Cognitive Computation (2024) 16:641–653
https://doi.org/10.1007/s12559-023-10220-w

A Novel Regularized Extreme Learning Machine Based on L_1-Norm and L_2-Norm: a Sparsity Solution Alternative to Lasso and Elastic Net

Hasan Yıldırım[1] · **M. Revan Özkale**[2]

Received: 9 September 2022 / Accepted: 6 November 2023 / Published online: 1 December 2023

Abstract
The aim of this study is to present a new regularized extreme learning machine (ELM) algorithm that can perform variable selection based on the simultaneous use of both ridge and Liu regressions in order to cope with some disadvantages of ELM and its variants such as instability and poor generalization performance and lack of sparsity. The proposed algorithm was compared with the classical ELM as well as the variants based on ridge, Liu, Lasso and Elastic Net approaches by cross-validation process and best tuning parameter over seven different real-world applications and their performances were presented comparatively. The proposed algorithm outperformed ridge, Lasso and Elastic Net algorithms in training performance prediction (average 40%) and stability (average 80%) and in test performance prediction (average 20%) and stability (60%) in the majority of the data. In addition, the proposed ELM was found to be more compact (better sparsity capability) with lower norm values. The results confirmed that the proposed ELM presents more stable and sparse solutions with better generalization performance than any other algorithm under favorable conditions. The findings based on experimental study via real-world applications indicate that the proposed ELM provides effective solutions to the mentioned drawbacks and yields more stable and sparse performance with better generalization capability than its competitors. Consequently, the proposed algorithm represents a powerful alternative both regression and classification tasks in machine learning field due to its theoretical flexibility.

Keywords Extreme learning machine · Sparsity · Liu regression · Tikhonov regularization · Ill-posed problems

Introduction

Recent developments in information retrieving field (particularly machine learning) have heightened the need for faster, reliable and generalizable algorithms. Obviously, artificial neural networks is the leading field to achieve these purposes due to the capabilities on modelling complex problem, adaptability, speed and flexibility. Although the training algorithms like back-propagation (BP) are the core components on the performance of any neural network, they have some shortcomings such as local optima. Additionally, these algorithms may cause higher computation-cost because of the need of extensive parameter tuning like weights, biases, momentum, and learning rate. As a solution to deal with these issues, extreme learning machine (ELM) was proposed by Huang et al. [1, 2].

ELM is a single layer feed-forward neural network (SLFN) as an alternative to the conventional learning algorithms (like BP) by considering a simple but elegant idea into the training process of the network. This idea is based on assigning some parameters including the input weights and biases randomly and reducing the problem to a simple linear system which can be obtained via ordinary least squares (OLS) method. ELM has obtained a large and growing attention due to some useful properties: (i) to have a closed-form solution which can be found via an OLS solution, (ii) to eliminate parameter tuning like input weights, biases, momentum, and learning rate. because of the random assignment procedure, (iii) to provide successful results for both regression and classification tasks, (iv) to present faster performance unlike its competitors (i.e., BP), (v) to need less human intervention during training process. Over the past decade, ELM has received major scientific attention as a powerful instrument

✉ Hasan Yıldırım
hasanyildirimstat@gmail.com

M. Revan Özkale
mrevan@cu.edu.tr

[1] Mathematics, Karamanoğlu Mehmetbey University, Karaman 70200, Türkiye

[2] Statistics, Çukurova University, Adana 01330, Türkiye

for modelling various types of problems in supervised, unsupervised and semi-supervised learning.

ELM has some advantages not only from practical capabilities but also theoretical properties including interpolation, universal approximation capability, and generalizability [1, 2]. In addition to these properties, ELM can be used both for regression and classification tasks in real-world applications. Although ELM has been considered in many fields, it has drawbacks like the instability, poor generalizability and insufficient sparsity due to the structural risk minimization which is the key concept of ELM's learning process. The structure of the hidden layer matrix of ELM seems to be the main reason of these drawbacks. If the hidden layer matrix suffers from the ill-conditioning problem, the performance of ELM is dramatically affected with the non-optimal solution; that is, the ELM solution may have a large variance and the distance from the ELM estimates to the true values may be large. As a result of ill-conditioning, as opposed to Huang et al. [3], ELM does not have the smallest training error and the smallest norm of output weights. The natural result of this particular problem is to produce any solution other than ordinary ELM method.

A number of studies have been carried out to explore the effects of these drawbacks, and attempts have been made to improve ELM to deal with these drawbacks. In the linear regression field, ridge estimator proposed by Hoerl and Kennard [4] is one of the most well-known methods to overcome the ill-conditioning problem. Therefore, Deng et al. [5], Li and Niu [6], Huang and Sun [7], Shao and Er [8], Chen and Wang [9], Yu et al. [10], Wang and Li [11], Yıldırım and Özkale [12], and Luo et al. [13] proposed some alternative algorithms based on the ridge estimator to take advantage of its ability to deal with the ill-conditioning problem, and showed that it generally contributes positively to the performance of the ELM in practical applications. Although ridge estimators is the most used shrinkage method in the literature, Liu estimator proposed by Keijian [14] is considered as a powerful alternative to ridge estimator. Because the ridge estimator is not linearly dependent, while the Liu estimator is linearly dependent on the tuning parameter which makes it easier to calculate the Liu estimator than the ridge estimator. In the context of ELM, Yıldırım and Özkale [15] proposed ELM based on Liu estimator with various tuning parameter selection techniques and proved that Liu estimator makes the ELM performance more stable and generalizable.

In addition to ill-conditioning problem, sparsity is another problem. The sparsity is critical property of any algorithm from the perspective of speed and compactness which is simply refers to a matrix of numbers (i.e., matrix of predictors in linear regression) that includes many zeros or values that will not significantly impact the calculation. However, ELM does not have the sparsity property and its variants based on ridge or Liu estimator are not the solutions at this point. Since Lasso regression proposed by Tibshirani [16] in linear regression is a remedy to provide sparse solutions by shrinking some of regression coefficients to exactly zero, Miche et al. [17, 18], Martínez-Martínez et al. [19], Luo et al. [13], Shan et al. [20], Li et al. [21] and Preeti et al. [22] developed ELM-based Lasso algorithms to improve the ELM performance via pruning the irrelevant nodes. Although Lasso enjoys sparsity, it selects as many predictors (p) as the maximum number of observations (n) in the $p > n$ case, it does not have grouping effect and its performance is dominated by ridge regression if there exists high correlation between predictors. Because of these reasons, Zou and Hastie [23] introduced Elastic Net (ENet) in linear regression. Martínez-Martínez et al. [19] considered the ENet approach which can be seen as the cascade of the ridge and Lasso estimators into the learning process of ELM. Yıldırım and Özkale [24] proposed a novel regularized algorithm called as LL-ELM based on both Liu and Lasso regressions as alternative to ENet.

From a statistical point of view, Liu and ridge estimators have their own characteristics in shrinking the hidden layer structure. They have different objective functions which affect their ability to learn the inherent patterns in the data. Therefore, ridge and Liu-based ELM algorithms act differently, affecting generalization performance [15].

Research Motivation and Enthusiasm

The motivation of this study is based on the idea that a new regularized ELM algorithm with sparsity property can be developed by simultaneously taking advantage of the different properties of the ridge and Liu estimators. Influenced with this idea, we propose a novel regularized ELM algorithm based on the combination of Liu, ridge and Lasso regressions which we call as GO-ELM. The GO-ELM has following major contributions into the existing literature:

- GO-ELM can easily deal with the multicollinearity problem (i.e., the ill-conditioning).
- GO-ELM has the sparsity property as a result of L_1 norm regularization and may provide more compact and better generalization performance.
- GO-ELM carries the advantages of ridge and Liu regressions since it carries the shrinkage idea of both ridge and Liu.
- GO-ELM has the grouping effect as ENet so that it overcomes Lasso which selects one predictor from the highly correlated group.
- Although GO-ELM has the sparsity feature like ENet and Lasso, the shrinkage effect is different from ENet and Lasso. Such that while ENet and Lasso shrink the weights to the origin, GO-ELM shrinks to $d\hat{\boldsymbol{\beta}}_{ELM}$, $0 < d < 1$.

- GO-ELM can be extended to the classification problems. This property makes it usable for various kinds of data-driven machine learning problems.

