Online Sequential Extreme Learning Machine With Kernels

Simone Scardapane, Danilo Comminiello, Michele Scarpiniti, and Aurelio Uncini

***Abstract*— The extreme learning machine (ELM) was recently proposed as a unifying framework for different families of learning algorithms. The classical ELM model consists of a linear combination of a fixed number of nonlinear expansions of the input vector. Learning in ELM is hence equivalent to finding the optimal weights that minimize the error on a dataset. The update works in batch mode, either with explicit feature mappings or with implicit mappings defined by kernels. Although an online version has been proposed for the former, no work has been done up to this point for the latter, and whether an efficient learning algorithm for online kernel-based ELM exists remains an open problem. By explicating some connections between nonlinear adaptive filtering and ELM theory, in this brief, we present an algorithm for this task. In particular, we propose a straightforward extension of the well-known kernel recursive least-squares, belonging to the kernel adaptive filtering (KAF) family, to the ELM framework. We call the resulting algorithm the kernel online sequential ELM (KOS-ELM). Moreover, we consider two different criteria used in the KAF field to obtain sparse filters and extend them to our context. We show that KOS-ELM, with their integration, can result in a highly efficient algorithm, both in terms of obtained generalization error and training time. Empirical evaluations demonstrate interesting results on some benchmarking datasets.**

***Index Terms*— Extreme learning machine (ELM), kernel, online learning, recursive least square (RLS).**

I. Introduction

Over the last decade, extreme learning machine (ELM) theory [1]–[4] has gained increasing attention in the Machine Learning research community. From a theoretical perspective, ELM provides an interesting unified formulation to a wide range of learning models, including single hidden layer feedforward networks (SLFNs) [5], support vector machines (SVMs) [6], and regularization networks (RNs) [6]. The optimization problem arising in learning the parameters of an ELM model can be solved analytically, resulting in a closed form involving only matrix multiplication and inversion. Hence, the learning process can be carried out in an efficient way without requiring an iterative algorithm, such as backpropagation, or the solution to a quadratic programming problem as in the standard formulation of SVM. From a practical point of view, thus, ELM provides an effective family of learning models, whose parameters can be found in a more efficient way with respect to other approaches. Moreover, empirical simulations show that SVM and SLFN (trained with backpropagation) provide a suboptimal solution to many learning problems when compared with ELM [1], [7].

The classical ELM optimization problem is formulated in batch mode, considering all the available data at once. This approach puts some limitations on the number of inputs that can be processed by the algorithm, mainly due to the computational cost and spatial requirements needed for inverting a matrix. In addition, in some cases data may arrive sequentially, and in this context, it would be better to work with a sequential algorithm instead of collecting data before processing. For these reasons, several extensions have been proposed to enhance the ELM robustness with respect to irrelevant data and to extend its capabilities in an online sequential setting. For example, Liang *et al*. [8] proposed an online version of ELM, called online sequential ELM (OS-ELM), that deals with data in a sequential fashion, one by one, or using chunks of (possibly) different sizes. Miche *et al.* [9] investigated a way of pruning the nodes of the network and derived the optimally pruned ELM (OP-ELM). Despite the simplicity of the algorithm, OP-ELM was found to be in many cases faster and more accurate than its classical counterpart, especially when dealing with redundant data, showing the importance that effective *sparsification* of the obtained model can have in a real-world implementation of ELM techniques (sparse data classification in ELM is instead explored in [10] and [11]). Previously to OP-ELM, pruning of an ELM network was explored in the fast pruned-ELM (P-ELM) [12], using statistical tests and only in the classification case. Other variants of the standard ELM model that have investigated similar issues are the convex incremental ELM (CI-ELM) [13], and the OS fuzzy ELM (OS-Fuzzy-ELM) [14].

The above works focused on the case of an explicit feature mapping. Only recently Huang *et al.* [1] extended ELM using kernel functions, explicating the connections with SVM and other kernel methods, and derived what we denote as the kernel ELM (K-ELM). Thus, no studies have been made regarding online ELM learning with the use of kernels, which up to now remains an open problem in the ELM theory. To this end, we propose to extend OS-ELM to the case of implicit feature mappings. Since the OS-ELM formulation is very similar to the recursive least square (RLS) algorithm, in this brief, we show how the algorithm known as kernel RLS (KRLS), belonging to the kernel adaptive filtering (KAF) family [15], can be used efficiently in this situation.[1] To be consistent with ELM notation, we call our resulting algorithm the kernel OS-ELM (KOS-ELM). Since most kernel models grow linearly with the number of processed inputs, computational time is the major issue to be confronted and solved in our context [16]. Hence, this problem needs to be addressed to consider KOS-ELM really useful in a realistic context. Here, we again exploit the connections with the KAF world, where this problem has been extensively studied. In particular, we provide the application of two specific criteria for the sparsification of input patterns, namely, the approximate linear dependency (ALD) [15] and the fixed-budget (FB) [17].

A noncomprehensive list of the families of ELM models previously discussed is summarized in Fig. 1, where our contribution is highlighted in boldface red, whereas some specific variations of the basic models are shown between brackets. Deformed K-ELM (DK-ELM) is presented in [18]. We refer to the

Manuscript received September 5, 2013; revised May 26, 2014, September 4, 2014, and November 15, 2014; accepted December 11, 2014. Date of publication December 31, 2014; date of current version August 17, 2015.

