Engineering Applications of Artificial Intelligence 44 (2015) 101–110

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

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

Kalman filter-based method for Online Sequential Extreme Learning Machine for regression problems

Jarley Palmeira Nobrega *, Adriano L.I. Oliveira

[a] Centro de Informática, Universidade Federal de Pernambuco (UFPE), Cidade Universitária, 50740-560 Recife, PE, Brazil

ARTICLE INFO

Article history:
Received 12 December 2014
Received in revised form
13 April 2015
Accepted 19 May 2015
Available online 6 June 2015

Keywords:
Online sequential learning
Extreme learning machine
Online Sequential Extreme Learning Machine
Kalman filter regression
Multicollinearity

ABSTRACT

In this paper, a new sequential learning algorithm is constructed by combining the Online Sequential Extreme Learning Machine (OS-ELM) and Kalman filter regression. The Kalman Online Sequential Extreme Learning Machine (KOSELM) handles the problem of multicollinearity of the OS-ELM, which can generate poor predictions and unstable models. The KOSELM learns the training data one-by-one or chunk-by-chunk by adjusting the variance of the output weights through the Kalman filter. The performance of the proposed algorithm has been validated on benchmark regression datasets, and the results show that KOSELM can achieve a higher learning accuracy than OS-ELM and its related extensions. A statistical validation for the differences of the accuracy for all algorithms is performed, and the results confirm that KOSELM has better stability than ReOS-ELM, TOSELM and LS-IELM.

1. Introduction

There has been growing interest in the development of machine learning methods for intelligently recognizing complex patterns even when all training data is not available. Over the past few decades, a large number of batch learning algorithms have been discussed and studied thoroughly, including the classical learning methods based on Backpropagation (BP) (Ruhmelhart et al., 1986) and Support Vector Machine (SVM) (Drucker et al., 1997). These methods can approximate complex nonlinear functions from the input samples without knowing the internal structure of the training data. However, it is known that the classical methods require a long time for tuning the parameters, and the learning process can be inefficient when the amount of training data is very large. Retraining the model using all of the data with the batch learning algorithms is a time consuming task which is unacceptable for many real-world applications that require fast learning speed and high accuracy. To overcome the weakness of traditional neural networks, Extreme Learning Machine (ELM) was introduced by Huang et al. (2006) to this end. One advantage of ELM over traditional neural networks is that it is not necessary to adjust parameters iteratively. Thus, ELM has a much faster training process with better performance in some cases when compared with traditional neural networks (Huang et al., 2015). This has been a motivation for the application of ELM to many practical problems, for instance, image classification (Luo and Zhang, 2014), multi-step-ahead time series prediction (Grigorievskiy et al., 2014) and fault section identification in transmission lines (Malathi et al., 2011).

In order to decrease the training time for batch algorithms, new classes of sequential learning methods were introduced (Platt, 1991; Kadirkamanathan and Niranjan, 1993; Yingwei et al., 1998; Huang et al., 2005). In some applications, these methods are preferred over batch algorithms as they do not need to retrain the model whenever a new training sample is received. In Liang et al. (2006), a sequential version of ELM has been developed to deal with the problem where training samples are received one-by-one or chunk-by-chunk. The Online Sequential Extreme Learning Machine (OS-ELM) is based on a single layer feedforward neural network (SLFN), in which the input weights and hidden layer biases are chosen randomly and the output weights are determined by the pseudo-inverse of the hidden layer matrix. The output weight estimation of ELM can be expressed as an ordinary least squares (OLS) solution. However, the OLS estimator can generate poor predictions in the presence of multicollinearity due to their large variance which decreases the model stability (Foucart, 1999).

To overcome this problem, some related work should be mentioned. In Huynh and Won (2011), the ReOS-ELM is

* Corresponding author.
E-mail addresses: jpn@cin.ufpe.br (J.P. Nobrega), alio@cin.ufpe.br (A.L.I. Oliveira).

http://dx.doi.org/10.1016/j.engappai.2015.05.010
0952-1976/

proposed with the use of a bi-objective optimization to obtain the small norm of the output weights. In this work, the problem of multicollinearity is controlled by applying a regularization parameter to update the weights. However, this method has some drawbacks. First, there is an additional parameter to adjust, the regularization constant, which makes the output of the algorithm dependent on its correct estimation. In addition, the optimization of the output weights is also dependent on the size of the sequential chunk data. In Gu et al. (2014), the stability and the convergence of the model are improved by the use of an adaptive scheme to update the output weights. This method, namely TOSELM, calculates the mean and variance of the sequential training data and adjusts the output weights by using an exponential function of the distribution of the data. In spite of the results reported in this work there has been a decrease indicated in the variance. This approach increases the computational complexity of the sequential learning algorithm since an additional iterative method is applied during the regular cycle of learning. Moreover, the stopping condition for the weight adjustment requires an additional parameter for the tolerance of the convergence, and this parameter must be tuned before the sequential training phase. Two similar approaches have been recently presented in Ye et al. (2013) and Guo et al. (2014). In the first work, the input and output weights are changing over time for both training and testing phases. The proposed OS-ELM-TV does not directly handled the problem of multicollinearity, but it reduces the variance of the estimations by using an output basis function to calculate a linear combination of the weights. This work present a fast training step and a stable model, despite the fact that the limitation of this approach is the correct choice of the output basis function, such as Fourier or Legendre functions, which demands an additional optimization step. The second approach, namely LS-IELM, has also an additional parameter to adjust, which is defined as a cost constant. This approach is similar to the ReOS-ELM, since the cost constant can be considered as a regularization parameter for the least squares solution of OS-ELM. In the LS-IELM, the problem of multicollinearity is not controlled directly since it updates the output weights by using the same recursive procedure of OS-ELM. A limitation of this method is the requirement of the optimization of the cost constant using a portion of the training data.

The limitations of the cited related work indicate that the problem of multicollinearity should be addressed without the optimization of additional parameters since they increase the computational complexity of the algorithm. It is known in the literature that the adjustment of the variance for an OLS based model can be modeled as a state space problem (Watson, 1983). In this context, a Kalman filter regression model could be applied to adjust the output weights of the OS-ELM with no additional parameters to be optimized. Since the Kalman filter applies a recursive least squares (RLS) solution in a similar way as the OS-ELM, an adjustment coefficient could be updated for each cycle of the sequential learning step. At the end of this process, the output weights are adjusted to avoid the effects of multicollinearity in the variance of the estimations.

The main motivation behind this paper is to investigate the limitations of OS-ELM and the related extensions with respect to the problem of multicollinearity. Based on a state space model, this paper proposes an improvement of OS-ELM, namely Kalman Online Sequential Extreme Learning Machine (KOSELM), in which the Kalman filter regression is applied to update the estimates of the output weights by adjusting its variance. In KOSELM, we use the capacity of the Kalman filter to handle the multicollinearity and adjust the variance of the estimated state. We perform a filtering procedure for the estimated output weight matrix during the update cycle. Next, the estimation is adjusted through the regression coefficient provided by the filtering procedure.

The main contributions of this paper are enumerated as follows:

1. The proposed KOSELM algorithm to the sequential learning problem in both one-by-one and chunk-by-chunk modes.
2. A weight update scheme using the Kalman filter regression for variance adjustment of the output weights.
3. An empirical comparison between the KOSELM and the related work for benchmark regression problems. Additionally, a statistical validation was performed for the differences of the accuracy of the studied algorithms.

This paper is organized as follows. Section 2 gives a brief overview of ELM, OS-ELM and Kalman filter. In Section 3, the proposed method for variance adjustment is presented. In Section 4, the description of the sample data, the experimental settings and the statistical validation of KOSELM are presented. Section 5 concludes the paper.

2. Fundamentals

In this section, the concepts of the batch version of ELM, the OS-ELM and Kalman filter regression are briefly reviewed.