Review of ELM and Its Variants

ELM is based on a classical SLFN but differs from the point that during the training process, the parameters like input weights and biases in the hidden layer are selected arbitrarily unlike the iterative process in any other neural network. As a result of fixed parameters, the learning problem (classification or regression) is reduced to a system which consists of a series of linear equations. For any linear system, OLS method can be used to find the solution. A SLFN can be written as

$$\sum_{i=1}^{\varphi} \boldsymbol{\beta}_i g\left(\boldsymbol{\omega}_i.\mathbf{x}_j + b_i\right) = \mathbf{y}_j,\ \ j = 1, 2, ..., N, \tag{1}$$

where $\left(\mathbf{x}_j^T, \mathbf{y}_j^T\right)$ is the set of N distinct patterns with $\mathbf{x}_j \in R^p$ and $\mathbf{y}_j \in R^m$, $\mathbf{y}_j$ is the m-dimensional network output, $\boldsymbol{\omega}_i$ is the input weight between the i-th hidden neuron and the hidden layer, b_i is the bias, φ is the number of hidden neurons, $g(.)$ is the activation function and $\boldsymbol{\beta}_i$ is the output weights [1, 2].

Equation (1) can be written in a matrix form as

$$\mathbf{S}\boldsymbol{\beta} = \mathbf{Y} \tag{2}$$

where

$$\mathbf{S} = \begin{bmatrix} g\left(\boldsymbol{\omega}_1.\mathbf{x}_1 + b_1\right) & \ldots & g\left(\boldsymbol{\omega}_\varphi.\mathbf{x}_1 + b_\varphi\right) \\ \vdots & \ldots & \vdots \\ g\left(\boldsymbol{\omega}_1.\mathbf{x}_N + b_1\right) & \ldots & g\left(\boldsymbol{\omega}_\varphi.\mathbf{x}_N + b_\varphi\right) \end{bmatrix}_{N\times\varphi} = \left[\mathbf{h}_1 \ldots \mathbf{h}_\varphi\right]_{N\times\varphi}$$

is the output matrix of hidden layer, $\boldsymbol{\beta}_{\varphi\times m} = \left(\beta_1, ..., \beta_\varphi\right)^T$ and $\boldsymbol{Y}_{N\times m} = \left(\mathbf{y}_1, ..., \mathbf{y}_N\right)^T$ are the output weight vector and output value vector, respectively. Here, m corresponds to the number of output layer neurons which is commonly equal to the number of target variable and fixed as 1 in most practical applications. To estimate the β parameter, the objective form of Eq. (2) is defined as:

$$\|\mathbf{S}\boldsymbol{\beta} - \mathbf{Y}\|_2 = \left(\mathbf{S}\,\boldsymbol{\beta} - \mathbf{Y}\right)^T \left(\mathbf{S}\,\boldsymbol{\beta} - \mathbf{Y}\right). \tag{3}$$

The $\boldsymbol{\beta}$ estimator minimizing Eq. (3) is obtained as $\widehat{\boldsymbol{\beta}}_{ELM} = \mathbf{S}^{+}\mathbf{Y}$ where $\mathbf{S}^{+}$ is the Moore-Penrose inverse of matrix $\mathbf{S}$ [2]. Some popular ways of calculating the Moore-Penrose inverse are the orthogonal projection method, iterative methods and singular value decomposition [25, 26]. According to the orthogonal projection method, $\mathbf{S}^{+}$ is calculated via $\mathbf{S}^T\left(\mathbf{S}\mathbf{S}^T\right)^{-1}$ if $\mathbf{S}$ is full row rank, else $\mathbf{S}^{+} = \left(\mathbf{S}^T\mathbf{S}\right)^{-1}\mathbf{S}^T$ if $\mathbf{S}$ is full column rank.

When there is a multicollinearity problem, inverting the matrix $\left(\mathbf{S}^T\mathbf{S}\right)^{-1}$ can sometimes be impossible and sometimes unstable. Therefore, alternative methods to ordinary ELM are recommended. On the other hand, ordinary ELM does not have sparsity property; that is, does not do variable selection. The methods proposed on the basis of ELM as a solution to these two problems are as follows:

Li and Niu [6] considered ridge regression in the context of ELM which was originally proposed by Hoerl and Kennard [4] and defined the ridge-ELM as

$$\hat{\boldsymbol{\beta}}_{R-ELM}^{k} = \left(\mathbf{S}^T\mathbf{S} + k\mathbf{I}_\varphi\right)^{-1}\mathbf{S}^T\mathbf{Y},\ \ k \geq \mathbf{0}$$

where k is the ridge tuning parameter and $\mathbf{I}_\varphi$ is the identity matrix with dimension φ. Although the value of k affects the performance of R-ELM, there is no single best and suitable method for selecting the ridge tuning parameter. Regression studies literature [6, 27] indicates that R-ELM can provide smaller error than ELM if k is correctly determined.

Yıldırım and Özkale [15] considered the Liu regression in the context of ELM which was originally defined by Kejian [14] and defined the Liu-ELM as

$$\hat{\boldsymbol{\beta}}_{Liu-ELM}^{d} = \left(\mathbf{S}^T\mathbf{S} + \mathbf{I}_\varphi\right)^{-1}\left(\mathbf{S}^T\mathbf{Y} + d\hat{\boldsymbol{\beta}}_{ELM}\right) \tag{4}$$

where $0 < d < 1$ is the Liu tuning parameter which shrinks each element of $\hat{\boldsymbol{\beta}}_{ELM}$ by the same d value. Since $\hat{\boldsymbol{\beta}}_{Liu-ELM}^{d}$ is in linear form of d, computing $\hat{\boldsymbol{\beta}}_{Liu-ELM}^{d}$ is much easier and faster than $\hat{\boldsymbol{\beta}}_{R-ELM}^{k}$ which provides computationally effective and cost minimized solutions for machine learning. $\hat{\boldsymbol{\beta}}_{Liu-ELM}^{d}$ is also a convex combination of $\hat{\boldsymbol{\beta}}_{ELM}$ and $\hat{\boldsymbol{\beta}}_{R-ELM}^{k}$ for $k = 1$ which claims that $\hat{\boldsymbol{\beta}}_{Liu-ELM}^{d}$ provides solution paths between $\hat{\boldsymbol{\beta}}_{ELM}$ and $\hat{\boldsymbol{\beta}}_{R}^{(k=1)}$ as d goes from 1 to 0:

$$\hat{\boldsymbol{\beta}}_{Liu-ELM}^{d} = d\hat{\boldsymbol{\beta}}_{ELM} + (1-d)\hat{\boldsymbol{\beta}}_{R-ELM}^{(k=1)}.$$

Yıldırım and Özkale [28] proposed OK-ELM which takes the advantages of ridge and Liu regressions as

$$\widehat{\boldsymbol{\beta}}_{OK-ELM} = \left(\mathbf{S}^T\mathbf{S} + k\mathbf{I}_\varphi\right)^{-1}\left(\mathbf{S}^T\mathbf{Y} + kd\hat{\boldsymbol{\beta}}_{ELM}\right)$$

where $0 < d < 1$ and $k > 0$ are the tuning parameters. OK-ELM is a convex combination of R-ELM and ELM:

$$\widehat{\boldsymbol{\beta}}_{OK-ELM} = d\hat{\boldsymbol{\beta}}_{ELM} + (1-d)\,\widehat{\boldsymbol{\beta}}^{(k)}_{R-ELM},$$

where d serves as a balance parameter between ELM and R-ELM and refers the relative contributions of them. Basically, while the higher d towards to 1 yields more contribution in favor of ELM, the lower d increases the effect of R-ELM to the solution.

In order to gain the sparsity property to ELM, Miche et al. [17] and Martínez-Martínez et al. [19] defined ELM based on Lasso ($\hat{\boldsymbol{\beta}}_{Lasso-ELM}$) as the solution of the objective function

$$\|\mathbf{S}\boldsymbol{\beta} - \mathbf{Y}\|_2 + \lambda\,\|\boldsymbol{\beta}\|_1 .$$

The main idea behind $Lasso-ELM$ is to select the nodes which minimize the error by shrinking some of the nodes to exactly zero. The sparsity property provides (i) compactness, (ii) stability, (iii) generalization ability, (iv) interpretability and (v) speed. The solution of $Lasso-ELM$ can be obtained via LARS-EN algorithm [23] or algorithm of Sjöstrand et al. [29] which is based on the LARS algorithm [30] and the algorithm with piecewise linear regularization path proposed by Rosset and Zhu [31]. The details of these algorithms can be found in Sjöstrand et al. [29].