The authors are with the Department of Information Engineering, Electronics and Telecommunications, Sapienza University of Rome, Rome 00185, Italy (e-mail: simonescardapane@gmail.com; danilo.comminiello@uniroma1.it; michele.scarpiniti@uniroma1.it; aurel@ieee.org).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2014.2382094

[1]For the interested reader, the connections between the online solution of regularized cost functions and adaptive filters have been already explored in previous works [16].

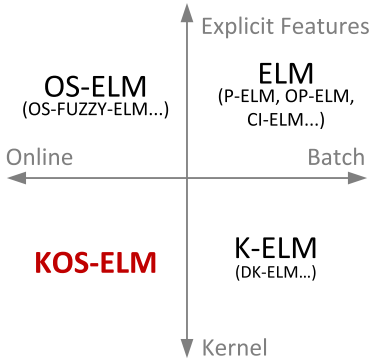

Fig. 1. Noncomprehensive list of ELM models. The contribution of this brief is highlighted in boldface red.

original papers for details on the domains in which each algorithm has been successfully applied: (ELM and K-ELM) [1], (OS-ELM) [8], (OP-ELM) [9], (P-ELM) [12], (CI-ELM) [13], (OS-Fuzzy-ELM) [14], (DK-ELM) [18]. Clearly, these are only a subset of all the ELM variations, which have been proposed recently. Because a complete survey of them goes beyond the scope of this brief, we refer the reader to [2] for an entry point into this growing body of literature.

We expect KOS-ELM to be beneficial in contexts where K-ELM works better than ELM [18], but one of the following situations occur.

1) The computational cost of the matrix inversion required by K-ELM is too large.
2) Data is arriving in a streaming fashion (time-series prediction).

We present examples of both these situations in our simulations. The experimental results show that KOS-ELM, with the incorporation of a sparsification procedure, is able to obtain comparable or higher accuracy and training time with respect to the other members of the ELM family in the datasets we considered.

The rest of this brief is organized as follows. In Section II, we outline the basis of ELM theory. The main contribution of this brief, the KOS-ELM algorithm, is presented in Section III, while in Section IV, we provide a brief description of the two sparsification criteria used in our experiments. Section V details some experimental results, and finally, Section VI outlines future lines of work.

II. Extreme Learning Machine

Consider the classical learning problem [5] of estimating an unknown relation between elements of an *input space* $X \subseteq \mathbb{R}^d$ and elements of an *output space*[2] $Y \subseteq \mathbb{R}$, whose relation is fully described by the unknown joint probability distribution $p(\mathbf{x}, y), \mathbf{x} \in X, y \in Y$. Given a class $H = \{f : X \to Y\}$ of possible models, the goal of learning is finding a good *predictive* function $f^* \in H$ such that $f^*(\mathbf{x}) \approx y$, given N independent and identically distributed samples $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ drawn from the distribution. The set S is known as the *dataset* and each element as an *example*.

An ELM [1] is a linear combination of L activation functions

$$f(\mathbf{x}) = \sum_{i=1}^{L} h_i(\mathbf{x})\beta_i = \mathbf{h}^T(\mathbf{x})\boldsymbol{\beta} \tag{1}$$

where $\mathbf{h}(\mathbf{x}) = [h_1(\mathbf{x}), \ldots, h_L(\mathbf{x})]^T$ is called the *ELM feature vector*. Thus, ELM has the classical three-layered structure of SLFNs. In Fig. 2, we show a simple graphical depiction of an ELM with one input and a single output. Differently from conventional SLFNs, however, the activation functions of ELM are considered fixed during training (with no tunable parameters), so the learning problem is reduced to that of estimating the optimal weight vector $\boldsymbol{\beta}$. Parameters of any standard activation function can be generated randomly from a uniform probability distribution. The main result of the ELM theory (see [1]) is that almost any nonlinear, piecewise continuous function can be used in (1), and the resulting network will have universal approximation capabilities.

[2]For simplicity, we consider here the case where all elements are real and we have a single output. The formulation can be easily extended to other situations.

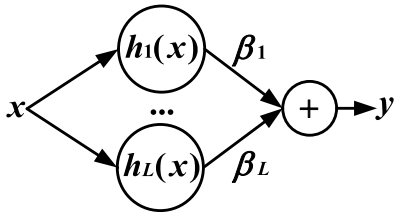

Fig. 2. Architecture of a basic ELM model. For simplicity, a single input and output are shown.

The optimal weight vector is given by the solution to the L2-regularized optimization problem [1]

$$\begin{aligned} &\min_{\boldsymbol{\beta}} \ \frac{1}{2}\|\boldsymbol{\beta}\|_2^2 + \frac{C}{2}\sum_{i=1}^{N}\zeta_i^2 \\ &\text{s.t. } \mathbf{h}^T(x_i)\boldsymbol{\beta} = y_i - \zeta_i, \quad i = 1, \ldots, N \end{aligned} \tag{2}$$

where C is a regularization parameter that can be adjusted by the user and $\zeta_i, i = 1, \ldots, N$ are *slack variables* that measure the error between desired and predicted output. Equation (2) is similar to the classical optimization problem of SVM [6], but has simpler constraints (equalities instead of inequalities) and is valid for regression, binary, and multiclass cases. Moreover, while SVM learning results in a quadratic optimization problem, (2) has a solution in closed form

$$\boldsymbol{\beta} = \mathbf{H}^T\left(\frac{1}{C}\mathbf{I}_N + \mathbf{H}\mathbf{H}^T\right)^{-1}\mathbf{Y} \tag{3}$$

where we defined the hidden matrix $\mathbf{H} = [\mathbf{h}(\mathbf{x}_1), \ldots, \mathbf{h}(\mathbf{x}_N)]$, the output vector $\mathbf{Y} = [y_1, \ldots, y_N]^T$, and $\mathbf{I}_N$ is the $N \times N$ identity matrix.