2.1. Extreme Learning Machine

Unlike the traditional learning algorithms for neural networks, the main characteristic of ELM is learning without iterative training as proposed by Huang et al. (2006). This makes the learning process faster when compared to the traditional algorithms for the training of neural networks. Let the training set be $\{(\mathbf{x}_i, \mathbf{t}_i)\}, \mathbf{x}_i \in \mathbb{R}^n, \mathbf{t}_i \in \mathbb{R}^m, i = 1, \ldots, N$, where $\mathbf{x}_i$ is an $n \times 1$ input vector and $\mathbf{t}_i$ is a $m \times 1$ target vector. The training process is briefly described as follows.

Step 1: Randomly assign values to the inputs weights and hidden neuron biases, a_i and b_i.

Step 2: The output weights are analytically determined through the generalized inverse operation of the hidden layer matrices (Huang et al., 2006), according to the following equation:

$$\sum_{i=1}^{L} \beta_i G(\mathbf{a}_i, b_i, \mathbf{x}_j) = \mathbf{t}_j, \quad j = 1, \ldots, N \tag{1}$$

where a_i are the input weights, b_i are the hidden layer biases, $\boldsymbol{\beta}_i$ is the output weight that connects the ith hidden node and the output node, and G is the activation function. L is the number of hidden neurons. N is the number of distinct input or output data. This is equivalent to $\mathbf{H}\boldsymbol{\beta} = \mathbf{T}$, where

$$\mathbf{H} = \begin{bmatrix} G(\mathbf{a}_1, b_1, \mathbf{x}_1) & \cdots & G(\mathbf{a}_L, b_L, \mathbf{x}_L) \\ \vdots & \ddots & \vdots \\ G(\mathbf{a}_1, b_1, \mathbf{x}_N) & \cdots & G(\mathbf{a}_L, b_L, \mathbf{x}_N) \end{bmatrix}_{N \times L}, \tag{2}$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_L^T \end{bmatrix}_{L \times m}, \tag{3}$$

$$\mathbf{T} = \begin{bmatrix} \mathbf{T}_1^T \\ \vdots \\ \mathbf{T}_L^T \end{bmatrix}_{N\times m}. \quad (4)$$

H is the hidden layer output matrix. The activation function $G(x)$ should be assigned before training is carried out. For instance, the sigmoid function is one the most commonly used. It is given by

$$G(\mathbf{a}_i, b_i, \mathbf{x}_i) = \left(1 + e^{-(\mathbf{a}_i\mathbf{x}_i + b_i)}\right)^{-1}, \quad i = 1, \ldots, N \quad (5)$$

Step 3: In order to calculate the output weight, it is necessary to find the least-squares solution for $\boldsymbol{\beta}$ from the linear system $\mathbf{H}\boldsymbol{\beta} = \mathbf{T}$. In most cases, the number of hidden nodes L is much less than the number of samples N, and a non-square matrix **H** may not exist such as $\mathbf{H}\boldsymbol{\beta} = \mathbf{T}$. According to Huang et al. (2006), the smallest norm least-squares solution for that linear system can be calculated by the following expression:

$$\boldsymbol{\beta} = \mathbf{H}^{\dagger}\mathbf{T} \quad (6)$$

where $\mathbf{H}^{\dagger}$ is the Moore–Penrose generalized inverse of **H** (Rao and Mitra, 1971). There are several ways to calculate the Moore–Penrose generalized inverse of a matrix, including orthogonal projection method, orthogonalization method and singular value decomposition (SVD) (Rao and Mitra, 1971). On most implementations of ELM, the SVD can always be used to calculate the Moore–Penrose generalized inverse of **H**.

The parameters of hidden neuron $\mathbf{a}_i$ and b_i do not need to be adjusted, since they are randomly assigned in the first step of the training phase. Eq. (6) can be seen as a linear system and it can be expressed in terms of its least squares solution according to the following equation:

$$\boldsymbol{\beta} = \mathbf{H}^{\dagger}\mathbf{T} = \left(\mathbf{H}^{\mathbf{T}}\mathbf{H}\right)^{-1}\mathbf{H}^{\mathbf{T}}\mathbf{T} \quad (7)$$

2.2. Online Sequential Extreme Learning Machine

Sequential learning algorithms are designed for time-varying system dynamics by constructing the model with a variable structure (Huang et al., 2012). This class of algorithms often generates a variable structure at each step upon learning samples sequentially, and the parameters are tuned accordingly. The sequential learning algorithm for ELM is originated from the resource allocation network (RAN) (Platt, 1991) algorithm which learns samples sequentially and adds neurons accordingly at each step. OS-ELM has been applied to many problems and simulation results indicate that it produces better generalization performance with lower training time compared with other sequential learning algorithms such as the resource allocation network (RAN) and its extensions (Huang et al., 2011).

The training process of OS-ELM consists of two phases: (*i*) an initialization phase and (*ii*) a sequential learning phase (Liang et al., 2006). In the initialization phase, a small portion of the training data feed the algorithm to estimate an initial matrix H_0 and the initial output weight matrix $\boldsymbol{\beta}^{(0)}$. According to Huang et al. (2006), the number of data required to estimate $\mathbf{H}_0$ should be at least equal to the number of hidden nodes, since the ELM algorithm can learn L distinct samples with zero error when the number of hidden nodes is equal to L. However, in real applications the number of hidden nodes is always less than the number of training samples, and the training error cannot be zero, but it can be approximated to a non-zero training error ϵ. The rank of $\mathbf{H}_0$ is equal to the number of hidden nodes if the first training data are distinct.

After the initialization, the learning phase starts either on a one-by-one or a chunk-by-chunk mode. Once an instance of data is used in the update of the output weight matrix, it is discarded and not used any more. The training process for OS-ELM is briefly described as follows.

Step 1 – *Initialization phase*: Let $\aleph_0 = [(\mathbf{x}_i, \mathbf{t}_i)]_{i=1}^{N_0}$ be a subset of the training set $\aleph = \{(\mathbf{x}_i, \mathbf{t}_i) | \mathbf{x}_i \in \mathbb{R}^n, \mathbf{t}_i \in \mathbb{R}^m | i = 1, \ldots, L\}, N_0 \geq L$. Assign random input weights $\mathbf{a}_i$ and bias b_i for sigmoid activation function. For RBF hidden nodes $\mathbf{a}_i$ and b_i is the center and impact factor, respectively. Next, calculate the initial hidden layer output matrix $\mathbf{H}_0$:

$$\mathbf{H}_0 = \begin{bmatrix} G(\mathbf{a}_1, b_1, \mathbf{x}_1) & \cdots & G(\mathbf{a}_L, b_L, \mathbf{x}_1) \\ \vdots & \ddots & \vdots \\ G(\mathbf{a}_1, b_1, \mathbf{x}_{N_0}) & \cdots & G(\mathbf{a}_L, b_L, \mathbf{x}_{N_0}) \end{bmatrix}_{N_0\times L}. \quad (8)$$

Next, use Eq. (7) to estimate the initial output weight matrix $\boldsymbol{\beta}^{(0)} = \mathbf{P}_0\mathbf{H}_0^T\mathbf{T}$, where $\mathbf{P}_0 = \left(\mathbf{H}_0^T\mathbf{H}_0\right)^{-1}$ and $\mathbf{T}_0 = \left[\mathbf{t}_1, \ldots, \mathbf{t}_{N_0}\right]^T$. At the end of this step, let $k=0$.