Although Lasso does shrinkage and automatic variable selection simultaneously, it has disadvantages in linear regression. Similarly, in ELM-based studies, Lasso-ELM has same disadvantages. Therefore, Miche et al. [17] proposed ELM based on ENet which has the grouping effect and simultaneously the shrinkage and automatic variable selection properties and has the objective function

$$\|\mathbf{S}\boldsymbol{\beta} - \mathbf{Y}\|_2 + \frac{k}{2}\|\boldsymbol{\beta}\|_2 + \lambda\|\boldsymbol{\beta}\|_1.$$

To solve the ENet-ELM, Miche et al. [17] and Luo et al. [13] applied the LARS-EN algorithm.

The Proposed Algorithm: GO-ELM

An algorithm holding the sparsity property can easily outperform any other algorithm in terms of speed, compactness and generalization performance. To gain OK-ELM the sparsity capability, we propose a new algorithm called GO-ELM which has triple shrinkage. The first two shrinkage effects are same with those of ridge and Lasso as in ENet and the third shrinkage factor makes the result closer to the real value than ENet. We define the objective function of the GO-ELM as:

$$D(\beta,k,\lambda) = \|\mathbf{S}\boldsymbol{\beta} - \mathbf{Y}\|_2 + \frac{k}{2}\left\|d\widehat{\beta}_{ELM} - \boldsymbol{\beta}\right\|_2 + \lambda\|\boldsymbol{\beta}\|_1. \quad (5)$$

In the proposed approach, the characteristic features of both ridge and Liu estimators to handle the multicollinearity problem occurring in the hidden layer of output matrix, are considered in learning process of algorithm. Additionally, the sparsity capability has been added via L_1-norm simultaneously with ridge and Liu estimators.

The objective function of GO-ELM can be written in augmented form given in Eq. (5) as follows:

$$D(\beta,k,\lambda) = \left\|\tilde{\mathbf{S}}\boldsymbol{\beta} - \tilde{\mathbf{Y}}\right\|_2 + \lambda\|\boldsymbol{\beta}\|_1 \quad (6)$$

where $\tilde{\mathbf{S}} = \begin{pmatrix}\mathbf{S}\\ \sqrt{k}\mathbf{I}_{\varphi}\end{pmatrix}$ and $\tilde{\mathbf{Y}} = \begin{pmatrix}\mathbf{Y}\\ \sqrt{k}d\hat{\boldsymbol{\beta}}_{ELM}\end{pmatrix}$ are the augmented form of the hidden layer and output matrices. The advantage of writing Eq. (6) in form Eq. (5) is to solve the GO-ELM problem as an L_1-norm problem. Therefore, various types of algorithm like LARS can be used to obtain the optimal solution of Eq. (6) via Eq. (5). The GO-ELM has the following properties which show that it is a generalization of ENet-ELM, Lasso-ELM, Liu-ELM, R-ELM and OK-ELM:

- As d goes to 0, GO-ELM behaves similar to ENet-ELM
- When $\lambda = 0$, GO-ELM and OK-ELM give same solutions.
- When $k = 1$, GO-ELM solution is same with Liu-ELM solution and when $k = 0$, GO-ELM solution is same with Lasso-ELM solution. Then, GO-ELM solutions trace a curve path through the parameter space from the Liu-ELM to Lasso-ELM as k goes from 1 to 0.
- When $\lambda = 0$ and $d = 0$, GO-ELM gives R-ELM solutions
- Since GO-ELM solutions can also be obtained from Eq. (6), it does variable selection as Lasso-ELM and ENet-ELM.
- The GO-ELM is strictly convex which can be seen from the equivalent objective function:

  $$\|\mathbf{S}\boldsymbol{\beta} - \mathbf{Y}\|_2 + \lambda_1\left(\frac{1-\alpha}{2}\left\|d\widehat{\beta}_{ELM} - \boldsymbol{\beta}\right\|_2 + \alpha\|\boldsymbol{\beta}\|_1\right)$$

  where $\lambda_1 = k+\lambda, \alpha = \lambda/(\lambda+k)$ and $\lambda_1 \geq 0, 0 \leq \alpha \leq 1$ are tuning parameters. The GO-ELM solutions trace a curve path through the parameter space from the OK-ELM to Lasso-ELM as α goes from 0 to 1.

Choice of Tuning Parameters

GO-ELM depends on three tuning parameters which can be selected similarly to Lasso-ELM and ENet-ELM. First, the d values in the interval (0,1) are selected, and for each d value, a k value is selected for the k grid values in the specified range. Then, for each (d,k) pair, GO-ELM is solved by any algorithm such as LARS-EN [23, 29] to select λ via

K-fold cross-validation. The cross-validation is done on a three-dimensional surface. By cross-validating the training data by considering every possible combination of three different tuning parameters, the best performing values were determined as optimum for the current trail. The tuning parameters are then chosen so as to give the smallest cross-validation error.

The pseudo-code for GO-ELM algorithm is as given by Algorithm 1.

Algorithm 1 GO-ELM Algorithm

Require: Training set $\{(x_i, y_i)\}$, the maximum number of hidden neurons $\{\omega\}$, an activation function $\{g(.)\}$, number of trials $\{L\}$.

Ensure: The β weight matrix.

1: Randomly generate the initial parameters $\mathbf{w}_i$ and b_i, $1 \leq i \leq \omega$.
2: Obtain $\mathbf{S}$ and the basic ELM solution via Eq. (2).
3: Find the solution of the objective function of GO-ELM given by Eq. (6) via LARS-EN algorithm as $\boldsymbol{\beta}$ =LARS-EN$\left(\tilde{\mathbf{S}}, \tilde{\mathbf{Y}}, \hat{d}, \hat{k}, \hat{\lambda}\right)$. for $1 \leq i \leq$L).
4: **repeat**
5: Calculate $\boldsymbol{\beta}$ solution for each combination k, d, and λ grid parameters.
6: Find the optimal solution providing smallest error based on LARS-EN algorithm.
7: **until** Convergence
8: Repeat this process via k-fold cross-validation for L trials.
9: Calculate the global optimum weight matrix and its performance metrics.

Sparsity Property

Figure 1 is given to show the sparsity property of GO-ELM and to compare the coefficient paths for GO-ELM, Lasso-ELM and ENet-ELM under Body Fat data set. In order to give better insights as visually, the number of hidden layer is fixed to 20. The differences between the coefficient paths of Lasso-ELM, ENet-ELM and GO-ELM algorithms are noticeable, but (ii) and (iii) exhibit a similar structure. This is because the value of d is a small value. While k is constant, the value of d is effective in determining the difference between ENet-ELM and GO-ELM algorithms. Figure 1 proves that all three algorithms do variable selection and the ways to select variables are different from each other.

Grouping Property

Grouping effect property of the ENet states that highly correlated features will have similar estimated coefficients. The property is described formally in terms of an upper bound on the difference between two coefficients as it relates to the correlation between the predictors (e.g., Zou and Hastie [23], Theorem 1, Zhou [32], Theorem 2).

The grouping effect of GO-ELM is encouraged from the tuning parameter k and it is stated by Theorem 1.

