electronics

MDPI

Article

An Advanced Pruning Method in the Architecture of Extreme Learning Machines Using L1-Regularization and Bootstrapping

Paulo Vitor de Campos Souza [1,*], Luiz Carlos Bambirra Torres [2], Gustavo Rodrigues Lacerda Silva [3], Antonio de Padua Braga [3] and Edwin Lughofer [1]

1 JKU-Department of Knowledge-Based Mathematical Systems, Johannes Kepler University Linz, 4203 Linz, Austria; edwin.lughofer@jku.at
2 UFOP-Department of Computing and Systems, Federal University of Ouro Preto, Rua 36, 115-Loanda, João Monlevade, Minas Geras 30360-740, Brazil; luiz.torres@ufop.edu.br
3 UFMG-Av. Antônio Carlos 6627, Federal University of Minas Gerais, Belo Horizonte, Minas Gerais 31270-901, Brazil; gustavolacerdas@gmail.com (G.R.L.S.); apbraga@cpdee.ufmg.br (A.d.P.B.)
* Correspondence: paulo.de_ campos_souza@jku.at

Received: 15 April 2020; Accepted: 8 May 2020; Published: 15 May 2020

Abstract: Extreme learning machines (ELMs) are efficient for classification, regression, and time series prediction, as well as being a clear solution to backpropagation structures to determine values in intermediate layers of the learning model. One of the problems that an ELM may face is due to a large number of neurons in the hidden layer, making the expert model a specific data set. With a large number of neurons in the hidden layer, overfitting is more likely and thus unnecessary information can deterioriate the performance of the neural network. To solve this problem, a pruning method is proposed, called Pruning ELM Using Bootstrapped Lasso BR-ELM, which is based on regularization and resampling techniques, to select the most representative neurons for the model response. This method is based on an ensembled variant of Lasso (achieved through bootstrap replications) and aims to shrink the output weight parameters of the neurons to 0 as many and as much as possible. According to a subset of candidate regressors having significant coefficient values (greater than 0), it is possible to select the best neurons in the hidden layer of the ELM. Finally, pattern classification tests and benchmark regression tests of complex real-world problems are performed by comparing the proposed approach to other pruning models for ELMs. It can be seen that statistically BR-ELM can outperform several related state-of-the-art methods in terms of classification accuracies and model errors (while performing equally to Pruning-ELM P-ELM), and this with a significantly reduced number of finally selected neurons.

Keywords: extreme learning machine; lasso with bootstrapping; pruning of neurons; least angle regression

1. Introduction

The idea of randomly selecting hidden neurons for neural networks has been known for a few decades [1], and the Random Vector Functional-Link proposed by Pao et al. [2] introduces the idea of random weight assignment. Later, the concept was popularized by Huang et al. [3] who proposed and named the Extreme Learning Machine (ELM), which is a learning algorithm for single-hidden layer feedforward network (SLFNs) with low computational complexity, suitable for training with high-level datasets. The ELM algorithm allows models to be trained in a single interaction: the random definition of the hidden layer parameters is followed by the least-squares calculation (using the pseudoinverse

matrix) of the linear output weights. This ensures rapid convergence and proper performance of the trained neural network.

However, ELM may suffer from a loss of generalization capability when the amount of neurons in the hidden layer is too high, incurring in a situation known as overfitting. Several approaches to deal with overfitting were proposed in the literature, typically based on trial and error runs to eliminate elements of the ELM architecture [4,5].

Thus, pruning methods have been designed in the past which have to identify how relevant each neuron is to the model and thus which neurons can be eliminated. For instance, Miche et al. [4] proposed a pruning technique in which neurons are ranked by the Multi-Response Sparse Regression (MRSR) algorithm [6] and then selected for elimination after leave-one-out cross validation. In the work of Rong et al. [5], a measure of mutual information between hidden neurons is used for pruning. More related work with a detailed description will be presented in the subsequent section.

This paper presents a new methodology for pruning the architecture of an ELM using regression and re-sampling techniques to select the most relevant neurons to the output of the model. For this purpose, a method proposed by Reference [7] called LARS (Least Angle Regression), which is a multi-variate regression algorithm, will be used in order to robustly estimate the regression coefficients and to shrink the subset of candidate regressors to be included in the final model. Therefore, we will demonstrate the usage of Bolasso method (as proposed in Reference [8]) to select the most relevant neurons in the hidden layer of ELMs. The main contribution to science that this paper proposes is to perform a reliable statistical evaluation in a practical way to define the optimal amount of neurons in the hidden layer of ELMs, allowing the training to be assertive and to improve the final model accuracy on separate test data due to more compact models, decreasing the likelihood of overfitting. The optimal amount is thereby steered by the outcome of LARS in combination with several bootstrap replications, which makes neuron selection more robust against noise in the data and a small amount of training samples (due to the consideration made in Reference [9] regarding the purpose and aims of boot-strapping in general). Based on extensive evaluations carried out on several complex real-world pattern classification and regression problems, we can clearly demonstrate a statistical significant improvement in model performance with a lower number neurons, compared to conventional ELMs, but also compared to several related pruning methods for ELMs.

The remainder of the article is organized as follows. Section 2 presents the theoretical concepts involved in extreme learning machines, pruning methods proposed in the literature, and the idea of simple linear regression. Section 3 describes the methodologies used for the construction of the proposed algorithm. In Section 4, the tests performed in the proposed pruning methodology to verify the efficiency in finding the ELM architecture in combination with various statistical tests are exposed to the reader, also by comparison of BR-ELM with related SoA techniques. Finally, Section 5 provides conclusions and future works.

2. Related Work

2.1. Extreme Learning Machines

The ELM (Extreme Learning Machine) is a training algorithm developed for a single hidden layer feedforward network (SLFN), in which random values are assigned to hidden layer weights, and output layer weights are estimated analytically [3,10].

For N given arbitrary training samples, with m features and associated outputs, composing pairs of type $(\mathbf{x}_i, y_i)_{i=1}^{N} \in \mathbb{R}^m$, an SLFN with k hidden nodes and activation functions $\mathrm{f}(\cdot)$ can be defined as follows: [3]:

$$y_i = \sum_{j=1}^{k} h_j f(\mathbf{w_j}, x_i, b_j) \quad i = 1, \ldots, N, \tag{1}$$

where $\mathbf{w_j}$ is the randomly assigned weight vector of the connections between the m inputs and the hidden j-th neuron, h_j is the vector of weights of the connections between the j-th hidden neuron and

the neurons of the network output, and b_j is the bias of the j-hidden neuron. $f(\cdot)$ is the activation function applied to the scalar product of the input vector and the hidden weights. The activation function can be sigmoidal, sine, hyperbolic or Gaussian tangent [3,10]. Using the model defined in (1), the output vector y can be written as $y = G^*h$. G is defined as [2,3]:

$$G = \begin{bmatrix} f(\vec{w}_1, x_1 + b_1) & \dots & f(\vec{w}_k, x_1 + b_k) \\ f(\vec{w}_1, x_2 + b_1) & \dots & f(\vec{w}_k, x_2 + b_k) \\ f(\vec{w}_1, x_n + b_1) & \dots & f(\vec{w}_k, x_N + b_k) \end{bmatrix}_{N \times k} . \quad (2)$$

The columns of the matrix G, defined in (2), correspond to the outputs of the hidden neurons of the network with respect to the input $X = [x_1, x_2, \dots, x_N]^T_{m \times N}$. The ELM algorithm proposes the initialization of the weights of the hidden layer, w_k, at random. Then, the weights of the output layer are obtained through the Moore-Penrose pseudoinverse of G according to the expression [3,11]:

$$h = G^+ y, \quad (3)$$

with $G^+ = (G^T G)^{-1}$.

The output weights obtained in (3) correspond to the minimum norm solution of the least squares problem defined as $min \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$, in which $\hat{y}$ represent the predicted outputs [12]. The number of hidden neurons affects the performance of the ELMs, especially regularization algorithms have been proposed to improve accuracy.

2.2. Pruning and Regularized Methods for Extreme Learning Machines

A neural network may suffer from lack or excess of hidden layer neurons, which can considerably damage model performance. When too many neurons are used, the model may present poor testing results despite a very good (sometimes perfect) training performance, a situation which is known as overfitting [13]. Therefore, the definition of its architecture is critical in obtaining a stable model. Two approaches have been followed when dealing with this subject: Tikhonov's regularization theory [14] that allows the resolution of ill-posed problems and the technique of stabilization of structures, which consists of identifying the ideal structure for each problem by reducing the influence of unnecessary regressors (= columns of G in our case) towards 0.

The approach based on Tikhonov's regularization theory [14] consists of treating the learning problem as an ill-posed inverse problem, and then inserting prior knowledge in order to make the problem well-posed. This prior knowledge corresponds to a smoothing term αI with I the identity matrix and α a regularization parameter that balances the complexity and fit of the model to the data (also known as the bias-variance tradeoff [13]). The main difficulty with this kind of approach is finding the optimal balance. Therefore, various regularization parameter choice techniques have been proposed in the past, see Reference [15] for a comprehensive survey.

The techniques for stabilizing structures are based on problems that seek to approximate or interpolate a continuous function defined by a parameter vector. The theory of approximation [16] defines that models that have a single hidden layer, which has the ideal amount of neurons in the hidden layer, can act efficiently as a universal approximator. Therefore, the architecture stabilization theory of intelligent models is based on obtaining optimal neurons so that models that work with regression and classification of patterns can perform tasks with adequate responses. To find this structure, we aim to compare structures with different amounts of neurons in the hidden layer and compare their results. This approach can become very time-consuming if the number of neurons and statistical techniques to the aid decision making are huge. Another variant is to assign a high value to the neurons of the hidden layer and to eliminate unnecessary neurons through statistical approaches. Finally, the last approach is the opposite of the previous one, since the model starts with a simple structure and new neurons are added to find the best answer (bottom-up versus top-down). Both approaches may lead to inadequate responses when, in the top-down variant, pruning is rigid

or very restricted, or when, in the bottom-up variant, the inclusion of neurons does not have a well-established stopping criterion.