Step 2 – *Sequential learning phase*: When the new sequence of data is available the sequential phase begins as follows. Let the $(k+1)$th block of data

$$\aleph_{k+1} = \{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=\left(\sum_{j=0}^{k} N_j\right)+1}^{\sum_{j=0}^{k+1} N_j},$$

where N_{k+1} is the number of observations in the $(k+1)$th block of data. Next, calculate the partial hidden layer output matrix $\mathbf{H}_{k+1}$ for the $(k+1)$th block $\aleph_{k+1}$, according to the following expression:

$$\mathbf{H}_{k+1} = \begin{bmatrix} G\left(\mathbf{a}_1, b_1, \mathbf{x}_{\left(\sum_{j=0}^{k} N_j\right)+1}\right) & \cdots & G\left(\mathbf{a}_L, b_L, \mathbf{x}_{\left(\sum_{j=0}^{k} N_j\right)+1}\right) \\ \vdots & \ddots & \vdots \\ \left(\mathbf{a}_1, b_1, \mathbf{x}_{\left(\sum_{j=0}^{k+1} N_j\right)}\right) & \cdots & G\left(\mathbf{a}_L, b_L, \mathbf{x}_{\left(\sum_{j=0}^{k+1} N_j\right)}\right) \end{bmatrix}_{N_{k+1}\times L} \quad (9)$$

Next, set $\mathbf{T}_{k+1}$ according to the following equation:

$$\mathbf{T}_{k+1} = \left[\mathbf{t}_{\left(\sum_{j=0}^{k} N_j\right)+1,\ldots}, \mathbf{t}_{\left(\sum_{j=0}^{k+1} N_j\right)}\right]^T \quad (10)$$

and calculate the output weight $\boldsymbol{\beta}^{(k+1)}$:

$$\mathbf{M}_{k+1} = \mathbf{M}_k - \mathbf{M}_k\mathbf{H}_{k+1}^T\left(\mathbf{I} + \mathbf{H}_{k+1}\mathbf{M}_k\mathbf{H}_{k+1}^T\right)^{-1}\mathbf{H}_{k+1}\mathbf{M}_k \quad (11)$$

$$\boldsymbol{\beta}^{(k+1)} = \boldsymbol{\beta}^{(k)} + \mathbf{M}_{k+1}\mathbf{H}_{k+1}^T\left(\mathbf{T}_{k+1} - \mathbf{H}_{k+1}\boldsymbol{\beta}^{(k)}\right). \quad (12)$$

Finally, let $k = k+1$ and repeat the Step 2.

The sequential implementation of this least squares solution has the same convergence performance as the recursive least squares (RLS) solution (Chong and Zak, 2013).

2.3. Kalman filter regression

The Kalman filter for regression can be described as a recursive method to estimate the state of a dynamic system from a series of incomplete and noisy measurements (Harvey, 1990). The state representation of the dynamics for the time-varying regression

coefficients is given by the following system of equations:

$$\mathbf{T}_t = \mathbf{H}_t \mathbf{x}_t + \varepsilon_t \tag{13}$$

$$\mathbf{x}_t = \mathbf{A}_{t-1} \mathbf{x}_{t-1} + \omega_t \tag{14}$$

where $\mathbf{T}_t$ is the dependent variable, $\mathbf{x}_t$ is a time-varying regression coefficient, $\mathbf{A}_t$ is the transition matrix of the dynamic model, and $\mathbf{H}_t$ is the independent variable at time t, respectively. ε_t and ω_t are independent uncorrelated error terms with standard variances σ_ε^2 and σ_ω^2, respectively. Eq. (13) is also called the measurement equation and Eq. (14) the state equation, which defines the regression coefficient as a simple random walk. Here, the variances of the noise process and other unknown parameters have to be estimated. This is accomplished by maximizing the following likelihood function:

$$\log\ L(\theta) = -\frac{1}{2}\sum_{t=1}^{T} \ln|\mathbf{F}_t| - \frac{1}{2}\sum_{t=1}^{T} \upsilon_t^T \mathbf{F}_t^{-1} \upsilon_t \tag{15}$$

where θ is the set of parameters to be estimated, υ_t is the one-step ahead residual vector and $\mathbf{F}_t$ is the covariance matrix of the innovations at time t. N and T are the number of columns of $\mathbf{H}_t$ and the number of elements of $\mathbf{T}_t$, respectively. When the Kalman filter is applied, the parameters A, ε_t and ω_t are estimated by maximizing Eq. (15) with a numerical algorithm based on ω_t. The coefficients are estimated at time t based on the new observations and the states estimates are propagated in time $t+1$. The Kalman filter can be described as a two-step process of prediction and correction (Harvey, 1990). Fig. 1 presents a summary of the iterative approach for Kalman filter. In the first step, the next state of the model is predicted based on the vector of parameters of the current state. Along the step the prediction error is estimated. Next, the model can be observed as it has transitioned to a new state. We also estimated the observation error in this step. Finally, the predicted estimate is adjusted based on this observation. The Kalman filter is an iterative prediction-correction method. In order to estimate the next sequence of states, the two-step procedure must be repeated. The recursive form of Eqs. (12) and (14) can be described as follows:

Prediction step:

$$\begin{aligned} \hat{x}_t &= \mathbf{A}_{t-1}\mathbf{x}_t \\ \hat{R}_t &= \mathbf{A}_{t-1}\mathbf{R}_{t-1}\mathbf{A}_{t-1}^T + \mathbf{Q}_{t-1} \end{aligned} \tag{16}$$

Correction step:

$$\begin{aligned} v_t &= \mathbf{T}_t - \mathbf{H}_t\hat{x}_t \\ \mathbf{S}_t &= \mathbf{H}_t\hat{R}_t\mathbf{H}_t^T + \mathbf{V}_t \\ \mathbf{K}_t &= \hat{R}_t\mathbf{H}_t^T\mathbf{S}_t^{-1} \\ \mathbf{x}_t &= \hat{x}_t + \mathbf{K}_t v_t \\ \mathbf{R}_t &= \hat{R}_t - \mathbf{K}_t\mathbf{S}_t\mathbf{K}_t^T \end{aligned} \tag{17}$$

where

- $\hat{x}_t$ and $\hat{R}_t$ are the predicted mean and covariance of the state, respectively, on the time step t before seeing the measurement;
- $\mathbf{x}_t$ and $\mathbf{R}_t$ are the estimated mean and covariance of the state, respectively, on time step t after seeing the measurement;
- v_t is the innovation or the measurement residual on time step t;
- $\mathbf{S}_t$ is the measurement prediction covariance on the time step t;
- $\mathbf{K}_t$ is the Kalman gain, which tells how much the predictions should be corrected on time step t;
- $\mathbf{Q}_t$ and $\mathbf{V}_t$ are the covariances of the state transition and measurement equations, respectively, on the time step t.

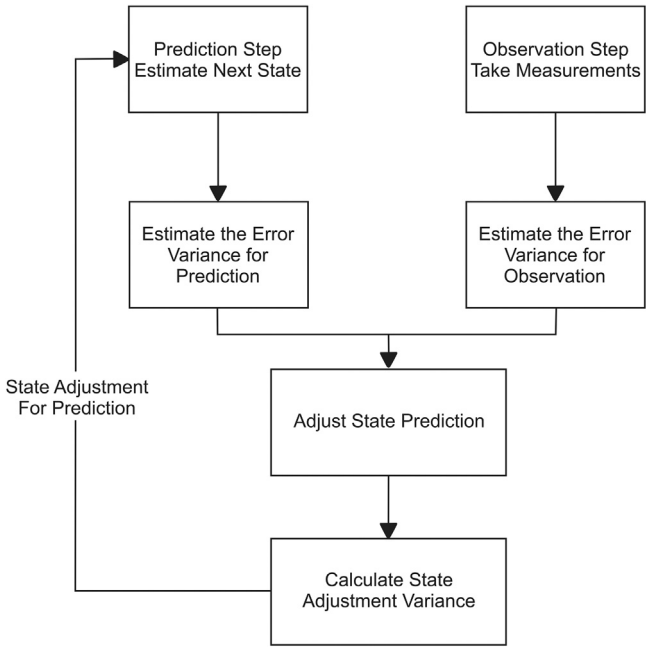

Fig. 1. The Kalman filter iterative process.

3. Proposed method

In this section, the KOSELM algorithm is proposed for sequential data. The KOSELM algorithm can learn data in both the chunk-by-chunk and one-by-one modes. For simplicity, in this paper the one-by-one mode is adopted in order to simplify the description of the algorithm. In KOSELM, the output weights of an OS-ELM are tuned by a Kalman filter regression model, which adjusts the estimate of the output matrix to find a new solution aiming to minimize the norm of $\|\mathbf{H}_t\boldsymbol{\beta}_t - \mathbf{T}_t\|$. Before formally proposing KOSELM, the method to update the output weights is briefly discussed.