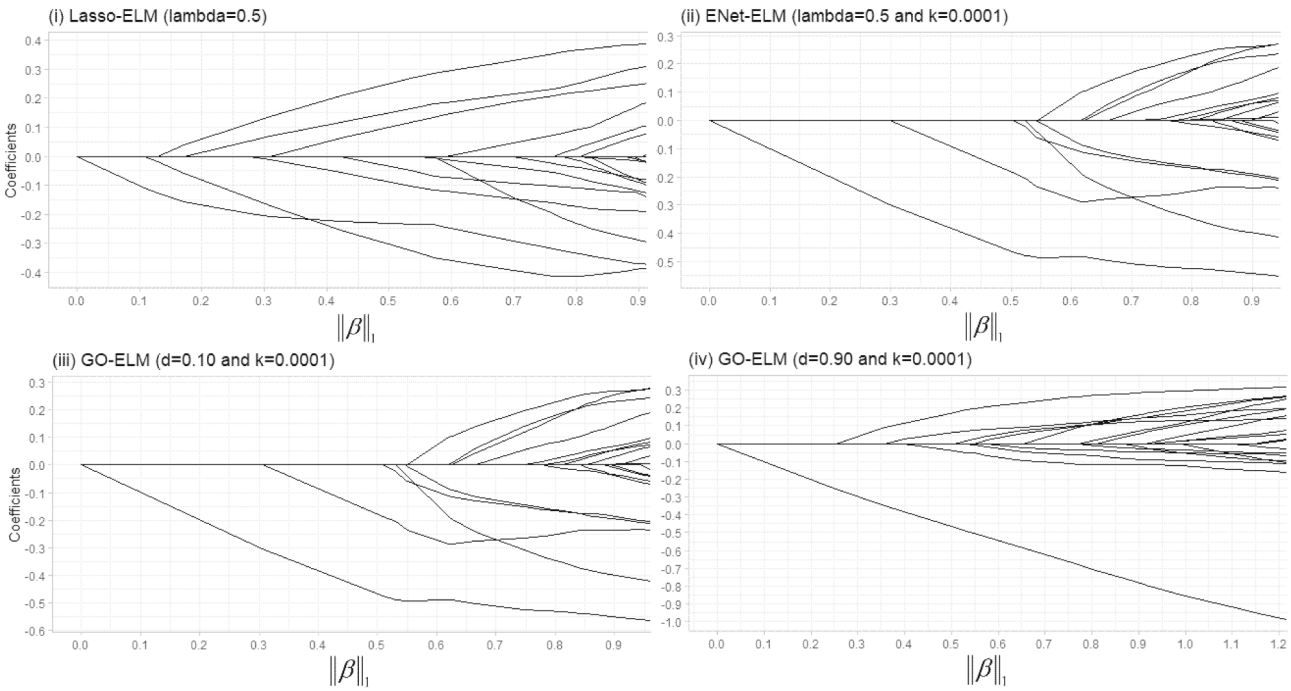

Fig. 1 A comparison of the coefficient paths for variable selection

Theorem 1 *Given* $(\mathbf{Y}, \mathbf{S})$ *and parameters* λ, k, *and* d *are fixed. If* $\widehat{\beta}_i \widehat{\beta}_t > 0$, *then*

$$\begin{aligned}&\left(\widehat{\beta}_i - \widehat{\beta}_t - d(\widehat{\beta}_{ELM_i} - \widehat{\beta}_{ELM_j})\right)\\ &\quad \leq \frac{1}{k}\|h_i - h_t\|_2 \left(\|\mathbf{Y}\|_2 + \frac{kd^2}{2}\|\widehat{\beta}_{ELM}\|_2\right)\end{aligned}$$

where $\widehat{\beta}$ *is the GO-ELM solution.*

Proof Let us first take the derivative of Eq.(5) with respect to β_i and β_t and set the results to zero:

$$\begin{aligned}\left.\frac{\partial D(.)}{\partial \beta_i}\right|_{\beta_i=\widehat{\beta}_i} &= 2h_i^T(\mathbf{S}\boldsymbol{\beta} - \mathbf{Y})\\ &\quad + k\left(\widehat{\beta}_i - d\widehat{\beta}_{ELM_i}\right)\\ &\quad + \lambda s_i = 0\end{aligned} \tag{7}$$

$$\begin{aligned}\left.\frac{\partial D(.)}{\partial \beta_t}\right|_{\beta_t=\widehat{\beta}_t} &= 2h_t^T(\mathbf{S}\boldsymbol{\beta} - \mathbf{Y})\\ &\quad + k\left(\widehat{\beta}_t - d\widehat{\beta}_{ELM_t}\right)\\ &\quad + \lambda s_t = 0\end{aligned} \tag{8}$$

where

$$s_i = \begin{cases} -1, & z < 0 \\ [-1, 1], & z = 0 \\ 1, & z > 0 \end{cases}$$

and s_i and s_t have the same sign. Subtracting Eq. (8) from Eq. (7), we obtain

$$\begin{aligned}&\left(\widehat{\beta}_t - d\widehat{\beta}_{ELM_t}\right) - \left(\widehat{\beta}_i - d\widehat{\beta}_{ELM_i}\right)\\ &\quad = \frac{1}{k}\left(h_t^T - h_i^T\right)(\mathbf{S}\boldsymbol{\beta} - \mathbf{Y}).\end{aligned} \tag{9}$$

Applying the Cauchy-Schwarz inequality to Eq. (9), we get

$$\begin{aligned}&\left|\left(\widehat{\beta}_t - \widehat{\beta}_i\right) - d\left(\widehat{\beta}_{ELM_t} - \widehat{\beta}_{ELM_i}\right)\right|\\ &\leq \frac{1}{k}\|h_t - h_i\|_2 \|\mathbf{S}\boldsymbol{\beta} - \mathbf{Y}\|_2.\end{aligned}$$

Since $\widehat{\beta}$ is a minimizer of Eq. (6), by taking $\beta = 0$, we see that $D(\widehat{\beta}, k, \lambda) \leq \|\mathbf{Y}\|_2 + \frac{k}{2}\left\|d\widehat{\beta}_{ELM}\right\|_2$ and we write the inequality

$$\|\mathbf{S}\boldsymbol{\beta} - \mathbf{Y}\|_2 \leq D(\widehat{\beta}, k, \lambda) \leq \|\mathbf{Y}\|_2 + \frac{k}{2}d^2\left\|\widehat{\beta}_{ELM}\right\|_2. \tag{10}$$

With respect to Eq. (10), we complete the proof.

If h_i, $i = 1, \ldots, \varphi$ are standardized as $\sum_{j=1}^{N} h_{ij} = \sum_{j=1}^{N} g\left(\omega_i.\mathbf{x}_j + b_i\right) = 0$ and $\sum_{j=1}^{N} h_{ij}^2 = \sum_{j=1}^{N} g^2\left(\omega_i.\mathbf{x}_j + b_i\right) = 1$, the inequality in Theorem 1 reduces to

$$\begin{aligned}&\left|\left(\widehat{\beta}_i - \widehat{\beta}_t\right) - d\left(\widehat{\beta}_{ELM_i} - \widehat{\beta}_{ELM_t}\right)\right|\\ &\quad \leq \frac{1}{k}\sqrt{2\left(1 - h_i^T h_t\right)}\left(\|\mathbf{Y}\|_2 + + \frac{kd^2}{2}\left\|\widehat{\beta}_{ELM}\right\|_2\right)\end{aligned}$$

where $h_i^T h_t$ is the correlation between h_i and h_t.

When h_i and h_t are highly correlated, $h_i^T h_t$ goes to 1 (if it goes to -1 then consider $-h_t$). In this case, Theorem 1 says that the difference between the coefficient paths of h_i and h_t is almost 0 (this result is the same with the linear regression as Zou and Hastie [23] and Zhou [32] stated).

Performance Evaluation

Data Sources

In this study, Body Fat, Boston Housing, Machine CPU, Servo, Fish, Heart and Abalone data sets have been retrieved from the UCI database and have been considered to compare the performance of all algorithms. Their properties are tabulated in Table 1.

Experimental Settings

The design of the experiment plays a critical role on the performance of each algorithm considered in this study. The process followed in this study can be summarized as follows:

- The inputs were standardized with a mean of zero and a variance of one so that possible performance losses due to data structure (such as scale and unit properties) was avoided.

Table 1 The properties of data sets used in this study

Data sets	Sample Size	Attributes	Training Data Size	Test Data Size
Body Fat	252	15	177	75
Boston Housing	506	14	355	151
Machine CPU	209	7	146	63
Servo	167	5	117	50
Fish	908	7	636	272
Heart	303	14	212	91
Abalone	4177	9	2924	1253

- Model training was initialized with random weight and bias values. In order to eliminate the effect of this randomness, each trial was performed using fixed starting points (i.e., seed values) by grid searching over the same range of tuning parameters depending on each algorithm.
- The sine function, which is one of the commonly used functions in ELM models, was used as the activation function.
- The data set was split into 70% and 30% training and test data, respectively. According to this splitting, the sample size of the training and test data sets for each data set is given in Table 1. The algorithms were trained using the training data and a 5-fold cross-validation (CV) was applied to obtain the tuning parameters on the training data. Then, the performance analysis on the test data was obtained.
- Too few neurons in the hidden layers to adequately detect signals in a complex data set can cause poor fit. An excessive number of neurons in the hidden layers can increase the time required to train the ELM. Training time can increase to a point where it is impossible to adequately train the ELM. To reach a compromise between too many and too few neurons in the hidden layers, the number of hidden layer neurons was taken as 100 and fixed across all data sets.
- The number of trials was set at twenty and the average of the trial results was reported.
- A grid search approach was used to determine the tuning parameters. Since different parameters are effective in training each algorithm, all algorithms were trained as a whole for all possible combinations of (d, k and λ) values. In Lasso, the conventional tuning parameter is the L_1-norm of the coefficients or the fraction of each regression coefficient estimates to the L_1-norm (s) (e.g., [23]). The advantage of s, is that it is valued within [0, 1]. Therefore, Lasso tuning parameter λ is reported through s. The grid of values for the parameters are picked as:

$$d \in \left[10^{-5}, 10^{-4}, .., 10^{-2}, 0.02, 0.03..., 0.1, 0.15, 0.2, ..., 1\right],$$

$$k \in \left[10^{-5}, 10^{-4}, .., 10^{-2}, 0.02, ..., 0.1, 0.2, ..., 1, 1.5, 2, ..., 5\right],$$

$$s \in \left[10^{-5}, 10^{-4}, .., 10^{-2}, 0.02, 0.03..., 0.1, 0.15, 0.2, ..., 1\right].$$

To get the optimum (d, k) pairs for the OK-ELM, 5-fold CV is conducted on two-dimensional space and to get the optimum (d, k, s) pairs for the GO-ELM, 5-fold CV is conducted on three-dimensional space. In each fold, the tuning parameters minimizing the CV error are determined as optimal for the corresponding fold. This process is repeated for twenty trials and the mean values of the tuning parameters calculated as overall for all folds are reported.
- The Norm (Mean), Norm (SD), RMSE and SD criteria were used for the performance evaluation. The optimal tuning parameters found through the training data set are used to produce the lowest RMSE value in each trial and the average of the statistics are presented in Table 2.