Several training strategies employing those two approaches (theory of regularization and definition of architectures) were proposed in order to solve the problem of finding the optimal number of hidden ELM neurons. Concepts such as Lasso, Ridge Regression, and elastic net [11], Chi-Squared statistics, and information gain criteria [5], LARS and Tikhonov regularization [17] were used, as well as linear and quadratic regularization methods for ELM with incomplete data [18]. A new version of the original ELM was also formulated using the LARS regression method [4].

The algorithm developed by Reference [4] attempts to increase the robustness of the ELM by eliminating irrelevant neurons. Pruning is achieved with Multiresponse Sparse Regression (MRSR), along with Leave-One-Out (LOO) cross-validation.

A disadvantage of MRSR, however, may be a possibly too optimistic selection achieved through LOO-error, because this considers only one left out sample in each run for which then prediction often leads to a too optimistic error close to the training error, see Reference [19] (Chapter 7) for a detailed analysis of this issue.

In Rong et al. [5], an algorithm was proposed, which identifies the importance of each of the hidden neurons of the extreme learning machine to the class labels using statistical techniques that allow the removal of unnecessary information.

The main idea proposed by Reference [11] is to identify the degree of relevance of the weight that links the k-th hidden element with the ELM output layer through regression methods, in particular, the regularized least squares regression method is used that applies penalties to the coefficient vector. Less relevant neurons can be identified and removed from the model, allowing for better generalization capability of the neural network. This method is applied to ELM models that contain several neurons in the hidden layer whose number is greater than or equal to the number of training samples [11] (leading to an over-determined equation system to solve $\rightarrow$ no unique stable solution).

Another method for regularizing ELM based on ridge, lasso, and elastic net regression was developed by Reference [20], but this process is based on the formation of a committee (an ensemble) of extreme learning machines. This methodology deals with the problem that some parameters remain unchanged after their values are chosen at random because, in many cases, they do not translate an optimal value for the operations that will be performed by the ELM. The proposal to solve this problem rests on using a set of ELM networks in which their parameters are initialized autonomously, and the combination of their predictions is responsible for the final output of the model (with various combination techniques possible, see Reference [21]). This work makes use of regularization methods to automatically select the members who will be part of the committee [20].

Recently, studies have been conducted to improve the internal architecture of the ELM. Reference [22] has proposed a new sparse extreme learning machine (SPI-ELM) based on the condition number of the regression matrix and found a lemma to relate the condition number with the number of hidden neurons. SPI-ELM exhibits better generalization performance and lower run-time complexity compared with ELM and also demonstrates a beneficial relationship between the condition number in the ELM design matrix and the number of hidden neurons. This relationship helps to understand the random weights and non-linear activation functions in ELMs.

In Reference [23], two greedy learning algorithms are proposed: the forward feature selection algorithm (FFS-GELM) and a backward feature selection algorithm (BFS-GELM), which are meant to tackle the problem of the number of neurons in the hidden layer of ELMs. To reduce the computational complexity, an iterative strategy is used in FFS-GELM, and its convergence is proven. In BFS-GELM, the decreasing iteration is applied to decay this model, and in this process, the accelerating scheme was proposed to speed up the computation of removing the insignificant or redundant features.

Finally, DELM proposed by Reference [24] uses an online learning mechanism to train ELM as a primary classifier and train a double hidden layer structure to improve the performance of ELM. When an alert about concept drift is set, more hidden layer nodes are added to ELM to improve the

generalization ability of the classifier. If the value measuring concept drift exceeds the upper limit, or the accuracy of ELM is at a low level, the current classifier will be deleted, and the algorithm will use new data to train a new classifier to learn the new concept.

2.3. Extreme Learning Machine and Pattern Recognition Problems

When the extreme learning machine acts on classification problems, it can also be regularized. Peng et al. [25] developed a method that is based on the idea that similar samples in an ELM should share similar properties, forming a regularized discriminative graph for extreme learning machines. In this method, to regularize ELM-based models for facial recognition, the constraint imposed on the output weights forces the output of the samples from the same class to be similar. This constraint is formulated with a regularization term that is added to the fundamental objective of an ELM model, causing the output layer weights to be solved analytically [25].

The work of Huang et al. [26] uses regularized ELM for problems of pattern classification and regression using semi-supervised and unsupervised learning. In a different way to the other related works, they introduce a form of regularization called manifold regularization [27], allowing its use in multiclass classification or multicluster grouping. The semi-supervised ELM proposal incorporates multiple regularizations to take advantage of unlabeled data to improve classification accuracy when they are scarce. In the unsupervised training scenario, the target is to find the data structure adjacent to the original data [26].

The model proposed by Silvestre et al. [28] uses a priori spatial information expressed by an affinity matrix to carry out the regularization, performed without any need for parameter tuning of the ELM. In Pinto et al. [29], the regularization is based on ranking and selection of best values of a priori information expressed by affinity matrices of the hidden neurons of ELM in pattern recognition in classification problems. Mohammed et al. proposed intelligent models based on the concepts of ELM and Bidirectional Principal Component Analysis [30] to act in the curvelet image decomposition of human faces. A subband that exhibits the maximum standard deviation is dimensionally reduced.

The variety of pattern identification tasks that can be performed with models trained by extreme learning machine is wide. In the Cao et al. model [31], the concept of sparse representation classification (SRC) in the area of image classification is applied in areas such as handwritten digit classification, landmark recognition and face recognition with satisfactory pattern identification results. The definition of the ELM space based on low classification decomposition is used for the treatment of data that are inserted into an intelligent model in the work proposed by Iosifidis et al. [32]. In the proposal of Jin et al. [33], ELM is adapted to work in conjunction with the MapReduce framework to classify patterns in Big Data problems. As in the Musikawan et al. [34] model, a parallelized metaheuristic proposal is used to identify existing patterns in regression problems.

The work proposed by Liangjun et al. [35] seeks to reduce the sensitivity of outliers and impulsive noises in the intelligent models through the correntropy. To confirm the results obtained by the model, datasets were submitted to standard classification tests with a significant improvement in the accuracy of the results obtained. The same concept of correntropy is also applied for robust ELM training in the model of Chen et al. [36] to act in function approximation and regression problems. Cybersecurity is also aided by models trained with ELM and feature selection methods. In the Gao et al. [37] model, an ELM incremental model combined with Adaptive Principal Component Analysis concepts is used to determine patterns of intrusions and intruders in computer networks. In the context of models using ELMs, Campos Souza et al. [38] built a model that performs pruning of less significant neurons using the concepts of Automatic Relevance Determination. There a strategy was suggested which is capable of defining the ELM hidden layer neurons that best contribute to the model with the usage of partial least squares regression [39]. The proposal in this paper differs from this approach due to the methodology of selection for the identification and construction of hidden layer neurons. In particular, the use of the LARS method in concomittance with bootstrap replications ensures the use of statistical techniques with various simulation scenarios. This technique seeks solid training to define the best

subgroup of neurons capable of optimizing the model's outputs through a combination of replications, statistical techniques, and selection of characteristics. In the Partial Least Squares (PLS) technique, only a single evaluation is used, which can be efficient in improving the performance of the model, but not in the selection of the best neuron selection group.

Recent work indeed also addresses pruning techniques using L_1-regularization in neural networks (see, e.g., He et al. [40], Fan et al. [41], Chang et al. [42], and Alemu et al. [43]) together with their benefit to shrink as much regression coefficients as possible to 0, but unlike the approach suggested in this paper, they do not apply bootstrapping, neither any resampling technique [44] to elicit the hidden neurons actually needed. Such replications should make the selection more stable, especially in the case of a low amount of samples and/or significant noise in the data. This is because in the case of noisy data samples, regression fits may often provide instable solutions, even though regularization may be involved [45] (such as based on L_1-norm, which is used in our approach for neuron selection). That is, the regression fits tend to over-fit due to fitting more the noise than the real underlying functional approximation trends, increasing the variance error much more than reducing the bias error [19]. Bagging short for Boostrap aggregating, invented by Leo Breiman in the 1990s [9], is a well-known and widely used method for reducing over-fitting effect due to noisy data samples [9,46]. Here, we use the bootstraps for providing a 'best of' selection rather than combining various (ELM) models trained on bags, but the spirit of variance reduction and thus overall stability increase remains the same when conducting lasso regression on the various bootstrap samples (and combining their outputs, afterwards).

2.4. Determining Coefficients of Linear Regression in a Robust Manner with L1-Regularization

LARS [7] is a regression algorithm for high-dimensional data which estimates the regression coefficients and the subset of candidate regressors to be included in the final model. When we evaluate a set of N distinct samples (x_i, y_i), where $x_i = [x_{i1}, x_{i2} \dots x_{id}]^T \in \mathbb{R}^d$ where d is the dimension of x_i and $y_i \in \mathbb{R}$ for $i = 1...N$, the cost function of this regression algorithm can be defined as:

$$\sum_{i=1}^{N} \parallel G(x_i)h - y_i \parallel_2 + \lambda \parallel h \parallel_1, \tag{4}$$

where λ is a regularization parameter usually estimated using cross-validation [47]. The first term (4) corresponds to the residual sum of squares (RSS). This term decreases as the training error decreases. The second is the L_1 regularization term. First, it improves network generalization, avoiding overfitting [48]. Second, it can be used to generate sparse models [47], as it aims to force as much regression coefficients to 0 [49]. To understand why LARS can be used as a variable selection algorithm, Equation (4) can be rewritten as:

$$\begin{aligned} \min_{v} \quad & RSS(h) \\ \text{s.t.} \quad & ||h||_1 \leq B, \end{aligned} \tag{5}$$

where B is an upper-bound on the L_1-norm of the weights. A small value of B corresponds to a high value of λ, and vice versa. This equation is known as "least absolute shrinkage and selection operator" [19], or lasso. Figure 1 illustrates the contours of the RSS objective function, as well as the $L1$ constraint surface. It is well known from the theory of constrained optimization that the optimal solution corresponds to the point where the lowest level of the objective function intersects the constraint surface. Thus, one should note that, when B grows until it meets the objective function, the points in the corners of the L_1 surface (i.e., the points on the coordinate axes) are more likely to intersect the RSS ellipse than those of the sides [47].