3.1. Weight update

The basis for the proposed method is the update of the output weights of OS-ELM through a state space model. This paper aims to estimate the optimal output weight $\boldsymbol{\beta}_t$ by performing a two-step procedure which includes predicting the next state for the time-varying regression coefficient of the output matrix $\boldsymbol{\beta}_t$ and filtering the estimation based on the variance of the observation. To compute the estimates for all iterations of the OS-ELM algorithm, the Kalman filter is applied for sequential problems. The first step, state prediction, is responsible for gaining the apriori estimation and error covariance of the current state. The second step, the measurement update, incorporates the apriori prediction into the real observations to calculate the optimal estimation for the regression coefficient. The regression coefficients of this recursive approach are used to adjust the estimate of the output weights matrix. The state space model described in this paper is based on the following linear function that relates $\boldsymbol{\beta}_t$ and $\hat{\boldsymbol{\beta}}_t$:

$$\boldsymbol{\beta}_t = \mathbf{A}\hat{\boldsymbol{\beta}}_t\alpha_t + \varepsilon_t \tag{18}$$

$$\alpha_t = \alpha_{t-1} + \omega_t \tag{19}$$

where $\boldsymbol{\beta}_t$ is the dependent variable at time t, $\hat{\boldsymbol{\beta}}_t$ is the estimate of $\boldsymbol{\beta}_t$ at time t, α_t is the time-varying regression coefficient and $\mathbf{A}$ is

the transition matrix. ε and ω are Gaussian noises with variance $\mathbf{v}_\varepsilon$ and $\mathbf{v}_\omega$, respectively. To avoid unnecessary iterations, the minimal norm solution is applied for weight update in the same loop of the Kalman filter algorithm using the Eq. (6) of the batch version of ELM.

The main motivation behind the proposed method is the superiority of the Kalman filter over the recursive least squares (RLS) for estimating the unknown coefficients of the regression model described by Eq. (13). The Kalman filter can be defined as an optimal method to provide minimum variance unbiased estimators of the unknown coefficients (Watson, 1983). In regression analysis, when two or more of the predictors in a regression model are highly correlated, a statistical phenomenon known as multicollinearity can be referred to. The main consequence of multicollinearity is to increase the standard errors of the coefficients which means that it makes some variables statistically insignificant when they should be significant (Farrar and Glauber, 1967).

It was shown in Li and Niu (2013) that the stability and generalization capability of the ELM could be strongly influenced by the singularity and ill-conditioning of the hidden layer output matrix. Multicollinearity of the hidden layer output in ELM may cause a large change of regression coefficients so that output weights may be rather unstable (Zhao et al., 2013). Both the Kalman filter and RLS suffer from the presence of multicollinearity, but the first method is capable of detecting the presence and severity of multicollinearity as well as adjusting the estimates in a fashion similar to ridge regression since the filter is less data sensitive, and the estimates are more reliable. A detailed discussion over the superiority of Kalman filter to handle the multicollinearity effects can be found in Watson (1983).

The proposed output weight update algorithm can be summarized as follows:

1. In the initialization step, after the estimation of the initial output weight $\boldsymbol{\beta}^{(0)}$ at time t, estimate the transition matrix $\mathbf{A}$ and the covariance matrices $\mathbf{v}_\varepsilon$ and $\mathbf{v}_\omega$ by the maximum likelihood method, using Eq. (15). These parameters are an important part of the Kalman filter and they will be used to adjust the gain matrix in the following steps.
2. After presenting the $(t+1)$th training sample
$$n_{t+1}=\{\mathbf{x}_i,\mathbf{y}_i\}, i=\left(\sum_{l=0}^{t}N_l\right)+1,\ldots,\left(\sum_{l=0}^{t+1}N_l\right),$$
calculate $\mathbf{H}_{t+1}$ using Eq. (9) and set $\mathbf{T}_{t+1}$ using Eq. (10).
3. For the $(t+1)$th weight estimation $\boldsymbol{\beta}_{t+1}$ using Eq. (11), filter α_t by the recursive equations of Kalman filter (Harvey, 1990). Evaluate $\hat{\alpha}_{t+1|t}$ and $\hat{R}_{t+1|t}$ using the state equation:
$$\hat{\alpha}_{t+1|t}=\mathbf{A}\alpha_{t|t} \tag{20}$$
$$\hat{R}_{t+1|t}=\mathbf{A}\mathbf{R}_{t|t}\mathbf{A}^T+\mathbf{v}_\omega \tag{21}$$
4. Find the observation $\hat{\boldsymbol{\beta}}_{t+1}$ using $\mathbf{H}_{t+1}$ and $\mathbf{T}_{t+1}$, by observing the system as follows:
$$\hat{\boldsymbol{\beta}}_{t+1}=\mathbf{H}_{t+1}^{+}\mathbf{T}_{t+1}\hat{\alpha}_{t+1|t} \tag{22}$$
5. Compute the Kalman gain $\mathbf{K}_{t+1}$, which will be used to obtain the estimate of the linear minimum error variance:
$$\mathbf{K}_{t+1}=\hat{R}_{t+1|t}\mathbf{H}_{t+1}^T\left(\mathbf{H}_{t+1}\hat{R}_{t+1|t}\mathbf{H}_{t+1}^T+\mathbf{v}_\varepsilon\right)^{-1} \tag{23}$$
6. Update the state $\alpha_{t+1|t+1}$ using the Kalman gain by the following equation:
$$\alpha_{t+1|t+1}=\hat{\alpha}_{t+1|t}+\mathbf{K}_{t+1}\left(\boldsymbol{\beta}_{t+1}-\hat{\boldsymbol{\beta}}_{t+1}\right) \tag{24}$$
7. Update the state covariance matrix $\mathbf{R}_{t+1|t+1}$ as follows:
$$\mathbf{R}_{t+1|t+1}=\hat{R}_{t+1|t}-\mathbf{K}_{t+1}\left(\mathbf{H}_{t+1}\hat{R}_{t+1|t}\mathbf{H}_{t+1}^T+\mathbf{v}_\varepsilon\right)\mathbf{K}_{t+1}^T \tag{25}$$
8. According to Harvey (1990), steps 2–7 are a recursive least squares solution for the state space model. Repeat steps 2–7 until the end of sequential training data.
9. Assign α as the resulting coefficient of the regression performed by the Kalman filter.
10. Finally, estimate the adjusted output weight matrix by the following linear equation:
$$\boldsymbol{\beta}^*=\boldsymbol{\beta}\alpha \tag{26}$$

In Nobrega and Oliveira (2013, 2014), evidences were presented that the linear combination of individual forecasts using a Kalman filter regression model can increase the forecast accuracy when applied for time-varying data. In that approach, two different classes of learning methods were combined using a state space model in order to adjust for the variance of individual forecasts. The proposed weight update algorithm uses the same method to adjust the estimation of the regression coefficient by filtering and adjusting the covariance of the state prediction for each cycle of the original OS-ELM. The novelty of this research is the estimation of the output weight matrix $\boldsymbol{\beta}^*$ using the adjusted regression coefficient since the norm of the vector $\left\|\mathbf{H}\boldsymbol{\beta}^*-\mathbf{T}\right\|$ can be minimized when the variance of one of its components decreases during the filtering process. The solution for Eq. (26) still has the smallest norm among all the least squares solutions $\mathbf{H}\boldsymbol{\beta}^*=\mathbf{T}$, according to the formal proof presented by Huang et al. (2006).

3.2. Kalman Online Sequential Extreme Learning Machine (KOSELM)

The KOSELM algorithm is summarized in Algorithm 1. The first part of the algorithm, the Initialization Step, is the same as the OS-ELM. The last part, the Sequential Learning Step, incorporates the filtering of the regression coefficient $\hat{\alpha}_{t+1}$ in order to adjust the variance of the $\hat{\boldsymbol{\beta}}_{t+1}$ estimate using the proposed weight update method.

Algorithm 1. Kalman Online Sequential Extreme Learning Machine (KOSELM).