The number of nodes, the norm values and standard deviations of the obtained coefficients are presented in Table 2.

The changes in the comparisons are also presented by Figs. 2 and 3 where the relative reduction rate is as computed as

$$RR = \frac{RMSE_{any\ alg.} - RMSE_{GO-ELM}}{RMSE_{any\ alg.}} \times 100.$$

Some of the results provided through Figs. 2 and 3 with Table 2 are listed as follows:

- From the view of training RMSE, the GO-ELM algorithm outperforms the ELM, R-ELM, Lasso-ELM and ENet-ELM algorithms for all data sets except the Heart data. The superiority of GO-ELM over ELM, R-ELM, Lasso-ELM and ENet-ELM is more than by an average of 40% for RMSE (compared to other algorithms, GO-ELM gives a minimum of 7% and a maximum of 95% improvement) and more than by an average of 80% for SD (compared to other algorithms, GO-ELM gives a minimum of 25% and a maximum of 99% improvement).
- GO-ELM is superior over ELM, R-ELM, Lasso-ELM and ENet-ELM with more than by an average of 20% in the sense of testing RMSE (minimum 2%, maximum 92% improvement), this superiority is more than by an average of 60% in the sense of SD (minimum 54%, maximum 99% improvement).

Results and Discussion

Table 2 gives the performance comparison of all algorithms based on the fixed parameters for the proposed algorithm and the optimal tuning parameter values for each algorithm itself, respectively. The performance measures such as Norm (Mean), Norm (SD), RMSE and SD were all computed on the test data sets using the optimal tuning parameters obtained through training data set. Since the training data sets are used for optimum tuning parameter selection, the comparisons will be made according to the test data. When the performance comparison is performed using the parameters for which the proposed algorithm is optimal, the results given in Table 2 are achieved.

Table 2 The performance comparisons of algorithms over their own optimum tuning parameters

Data	Algorithm	*d*	*k*	*s*	# of Nodes[a]	Norm (Mean)	Norm (SD)	Training Data		Test Data	
								RMSE	SD	RMSE	SD
Body Fat	ELM	*	*	*	100.00	1.7549	0.2014	0.9632	0.2377	0.9579	0.2325
	R-ELM	*	5	*	100.00	1.2152	**0.0840**	0.8001	0.1438	0.7931	0.1458
	OK-ELM	0.0001	5	*	100.00	1.2052	0.0844	**0.2990**	**0.0237**	0.5791	0.0686
	Lasso-ELM	*	*	0.47	39.08	1.4612	0.1908	0.6903	0.2033	0.6795	0.2082
	ENet-ELM	*	0.0001	0.59	44.60	1.6518	0.2242	0.5885	0.2045	0.6816	0.1922
	GO-ELM	0.3	0.00001	0.53	35.16	**1.0823**	0.1535	0.3437	0.0381	**0.5283**	**0.0664**
Boston	ELM	*	*	*	100.00	2.0858	0.1596	1.2510	0.4625	1.2318	0.4586
	R-ELM	*	5	*	100.00	1.5605	0.0909	1.1038	0.3671	1.0872	0.3554
	OK-ELM	0.0001	5	*	100.00	1.5606	**0.0900**	**0.4067**	**0.0327**	0.6019	**0.0434**
	Lasso-ELM	*	*	0.26	39.96	1.4694	0.1775	0.9395	0.1913	0.9200	0.1847
	ENet-ELM	*	0.001	0.33	41.28	1.4987	0.1782	0.8892	0.2114	0.9874	0.1741
	GO-ELM	0.95	0.01	0.53	38.64	**1.2801**	0.1839	0.4468	0.0440	**0.5962**	0.0481
MachineCPU	ELM	*	*	*	100.00	3.1271	0.7724	1.2829	0.6026	1.0442	0.0742
	R-ELM	*	5	*	100.00	1.6069	**0.0883**	1.1465	0.2474	0.7513	0.3340
	OK-ELM	0.03	5	*	100.00	1.8895	0.1304	0.2115	**0.0156**	**0.2553**	0.0413
	Lasso-ELM	*	*	0.09	38.64	1.0299	0.3677	1.0093	0.1277	0.5483	0.1893
	ENet-ELM	*	0.001	0.19	38.60	**1.0278**	0.3942	0.9893	0.1102	0.5199	0.1692
	GO-ELM	0.45	0.01	0.62	54.35	2.0779	0.1416	**0.1844**	0.0227	0.2554	**0.0399**
Servo	ELM	*	*	*	100.00	10.7167	1.8277	4.7303	2.5689	4.7282	2.5862
	R-ELM	*	5	*	100.00	**0.9697**	**0.0563**	0.8932	0.2138	0.9093	0.2155
	OK-ELM	0.05	1	*	100.00	5.6507	0.8669	**0.2529**	**0.0058**	0.3383	0.0184
	Lasso-ELM	*	*	0.26	41.20	1.4476	0.2090	0.9192	0.2237	0.9670	0.2220
	ENet-ELM	*	0.1	0.45	38.63	1.3211	0.2111	0.7864	0.2302	0.9009	0.1689
	GO-ELM	0.45	0.02	0.81	80.79	6.1466	1.4467	0.2591	0.0089	**0.3373**	**0.0182**
Fish	ELM	*	*	*	100.00	4.0606	0.8500	2.3368	1.1685	2.3560	1.1540
	R-ELM	*	0.88	*	100.00	2.4286	0.1393	1.6976	0.6028	1.7160	0.6055
	OK-ELM	0.25	0.90	*	100.00	2.4298	0.1398	0.5750	0.0085	0.7145	0.0473
	Lasso-ELM	*	*	0.18	33.84	1.1287	0.2421	0.8688	0.1202	0.9068	0.1060
	ENet-ELM	*	0.01	0.24	35.40	1.1677	0.2506	0.7660	0.1196	0.9220	0.1119
	GO-ELM	0.62	0.02	0.81	27.84	**0.8702**	**0.1223**	**0.5077**	**0.0058**	**0.6004**	**0.0232**
Heart	ELM	*	*	*	100.00	1.4190	0.1201	1.0909	0.1628	1.0673	0.1829
	R-ELM	*	1.30	*	100.00	1.3558	0.1068	1.0818	0.1545	1.0593	0.1742
	OK-ELM	0.09	0.80	*	100.00	1.3550	**0.1059**	1.0747	0.0298	1.1960	0.0628
	Lasso-ELM	*	*	0.27	33.12	0.6143	0.1177	**0.9831**	0.0346	**0.9973**	**0.0290**
	ENet-ELM	*	0.001	0.11	33.55	0.6210	0.1108	0.9880	0.0344	0.9987	0.0311
	GO-ELM	0.001	1.26	0.28	33.16	**0.6074**	0.1131	0.9963	**0.0258**	1.0351	0.0344
Abalone	ELM	*	*	*	100.00	3.6096	0.4124	1.3704	0.3651	1.3809	0.3869
	R-ELM	*	0.62	*	100.00	2.9142	**0.2143**	1.2118	0.2542	1.2213	0.2759
	OK-ELM	0.16	0.58	*	100.00	2.8848	0.2144	**0.6381**	0.0065	0.6570	0.0074
	Lasso-ELM	*	*	0.09	35.08	1.1570	0.2565	0.9836	0.1762	0.9979	0.1726
	ENet-ELM	*	0.01	0.11	34.96	1.1488	0.2561	0.9987	0.1849	0.9892	0.1706
	GO-ELM	0.01	0.74	0.32	34.72	**1.0561**	0.2596	0.6459	**0.0029**	**0.6278**	**0.0044**