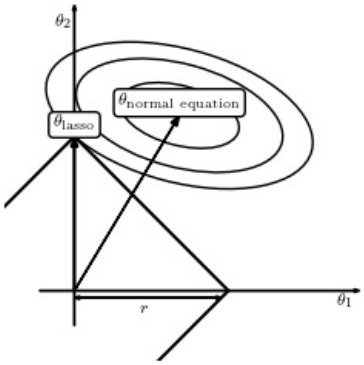

Figure 1. Visualization of the effect of L_1 regularization: solution provided by lasso ϕ_{lasso} versus the unconstrained (regular) solution $\phi_{normalequation}$.

3. Pruning ELM Using Bootstrapped Lasso (BR-ELM)

When analyzing the behavior of ELM, we propose a method to choose the more significant neurons that actually contribute to the structure of the neural network for the actual learning problem at hand and thus to improve the prediction performance of the model due to reducing the likelihood of overfitting by more compact models. In this context, we will use regularization techniques to identify those elements which have a greater relevance to the hidden layer of the ELM and later apply a pruning technique to optimize the architecture of the network with the chosen neurons. Therefore, this approach is based on the choice of a large number of neurons in the hidden layer, and based on regularization and resampling techniques, we find the optimal set of neurons to solve the actual learning problem. In other words, the final network architecture in ELM is defined through a neuron shrinkage technique based on a combination of L_1 regularization and bootstrap replications.

For the problem considered in this paper, the regressors G_{lp} are the activation levels of the neurons, where each column of G can be associated with the activation level of one neuron on all training samples (rows). Thus, the LARS algorithm can be used to select an optimal subset of the significant neurons (L_p) that minimize Equation (4) for a given value of λ. This is because LARS tries to shrink as many coefficients as possible to 0, where one coefficient can be directly associated with a neuron (and its activation levels). Thus, all neurons leading to coefficients with value of 0 can be finally neglected in the final structure of the model. The amount of shrinkage thereby depends on the regularization parameter λ: the higher it becomes, the more coefficients will be shrinked to 0. In order to select an optimal λ, cross-validation (CV) based tuning (in a grid) can be applied, for instance, or also different direct choice techniques as proposed in Reference [15], circumventing time-intensive CV operations—for example, the method of hardened-balancing could be shown that it approximates the outcome of CV pretty well and often comes even closer to the optimal value of λ by avoiding distinct grid points. A final remark is that the application of LARS results in an embedded selection approach, which is closely linked to the supervised learning problem at hand, much more than a filter-based selection approach is (which usually relies on some information-theoretic measures, which are usually carried out in advance and not handled in combination with the current model and its learning method under investigation).

The approach used to increase the stability of the model selection algorithm is the use of re-sampling through bootstrapping, in combination with lasso regression also termed as Bolasso. Its principal procedure is summarized in Algorithm 1 [8].

Algorithm 1 Bolasso-bootstrap-enhanced least absolute shrinkage operator

(1) Let n be the number of examples, (lines) in X and Y:
(2) Draw n examples from (X, Y), uniformly and with replacement, termed as $(Xsamp, ysamp)$.
(3) Perform LARS on $(Xsamp, ysamp)$ with a given value of λ to estimate the regression coefficients $\vec{h}$.
(4) Determine which coefficients are nonzero.
(5) Repeat steps (2) to (4) for a specified number of bootstraps b_t.
(6) Take the intersection of the non-zero coefficients from all bootstrap replications as significant variables to be selected (100% consensus).
(7) Revise using the variables selected via non-regularized least squares regression (if requested).
(8) Calculate the expected error of the model on a separate validation set.
(9) Repeat the procedure for each value of b_t bootstraps and λ (actually done more efficiently by collecting interim results).
(10) Determine "optimal" values for λ and b through eliciting the minimal expected error of the model on a separate validation set.

In this procedure, the LARS algorithm runs on several bootstrap replications of the training data set and thus enjoys the benefits of bootstrap replications (noise reduction, sample significance increase and so forth [9]). For each repetition, a distinct subset of regressors is selected. The regressors to be included in the final model are defined according to the frequency with which each of them is chosen through different tests. In a more realistic variant, a consensus threshold smaller than 100% may be determined, say $\gamma = X\%$, and a regressor is then included, if selected in at least $X\%$ of the assays. X can be set according to statistical significance level considerations, thus being typically in the range $[90, 99]$ (e.g., 95 would belong to a significance level of 0.05). This means that a neuron has to be selected in most of the bootstrap replications such that it can be safely accepted that it is significantly important for the model structure to explain the modeling target.

Consider n independent and identically distributed observations $(x_i, y_i) \in \mathbb{R}^d \times \mathbb{R}$, $i = 1, \ldots, n$, provided by the matrices $\mathbf{X} \in \mathbb{N}^{n \times d}$ and $\mathbf{Y} \in \mathbb{R}^n$, admit b_t bootstrap replications of n data points [8,50]; that is, for $k = 1, \ldots b_t$, we suppose an auxiliary sample $(x_i^k, y_i^k) \in \mathbb{R}^d \times \mathbb{R}$, $i = 1, \ldots, n$, given by the matrices $\mathbf{X}^k \in \mathbb{R}^{nxp}$ and $\mathbf{Y}^k \in \mathbb{R}^n$. The n pairs (x_i^k, y_i^k), $i = 1, \ldots, n$, are randomly drawn from the original n pairs in (**X, Y**) with replacement, subtracting the evaluations not to be biased. Thus, we determine the distribution of the auxiliary sample $(\mathbf{X}^*, \mathbf{Y}^*)$ by sampling n points with the substitution of (**X, Y**) and, given (**X, Y**), the auxiliary samples are independently sampled of $(\mathbf{X}^*, \mathbf{Y}^*)$ [8].

It is suggested to estimate the supports $J_k = j$, $w_j^k \neq 0$ of the Lasso estimates $\vec{h}_k$ for the bootstrap samples, $k = 1, \ldots, b_t$, and to cross them to define the estimate of the support: $\mathbf{J} = \bigcap_{k=1}^{b_t} A_i \; J_k$, for $\mathrm{A} \in \mathbb{R}^{pxp}$ is the weight. Since **J** is selected, we estimate $\vec{h}$ by the unregulated least-squares adjustment restricted to variables in **J**. This estimation can be computed simultaneously for a large number of regularization parameters due to the efficiency of the LARS algorithm, which allows us to find the whole regularization path for the Lasso at the empirical cost of a single matrix inversion [7]. Even with an order of high complexity, the bootstrapped lasso may be able to use the training samples and define the relevance of that evaluated neuron to the set of final responses by using resampling. As replications are random, the results are free of tendencies, and the pruning of less relevant neurons according to the consensus threshold γ can be performed in the following way:

$$\frac{|\{h_i(b)|b = 1, ..., b_t \wedge v_i(b) > 0\}|}{b_t} > threshold, \tag{6}$$

with $h_i(b)$ the coefficient (output weight) of neuron i estimated through LARS on the b-th bootstrap replication and *threshold* the consensus threshold ($X\%$ belongs to $X/100$). If this condition holds, then variable i is selected. Thus, the finally selected set of neurons becomes

$$G_{sel} = \{g_i|i = 1, ..., k \wedge (6) \text{ holds}\} \tag{7}$$

with $L_p = |G_{sel}|$.

In our case, the learning algorithm assumes that the output hidden layer composed of the candidate neurons can be written as:

$$f(x_i) = \sum_{l=0}^{Lp} G_l(x_i)h_l = G(x_i)h, \tag{8}$$

where $h = [h_0, h_1, h_2, \ldots, h_{L\rho}]$ is the weight vector of the output layer and $G(x_i) = [G_0, G_1(x_i), G_2(x_i)]$ the output vector of the second layer, for $G_0 = 1$. In this context, $G(x_i)$ is considered as the non-linear mapping of the input space for a feature space of dimension $L\rho$. Since the weights connecting the first two layers are randomly assigned, the only parameters to be estimated are the weights of the output layer. Thus, the problem of network parameter estimation can be seen as a linear regression problem, allowing the use of regression techniques [19] for estimating parameters and selecting candidate neurons. The benefit of using a bootstrap approach combined with the LARS procedure is the more coherent choice of a subset capable of better representing a problem's data, due to the set of combinations that can be generated in order to validate the efficiency of an analyzed neuron. Despite several combinations, the impacts on the model's outputs are considerable, as they are more assertive, even with a substantial increase in the complexity of the model's training. The proposed method then consists of computing the matrix of the activation functions of the hidden layer G and then calculating the regression coefficients using the Lasso method on different bootstrap replications, and finally selecting the most significant number of neurons L_p through consensus threshold (see above). The combined procedure for pruning ELMs is summarized in Algorithm 2 and should be self-explainable. The algorithm has three parameters:

- The number of bootstrap replicates, b_t;
- The consensus threshold, γ.
- The initial amount of hidden neurons, k

Algorithm 2 Pruning algorithm for Extreme Learning Machines

(1) Define b_t number of bootstrap replicates.
(2) Define the consensus threshold γ.
(3) Create initial hidden neurons k following the conventional strategy used in ELMs.
(4) Assign random values to w_j and b_j, for $j = 1, \ldots, k$.
(5) For $m = 1, \ldots, b_t$:
(6) Draw n examples from (X, Y), uniformly and with replacement, termed as (X_m, y_m).
(7) Compute G_m using Equation (2).
(8) Perform LARS algorithm in order to solve (4) for yielding $\vec{h}_m$ (internally, apply regularization parameter choice technique to elicit the optimal λ).
(9) EndFor
(10) Select the final L_p neurons using (7) through (6) with consensus threshold γ.
(11) Construct matrix G_{sel} from the L_p selected neurons.
(11) Estimate the final output layer weights $\vec{h}$ by applying (3) using G_{sel} instead of G.
(12) Define the output hidden layer using (8).