Data: $\{\mathbf{x}_i, \mathbf{y}_i\}, \mathbf{x}_i \in \mathbb{R}^d,\ \mathbf{y}_i \in \mathbb{R}^q,\ i = 1, \ldots, N$
Result: $\boldsymbol{\beta}^*$
/* Initialization Step */
1 **begin**
2 Initial training dataset: $n_0 = \{\mathbf{x}_i, \mathbf{y}_i\}, \mathbf{x}_i \in \mathbb{R}^d$ and $\mathbf{y}_i \in \mathbb{R}^q$, $i = 1, \ldots, N_0$, with $N_0 \in N$.
3 Set hidden nodes L, with $N_0 \geq L$.
4 Choose random hidden node parameters: $\mathbf{a}_j \in \mathbb{R}^d$ and $b_j \in \mathbb{R}^1, j = 1, \ldots, L$.
5 Calculate the initial hidden layer output matrix:

$$\mathbf{H}_0 = \begin{bmatrix} G(\mathbf{a}_1, b_1, \mathbf{x}_1) & \cdots & G(\mathbf{a}_L, b_L, \mathbf{x}_1) \\ \vdots & \ddots & \vdots \\ G(\mathbf{a}_1, b_1, \mathbf{x}_{N_0}) & \cdots & G(\mathbf{a}_L, b_L, \mathbf{x}_{N_0}) \end{bmatrix}_{N_0 \times L}.$$

6 Inverse $\mathbf{P}_0 : \mathbf{P}_0 = \left(\mathbf{H}_0^T \mathbf{H}_0\right)^{-1}$.
7 Estimate initial output weight $\boldsymbol{\beta}^{(0)} = \mathbf{P}_0 \mathbf{H}_0^T \mathbf{T}$.
8 Assign $\alpha_{t|t} = 1$ and $\mathbf{R}_{t|t} = 1$.
9 Assign $\mathbf{M}_{t|t} = \mathbf{P}_0$.
10 Assign $t = N_0$.
11 Estimate the covariances matrices $\mathbf{Q}_t$ and $\mathbf{R}_t$ and the transition matrix $\mathbf{A}$ using Equation 15.
12 **end**
/* Sequential Learning Step */
13 **repeat**
14 Sequential training data:

$$n_{t+1} = \{\mathbf{x}_i, \mathbf{y}_i\}, i = \left(\sum_{l=0}^{t} N_l\right) + 1, \ldots, \left(\sum_{l=0}^{t+1} N_l\right).$$

15 Calculate $\mathbf{H}_{t+1}$ using Equation 9.
16 Set $\mathbf{T}_{t+1}$ using Equation 10.
17 Calculate $\mathbf{M}_{t+1} = \mathbf{M}_t - \mathbf{M}_t \mathbf{H}_{t+1}^T \left(\mathbf{I} + \mathbf{H}_{t+1} \mathbf{M}_t \mathbf{H}_{t+1}^T\right)^{-1} \mathbf{H}_{t+1} \mathbf{M}_t$.
18 Estimate the output weight $\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t + \mathbf{M}_{t+1} \mathbf{H}_{t+1}^T (\mathbf{T}_{t+1} - \mathbf{H}_{t+1} \boldsymbol{\beta}_t)$.
19 Update $\hat{\alpha}_{t+1|t} = \mathbf{A} \alpha_{t|t}$ and $\hat{\boldsymbol{R}}_{t+1|t} = \mathbf{A} \mathbf{R}_{t|t} \mathbf{A}^T + \mathbf{v}_\omega$.
20 Find the observation $\hat{\boldsymbol{\beta}}_{t+1} = \mathbf{H}_{t+1}^{\dagger} \mathbf{T}_{t+1} \hat{\alpha}_{t+1|t}$.
21 Compute the Kalman gain

$$\mathbf{K}_{t+1} = \hat{\boldsymbol{R}}_{t+1|t} \mathbf{H}_{t+1}^T \left(\mathbf{H}_{t+1} \hat{\boldsymbol{R}}_{t+1|t} \mathbf{H}_{t+1}^T + \mathbf{v}_\varepsilon\right)^{-1}.$$

22 Update the state $\alpha_{t+1|t+1} = \hat{\alpha}_{t+1|t} + \mathbf{K}_{t+1} \left(\boldsymbol{\beta}_{t+1} - \hat{\boldsymbol{\beta}}_{t+1}\right)$.
23 Update the state covariance matrix

$$\mathbf{R}_{t+1|t+1} = \hat{\boldsymbol{R}}_{t+1|t} - \mathbf{K}_{t+1} \left(\mathbf{H}_{t+1} \hat{\boldsymbol{R}}_{t+1|t} \mathbf{H}_{t+1}^T + \mathbf{v}_\varepsilon\right) \mathbf{K}_{t+1}^T.$$

24 **until** $t = N$;
25 Assign $\hat{\boldsymbol{\beta}} = \boldsymbol{\beta}_N$, where $\boldsymbol{\beta}_N$ is the last output weight estimated in the sequential learning step.

3.3. Computational complexity

The computational complexity of the proposed KOSELM algorithm must be separated into two parts in order to be analyzed: (i) the initialization step and (ii) the sequential learning step. Assuming the computation with individual elements has complexity $O(1)$, the complexity of multiplication of one $m \times n$-matrix and one $n \times$ p-matrix is $O(mnp)$. As $\mathbf{H} \in \mathbb{R}^{N \times L}$, the computational complexity of two matrix multiplications and one matrix inversion in the step 6 of the Initialization step are $O\left(L^2 N\right)$ and $O\left(L^3\right)$, respectively. To calculate the output weight matrix $\boldsymbol{\beta}^{(0)}$, with $\mathbf{H}_0^{\dagger} \in \mathbb{R}^{L \times N}$ and $\mathbf{T}_0 \in \mathbb{R}^{N \times L}$, the computational complexity is $O\left(L^2 N\right)$.

In the second part, the sequential learning step, when the $(t+1)$th instance of data arrives, steps 17 and 18 are used to update the output weight matrix. As $\mathbf{M}_t \in \mathbb{R}^{L \times L}$ and $\mathbf{H}_{t+1} \in \mathbb{R}^{N \times L}$, the computational complexity to calculate $\mathbf{M}_{t+1}$ is $O\left(L^3 + L^2 N + L N^2\right)$. Similarly, as $\boldsymbol{\beta}_t \in \mathbb{R}^{N \times L}$, the computational complexity to calculate $\boldsymbol{\beta}_{t+1}$ is $O\left(L^2 N + L N^2\right)$. In step 20, the aim is to calculate the observation and the complexity as $O(LN)$. Since additional steps were introduced in the original OS-ELM algorithm, the

computational complexity of the Kalman filter regression involves matrices inversion and multiplication in steps 21–23, which results in a complexity of $O\left(LN^2\right)$, $O(LN)$ and $O\left(L^2N\right)$, respectively. Thus, the total computational complexity of the KOSELM algorithm is $O\left(L^3+L^2N+LN^2+LN\right)$. It can be seen that since the OS-ELM algorithm does not perform the additional filtering steps of KOSELM, the learning rules of OS-ELM for initialization and sequential steps require the computational complexity of $O\left(L^3+L^2N+LN^2\right)$.

4. Experiments

4.1. Datasets

In this section, the performance of KOSELM is evaluated using four regression benchmark problems selected from the UCI repository (Bache and Lichman, 2013): Auto MPG, Abalone, California housing and Mackey–Glass. These datasets were used to evaluate the OS-ELM in Liang et al. (2006), and in this paper, the same experimental approach is applied to evaluate KOSELM. The accuracy of KOSELM is compared with the original OS-ELM and other sequential learning algorithms, namely, ReOS-ELM (Huynh and Won, 2011), TOSELM (Gu et al., 2014), OS-ELM-TV (Ye et al., 2013) and LS-IELM (Guo et al., 2014).