Connections with SVM and other kernel methods become apparent when, instead of an explicit feature vector $\mathbf{h}(\mathbf{x})$, we consider a *kernel function* $k(\mathbf{x}, \mathbf{y}) = \langle \mathbf{h}(\mathbf{x}), \mathbf{h}(y) \rangle$, where $\langle \cdot, \cdot \rangle$ represents the inner product in the ELM feature space. Function $k(\cdot, \cdot)$ is named *ELM kernel*, and is a kernel by definition. By explicating the *kernel matrix* $\boldsymbol{\Omega} = \mathbf{H}\mathbf{H}^T : \Omega_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$, the ELM model can be rewritten as

$$f(\mathbf{x}) = [k(\mathbf{x}, \mathbf{x}_1), \ldots, k(\mathbf{x}, \mathbf{x}_N)]^T\left(\frac{1}{C}\mathbf{I}_N + \boldsymbol{\Omega}\right)^{-1}\mathbf{Y}. \tag{4}$$

Equation (4) is similar to the SVM model, but provides a closed-form expression for the kernel coefficients. The resulting formulation is closely related to the RN, as described for example in [6] and [10]. Thus, ELM can be seen as connected to SVMs, SLFNs, and RNs. Differently from the first two models, however, it provides a more efficient solution to the computation of the model parameters [1].

A. Online Sequential ELM

To ease the computation of the inverse matrix in (3), and to extend ELM to the online setting, Liang *et al.* [8] proposed a sequential learning algorithm, called *OS-ELM*. Suppose that the dataset is

presented in successive *mini-batches* S_i, $i = 1, \ldots, B$, such that $S = \bigcup_{i=1}^{B} S_i$. We denote by $\mathbf{H}_i$ and $\mathbf{Y}_i$ the hidden matrix and output vector restricted to the mini-batch i. A first approximation of the weights is computed from the first mini-batch

$$\mathbf{P}_1 = \left(\mathbf{H}_1^T \mathbf{H}_1\right)^{-1} \tag{5}$$

$$\boldsymbol{\beta}^{(1)} = \mathbf{P}_1 \mathbf{H}_1^T \mathbf{Y}_1. \tag{6}$$

For the inverse to exist, we require that $|S_1| \geq rank(\mathbf{H})$, where $|\cdot|$ denotes the cardinality of a set. At each new batch k, then weights are updated as

$$\mathbf{P}_{k+1} = \mathbf{P}_k - \mathbf{P}_k \mathbf{H}_{k+1}^T \left(\mathbf{I} + \mathbf{H}_{k+1} \mathbf{P}_k \mathbf{H}_{k+1}^T\right)^{-1} \mathbf{H}_{k+1} \mathbf{P}_k \tag{7}$$

$$\boldsymbol{\beta}^{(k+1)} = \boldsymbol{\beta}^{(k)} + \mathbf{P}_{k+1} \mathbf{H}_{k+1}^T (\mathbf{Y}_{k+1} - \mathbf{H}_{k+1} \boldsymbol{\beta}^{(k)}). \tag{8}$$

As pointed out in [8], the algorithm is equivalent to the known RLS applied to the vector $\mathbf{h}(\mathbf{x})$ and, for this reason, also possesses all its convergence properties [20].

III. Kernel OS-ELM

Over the last years, many attempts have been made at extending the capabilities of linear filters such as the RLS to general nonlinear mappings [21]. The simplest solution is to carry out the filtering process in two stages: 1) a nonlinear transformation of the input and 2) a subsequent linear filtering. If the input to the first step is a buffer of the last M elements of the signal $\mathbf{x}$: $\mathbf{x}_n = (x[n], x[n-1], \ldots, x[n-M+1])^T$, and we denote the nonlinearity as $\mathbf{h}(\cdot)$, the resulting model is

$$y[n] = \mathbf{h}^T(\mathbf{x}_n) \boldsymbol{\beta}_n. \tag{9}$$

By the equivalence of (1) and (9), the connection between online ELM and nonlinear adaptive filtering becomes noticeable. In particular, if the ELM feature vector is given by the nonlinear transformation applied to the signal and the RLS is used to adjust the weights, the resulting adaptive filtering algorithm is equivalent to OS-ELM.

An important class of nonlinear adaptive filters is the KAF family [15]. Although they also consider a nonlinear transformation of the original input, such transformation is never computed explicitly, but only through the use of a kernel function (similar to the difference between SLFN and SVM). Due to the properties of the underlying reproducing kernel hilbert space associated with the kernel, KAF algorithms exhibit interesting properties of convergence and have low computational requirements. For this reason, they are one of the most successful learning models dealing with kernels. RLS has an equivalent formulation in the KAF domain known as KRLS [15].