For the initial amount of neurons we can assume any high value (>100) sufficiently covering the degree of non-linearity contained in a learning problem. As LARS is a regularized procedure and thus able to deal with weakly-determined equation systems, this initial amount can be expected to be not really sensitive. The ideal number of bootstrap replicates could be estimated through the out-of-bag error (OOB) [8] by varying b_t and taking the minimal OOB value for final model training and pruning. The consensus threshold γ is typically set according statistical significance considerations (as discussed above).

4. Experiments

4.1. Synthetic Database Classification

To perform the following tests, four synthetic two-dimensional data sets were used to simulate the pattern classification for this type of hypothetical situations. The four data sets have 500 samples each and $d = 2$. Consider further the initial number of hidden neurons, $L = 500$, the number of bootstrap replicates, $b = 8$ and the pessimistic consensus threshold $\gamma = 90\%$ (thus, for all replications the estimated output weight of a neuron has to be non-zero to be selected).

In all the tests performed for the synthetic bases consider that the inputs were normalized, the outputs fall within the range of $[-1, 1]$ and the initial number of neurons is equal to the number of samples of each of the bases used in the test (which in this case was 500). The organization of the data sets in space is presented below. The data sets are shown as geometric outlines to evaluate the ability of the models to find those labels. Figure 2 shows the triangulation of Delaunay [x] for the bases; Figure 3 shows the separation between positive and negative classes. Table 1 presents the characteristics of each base and the percentage of ELM training (70%) and the test value (30%).

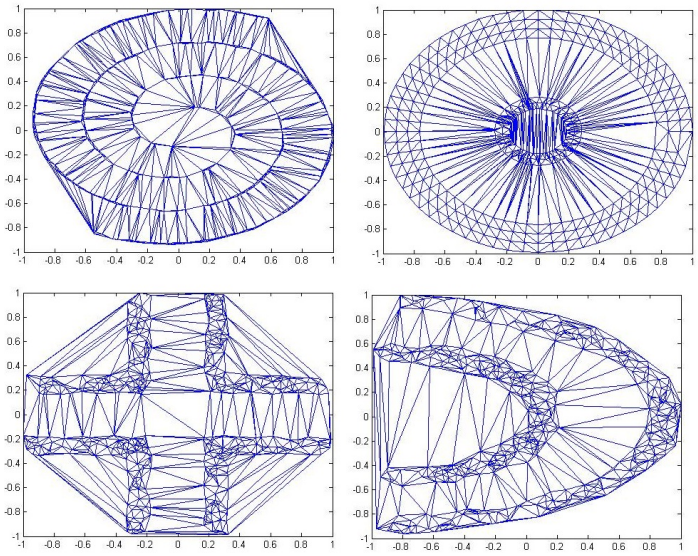

Figure 2. Bases of Delaunay triangulation.

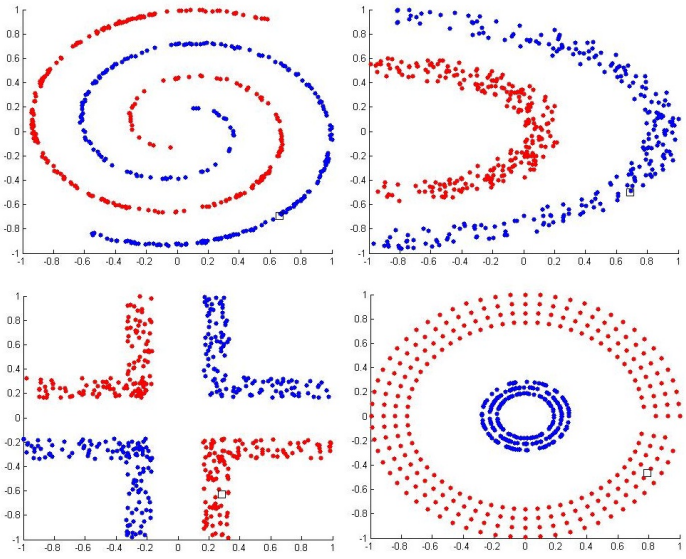

Figure 3. Positive and negative samples in each data set.

Table 1. Synthetic data set used in the experiments.

Dataset	Init.	Feature	Train	Test	Neurons (k)
Half Kernel	HKN	2	350	150	500
Spiral	SPR	2	350	150	500
Cluster	CLU	2	350	150	500
Corner	COR	2	350	150	500

In Tables 2 and 3, the average values of 30 repetitions were presented for each of the evaluated models. The highlighted values refer to the mean accuracy of the pattern classification for the first frame and the mean number of neurons used in the final architecture of the network. The values in parentheses represent the standard deviations of the tests performed.

Table 2. Accuracies of the model in the tests performed.

Dataset	BR-ELM	OP-ELM	P-ELM	ELM
HKN	100.0 (0)	100.00 (0)	92.73 (3.12)	97.89 (1.98)
SPR	100.0 (0)	99.65 (1.70)	93.45 (3.09)	99.72 (0.43)
CLU	100.0 (0)	93.19 (4,29)	91.98 (6.12)	92.11 (4.19)
COR	100.0 (0)	98.87 (3.08)	97.18 (3.21)	93.98 (2.87)

Table 3. Number of neurons selected in the test.

Dataset	Lp	BR-ELM	OP-ELM	P-ELM	ELM
HKN	500	88.43 (22.45)	137.83 (27.69)	291.10 (70.28)	500(0)
SPR	500	126.73 (35.99)	93.67 (29.59)	272.77 (78.78)	500(0)
CLU	500	245.71 (132.75)	157.00 (28.51)	249.90 (87.45)	500(0)
COR	500	79.50 (19.46)	118.17 (30.16)	286.60 (53.20)	500(0)

It can be verified by the synthetic bases test that the conventional ELM approach has a very high amount of neurons in the hidden layer, which does not necessarily lead to better accuracies (most probably it suffers from overfitting issues). Thus, pruning methods, especially our proposed algorithm BR-ELM, but also the related approach OP-ELM) help in the elimination of neurons that undermined the accuracy of the model in accomplishing the pattern classification.

The model proposed in the paper (BR-ELM) and the related state-of-the-art method OP-ELM in pruning of neurons had a statistically equivalent performance when dealing with accuracy, besides being superior to the P-ELM model and the original ELM, but our new approach used fewer neurons in three of the four tests performed. That emphasizes that the model of ELM based on regularization resampling allows a more efficient adaptation of the ELM architecture because it uses fewer neurons, reducing complexity and underlining the useability of the model in pattern classification. Figures 4 and 5 identify the classifier response for the classes used in each of the tests. Also, the separation of classes in space is presented. For all the figures shown, the accuracy of the test was 100% (also compared with Table 2).

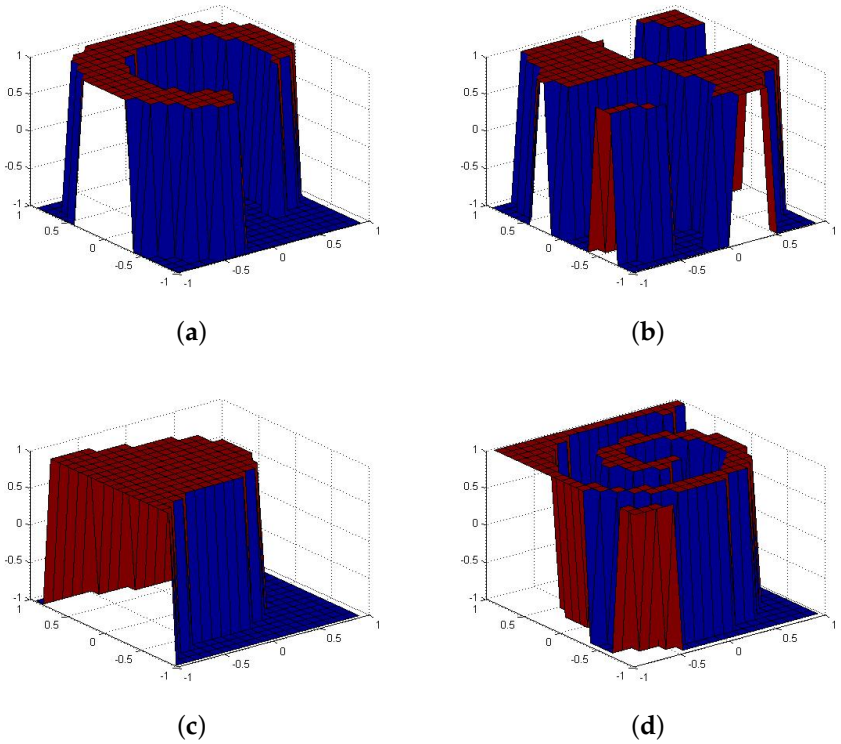

Figure 4. (**a**) Data set Cluster. (**b**) Data set Corners. (**c**) Data set Half Kernel (**d**) Data set Spiral.

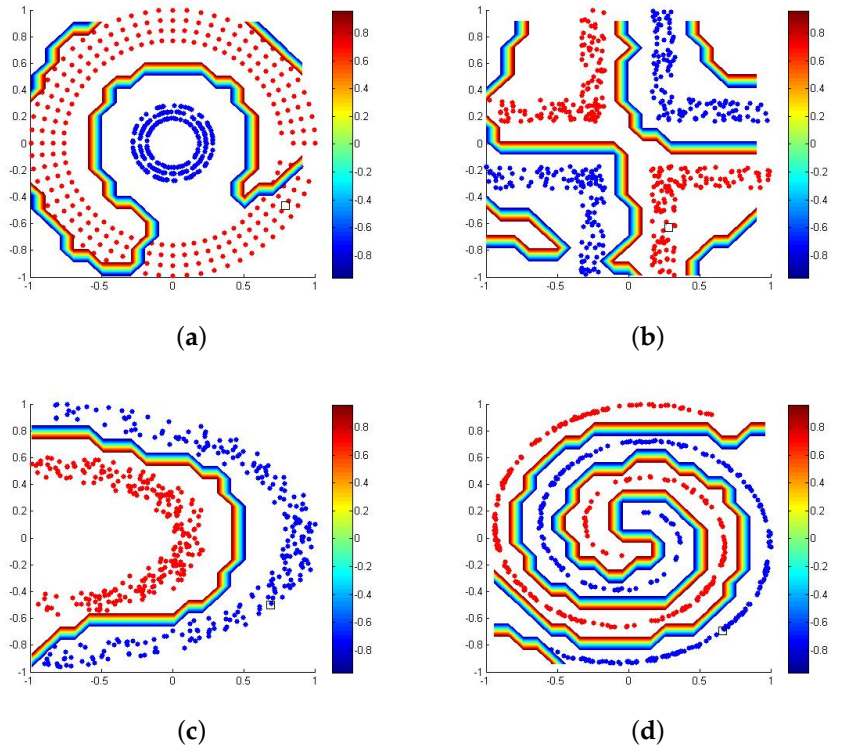

Figure 5. (**a**) Data set Cluster Classification. (**b**) Data set Corners Classification. (**c**) Data set Half Kernel Classification. (**d**) Data set Spiral Classification.