The Auto MPG problem consists of predicting the fuel consumption of different cars based on the features of each car model. The Abalone problem aims to predict the age of abalone using a set of physical features. The California housing problem aims to estimate the median price of houses using data collected from the 1990 census. The Mackey–Glass problem consists of the prediction for a chaotic time series generated by a nonstationary and nonlinear differential equation. For Mackey–Glass problem, the same set of parameters presented in Liang et al. (2006) are used to generate the time series. In Table 1, a summary of the regression datasets is described. The training and testing datasets are split the same as in the OS-ELM original paper. In order to keep the same settings adopted by Liang et al. (2006), the Gaussian RBF and the sigmoidal functions are used here. All the experiments are performed in MATLAB 8.1 running on a PC with 2.40 GHz and 6 GB RAM.

Table 1
Description of regression datasets.

Dataset	# Attributes	# Training data	# Testing data
Auto MPG	7	320	72
Abalone	8	3000	1177
California Housing	8	8000	12,640
Mackey–Glass	4	4000	500

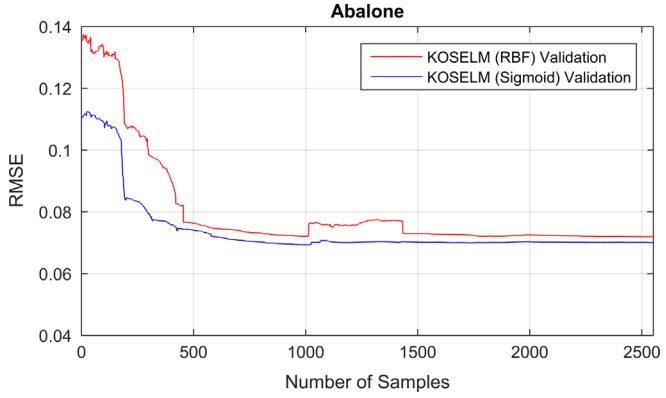

Fig. 2. Evolution of the validation error for KOSELM – Abalone dataset.

4.2. Experimental settings

The attributes of the regression datasets are normalized into the range of $[0, 1]$. The input weights and biases parameters are randomly chosen and normalized into the range of $[-1, 1]$ for both RBF and sigmoid hidden nodes. The estimation of the number of hidden nodes and the size of initial training data for KOSELM are determined by the split of the training dataset into two nonoverlapped random subsets: one for training and the last one for validation. The optimal set of parameters is chosen as same as in Liang et al. (2006), in which the lowest validation error is averaged over 50 trials. In Figs. 2–5, the evolution of the validation error for KOSELM is presented in terms of the number of sample data for all datasets. The number of hidden

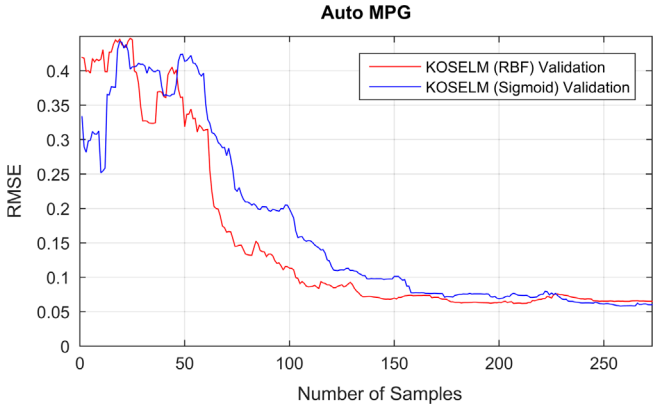

Fig. 3. Evolution of the validation error for KOSELM – Auto MPG dataset.

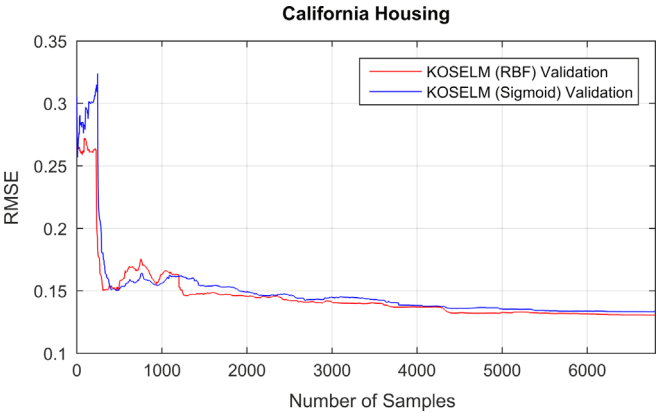

Fig. 4. Evolution of the validation error for KOSELM – California Housing dataset.

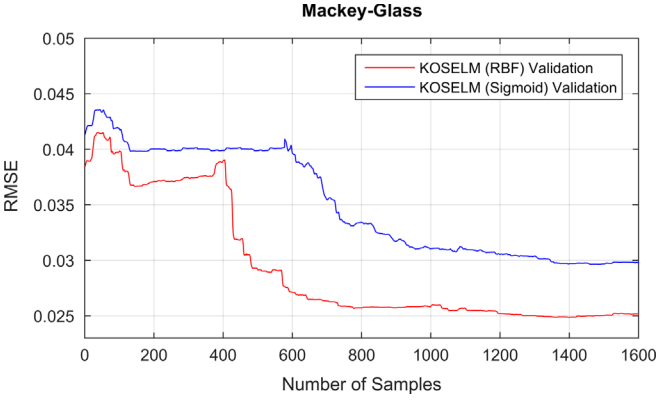

Fig. 5. Evolution of the validation error for KOSELM – Mackey–Glass dataset

nodes for all algorithms is optimized within the range of [1, 50] for Abalone, Auto MPG and California housing. For Mackey–Glass dataset, the range for optimization is [50, 150]. The number of initial training data has been taken as $N_0 = 75$ for Abalone and Auto MPG. For California Housing, the number of initial data is 100 and for Mackey–Glass $N_0 = 1620$. The algorithms TOSELM, LS-IELM and ReOS-ELM have specific parameters ϵ, υ and λ, respectively, and all of them are defined as same as in their original papers. A summary of the parameters is presented in Table 2.

Table 2
Optimal parameters for sequential learning algorithms.

Dataset	# Hidden nodes	# Initial training data
Abalone	25	75
Auto MPG	25	75
California Housing	50	100
Mackey–Glass	120	1620

4.3. Statistical evaluation

In order to evaluate the statistical performance of the proposed KOSELM, as well the comparison with OS-ELM, ReOS-ELM, TOSELM, OS-ELM-TV and LS-IELM, 50 trials for each algorithm were executed and the average training time and the average RMSE was computed for both training and testing datasets. The number of hidden nodes

Table 4
Average ranks for sequential learning algorithms.

Algorithm	Rank
KOSELM	1.3750
OS-ELM	3.1250
ReOS-ELM	4.3750
TOSELM	5.3750
OS-ELM-TV	2.8750
LS-IELM	3.8750

Table 3
Performance comparison between all algorithms in regression problems.