By the above discussion, it should be clear how the KRLS can be extended straightforwardly to OS-ELM learning with a generic ELM kernel. The result is a sequential online approximation of model in (4) that, in accordance with ELM terminology, we call KOS-ELM. In Table I, we report the pseudocode of the algorithm, adapted from [15]. We refer to [15] and [22] for a full derivation and an analysis of KRLS convergence properties that hold equivalently in this context. The parameters of the algorithm are the kernel function to be used, and a regularization parameter C. At each new example $(\mathbf{x}_i, y_i)$, the algorithm stores a new unit with center $\mathbf{x}_i$ and $r_i^{-1} e_i$ as its coefficient, where e_i denotes the error of the filter before the update. Moreover, the algorithm updates at the same time all the previous coefficients by a factor $-\mathbf{z}_i r_i^{-1} e_i$. Note that differently from OS-ELM, the algorithm is defined for inputs coming one at a time, since it needs to compute a new center for each of them. From a computational perspective, the time complexity of OS-ELM is related to inverting a $P \times P$ matrix, where P is the size of the mini-batch, hence between $\mathcal{O}(P^2)$ and $\mathcal{O}(P^3)$. Instead, the time complexity of KOS-ELM at iteration i is on the order of $\mathcal{O}(i^2)$, although this complexity can decrease by considering the sparsification criteria presented in the next section.

TABLE I
KOS-ELM Pseudocode

Input: A kernel function $k(\mathbf{x}, \mathbf{y})$, and a regularization parameter C.
Output: The expansion weights $\boldsymbol{\beta}$ computed after the last example.

$\mathbf{Q}_1 = (C^{-1} + k(\mathbf{x}_1, \mathbf{x}_1)^{-1})$
$\boldsymbol{\beta}_1 = \mathbf{Q}_1 y_1$
for $i = 2$ to N **do**
  $\mathbf{k}_i = [k(\mathbf{x}_i, \mathbf{x}_1), \ldots, k(\mathbf{x}_i, \mathbf{x}_{i-1})]^T$
  $\mathbf{z}_i = \mathbf{Q}_{i-1} \mathbf{k}_i$
  $r_i = C^{-1} + k(\mathbf{x}_i, \mathbf{x}_i) - \mathbf{z}_i^T \mathbf{k}_i$
  $\mathbf{Q}_i = r_i^{-1} \begin{bmatrix} \mathbf{Q}_{i-1} r_i + \mathbf{z}_i \mathbf{z}_i^T & -\mathbf{z}_i \\ -\mathbf{z}_i^T & 1 \end{bmatrix}$
  $e_i = y_i - \mathbf{k}_i^T \boldsymbol{\beta}_{i-1}$
  $\boldsymbol{\beta}_i = \begin{bmatrix} \boldsymbol{\beta}_{i-1} - \mathbf{z}_i r_i^{-1} e_i \\ r_i^{-1} e_i \end{bmatrix}$
end for
return $\boldsymbol{\beta}_N$

IV. Sparsification of KOS-ELM

It was already hinted in Section I that the ability of handling irrelevant and redundant data is of fundamental importance in ELM. This is more acute in the case of online kernel methods, whose models grow linearly with the number of examples that are presented to them. For this reason, the problem of *sparsification* of the filters, i.e., the problem of actively discarding nodes and selecting inputs to be processed, has been extensively studied in the KAF field.

The problem can be stated formally as follows. At iteration n, in general, we will have processed and stored a set D of Q inputs $D = \{\mathbf{x}_i\}_{i=1}^{Q}$, known as the *dictionary*, which is a subset of the original training data $\{\mathbf{x}_1, \ldots, \mathbf{x}_{n-1}\}$. A *constructive* sparsification criterion performs a given sequence of tests to choose whether the new input pair $(\mathbf{x}_n, y_n)$ should be processed or not to eliminate redundant data and enhance the generalization capabilities of the model. Ideally, such a test should also guarantee that the overall size of the network is bounded in the stationary case. On the contrary, a *pruning* criterion adds the new input without testing and, subsequently, discards inputs from the new dictionary if deemed necessary.

An example of the former case is the ALD criterion [15]. Using ALD, the input $\mathbf{x}_n$ is rejected if its distance to the linear span of the current dictionary in the feature space is less than a certain threshold δ, specified *a priori*

$$\text{dis} = \min_{\boldsymbol{\alpha}} \left\| \mathbf{h}(\mathbf{x}_n) - \sum_{\mathbf{x}_j \in D} \alpha_j \mathbf{h}(\mathbf{x}_j) \right\|_2 < \delta \tag{10}$$

where $\mathbf{h}(\cdot)$ denotes the feature space associated to the kernel and $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_Q]^T$ is the vector of coefficients of the linear

TABLE II
GENERAL DESCRIPTION OF THE DATASETS

Dataset name	Input features	Instances	Desired output	Task type
Sylva	216	14394	Forest Type (Ponderosa pine vs. everything else)	Classification
Calhousing	8	20640	Prediction of Median House Valu	Regression
WDBC	30	569	Diagnosis of Breast Cancer	Classification
Mackey-Glass	7	20000	Prediction of time-series	Regression

combination. It turns out that in KRLS, (10) can be simplified greatly [15]

$$\text{dis}^2 = k(\mathbf{x}_n, \mathbf{x}_n) - \mathbf{k}_n^T \boldsymbol{\Omega}_D^{-1} \mathbf{k}_n \tag{11}$$

where $\boldsymbol{\Omega}_D$ is the kernel matrix restricted to the dictionary and can be estimated efficiently [15].