4.2. Evaluation Measures Used for Tests on Real-World Data Sets

The proposed pruning algorithm described in the previous section is evaluated using binary pattern classification problems and regression problems. The performance analysis was performed based on data sets with small $(N \leq 6)$ and high $(N > 6)$ dimensions.The data sets also included a low number (<1000), a mean number (between 1001 and 5000), and a high number (above 5001) of samples.

The evaluation measures in the pattern classification problems were as follows:

$$accuracy = \frac{TP + TN}{TP + FN + TN + FP} \tag{9}$$

$$sensitivity = \frac{TP}{TP + FN} \tag{10}$$

$$specificity = \frac{TN}{TN + FP} \tag{11}$$

$$AUC = \frac{1}{2}(sensitivity + specificity), \tag{12}$$

where, TP = true positives, TN = true negatives, FN = false negatives and FP = false positives.

In regression tests, the methods were evaluated through the root-mean-square error (RMSE) as defined by:

$$RMSE = \sqrt{\frac{\sum_{q=1}^{N}(y_q - \hat{y}_q)^2}{N}}, \tag{13}$$

where y is the target, $\hat{y}$ is the estimation from the model, and N is the number of samples. That is a frequently used measure of the differences between values (sample or population values) predicted by a model and the values observed.

The pruning method proposed in this article (BR-ELM) was compared with other classifiers to verify their accuracies. In all tables, the principal value is the accuracy, and the highlighted value in the parentheses is the respective standard deviation. The comparison was done with conventional ELM method (without using any pruning technique), and with the recently proposed pruning-based approaches P-ELM [5] and OP-ELM [4], which are closely related to ours as also performing LARS algorithm for neuron selection (but without bootstrap replications).

The data samples were randomly selected. We collected 30 measurements of accuracy from each of the bases evaluated in each model analyzed. The variables involved in the process were normalized with mean zero and unit variance. All outputs of the model were normalized to the range $[-1, 1]$. The initial parameters of the hidden layer were sampled from a uniform distribution $U[-0,5; 0,5]$. Hyperbolic tangent type sigmoidal activation functions were adopted for all neurons in all ELM models used in the test. The initial number of neurons k was set to the same value for all methods in order to achieve a fair comparison, typically to 500, except when the data set has less samples, then it was set to the number of samples

4.3. Benchmark Classification Datasets

The data sets used were obtained from the UCI machine learning repository [51] and in some standard-rank surveys performed by Reference [52,53]. Note the Iris class that had three types of classification was re-adapted to have only two outputs. Here is some relevant information about the real bases used in the tests [51–53].

Table 4 details the basics including the number of samples used for training (70%) and performance evaluation (30%) in addition to the number of initial neurons in the hidden layer (k).

Table 4. Data sets used in the Classification experiments.

Dataset	Init.	Feature	Train	Test	Neurons (k)
Four Class	FCL	2	603	259	862
Haberman	HAB	3	214	92	306
Iris	IRI	4	105	45	150
Transfusion	TRA	4	523	225	748
Mammographic	MAM	5	581	249	830
Liver Disorder	LIV	6	242	103	345
Diabetes	DIA	8	538	230	768
Heart	HEA	13	189	81	270
German Credit	GER	14	390	168	558
Climate	CLI	18	377	163	540
Australian Credit	AUS	24	700	300	1000
Parkison	PAR	24	137	58	195
Ionosphere	ION	32	245	106	351
Spam	SPA	57	3221	1380	1000
Sonar	SON	60	146	62	208
Pulsar	PUL	8	12,529	5369	1000
Credit Rating	CDT	23	21,000	9000	1000

Table 5 shows the accuracy Equation(9) of results of the binary pattern classification problems methods, Table 6 the respective number of selected neurons. In Tables 7 and 8, the AUC and the time of execution of the tests are represented, respectively.

Table 5. Accuracies of the model in the tests performed.

Dataset	BR-ELM	OP-ELM	P-ELM	ELM
FCL	99.95 (0.15)	99.77 (0.41)	96.93 (1.27)	89.82 (2.14)
HAB	75.95 (5.39)	70.61 (4.35)	73.34 (3.03)	69.29 (3.25)
IRI	95.62 (2.57)	94.96 (3.39)	93.01 (0.44)	89.79 (0.37)
TRA	80.61 (2.60)	77.32 (7.28)	74.09 (5.51)	71.42 (1.57)
MAM	82.95 (2.21)	81.15 (2.85)	81.25 (1.54)	78.76 (2.51)
LIV	67.98 (3.94)	63.92 (4.66)	64.14 (6.21)	61.14 (3.42)
DIA	74.55 (2.52)	70.50 (2.85)	72.23 (2.76)	69.07 (4.13)
HEA	82.68 (3.46)	70.12 (5.35)	85.75 (1.99)	79.10 (1.50)
GER	81.65 (2.13)	73.53 (5.92)	80.20 (1.69)	64.78 (0.94)
CLI	90.92 (2.54)	93.37 (2.79)	92.28 (1.58)	82.36 (1.78)
AUS	68.00 (1.83)	61.31 (4.81)	67.12 (3.16)	60.65 (2.80)
PAR	89.03 (4.23)	80.79 (5.04)	84.37 (2.70)	81.13 (1.96)
ION	87.69 (3.70)	90.12 (1.14)	88.03 (3.13)	84.78 (0.93)
SPA	91.89 (0.54)	91.95 (0.47)	90.84 (2.66)	77.74 (2.17)
SON	78.44 (5.19)	70.05 (5.23)	72.07 (0.14)	71.89 (5.68)
PUL	97.91 (0.16)	97.75 (1.14)	92.07 (2.17)	81.89 (2.31)
CDT	81.32 (0.33)	80.97 (0.29)	78.22 (1.14)	73.15 (4.68)

When analyzing Tables 5 and 6, it can be seen that the BR-ELM model obtained better results than the other models analyzed in the accuracy test while the number of neurons selected (L_p) was much lower for BR-ELM. Statistical analysis was performed on the datasets, and the accuracy as the only factor evaluated, where each of the classification bases can be seen as the blocking factor. Figure 6 shows the graphical organization of algorithm performance based on blocking factors. It presents the average behavior of the algorithms in solving all problems and the normal behavior of the models in the evaluated datasets. To perform the statistical tests, the analysis of variance (ANOVA) on the results for each of the groups (algorithm x block factor) is used in the test. In general, it is verified that the test has 68 groups (4 algorithms and 17 bases). As a null hypothesis, we will consider that the four algorithms have the same mean accuracy to perform the pattern classification. As an alternative hypothesis, we define that they have a distinct average accuracy in performing the classification task

of the 17 data sets. Using this test, it is possible to conclude whether or not the performance of the algorithm proposed in this paper achieves an average performance higher than the state-of-the-art methods. After performing the ANOVA test with a significance level of $\alpha = 0.05$, it turned out that the null hypothesis of equality of the performance of the algorithms has to be rejected due to a p-value of (p-value = 1.21×10^{-7}). This is also the case for the equality of performance of the algorithms by the blocking factor (p-value = 2×10^{-16}). In Figure 7, we can observe the characteristics resulting from the ANOVA test for the database set evaluated. The factors presented in the figure represent the validations of the assumptions of the ANOVA test, proving normality (Shapiro-Wilk normality test), homogeneity of variances (Fligner-Killeen test) and independence (Durbin-Watson test) [54].

Table 6. Number of neurons selected in the test.

Dataset	Lp	BR-ELM	OP-ELM	P-ELM	ELM
FCL	862	200.40 (55.60)	153.50 (72.13)	167.73 (32.87)	862(0)
HAB	306	7.7 (2.99)	29.83 (23.97)	44.46 (22.43)	306(0)
IRI	150	9.53 (4.13)	25.00 (19.02)	32.87 (12.99)	150(0)
TRA	748	13.26 (5.69)	15.66 (6.12)	18.10 (5.64)	748(0)
MAM	830	13.06 (8.18)	58.00 (44.82)	22.84 (4.97)	830(0)
LIV	345	35.66 (8.56)	114.33 (43.10)	49.03 (6.29)	345(0)
DIA	768	106.78 (46.19)	127.00 (29.26)	220.33 (26.71)	768(0)
HEA	270	15.2 (8.05)	126 (37.31)	20.46 (6.64)	270(0)
GER	558	32.50 (9.79)	234.5 (17.68)	27.83 (10.07)	558(0)
CLI	540	8.80 (8.37)	9.16 (7.50)	9.01 (5.13)	540(0)
AUS	1000	56.40 (32.07)	502.16 (184.72)	29.36 (8.42)	1000(0)
PAR	195	41.03 (11.98)	93.33 (29.19)	18.03 (8.52)	195(0)
ION	351	50.60 (10.08)	21.83 (23.81)	18.7 (9.20)	351(0)
SPA	1000	117.73 (8.90)	345.76 (65.13)	183.75 (39.23)	1000(0)
SON	208	37.10 (9.46)	404.00 (24.08)	48.71 (43.65)	208(0)
PUL	1000	99.10 (27.17)	102.21 (5.65)	127.14 (43.65)	1000(0)
CDT	1000	85.00 (12.36)	76.00 (16.36)	108.71 (43.65)	1000(0)

Table 7. AUC .