Values in bold correspond to the best (optimal) values

[a]The node numbers of Lasso-ELM, ENet-ELM and GO-ELM are the average on twenty trails. Therefore, they are in decimal but can be rounded to integer

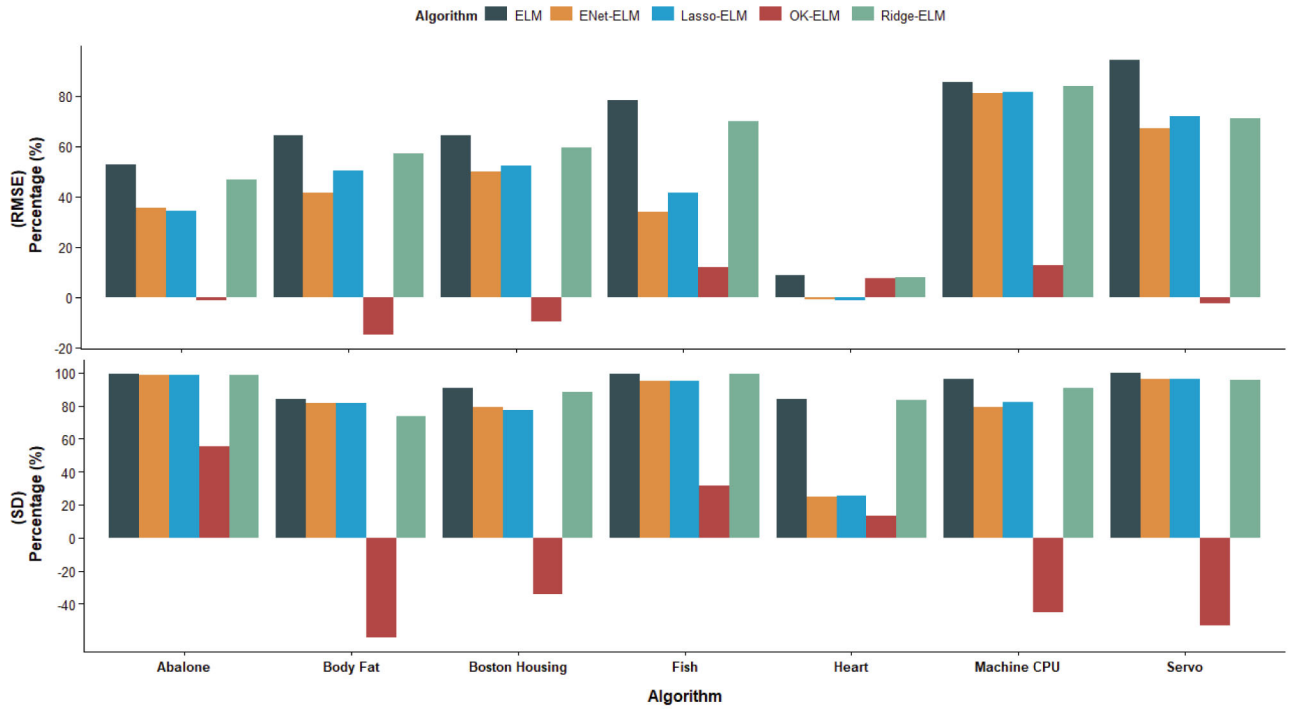

Fig. 2 Comparison of GO-ELM with other algorithms via RR percentage of RMSE and SD for training data set

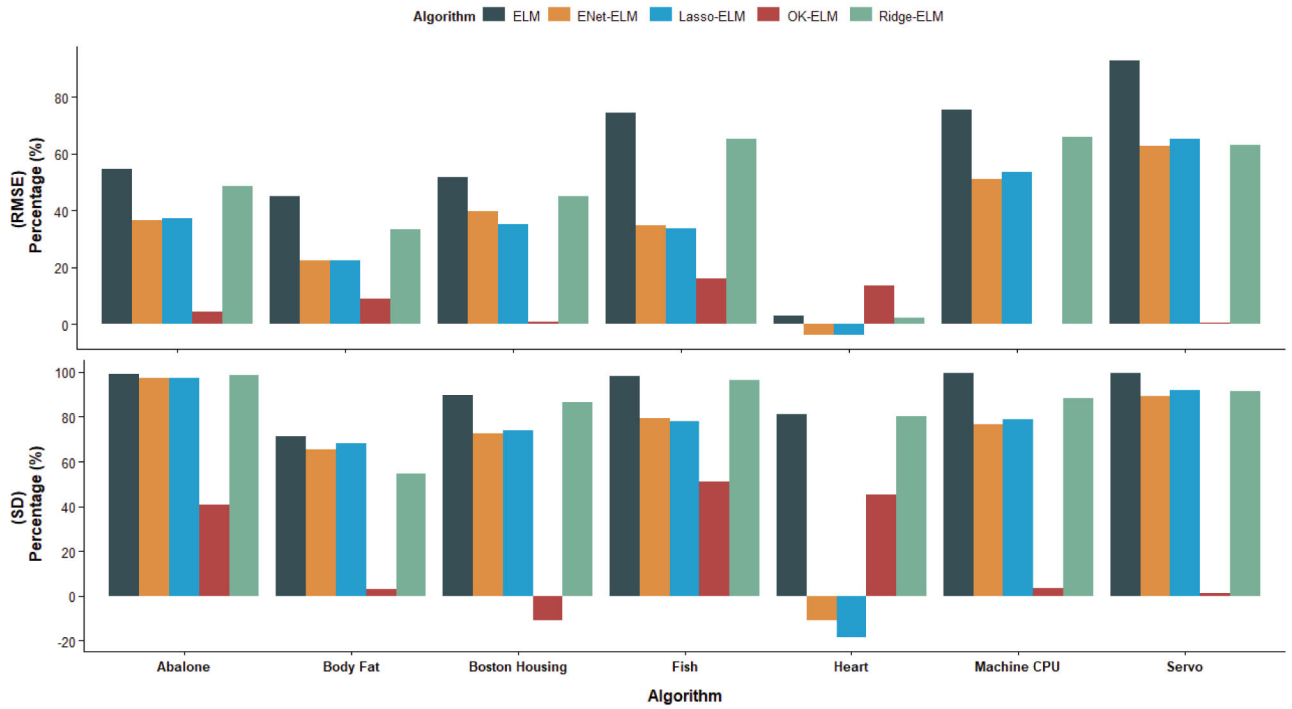

Fig. 3 Comparison of GO-ELM with other algorithms via RR percentage of RMSE and SD for test data set