An example of a pruning sparsification criterion, instead, is the FB criterion developed in [17]. The algorithm considers a maximum dimension F of the network. If the new input makes the size of the dictionary larger than F, the least significant pattern is removed from the dictionary, where significance is defined in terms of the error on the new pattern after removal. After the pruning step, the new Gram matrix is recomputed using a simple procedure based on the matrix inversion lemma. We refer to [17] for additional details.

In the following, we refer to KOS-ELM with the inclusion of the aforementioned criteria as ALD-KOS-ELM and FB-KOS-ELM, respectively. Although we have chosen only two examples of sparsification criteria, an ample literature exists on the topic, with more methods being published every month. Clearly, an exhaustive survey of them is beyond the scope of this brief. We refer to [23] and [24] for more recent studies in this context. Finally, we observe that by using a sparsification criteria, time complexity of KOS-ELM is effectively reduced to $\mathcal{O}(Q^2)$.

V. EXPERIMENTAL EVALUATION

A. Setup

To show the validity of the proposed algorithm, in this section, we present a comparison of ELM, K-ELM, OS-ELM, and the two variants of the KOS-ELM algorithm on four selected datasets. The raw KOS-ELM algorithm, without sparsification criteria, is excluded due to its very high training time. In all cases, to compute performance, we averaged over a stratified 10-fold cross validation on the available dataset, except for the Wisconsin database of breast cancer (WDBC) dataset, where we executed a three-fold cross validation due to the smaller size. All experiments are then averaged over 20 different runs. We tested K-ELM and KOS-ELM with a Gaussian kernel

$$k(\mathbf{x}, \mathbf{y}) = \exp(-\gamma \|\mathbf{x} - \mathbf{y}\|^2) \tag{12}$$

where γ is a parameter known as the *kernel bandwidth*. The Gaussian kernel of (12) belongs to the class of *universal* kernels that grants kernel methods with the universal approximation capability (provided a sufficiently large regularization parameter C). As a comparison, we tested ELM and OS-ELM using sigmoid additive activation functions

$$g(\mathbf{x}) = \frac{1}{1 + e^{-(\mathbf{a}\mathbf{x}+b)}} \tag{13}$$

where the parameters $\mathbf{a} \in \mathbb{R}^d, b \in \mathbb{R}$ of the function are extracted randomly from the uniform distribution $[-1, 1]$. Inputs to OS-ELM were presented by 200, and the algorithm was initialized with an initial batch of 250 elements. The two sparsification criteria were adapted from the Kernel Methods Toolbox [25]. Parameters of the algorithms were found by performing a grid search and measuring performance by an additional stratified three-fold cross validation on the training data. The following intervals were searched.

1) For K-ELM, the regularization parameter C and the bandwidth γ of the Gaussian kernel were searched in $\{2^{-8}, 2^{-4}, \ldots, 2^{13}\}$. These two parameters are shared with KOS-ELM.
2) For ELM, the regularization parameter was searched in the same interval detailed before, while hidden nodes in the interval from 100 to 1000 (by steps of 50). OS-ELM shares the same configuration of ELM.
3) The threshold δ of ALD was searched uniformly in $\{0.05, 0.1, \ldots, 0.8\}$, while the maximum size F of the fixed-budget criterion in $\{50, 100, \ldots, 400\}$.

TABLE III
OPTIMAL PARAMETERS FOUND BY THE GRID-SEARCH PROCEDURE, AVERAGED OVER THE DIFFERENT RUNS

Dataset	ELM	K-ELM	FB-KOS-ELM	ALD-KOS-ELM
Sylva	$L = 460$ $C = 40$	$\gamma = 0.03$ $C = 32$	$F = 200$	$\delta = 0.15$
CalHousing	$L = 750$ $C = 1024$	$\gamma = 5$ $C = 200$	$F = 400$	$\delta = 0.08$
WDBC	$L = 300$ $C = 192$	$\gamma = 0.3$ $C = 98$	$F = 200$	$\delta = 0.65$
Mackey-Glass	$L = 750$ $C = 1024$	$\gamma = 4$ $C = 1012$	$F = 300$	$\delta = 0.1$

The name of the datasets, number of input features, and expected output are described in Table II. We note that prior to training, the inputs to the system were normalized between 0 and 1. The averaged values of the optimal parameters found by the grid-search procedure are in Table III. All simulations were performed by MATLAB 2013a, on an Intel Core i3 3.07 GHz processor at 64 bit with 16 GB of RAM, using the Lynx MATLAB toolbox.[3] Detailed instructions to repeat them are available on the corresponding author's website.[4] Next, we analyze in detail the results on each dataset.

B. Results

In the first experiment, we have used the Sylva binary classification dataset,[5] which has several peculiar characteristics: 1) it is rather big (more than 14000 examples); 2) it has a large number of distracting features; and 3) it has a marked class imbalance, with only 6.15% of positive examples. Results of this experiment are shown in the first row of Table IV. Due to the class imbalance, we use the matthew correlation coefficient (MCC) [26] as a performance measure, since it is more robust to the skewness of the dataset. The MCC ranges

[3] https://github.com/ispamm/Lynx-Toolbox/

[4] http://ispac.ing.uniroma1.it/scardapane/software/lynx/online-kelm/

[5] http://www.causality.inf.ethz.ch/al_data/SYLVA.html

TABLE IV
EXPERIMENTAL RESULTS FOR THE TWO CLASSIFICATION DATASETS. WE SHOW THE AVERAGE AND STANDARD DEVIATION OF THE MCC, AND THE AVERAGE TRAINING TIME BETWEEN BRACKETS (IN SECONDS). BEST RESULTS ARE IN BOLD