Dataset	BR-ELM	OP-ELM	P-ELM	ELM
FCL	0.9997 (0.0010)	0.9965 (0.0058)	0.9812 (0.0052)	0.9012 (0.0065)
HAB	0.5745 (0.0500)	0.5752 (0.0442)	0.5214 (0.0072)	0.5126(0.0082)
IRI	0.9645 (0.0195)	0.9475 (0.0378)	0.9401 (0.0013)	0.9091(0.0082)
TRA	0.6379 (0.0320)	0.6021 (0.0620)	0.5861 (0.0283)	0.5417 (0.0032)
MAM	0.8260 (0.0214)	0.8119 (0.0288)	0.8109 (0.0145)	0.8006 (0.0014)
LIV	0.6704 (0.0484)	0.6105 (0.0438)	0.6235 (0.0809)	0.5807 (0.0121)
DIA	0.7577 (0.0279)	0.6735 (0.0314)	0.7345 (0.0198)	0.6651 (0.1008)
HEA	0.8183 (0.0333)	0.7012 (0.0624)	0.8227 (0.0391)	0.6549 (0.0280)
GER	0.8346 (0.0192)	0.7277 (0.0575)	0.8122 (0.0861)	0.7089 (0.1065)
CLI	0.8987 (0.0096)	0.9106 (0.0192)	0.9094 (0.0631)	0.8864 (0.0098)
AUS	0.7105 (0.0192)	0.5686 (0.0381)	0.6287 (0.0182)	0.6105 (0.0276)
PAR	0.8667 (0.0622)	0.7841 (0.0639)	0.7441 (0.0439)	0.7227 (0.0590)
ION	0.8264 (0.0479)	0.8423 (0.0156)	0.8365 (0.0179)	0.7811 (0.0349)
SPA	0.9104 (0.0054)	0.9133 (0.0012)	0.9087 (0.0160)	0.7823 (0.0523)
SON	0.7831 (0.0501)	0.7013 (0.0553)	0.6413 (0.0185)	0.6304 (0.0114)
PUL	0.9089 (0.0077)	0.9008 (0.0017)	0.8521 (0.0054)	0.8329 (0.0176)
CDT	0.6377 (0.0060)	0.6256 (0.0064)	0.6019 (0.0045)	0.5765 (0.0081)

Table 8. Time.

Dataset	BR-ELM	OP-ELM	P-ELM	ELM
FCL	2941.78 (92.70)	122.98 (3.75)	167.73 (32.87)	192.21 (14.76)
HAB	105.30 (8.61)	4.38 (0.19)	44.46 (22.43)	9.27 (0.54)
IRI	10.19 (1.05)	0.62 (0.14)	32.87 (12.99)	0.89 (0.16)
TRA	7.10 (4.98)	15.66 (6.12)	8.10 (5.64)	748(0)
MAM	833.86 (197.75)	106.01 (3.22)	671.43 (141.44)	465.91(132.17)
LIV	2.41 (0.008)	7.32 (0.42)	9.03 (6.29)	6.21 (0.33)
DIA	4219.17 (77.28)	31.54 (0.85)	20.33 (6.71)	39.87 (10.21)
HEA	4.68 (0.27)	3.39 (0.34)	20.46 (6.64)	14.09 (1.21)
GER	77.40 (14.71)	36.83 (1.80)	67.83 (10.07)	58.85 (12.87)
CLI	10.80 (8.37)	9.16 (7.50)	9.01 (5.13)	15.38 (6.71)
AUS	798.38 (26.32)	218.40 (6.90)	299.36 (8.42)	492.81 (41.21)
PAR	19.54 (2.81)	1.13 (0.12)	18.03 (8.52)	19.31 (2.12)
ION	167.84 (59.56)	21.83 (23.81)	18.70 (9.2)	35.16 (6.51)
SPA	293.33 (20.71)	1191.60 (15.23)	183.75 (39.23)	165.89 (73.21)
SON	110.27 (10.48)	1.26 (0.06)	48.71 (43.65)	20.89 (2.61)
PUL	440.17 (83.93)	79.64 (0.76)	148.71 (43.65)	189.76 (14.27)
CDT	846.69 (58.23)	91.60 (1.57)	95.89 (43.65)	112.70 (10.21)

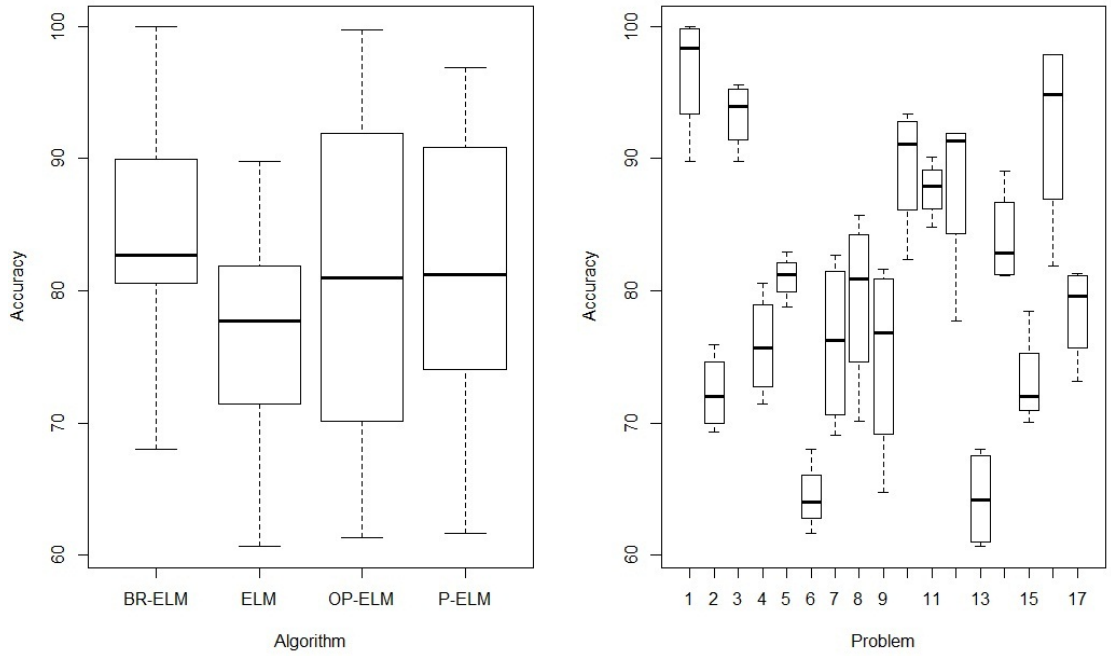

Figure 6. Plot of the average behavior of the algorithms in solving all problems.

Based on the results obtained by the ANOVA test and its due validations, we can conclude with 95% certainty that the algorithms evaluated in the test have different means to perform the block classification of these 17 databases (in this sense BR-ELM seems to be superior to the other methods as having the highest accuracy in most of the data sets and also having the highest one in average). We further verified the performance of the methods through statistical tests allowing multiple comparisons at the same time. Therefore, we performed a multiple comparisons post-hoc test called Tukey's test [54] that makes two-to-two comparisons between all the algorithms involved. The numerous comparisons are presented in Table 9 and there is enough evidence in the results obtained to reject the equivalence between BR-ELM and ELM as well as BR-ELM and OP-ELM with a significance level of 0.05 (as both achieving a p-value smaller than 0.05), but not between BR-ELM and P-ELM. Significant differences are also identified between the performance of P-ELM and OP-ELM

with ELM. Figure 8 and Table 9 presents a graphical evaluation of the Tukey test that can corroborate the statements presents in Table 9, especially in the demonstration of the lowest and highest values present in the comparison between the analyzed algorithms.

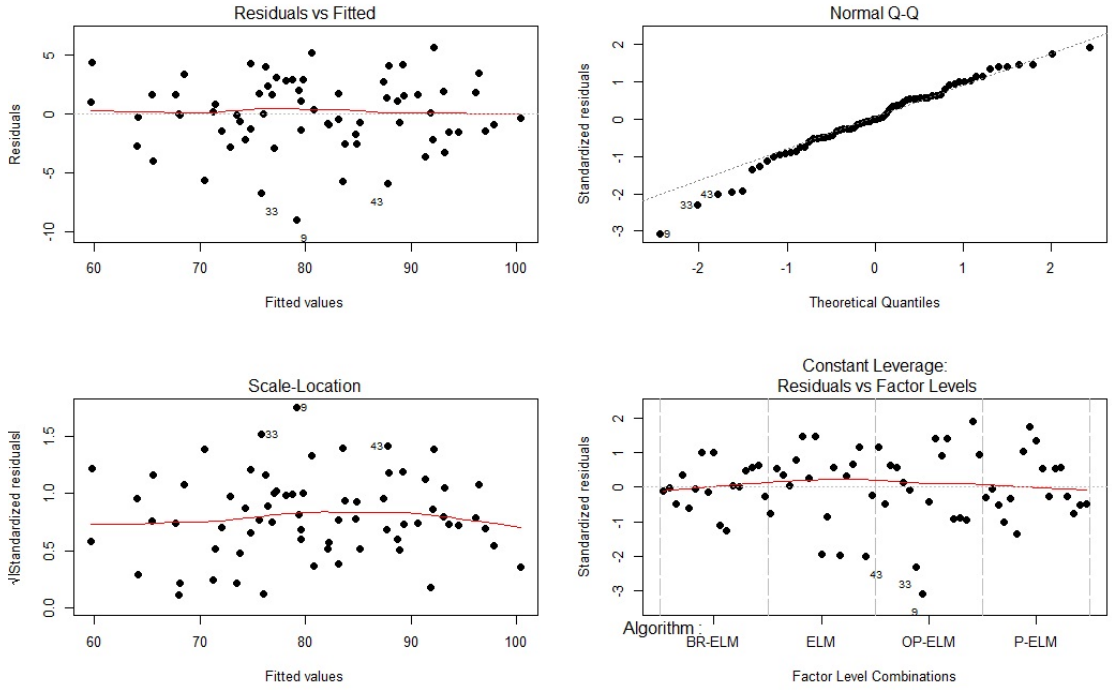

Figure 7. Validation results of the ANOVA test.