Dataset	Algorithm	Time (s)	Training	Testing
Abalone	KOSELM (sigmoid)	0.7366	0.0751 (0.0010)	0.0770 (0.0036)
	KOSELM (RBF)	0.9581	0.0749 (0.0008)	**0.0766** (0.0046)
	OS-ELM (sigmoid)	0.2831	0.0752 (0.0011)	0.0788 (0.0070)
	OS-ELM (RBF)	0.6319	0.0748 (0.0011)	0.0771 (0.0038)
	ReOS-ELM (sigmoid)	0.2684	0.0756 (0.0031)	0.0795 (0.0103)
	ReOS-ELM (RBF)	0.5925	0.0747 (0.0012)	0.0771 (0.0039)
	TOSELM (sigmoid)	0.7678	0.0786 (0.0056)	0.0815 (0.0113)
	TOSELM (RBF)	1.1947	0.0760 (0.0025)	0.0788 (0.0047)
	OS-ELM-TV (sigmoid)	0.4422	0.0749 (0.0010)	0.0778 (0.0048)
	OS-ELM-TV (RBF)	0.5406	0.0752 (0.0012)	0.0767 (0.0030)
	LS-IELM (sigmoid)	0.2769	0.0750 (0.0010)	0.0786 (0.0069)
	LS-IELM (RBF)	0.6288	0.0748 (0.0012)	0.0775 (0.0057)
Auto MPG	KOSELM (sigmoid)	0.1006	0.0674 (0.0024)	0.0761 (0.0097)
	KOSELM (RBF)	0.1500	0.0683 (0.0031)	**0.0738** (0.0102)
	OS-ELM (sigmoid)	0.0338	0.0677 (0.0035)	0.0763 (0.0102)
	OS-ELM (RBF)	0.1031	0.0671 (0.0029)	0.0753 (0.0100)
	ReOS-ELM (sigmoid)	0.0350	0.0698 (0.0042)	0.0769 (0.0134)
	ReOS-ELM (RBF)	0.1225	0.0700 (0.0039)	0.0780 (0.0108)
	TOSELM (sigmoid)	0.0803	0.0703 (0.0033)	0.0775 (0.0085)
	TOSELM (RBF)	0.1456	0.0692 (0.0030)	0.0776 (0.0113)
	OS-ELM-TV (sigmoid)	0.0928	0.0676 (0.0035)	0.0767 (0.0094)
	OS-ELM-TV (RBF)	0.1006	0.0680 (0.0035)	0.0787 (0.0090)
	LS-IELM (sigmoid)	0.0325	0.0674 (0.0028)	0.0766 (0.0092)
	LS-IELM (RBF)	0.0978	0.0671 (0.0027)	0.0756 (0.0116)
California Housing	KOSELM (sigmoid)	3.7475	0.1304 (0.0025)	0.1335 (0.0035)
	KOSELM (RBF)	8.9594	0.1297 (0.0022)	**0.1319** (0.0026)
	OS-ELM (sigmoid)	1.5850	0.1303 (0.0023)	0.1342 (0.0031)
	OS-ELM (RBF)	6.7941	0.1297 (0.0020)	0.1323 (0.0024)
	ReOS-ELM (sigmoid)	1.3559	0.1322 (0.0026)	0.1361 (0.0048)
	ReOS-ELM (RBF)	6.4406	0.1305 (0.0031)	0.1338 (0.0045)
	TOSELM (sigmoid)	5.4763	0.1338 (0.0029)	0.1388 (0.0060)
	TOSELM (RBF)	10.5097	0.1329 (0.0030)	0.1362 (0.0045)
	OS-ELM-TV (sigmoid)	1.1234	0.1307 (0.0017)	0.1329 (0.0032)
	OS-ELM-TV (RBF)	1.2941	0.1299 (0.0018)	0.1328 (0.0040)
	LS-IELM (sigmoid)	1.3266	0.1304 (0.0021)	0.1338 (0.0026)
	LS-IELM (RBF)	5.7244	0.1300 (0.0020)	0.1322 (0.0026)
Mackey–Glass	KOSELM (sigmoid)	2.0591	0.0255 (0.0010)	0.0200 (0.0009)
	KOSELM (RBF)	5.5012	0.0246 (0.0005)	**0.0189** (0.0007)
	OS-ELM (sigmoid)	1.3984	0.0252 (0.0005)	0.0198 (0.0007)
	OS-ELM (RBF)	4.8631	0.0247 (0.0007)	0.0191 (0.0007)
	ReOS-ELM (sigmoid)	1.3447	0.0254 (0.0007)	0.0200 (0.0008)
	ReOS-ELM (RBF)	4.7059	0.0245 (0.0006)	**0.0189** (0.0008)
	TOSELM (sigmoid)	1.8200	0.0254 (0.0005)	0.0201 (0.0007)
	TOSELM (RBF)	4.7284	0.0246 (0.0006)	0.0191 (0.0007)
	OS-ELM-TV (sigmoid)	2.4425	0.0251 (0.0005)	0.0198 (0.0007)
	OS-ELM-TV (RBF)	2.6897	0.0245 (0.0005)	0.0190 (0.0006)
	LS-IELM (sigmoid)	1.0962	0.0406 (0.0024)	0.0368 (0.0025)
	LS-IELM (RBF)	5.1622	0.0360 (0.0011)	0.0318 (0.0013)

and the number of initial training data were the same for all algorithms executions. Table 3 presents a summary of the results for each dataset in terms of training time and training and testing errors. For all datasets, the best RMSE is shown in bold face. For the sake of simplicity, only the one-by-one mode was tested here since the results presented in Liang et al. (2006) showed a similar performance when compared to the chunk-by-chunk mode.

As shown in Table 3, the additional computation for Kalman filtering of KOSELM increases the average training time when compared with the competing algorithms. For Abalone and California Housing datasets, the TOSELM algorithm was slower than KOSELM. This behavior was expected since the effort to perform the matrix computations for Kalman filter and to estimate the adjusted weight $\hat{\boldsymbol{\beta}}$ adds complexity to the sequential step of KOSELM. When comparing the performance of KOSELM versions for RBF and sigmoid, the similarities in the results in terms of accuracy can be highlighted. As observed from the training and testing RMSE of OS-ELM, the performance is comparable to the results shown in Liang et al. (2006). In comparison with OS-ELM, ReOS-ELM, TOSELM, OS-ELM-TV and LS-IELM, the proposed KOSELM presents better performance by showing the lowest RMSEs for all testing datasets.

To further confirm the results obtained from the statistical evaluation, a set of tests was performed to compare the difference between the residuals for all executions. The null hypothesis being tested here is that all algorithms perform the same, and the observed differences are merely random. The Friedman test (Friedman, 1940) was performed in order to analyze if there are statistically significant differences for all algorithms. All algorithms are ranked for each dataset separately, the best performing algorithm getting the lowest rank. If two or more algorithms present the same performance for a single dataset, the average rank is assigned. Further details of the process for comparison of multiple algorithms are given in Demšar (2006). In Table 4, the average rank for each algorithm is presented. The performance of two algorithms is significantly different at confidence level $\alpha=0.05$ if the difference of average ranks reaches a critical difference according to the following expression:

$$CD = q_\alpha \sqrt{\frac{c(c+1)}{6D}}, \tag{27}$$

where c is the number of algorithms, D is the number of datasets and q_α is the critical value. According to the Bonferroni–Dunn test (Dunn, 1961), the critical value for $\alpha=0.05$ is 2.576. Under the null hypothesis which states that all the algorithms are equivalent and their ranks are equal, the Friedman statistic is given by the following

$$\chi_F^2 = \frac{12D}{c(c+1)}\left[\sum_j R_j^2 - \frac{c(c+1)^2}{4}\right], \tag{28}$$

where R_j^2 is the j-th average rank of the algorithms. The statistic is distributed according to χ_F^2 with $c-1$ degrees of freedom. For the comparison of all algorithms with the Friedman test, the χ_F^2 statistic is 21.64 and the p-value is 0.0006, which rejects the null hypothesis that all algorithms have the same performance.

After the null hypothesis rejection, a post hoc test is performed to evaluate the pairwise performance when all algorithms are compared to each other. The Bonferroni–Dunn test with $\alpha=0.05$ was applied, and the results are presented in Table 5. According to the table, the critical differences are significant when comparing KOSELM with ReOS-ELM and TOSELM algorithms, with the corresponding p-values highlighted in bold. The null hypothesis for the remaining comparisons was not rejected since there is no statistically significant differences between them. According to Salzberg (1997), the Bonferroni–Dunn test can be considered a conservative test since it supposes the independence of the hypotheses, and it makes an adjustment to compensate for multiple comparisons. If the Tukey test (Tukey, 1949) is applied, the KOSELM outperforms ReOS-ELM, TOSELM and LS-IELM algorithms, since the critical differences are statistically significant. Moreover, the TOSELM algorithm is significantly inferior to OS-ELM and OS-ELM-TV. In Table 5, the p-values for Tukey test are presented, with the significant values highlighted in bold.

Table 5
Post hoc tests of algorithms performance.