	ELM	K-ELM	OS-ELM	FB-KOS-ELM	ALD-KOS-ELM
Sylva	0.88 ± 0.003 (0.20)	0.94 ± 0.002 (17.42)	0.88 ± 0.003 (11.98)	$\mathbf{0.97 \pm 0.004}$ (14.69)	0.92 ± 0.004 (9.23)
WDBC	0.93 ± 0.012 (0.01)	0.95 ± 0.011 (0.01)	0.74 ± 0.025 (0.05)	$\mathbf{0.98 \pm 0.005}$ (0.30)	0.89 ± 0.016 (0.07)

TABLE V
CONFUSION MATRICES FOR THE SYLVA DATASET, RELATIVE TO THE ELM, K-ELM, AND FB-KOS-ELM ALGORITHMS. RESULTS ARE AVERAGED OVER THE DIFFERENT RUNS, AND ROUNDED TO THE NEAREST INTEGER

			Actual class: Negative	Actual class: Positive
Predicted class	Negative	ELM	1345	6
		K-ELM	1346	5
		KOS-ELM	1350	2
	Positive	ELM	12	77
		K-ELM	5	84
		KOS-ELM	1	88

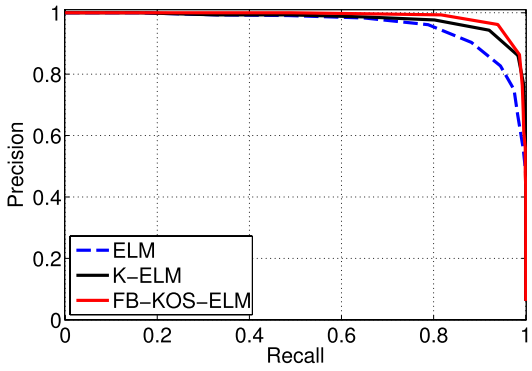

Fig. 3. Precision–recall curve for the Sylva dataset.

in $[-1, 1]$, with 1 indicating the perfect correlation between true class and output of the algorithm, and -1 the perfect negative correlation. Probably due to the aforementioned characteristics, K-ELM has a higher performance with respect to ELM, being on average 5% more correlated with the desired output. This is obtained at the cost of a larger training time: slightly more than 17 s against $1/5$ of a second for ELM. This shows that, even if ELM is one of the fastest learning models to train, in some situations, it can be suboptimal with respect to its kernel counterpart in terms of classification accuracy. OS-ELM reaches a similar performance to ELM, while FB-KOS-ELM obtains a higher MCC score with respect to K-ELM, and it is in total 9% higher than the standard ELM MCC score. Moreover, it is also able to leverage on the required training time of K-ELM, being 15% faster in average. ALD-KOS-ELM is in between, scoring better than ELM, but lower than K-ELM, although it needs only 9 s to train. Therefore, if our main aim is testing accuracy, FB-KOS-ELM would be our first choice in this situation, while ALD-KOS-ELM provides a good compromise between accuracy and required training time. To further strengthen this point, we show in Table V the confusion matrices for ELM, K-ELM, and FB-KOS-ELM. We can see that ELM has a high rate of false negatives that are almost absent for FB-KOS-ELM, and hence, the superior performance. This is also shown in Fig. 3, where we plotted the precision–recall curve [27] for the three algorithms. We see that the FB-KOS-ELM curve, shown in red, dominates the other two curves, signifying that the algorithm is outperforming ELM and K-ELM in terms of classification accuracy.

Even in situations in which KOS-ELM is not able to surpass the testing accuracy of K-ELM, it can be a valid alternative to it due to its lower training time, especially for very large datasets. Even worse, in some situations, the available hardware may not be enough to perform the matrix inversion required by K-ELM. We explore this situation in our second experiment, where we use the Calhousing dataset taken from the StatLib repository.[6] This is a regression task, whose dataset is larger than the previous task, containing more than 20 000 examples. Results of this simulation are presented in Table VI, where we use the normalized root mean-squared error (NRMSE) as a performance measure, defined for a testing set T as

$$\text{NRMSE}(T) = \sqrt{\frac{\sum_{(\mathbf{x}_i, y_i) \in T} (f(\mathbf{x}_i) - y_i)^2}{|T| \hat{\sigma}_y^2}} \tag{14}$$

where $|T|$ denotes the cardinality of the set T and $\hat{\sigma}_y^2$ is an empirical estimate of the variance of $\mathbf{y}$. We see that in this situation, K-ELM has a 10% decrease in testing error with respect to ELM, but the difference in training time is more pronounced than in the previous experiment: more than 43 s against 0.47 s of ELM. Moreover, in a computer with a similar configuration but a lower amount of RAM (4 GB), MATLAB was not able to perform the matrix inversion due to the excessive requirements in terms of memory storage. In this experiment, we observe from Table VI that ALD-KOS-ELM is able to obtain equivalent performance with respect to K-ELM, using only a third of its training time, slightly more than 16 s. Moreover, there is no need of storing the full kernel matrix, since the final network was always restricted to a few hundred elements. In this experiment, OS-ELM presented several problems of numerical instability, and for this reason, we decided to exclude it from the results. In addition, we also note that the FB criterion is working relatively poorly here, with a large increase in training time and testing error. Using $F = 800$, we were able to obtain a similar performance with respect to K-ELM, although training time was around 100 s. This exemplifies that in some datasets, different sparsification criteria can have unequal performance. However, in our experience, in typical machine learning tasks in general, it is possible to obtain very good results using one of the two criteria we detailed here.