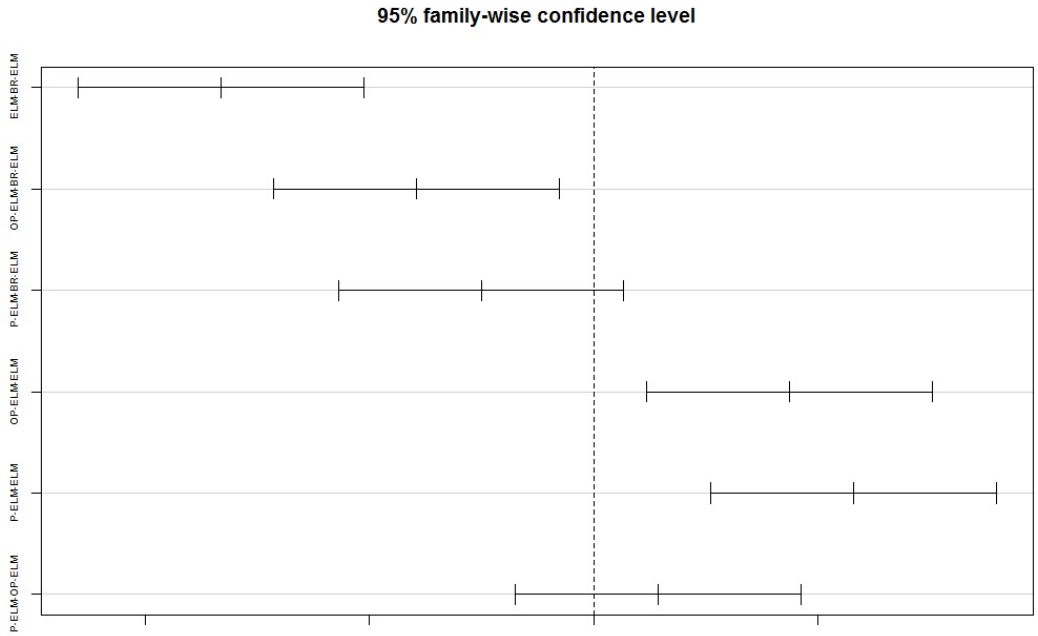

Figure 8. Multiple comparisons of Tukey.

Table 9. Tukey multiple comparisons of means. 95% family-wise confidence level.

Algorithm	diff	lwr	upr.	p-adj
ELM-BR-ELM	−8.304	−11.486961	−5.1212742	0.0000001
OP-ELM-BR-ELM	−3.951	−7.134020	−0.7683330	0.0094658
P-ELM-BR-ELM	−2.512	−5.695785	0.6699022	0.1673468
OP-ELM-ELM	4.352	1.170098	7.5357846	0.0036086
P-ELM-ELM	5.791	2.608333	8.9740199	0.0000793
P-ELM-OP-ELM	1.438	−1.744608	4.6210787	0.6282214

Another way to visualize the performance of the model proposed is to perform the multiple comparisons of Dunnett [54], where one of the algorithms is highlighted as reference, and all comparisons can be made with this reference. Table 10 shows the result of the comparisons of this test and Figure 9 is the graph result obtained by the multiple comparisons; again, a value of $Pr < 0.05$ indicates statistical significance for rejecting the equal performance assumption. Thus, ELM and OP-ELM can be clearly outperformed by BR-ELM. Although P-ELM can be not outperformed it is remarkable the BR-ELM leads to a lower number of neurons than P-ELM in 13 out of 17 data sets and in average to a significantly lower number (54.69 vs. 143.42). Figure 9 showing the comparison of the values present in Table 10.

Table 10. Multiple Comparisons of Means: Dunnett Contrasts.

Algorithm	Estimate	Std. Error	t Value.	Pr (> | t |)
ELM-BR-ELM	−8.304	1.196	−6.944	0.001
OP-ELM-BR-ELM	−3.951	1.196	−3.304	0.00512
P-ELM-BR-ELM	−2.513	1.196	−2.101	0.10281

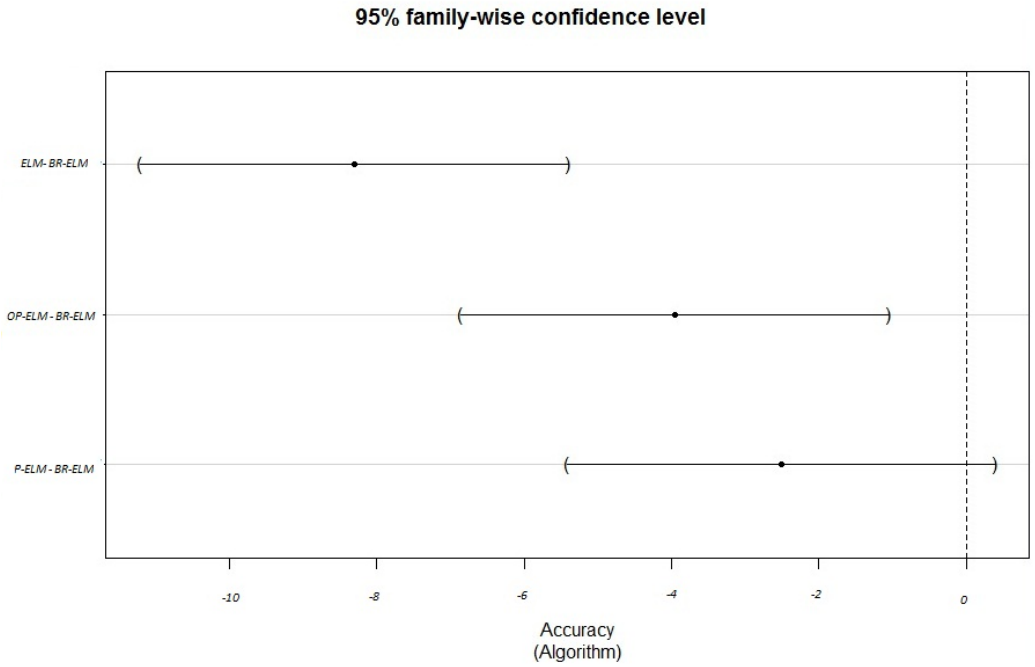

Figure 9. Dunnett Contrasts.

Through this multiple comparison tests we can thus affirm with 95% confidence that ELM and OP-ELM can be clearly outperformed by BR-ELM. Although P-ELM cannot be outperformed, it is remarkable that BR-ELM leads to a lower number of neurons than P-ELM in 13 out of 17 data sets and in average to a significantly lower number (54.69 vs. 143.42).

4.4. Benchmark Regression Datasets

To verify the ability of the BR-ELM model in the regression tests, it was performed with bases commonly used in problems for this purpose. The model was again compared to the OP-ELM approach. The seven regression sets use in these tests (cf. Table 11) can be classified into three groups of data:

- Datasets with relatively small size and low dimensions, for example, Cloud [55] and Container Crane Controller [56].
- Datasets with low size and low dimensions, for example, Carbon nanotubes [57].
- Datasets with relatively large size and low dimensions, for example, Quake [58], Abalone [55], Absenteeism at work [59] and Air Quality (Without features Date and Time.) [60] and Combined Cycle Power Plant [61]

Table 11. Data sets used in the experiments for regression problems.

Dataset	Acronym	Feature	Train	Test
Air Quality	AIQ	2	192	80
Container Crane Controller	CON	2	10	5
Carbon Nanotubes	CAR	5	6698	2870
Cloud	CLO	9	72	36
Absenteeism at work	ABS	21	518	222
Combined Cycle Power Plant	CPP	4	6698	2870
Abalone	ABA	8	2874	1393

In the regression test (Table 12), the BR-ELM results were the best in five of the seven bases evaluated. That shows that despite the longer time to solve problems, the approximation of the values is more consistent with the BR-ELM. In this context, it is possible to evaluate that the model, despite the time excess, presents a better presentation than the OP-ELM using a set of more significant neurons. In contrast, the number of neurons used in the models (Table 13) was much lower in BR-ELM in almost all experiments than the models were compared. This shows that the resampling technique can also identify the essential neurons, generating models with responses that are closer to what is intended to be evaluated.

Table 12. Result of regression test—RMSE.

Dataset	BR-ELM	OP-ELM
AIQ	7.2119 (0.4818)	4.1136 (0.6284)
CON	0.5552 (0.3863)	0.9054 (3.2029)
CAR	0.2399 (0.0055)	4.1892 (0.1252)
CLO	0.0123 (0.0093)	0.0223 (0.0113)
ABS	14.25 (2.47)	13.90 (1.64)
CPP	4.4263 (0.3261)	4.1645 (0.1262)
ABA	0.6598 (0.0179)	2.2073 (0.0493)

Table 13. Result of regression test—number of neurons.

Dataset	BR-ELM	OP-ELM
AIQ	82.26 (5.77)	172.66 (8.58)
CON	16.67 (7.50)	5.00 (0.00)
CAR	74.26 (9.93)	84.83 (14.65)
CLO	74.26 (11.44)	80.00 (6.29)
ABS	25.46 (10.41)	46.33 (14.79)
CPP	107.50 (21.71)	135.50 (12.95)
ABA	56.33 (8.99)	55.66 (7.73)

4.5. Pattern Classification Tests Using Complex Real Datasets

The BR-ELM model proposed in this paper will solve the real problems of various kinds. To this end, its performance will be compared with recent further approaches that also use techniques to select hidden layer neurons (statistical, bayesian, among others).