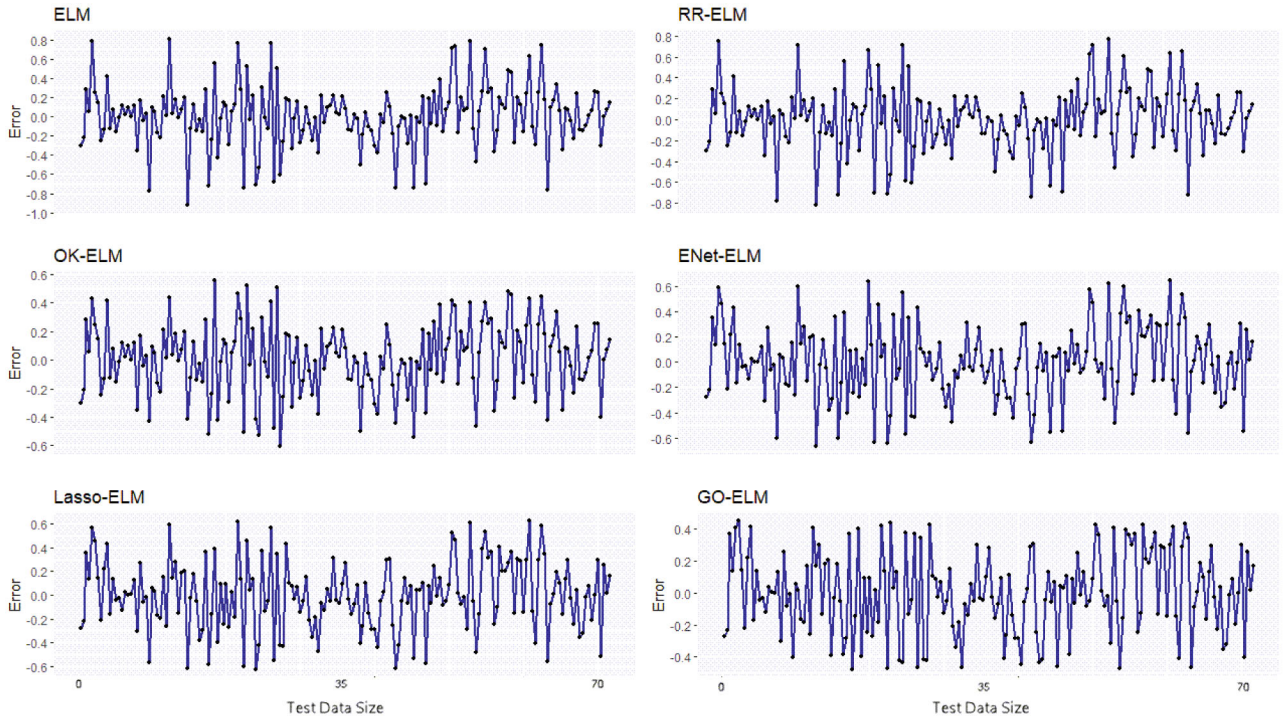

Fig. 4 The change of testing errors for Body fat data set

The comprehensive evaluation of the tuning parameters in the grid space using the optimal values for each algorithm is presented in Table 2. The important results from Table 2 are as follows:

- Considering the test data, GO-ELM outperforms OK-ELM on the RMSE criterion in all other data sets except the MachineCPU data set. The superiority percentage of OK-ELM over GO-ELM in the MachineCPU data set

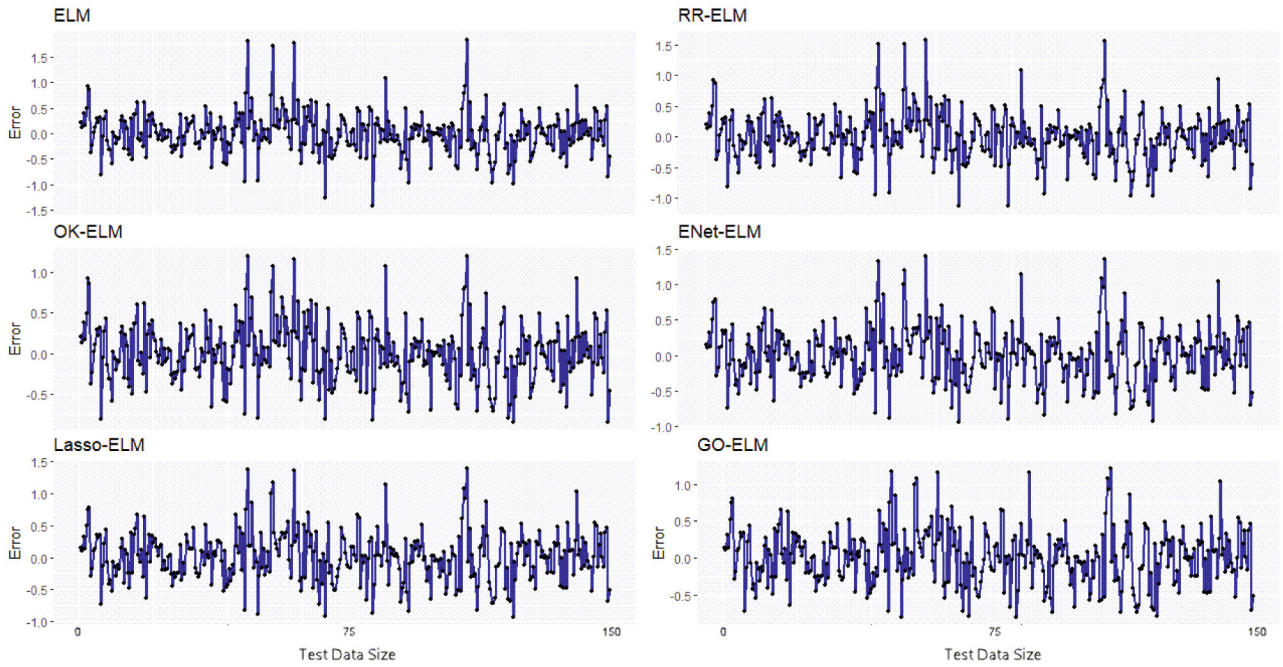

Fig. 5 The change of testing errors for Boston data set

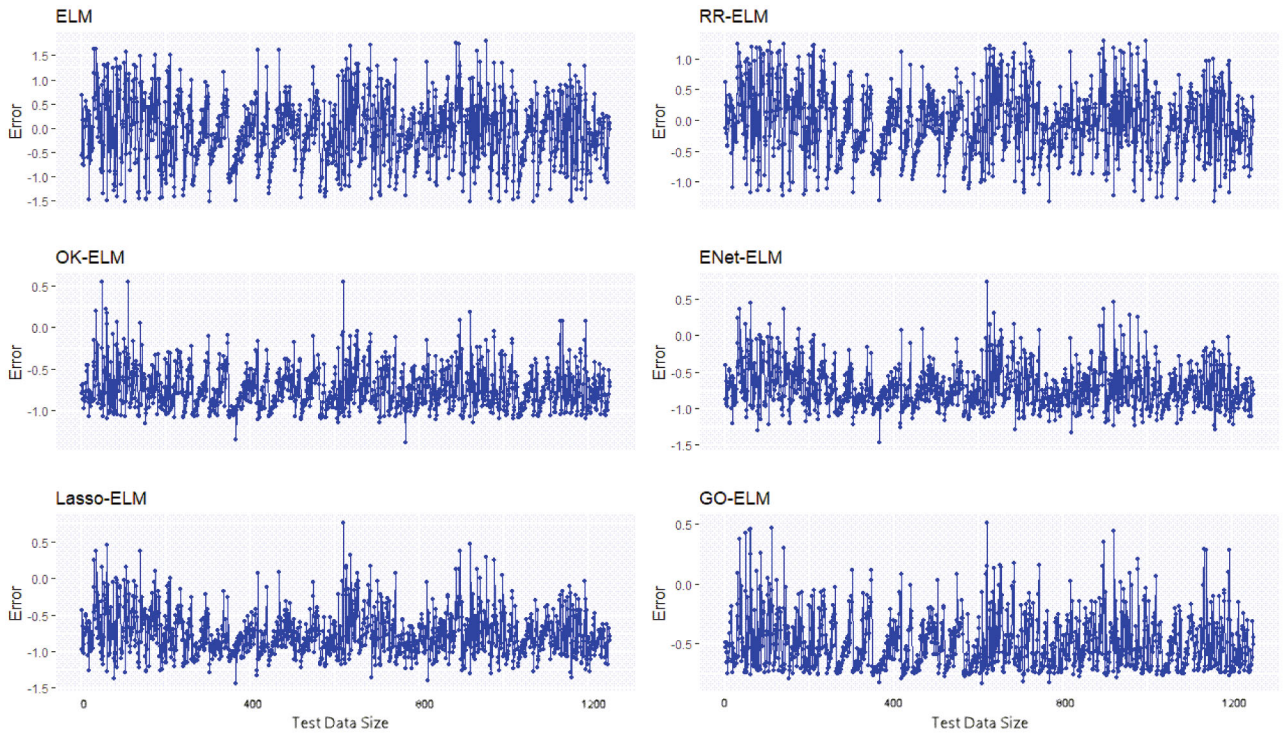

Fig. 6 The change of testing errors for Abalone data set

is only 0.039%. According to the test data set, except for the Boston data set, GO-ELM outperforms OK-ELM under the SD criterion. In all test data sets except the Heart test data set, GO-ELM shows better RMSE and SD performances than ELM, R-ELM, Lasso-ELM, and ENet-ELM. The testing RMSE error of GO-ELM for the data sets except Machine CPU and Heart is the smallest value when compared to its competitors.

- Considering the number of nodes for all data sets, it is seen that ELM, R-ELM and OK-ELM do not select nodes, while Lasso-ELM, ENet-ELM and GO-ELM do node selection. This is an indication that Lasso-ELM, ENet-ELM and GO-ELM have been proposed as solutions to the sparsity problem.
- The norm of GO-ELM is less than ELM in all data sets. Except for the Servo and MachineCPU data, the mean of the norm of GO-ELM is smaller than the other algorithms, even if the standard error of the norm is not small in all other data sets.

On the test data, the changes of the residual values of the algorithms trained at their optimum tuning parameters are presented in Figs. 4, 5, and 6 for Body, Boston and Abalone data sets.