Similar results are obtained for the second regression dataset, where the task is predicting the value of the Mackey-Glass time series, given the last seven samples of the series. The series is generated according to the procedure detailed in [24]. ALD-KOS-ELM is the best performing algorithm in this context

[6] http://www.dcc.fc.up.pt/~ltorgo/Regression/cal_housing.html

TABLE VI
EXPERIMENTAL RESULTS FOR THE CALHOUSING AND MACKEY-GLASS DATASETS. WE SHOW THE AVERAGE AND STANDARD DEVIATION OF THE NRMSE, AND THE AVERAGE TRAINING TIME BETWEEN BRACKETS (IN SECONDS). BEST RESULTS ARE IN BOLD

	ELM	K-ELM	OS-ELM	FB-KOS-ELM	ALD-KOS-ELM
Calhousing	0.53 ± 0.001 (0.47)	$\mathbf{0.48 \pm 0.001}$ (43.55)	$- - -$	0.82 ± 0.019 (87.30)	$\mathbf{0.48 \pm 0.002}$ (16.34)
Mackey-Glass	0.10 ± 0.01 (0.43)	0.06 ± 0.02 (41.26)	0.15 ± 0.03 (118.82)	0.11 ± 0.03 (30.13)	$\mathbf{0.05 \pm 0.01}$ (10.82)

TABLE VII
CONFUSION MATRICES FOR THE WDBC DATASET, RELATIVE TO THE ELM, K-ELM, AND FB-KOS-ELM ALGORITHMS. RESULTS ARE AVERAGED OVER THE DIFFERENT RUNS, AND ROUNDED TO THE NEAREST INTEGER

			Actual class: Positive	Actual class: Negative
Predicted class	Positive	ELM	65	6
		K-ELM	67	4
		KOS-ELM	69	2
	Negative	ELM	2	118
		K-ELM	2	118
		KOS-ELM	0	120

(with a marked improvement with respect to the standard ELM), and its training time is approximately $1/4$ with respect to that of the batch K-ELM.

Superior performance of KOS-ELM can be obtained even on small-sized datasets, possibly due to its robustness to uninformative patterns and outliers. In these cases, however, it is difficult to obtain significantly lower training times, since the single matrix inversion of K-ELM can be performed very efficiently for small kernel matrices. To show this, in our last experiment, we use the classification problem known as WDBC taken from the UCI repository.[7] It contains 569 examples distributed evenly in two classes, each one described by 30 features. The examples are extracted from real-world cases, and it is shown that only a small subset of the features is useful for classification purposes [28]. Results of the experiment are presented in the second row of Table IV, using the MCC as a performance measure. In terms of classification accuracy, it has a similar outcome with respect to the Sylva dataset. K-ELM works better than ELM, with a 2% increase in correlation with the output. FB-KOS-ELM is the best scoring algorithm, with an additional 3% increase with respect to K-ELM. Both OS-ELM and ALD-KOS-ELM have inferior performances here. As expected, we see that for small datasets, the use of KOS-ELM introduces an overhead in terms of computational time. However, we believe this is probably negligible at this order of magnitude (no algorithm took more than a second to train). To elaborate on this, we also show the confusion matrices for ELM, K-ELM, and FB-KOS-ELM in Table VII. As was the case for the previous classification experiment, we see that FB-KOS-ELM is able to reduce both false positives and false negatives, while in this case, ELM and K-ELM achieve similar performance.

VI. CONCLUSION

In this brief, we have presented an algorithm for online learning with a kernel-based ELM that we called KOS-ELM. In particular, the algorithm is a straightforward extension of the KRLS taken from the KAF family. Besides filling a gap in the ELM theory, KOS-ELM may provide a useful learning tool in situations where kernel-based methods perform well and an OS setting is required, or the dataset is too large for a single matrix inversion. Moreover, we hope that this brief may help in strengthening the connections between the Machine Learning and the Adaptive Filtering communities, in both of which kernel algorithms are in widespread use [29], and in providing a bridge for the circulation of additional results. As an example, we are eager to test time-variant versions of KOS-ELM in dynamic scenarios on which ELM is beginning to be adopted [30]. In fact, criteria such as the FB are especially designed to be performant in these situations, which are more recurrent in adaptive filtering applications.

[7] http://archive.ics.uci.edu/ml/index.html

ACKNOWLEDGMENT

The authors would like to thank Prof. G.-B. Huang for the helpful comments and support. They would also like to thank the anonymous reviewers for their highly appreciated suggestions.

