nonlinear system identification in the presence of outliers with randomized SLFNs. For that purpose, we decided to replace the original *batch* OLS-based learning rules of the RVFL and ELM networks with adaptive LMS-based ones, enabling recursive learning and training from larger datasets. Seeking resilience to outliers, a robust version of the LMS rule, known as LMM rule, was also considered as a learning algorithm.

We performed computational experiments with both an artificial and a real-world datasets using incremental levels of contamination by outliers. The achieved results are promising, with the evaluated robust randomized SLFNs being capable of fast learning of the system dynamics in an online fashion, i.e., without the need of multiple epochs, even in the presence of outliers.

However, despite the appealing results of our evaluation, further experiments are still needed in order to have a clear picture of the pros and cons of the proposed approach. For instance, hyperparameter optimization (i.e. number of hidden units and learning rates) remains an open issue, since it requires costly rounds of cross-validation. Furthermore, we were not able to obtain a strong resilience to outliers in the *Silver box* dataset, when compared to the good results we presented for the *Artificial* dataset.

In this regard, we continue to evaluate the proposed methodology in other benchmarking system identification datasets, as well as experimenting with other adaptive learning rules based, for instance, on the recursive least-squares (RLS) algorithm.

Acknowledgments

The first two authors thank the financial support of FUNCAP, CNPq (grant no. 309451/2015-9) and NUTEC. The third author acknowledges partial finantial support of Conicyt via Fondef Mineria Grant IT16M100008.

References

1. Billings, S.A.: Nonlinear System Identification NARMAX Methods in the Time, Frequency and Spatio-Temporal Domains. 1st edn. (2013)
2. Choi, Y.S., Shin, H.C., Son, W.J.: Robust regularization for normalized LMS algorithms. IEEE Transactions on Circuits and Systems-II 53(8), 627–631 (2006)
3. Huang, G., Huang, G.B., Song, S., You, K.: Trends in extreme learning machines: A review. Neural Networks 61(1), 32–48 (2015)
4. Huber, P.J., Ronchetti, E.M.: Robust Statistics. john Wiley & Sons, LTD (2009)
5. Li, M.B., Er, M.J.: Nonlinear system identification using extreme learning machine. In: Proceedings of the 9th International Conference on Control, Automation, Robotics and Vision (ICARCV)'06. pp. 1–4 (2006)
6. Marconato, A., Sjöberg, J., Suykens, J., Schoukens, J.: Identification of the silverbox benchmark using nonlinear state-space models. IFAC Proceedings Volumes 45(16), 632–637 (2012)
7. Narendra, K.S., Parthasarathy, K.: Identification and control of dynamical systems using neural networks. IEEE Transanctions on Neural Networks 1(1), 4–27 (1990)
8. Rousseeuw, P.J., Leroy, A.M.: Robust Regression and Outlier Detection. John Wiley & Sons (1987)
9. Salih, D.M., Noor, S.B.M., Merhaban, M.H., Kamil, R.M.: Wavelet network: Online sequential extreme learning machine for nonlinear dynamic systems identification. Advances in Artificial Intelligence 2015(184318), 1–10 (2015)
10. Schoukens, J., Nemeth, J.G., Crama, P., Rolain, Y., Pintelon, R.: Fast approximate identification of nonlinear systems. Automatica 39(7), 1267–1274 (2003)
11. Snoek, J., Larochelle, H., Adams, R.P.: Practical bayesian optimization of machine learning algorithms. In: Advances in neural information processing systems. pp. 2951–2959 (2012)
12. Tang, Y., Li, Z., Guan, X.: Identification of nonlinear system using extreme learning machine based Hammerstein model. Communications in Nonlinear Science and Numerical Simulation 19(9), 3171–3183 (2014)
13. Widrow, B.: Thinking about thinking: The discovery of the LMS algorithm. IEEE Signal Processing Magazine 22(1), 100–106 (2005)
14. Widrow, B., Greenblatt, A., Kim, Y., Park, D.: The No-Prop algorithm: A new learning algorithm for multilayer neural networks. Neural Networks 37, 182–188 (2013)
15. Y.-H. Pao, G.H.P., Sobajic, D.J.: Learning and generalization characteristics of the random vector functional-link net. Neurocomputing 6, 163–180 (1994)
16. Zhang, L., Suganthan, P.N.: A comprehensive evaluation of random vector functional link networks. Information Sciences 367–368, 1094–1105 (2016)
17. Zou, Y., Chan, S.C., Ng, T.S.: Least mean M-estimate algorithms for robust adaptive filtering in impulsive noise. IEEE Transactions on Circuits and Systems II 47(12), 1564–1569 (2000)