Comparison	Difference	p-value (Bonferroni–Dunn)	p-value (Tukey)
LS-IELM–KOSELM	2.50	0.0806	**0.0075**
OS-ELM–KOSELM	1.75	0.4202	0.0613
OS-ELM-TV–KOSELM	1.50	0.5961	0.1088
ReOS-ELM–KOSELM	3.00	**0.0167**	**0.0013**
TOSELM–KOSELM	4.00	**0.0002**	**1.9012e−5**
OS-ELM–LS-IELM	−0.75	0.9672	0.4226
OS-ELM-TV–LS-IELM	−1.00	0.8937	0.2850
ReOS-ELM–LS-IELM	0.50	0.9947	0.5929
TOSELM–LS-IELM	1.50	0.5962	0.1088
OS-ELM-TV–OS-ELM	−0.25	0.9998	0.7892
ReOS-ELM–OS-ELM	1.25	0.7647	0.1814
TOSELM–OS-ELM	2.25	0.1542	**0.0161**
ReOS-ELM–OS-ELM-TV	1.50	0.5961	0.1088
TOSELM–OS-ELM-TV	2.50	0.0808	**0.0075**
TOSELM–ReOS-ELM	1.00	0.8937	0.2850

5. Conclusions

In this paper, a novel online sequential algorithm for regression learning has been presented. The proposed Kalman Online Sequential Extreme Learning Machine (KOSELM) is different from the original OS-ELM since it can provide minimum variance estimators for the output weights. In regression problems where training samples are received sequentially, the KOSELM approach can statistically outperform ReOS-ELM, TOSELM and LS-IELM for both sigmoid and RBF hidden nodes. The performance of the proposed method was evaluated on benchmark datasets for regression problems. KOSELM performs better in terms of accuracy when compared with their counterparts. The superiority of KOSELM accuracy was confirmed by applying statistical tests to compare the residuals of the estimations. In terms of variance adjustment, KOSELM can handle the effect of multicollinearity of the hidden layer output. The smoothing nature of the Kalman filter is a key factor to build stable models based on sequential learning.

In future works, the authors intend to investigate other state space approaches in order to address the constraints of Kalman filter. For instance, the state transition model may be nonlinear, and a different approach could be applied for nonlinear estimation. The authors aim to extend this paper and investigate the performance of KOSELM for multi-step-ahead prediction for time series problems (Grigorievskiy et al., 2014; Wang and Han, 2015). The authors also plan to apply the proposed method to classification problems and perform a comparison with an extended set of sequential learning algorithms.

Acknowledgements

The authors would like to thank CNPq and FACEPE (Brazilian Agencies) for their financial support.

References

Bache, K., Lichman, M., 2013. Uci Machine Learning Repository, p. 901. URL ⟨http://archiveicsuciedu/ml⟩.

Chong, E.K., Zak, S.H., 2013. An Introduction to Optimization, vol. 76. John Wiley & Sons, Hoboken, New Jersey, USA.

Demsar, J., 2006. Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. 7, 1–30.

Drucker, H., Burges, C.J., Kaufman, L., Smola, A., Vapnik, V., 1997. Support vector regression machines. Adv. Neural Inf. Process. Syst. 9, 155–161.

Dunn, O.J., 1961. Multiple comparisons among means. J. Am. Stat. Assoc. 56 (293), 52–64.

Farrar, D.E., Glauber, R.R., 1967. Multicollinearity in regression analysis: the problem revisited. Rev. Econ. Stat., 92–107.

Foucart, T., 1999. Stability of the inverse correlation matrix. partial ridge regression. J. Stat. Plann. Inference 77 (1), 141–154.

Friedman, M., 1940. A comparison of alternative tests of significance for the problem of m rankings. Ann. Math. Stat. 11 (1), 86–92.

Grigorievskiy, A., Miche, Y., Ventel'´a, A.M., Séverin, E., Lendasse, A., 2014. Long-term time series prediction using op-elm. Neural Netw. 51, 50–56.

Gu, Y., Liu, J., Chen, Y., Jiang, X., Yu, H., 2014. Toselm: timeliness online sequential extreme learning machine. Neurocomputing 128, 119–127.

Guo, L., Hao, J.h., Liu, M., 2014. An incremental extreme learning machine for online sequential learning problems. Neurocomputing 128, 50–58.

Harvey, A.C., 1990. Forecasting, Structural Time Series Models and the Kalman Filter. Cambridge University Press.

Huang, G., Huang, G.B., Song, S., You, K., 2015. Trends in extreme learning machines: a review. Neural Netw. 61, 32–48.

Huang, G.B., Saratchandran, P., Sundararajan, N., 2005. A generalized growing and pruning rbf (ggap-rbf) neural network for function approximation. IEEE Trans. Neural Netw. 16 (1), 57–67.

Huang, G.B., Wang, D.H., Lan, Y., 2011. Extreme learning machines: a survey. Int. J. Mach. Learn. Cybern. 2 (2), 107–122.

Huang, G.B., Zhou, H., Ding, X., Zhang, R., 2012. Extreme learning machine for regression and multiclass classification. IEEE Trans. Syst., Man, Cybern., Part B: Cybern. 42 (2), 513–529.

Huang, G.B., Zhu, Q.Y., Siew, C.K., 2006. Extreme learning machine: theory and applications. Neurocomputing 70 (1), 489–501.

Huynh, H.T., Won, Y., 2011. Regularized online sequential learning algorithm for single-hidden layer feedforward neural networks. Pattern Recognit. Lett. 32 (14), 1930–1935.

Kadirkamanathan, V., Niranjan, M., 1993. A function estimation approach to sequential learning with neural networks. Neural Comput. 5 (6), 954–975.

Li, G., Niu, P., 2013. An enhanced extreme learning machine based on ridge regression for regression. Neural Comput. Appl. 22 (3–4), 803–810.

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

Luo, M., Zhang, K., 2014. A hybrid approach combining extreme learning machine and sparse representation for image classification. Eng. Appl. Artif. Intell. 27, 228–235.

Malathi, V., Marimuthu, N., Baskar, S., Ramar, K., 2011. Application of extreme learning machine for series compensated transmission line protection. Eng. Appl. Artif. Intell. 24 (5), 880–887.

Nobrega, J.P., Oliveira, A.L.I., 2013. Improving the statistical arbitrage strategy in intraday trading by combining extreme learning machine and support vector regression with linear regression models. In: 2013 IEEE 25th International Conference on Tools with Artificial Intelligence (ICTAI), IEEE, Washington, DC, USA, p. 182–188.

Nobrega, J.P., Oliveira, A.L.I., 2014. A combination forecasting model using machine learning and kalman filter for statistical arbitrage. In: 2014 IEEE International Conference on Systems, Man and Cybernetics (SMC), IEEE, San Diego, CA, USA, p. 1294–1299.

Platt, J., 1991. A resource-allocating network for function interpolation. Neural Comput. 3 (2), 213–225.

Rao, C.R., Mitra, S.K., 1971. Generalized Inverse of Matrices and its Applications, 7. Wiley, New York.

Ruhmelhart, D., Hinton, G., Wiliams, R., 1986. Learning representations by back-propagation errors. Nature 323, 533–536.

Salzberg, S.L., 1997. On comparing classifiers: pitfalls to avoid and a recommended approach. Data Min. Knowl. Discov. 1 (3), 317–328.

Tukey, J.W., 1949. Comparing individual means in the analysis of variance. Biometrics, 99–114.

Wang, X., Han, M., 2015. Improved extreme learning machine for multivariate time series online sequential prediction. Eng. Appl. Artif. Intell. 40, 28–36.

Watson, P., 1983. Kalman filtering as an alternative to ordinary least squaressome theoretical considerations and empirical results. Empir. Econ. 8 (2), 71–85.

Ye, Y., Squartini, S., Piazza, F., 2013. Online sequential extreme learning machine in nonstationary environments. Neurocomputing 116, 94–101.

Yingwei, L., Sundararajan, N., Saratchandran, P., 1998. Performance evaluation of a sequential minimal radial basis function (rbf) neural network learning algorithm. IEEE Trans. Neural Netw. 9 (2), 308–318.

Zhao, L., Wang, D., Chai, T., 2013. Estimation of effluent quality using pls-based extreme learning machines. Neural Comput. Appl. 22 (3–4), 509–519.