REFERENCES

[1] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 42, no. 2, pp. 513–529, Apr. 2012.
[2] G.-B. Huang, D. H. Wang, and Y. Lan, "Extreme learning machines: A survey," *Int. J. Mach. Learn. Cybern.*, vol. 2, no. 2, pp. 107–122, Jun. 2011.
[3] G.-B. Huang, L. Chen, and C.-K. Siew, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *IEEE Trans. Neural Netw.*, vol. 17, no. 4, pp. 879–892, Jul. 2006.
[4] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, nos. 1–3, pp. 489–501, Dec. 2006.
[5] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2004.
[6] T. Evgeniou, M. Pontil, and T. Poggio, "Regularization networks and support vector machines," *Adv. Comput. Math.*, vol. 13, no. 1, pp. 1–50, Apr. 2000.
[7] S. Scardapane, D. Comminiello, M. Scarpiniti, and A. Uncini, "Music classification using extreme learning machines," in *Proc. 8th Int. Symp. Image Signal Process. Anal. (ISPA)*, Trieste, Italy, Sep. 2013, pp. 377–381.
[8] N.-Y. Liang, G.-B. Huang, P. Saratchandran, and N. Sundararajan, "A fast and accurate online sequential learning algorithm for feedforward networks," *IEEE Trans. Neural Netw.*, vol. 17, no. 6, pp. 1411–1423, Nov. 2006.
[9] Y. Miche, A. Sorjamaa, P. Bas, O. Simula, C. Jutten, and A. Lendasse, "OP-ELM: Optimally pruned extreme learning machine," *IEEE Trans. Neural Netw.*, vol. 21, no. 1, pp. 158–162, Jan. 2010.
[10] S. Suresh, R. V. Babu, and H. J. Kim, "No-reference image quality assessment using modified extreme learning machine classifier," *Appl. Soft Comput.*, vol. 9, no. 2, pp. 541–552, Mar. 2009.
[11] S. Suresh, S. Saraswathi, and N. Sundararajan, "Performance enhancement of extreme learning machine for multi-category sparse data classification problems," *Eng. Appl. Artif. Intell.*, vol. 23, no. 7, pp. 1149–1157, Oct. 2010.

[12] H.-J. Rong, Y.-S. Ong, A.-H. Tan, and Z. Zhu, "A fast pruned-extreme learning machine for classification problem," *Neurocomputing*, vol. 72, nos. 1–3, pp. 359–366, Dec. 2008.
[13] G.-B. Huang and L. Chen, "Convex incremental extreme learning machine," *Neurocomputing*, vol. 70, nos. 16–18, pp. 3056–3062, Oct. 2007.
[14] H.-J. Rong, G.-B. Huang, N. Sundararajan, and P. Saratchandran, "Online sequential fuzzy extreme learning machine for function approximation and classification problems," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 39, no. 4, pp. 1067–1072, Aug. 2009.
[15] J. C. Príncipe, W. Liu, and S. Haykin, *Kernel Adaptive Filtering: A Comprehensive Introduction*. Hoboken, NJ, USA: Wiley, Mar. 2010.
[16] J. Kivinen, A. J. Smola, and R. C. Williamson, "Online learning with kernels," *IEEE Trans. Signal Process.*, vol. 52, no. 8, pp. 2165–2176, Aug. 2004.
[17] S. Van Vaerenbergh, I. Santamaria, W. Liu, and J. C. Príncipe, "Fixed-budget kernel recursive least-squares," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, Dallas, TX, USA, Mar. 2010, pp. 1882–1885.
[18] C. Zhang, X. S. Xia, and B. Liu, "Deformed kernel based extreme learning machine," *J. Comput.*, vol. 8, no. 6, pp. 1602–1609, Jun. 2013.
[19] S. Scardapane, D. Comminiello, M. Scarpiniti, and A. Uncini, "GP-based kernel evolution for L2-regularization networks," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Beijing, China, Jul. 2014, pp. 1674–1681.
[20] A. H. Sayed, *Fundamentals of Adaptive Filtering*. Hoboken, NJ, USA: Wiley, Jun. 2003.
[21] A. Uncini, *Fundamentals of Adaptive Signal Processing*. New York, NY, USA: Springer-Verlag.
[22] Y. Engel, S. Mannor, and R. Meir, "The kernel recursive least-squares algorithm," *IEEE Trans. Signal Process.*, vol. 52, no. 8, pp. 2275–2285, Aug. 2004.
[23] B. Chen, S. Zhao, P. Zhu, and J. C. Príncipe, "Quantized kernel recursive least squares algorithm," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 9, pp. 1484–1491, Sep. 2013.
[24] W. Liu, I. Park, and J. C. Príncipe, "An information theoretic approach of designing sparse kernel adaptive filters," *IEEE Trans. Neural Netw.*, vol. 20, no. 12, pp. 1950–1961, Dec. 2009.
[25] S. Van Vaerenbergh, "Kernel methods for nonlinear identification, equalization and separation of signals," Ph.D. dissertation, Univ. Cantabria, Cantabria, Spain, Feb. 2010.
[26] P. Baldi, S. Brunak, Y. Chauvin, C. A. F. Andersen, and H. Nielsen, "Assessing the accuracy of prediction algorithms for classification: An overview," *Bioinformatics*, vol. 16, no. 5, pp. 412–424, May 2000.
[27] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009.
[28] R. Setiono, "Generating concise and accurate classification rules for breast cancer diagnosis," *Artif. Intell. Med.*, vol. 18, no. 3, pp. 205–219, Mar. 2000.
[29] K.-R. Müller, T. Adali, K. Fukumizu, J. Principe, and S. Theodoridis, "Special issue on advances in kernel-based learning for signal processing," *IEEE Signal Process. Mag.*, vol. 30, no. 4, pp. 14–15, Jul. 2013.
[30] Y. Ye, S. Squartini, and F. Piazza, "Online sequential extreme learning machine in nonstationary environments," *Neurocomputing*, vol. 116, pp. 94–101, Sep. 2012.