In order to evaluate the stability of the algorithms, the narrow range of residual values is preferable. Figures 4, 5, and 6 show that the stability of GO-ELM algorithm over the other algorithms are obvious for Body Fat and Abalone data sets. Considering Fig. 5 for the Boston data set, GO-ELM and

Table 3 Statistical comparison results of GO-ELM algorithm with each algorithm based on test data predictions

Data	GO-ELM vs ELM	GO-ELM vs R-ELM	GO-ELM vs OK-ELM	GO-ELM vs Lasso-ELM	GO-ELM vs Enet-ELM
Body Fat	< 0.001	< 0.001	0.0046	0.001	0.0006
Boston	< 0.001	< 0.001	0.1392	< 0.001	< 0.001
MachineCPU	< 0.001	< 0.001	0.1988	< 0.001	< 0.001
Servo	< 0.001	< 0.001	0.1728	< 0.001	< 0.001
Fish	< 0.001	< 0.001	< 0.001	< 0.001	< 0.001
Heart	0.0896	0.1098	< 0.001	0.0002	0.0002
Abalone	< 0.001	< 0.001	< 0.001	< 0.001	< 0.001

OK-ELM algorithms show almost the same stability performance and it is seen that both algorithms outperform other algorithms. In order to determine whether the performance of the proposed algorithm is statistically significant compared to other algorithms, paired samples t-test with Bonferroni correction was used on the test data predictions (i.e., errors) and the results are given in Table 3. According to these results, it is mostly observed that the performance difference is statistically significant.

Conclusion

The aim of the current study is to present a new regularized ELM algorithm based on the simultaneous use of both ridge and Liu regression with Lasso regression to improve the ELM and its variants to cope with some drawbacks such as instability, poor generalization performance and lack of sparsity. This new algorithm has been proposed as a solution to multicollinearity and sparsity problems such as Lasso-ELM and ENet-ELM by both node selection and shrinkage. After conducting experimental study via real-world applications, the findings indicate that the proposed algorithm provides effective solutions to the mentioned drawbacks and yields more stable and sparse performance with better generalization capability than its competitors.

Limitations and Ideas for Future Works

Although the proposed algorithm provides significant improvements on the performance of ELM and its variants, the major limitation of this study is the method of selection of tuning parameters. Some solid and analytical or genetic algorithm-based approaches should be considered in the future works. Secondly, the proposed algorithm cannot be used in high-dimensional data and it should be possible to make an adaptation that can be used in high-dimensional data. Additionally, keeping the number of hidden neurons fixed can be considered as another limitation. Future studies may include finding the optimum number of neurons using ELM and then choosing a sparse model using the number of neurons found.

Author Contribution Hasan Yıldırım: methodology, investigation, software, data curation, formal analysis, visualization, writing — original draft. M. Revan Özkale: methodology, conceptualization, supervision, writing — review editing, validation.

Data Availability The datasets generated during and/or analyzed during the current study are available in the UCI repository (https://archive-beta.ics.uci.edu/).

Declarations

Competing Interest The authors declare no competing interests.

References

1. Huang GB, Zhu QY, Siew CK. Extreme learning machine: a new learning scheme of feedforward neural networks. In: 2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No. 04CH37541) (Vol. 2). IEEE; 2004. p. 985–90.
2. Huang GB, Zhu QY, Siew CK. Extreme learning machine: Theory and applications. Neurocomputing. 2006;70(1–3):489–501.
3. Huang GB, Zhou H, Ding X, Zhang R. Extreme learning machine for regression and multiclass classification. IEEE Trans Syst Man Cybernet B (Cybernet). 2011;42(2):513–29.
4. Hoerl AE, Kennard RW. Ridge regression: Applications to nonorthogonal problems. Technometrics. 1970;12(1):69–82.
5. Deng W, Zheng Q, Chen L. Regularized extreme learning machine. In: 2009 IEEE Symposium on Computational Intelligence and Data Mining. IEEE; 2009. p. 389–95
6. Li G, Niu P. An enhanced extreme learning machine based on ridge regression for regression. Neural Comput Appl. 2013;22:803–10.
7. Huang WB, Sun FC. Building feature space of extreme learning machine with sparse denoising stacked-autoencoder. Neurocomputing. 2016;22(174):60–71.
8. Shao Z, Er MJ. Efficient leave-one-out cross-validation-based regularized extreme learning machine. Neurocomputing. 2016; 19(194):260–70.
9. Chen YY, Wang ZB. Novel variable selection method based on uninformative variable elimination and ridge extreme learning machine: CO gas concentration retrieval trial. Guang pu xue yu guang pu fen xi= Guang pu. 2017;37(1):299–305.
10. Yu Q, Miche Y, Eirola E, Van Heeswijk M, Séverin E, Lendasse A. Regularized extreme learning machine for regression with missing data. Neurocomputing. 2013;15(102):45–51.
11. Wang H, Li G. Extreme learning machine Cox model for high-dimensional survival analysis. Stat Med. 2019;38(12):2139–56.
12. Yildirim H, Özkale MR. The performance of ELM based ridge regression via the regularization parameters. Expert Syst Appl. 2019;15(134):225–33.
13. Luo X, Chang X, Ban X. Regression and classification using extreme learning machine based on L1-norm and L2-norm. Neurocomputing. 2016;22(174):179–86.
14. Kejian L. A new class of blased estimate in linear regression. Commun Stat Theor Methods. 1993;22(2):393–402.
15. Yıldırım H, Özkale MR. An enhanced extreme learning machine based on Liu regression. Neural Process Lett. 2020;52:421–42.
16. Tibshirani R. Regression shrinkage and selection via the lasso. J R Stat Soc Ser B Stat Methodol. 1996;58(1):267–88.
17. Miche Y, Sorjamaa A, Bas P, Simula O, Jutten C, Lendasse A. OP-ELM: Optimally pruned extreme learning machine. IEEE Trans Neural Netw. 2009;21(1):158–62.
18. Miche Y, Van Heeswijk M, Bas P, Simula O, Lendasse A. TROP-ELM: a double-regularized ELM using LARS and Tikhonov regularization. Neurocomputing. 2011;74(16):2413–21.
19. Martínez-Martínez JM, Escandell-Montero P, Soria-Olivas E, Martín-Guerrero JD, Magdalena-Benedito R, Gómez-Sanchis J. Regularized extreme learning machine for regression problems. Neurocomputing. 2011;74(17):3716–21.
20. Shan P, Zhao Y, Sha X, Wang Q, Lv X, Peng S, Ying Y. Interval lasso regression based extreme learning machine for nonlinear multivariate calibration of near infrared spectroscopic datasets. Anal Methods. 2018;10(25):3011–22.
21. Li R, Wang X, Lei L, Song Y. $L_2$1-norm based loss function and regularization extreme learning machine. IEEE Access. 2018;18(7):6575–86.
22. Preeti, Bala R, Dagar A, Singh RP. A novel online sequential extreme learning machine with L 2, 1-norm regularization for prediction problems. Appl Intell. 2021;51:1669–89.

23. Zou H, Hastie T. Regularization and variable selection via the elastic net. J R Stat Soc Ser B Stat Methodol. 2005;67(2):301–20.
24. Yıldırım H, Özkale MR. LL-ELM: a regularized extreme learning machine based on L 1-norm and Liu estimator. Neural Comput Appl. 2021;33(16):10469–84.
25. Rao CR, Mitra SK. Generalized inverse of a matrix and its applications. In: Proceedings of the Sixth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Theory of Statistics (Vol. 6). University of California Press; 1972. p. 601–21.
26. Schott JR. Matrix analysis for statistics. John Wiley & Sons; 2016.
27. Tutz G, Binder H. Boosting ridge regression. Comput Stat Data Anal. 2007;51(12):6044–59.
28. Yıldırım H, Özkale MR. A combination of ridge and Liu regressions for extreme learning machine. Soft Comput. 2023; 27(5):2493–508.
29. Sjöstrand K, Clemmensen LH, Larsen R, Einarsson G, Ersbøll B. Spasm: a Matlab toolbox for sparse statistical modeling. J Stat Softw. 2018;23(84):1–37.
30. Efron B, Hastie T, Johnstone I, Tibshirani R. Least angle regression. Ann Stat. 2004;32(2):407–99.
31. Rosset S, Zhu J. Piecewise linear regularized solution paths. Ann Stat. 2007;1:1012–30.
32. Zhou DX. On grouping effect of elastic net. Stat Probab Lett. 2013;83(9):2108–12.