The models used for comparisons will be the correlation coefficient pruning (CE-ELM) [62], partial least squares regression in combination with ELMs (PS-ELM) [39], and Automatic Relevance Determination (ARD-ELM) [38] for pruning ELMs. All neurons in the hidden layer of all models are the same (sigmoidal). The number of hidden layer neurons for all models was set to 200 neurons, and the BR-ELM parameter values are the same as those elicited by cross-validation during the previous pattern classification experiments. These factors are essential to maintain the ability to evaluate improvement in resampling technique results along with regularization l1.

The evaluation criteria follow the same assumptions as in the standard classification or regression tests, and the machine configuration for the tests follows the same characteristics as listed above. Highlighted bold values represent the best values when comparing models. These models will act to solve complex problems of various kinds, as listed below.

4.5.1. Objectivity or Subjectivity of Sporting Articles

The dataset (sports articles) from this experiment represents 1000 English sports articles from 50 different sites. Professional journalists or sports blog fans wrote most of these articles. Articles ranged from 47 to 4283 words, with an average length of 697 words. For identification, we used Amazon's Mechanical Turk tool [63]. 658 articles were labeled as objective (1), and 342 were labeled as subjective (-1). The database is unbalanced and made up of articles from various sources and authors. Therefore, the linguistic structures of the articles vary significantly, making the classification task difficult. Most features extracted from the text are linked to the frequency of specific terms. All features can be viewed in the paper by Hall et al. [63]. Table 14 presents the results obtained by the pruned ELM models to identify whether or not the evaluated articles are objective.

Table 14. Objectivity or subjectivity of sporting articles.

Model	L_p	Sensibility	Specificity	AUC	Accuracy	Training and Test Time
BR-ELM	35.93 (14.13)	72.99 (3.59)	87.68 (2.81)	0.8034 (0.01)	85.34 (1.60)	60.01 (4.48)
CE-ELM	99.86 (6.88)	63.65 (25.66)	83.91 (8.37)	0.7420 (0.10)	74.83 (11.31)	0.08 (0.03)
PS-ELM	68.64 (16.14)	71.20 (1.16)	81.14 (0.98)	0.7617 (0.12)	79.23 (1.76)	54.14 (2.16)
ARD-ELM	71.20 (16.67)	73.06 (3.92)	87.03 (2.45)	0.8005 (0.02)	81.86 (2.04)	50.74 (3.87)
ELM	200.00 (0.00)	65.14 (8.37)	84.53 (3.31)	0.7483 (0.03)	77.36 (2.54)	0.01 (0.00)

The results of the BR-ELM model (Table 14) were superior in almost all evaluated items, except in its execution time. However, it should be noted that it was the model that used a simpler architecture to solve the problem and obtained the best results of correctness in the objectivity of sports articles. It is noteworthy that using all the features of the problem, the average results obtained by the BR-ELM model were superior to the state-of-the-art ([63]). That demonstrates that the model proposed in this paper has high accuracy in the evaluation of sports articles, with the lowest confusion rate between the possible variations between false positives and false negatives.

4.5.2. Suspicious Firm Classification

Companies that cheat markets are apparent these days. Therefore, databases for this purpose are relevant to be solved by intelligent models. The database uses a case study of an external audit firm that wants to improve its actions. Annual data were collected from 777 companies from 14 different sectors. The characteristics of the database can be viewed in the paper by Hooda et al. [64].

Table 15 presents the results of intelligent models in identifying companies with audit problems. The results presented by the BR-ELM were extremely satisfactory due to its high performance in the

configurations defined in the fraud detection tests in companies. A relevant factor is that the model was immensely superior to state of the art for the subject increasing by approximately 4% the ability to detect fraud in company audits (94% accuracy in the paper by Hooda et al. Versus 98.72% in this paper.), and this with the lowest number of neurons.

Table 15. Classification of fraud in companies.

Model	L_p	Sensibility	Specificity	AUC	Accuracy	Training and Test Time
BR-ELM	34.76 (16.80)	100.0 (0.00)	97.94 (1.45)	0.9897 (0.00)	98.72 (0.92)	58.85 (3.26)
CE-ELM	99.69 (4.56)	75.51 (11.12)	81.85 (34.31)	0.7936 (0.11)	76.75 (10.48)	0.08 (0.04)
PS-ELM	76.42 (5.82)	98.15 (0.07)	84.19 (0.14)	0.9122 (0.04)	91.18 (0.54)	63.21 (1.78)
ARD-ELM	111.50 (14.20)	100.00 (0.00)	97.82 (0.00)	0.9891 (0.01)	98.63 (0.75)	66.94 (6.61)
ELM	200.00 (0.00)	94.59 (2.60)	84.92 (4.46)	0.8975 (0.02)	88.46 (2.76)	0.11 (0.02)

5. Conclusions

After the tests were performed and the statistical evaluations completed, we can conclude that the pruning methods presented in this work maintain the average accuracy of data classification correctness when compared to an ELM structure with the complete neurons in the hidden layer, besides pruning methods commonly used in the literature. With fewer neurons, the probability of overfitting is lower, which in turn increases the likelihood that the model performs well new separate test data. Furthermore, it reduces the calculations of the data between the hidden layer and the output layer. The model proposed in this paper assertively acts on complex regression and pattern classification problems. Its main contribution to science is the use of resampling and LARS to define subgroups of neurons in the hidden layer of artificial neural networks. That allows the definition of neurons in the hidden layer of a model to be more consistent with the problems to be solved, ensuring assertive outputs. The model was confirmed as efficient in solving real complex problems, thus allowing its results to be better than those presented for the state of the art of the analyzed problems. The proposed methods and cut-off factors eliminate unnecessary neurons, whereas the BR-ELM method did so more efficiently than OP-ELM and P-ELM for most of the data sets, and this with achieving better model accuracies. It was verified in the course of the tests that pruning algorithms significantly improved the final response of the model, but some adjustments in the pruning variable can be performed to improve the algorithms further. Regarding the tests related to real and complex problems, the BR-ELM model was superior to the other pruning models used in the pattern classification tests, demonstrating that their results even surpass the state of the art of the problem. Bolasso's approach brought stability and more efficient answers to solve problems of various specificities.

For future work, optimization methods can be used to evaluate each case of the data set used in the model, finding the best pruning factor for the neurons in the hidden layer. The use of optimization methods can find hyperparameters more efficiently and obtain optimal results in the construction of the internal structures of the model (weights, bias, the first number of neurons). Studies on weight restrictions or limitations [65–69] as an alternative approach to solving the problem of overfitting may be targets for future implementations.

Author Contributions: Conceptualization, P.V.d.C.S. and L.C.B.T.; methodology, G.R.L.S.; software, P.V.d.C.S. and G.R.L.S.; validation, A.d.P.B., L.C.B.T. and E.L.; formal analysis, P.V.d.C.S.; investigation, L.C.B.T.; resources, P.V.d.C.S.; data curation, P.V.d.C.S. and A.d.P.B.; writing—original draft preparation, P.V.d.C.S.; writing—review and editing, P.V.d.C.S. and L.C.B.T; visualization, G.R.L.S.; supervision, P.V.d.C.S.; project administration, P.V.d.C.S.; funding acquisition, P.V.d.C.S. and E.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the Brazilian agency CAPES, in part by CNPq, and in part by FAPEMIG.

Acknowledgments: The fifth author acknowledges the support by the "LCM—K2 Center for Symbiotic Mechatronics" within the framework of the Austrian COMET-K2 program. Open Access Funding by the University of Linz.

Conflicts of Interest: The authors declare that there is no conflict of interest in the work.

References

1. Broomhead, D.S.; Lowe, D. *Radial Basis Functions, Multi-Variable Functional Interpolation and Adaptive Networks*; Technical Report; Royal Signals and Radar Establishment Malvern: Malvern, UK, 1988.
2. Pao, Y.H.; Park, G.H.; Sobajic, D.J. Learning and generalization characteristics of the random vector functional-link net. *Neurocomputing* **1994**, *6*, 163–180. [CrossRef]
3. Huang, G.B.; Zhu, Q.Y.; Siew, C.K. Extreme learning machine: Theory and applications. *Neurocomputing* **2006**, *70*, 489–501. [CrossRef]
4. Miche, Y.; Sorjamaa, A.; Bas, P.; Simula, O.; Jutten, C.; Lendasse, A. OP-ELM: Optimally pruned extreme learning machine. *IEEE Trans. Neural Netw.* **2010**, *21*, 158–162. [CrossRef] [PubMed]
5. Rong, H.J.; Ong, Y.S.; Tan, A.H.; Zhu, Z. A fast pruned-extreme learning machine for classification problem. *Neurocomputing* **2008**, *72*, 359–366. [CrossRef]
6. Similä, T.; Tikka, J. Multiresponse Sparse Regression with Application to Multidimensional Scaling. In *Artificial Neural Networks: Formal Models and Their Applications—ICANN 2005*; Duch, W., Kacprzyk, J., Oja, E., Zadrożny, S., Eds.; Springer: Berlin/Heidelberg, Germany, 2005; pp. 97–102.
7. Efron, B.; Hastie, T.; Johnstone, I.; Tibshirani, R. Least angle regression. *Ann. Stat.* **2004**, *32*, 407–499. [CrossRef]
8. Bach, F.R. Bolasso: Model Consistent Lasso Estimation Through the Bootstrap. In *Proceedings of the 25th International Conference on Machine Learning*; ACM: New York, NY, USA, 2008; pp. 33–40. [CrossRef]
9. Breiman, L. Bagging Predictors. *Mach. Learn.* **1996**, *24*, 123–140. [CrossRef]
10. Pao, Y.H.; Takefuji, Y. Functional-link net computing: Theory, system architecture, and functionalities. *Computer* **1992**, *25*, 76–79. [CrossRef]
11. Martínez-Martínez, J.M.; Escandell-Montero, P.; Soria-Olivas, E.; Martín-Guerrero, J.D.; Magdalena-Benedito, R.; Gómez-Sanchis, J. Regularized extreme learning machine for regression problems. *Neurocomputing* **2011**, *74*, 3716–3721. [CrossRef]
12. Ljung, L. *System Identification: Theory for the User*; Prentice Hall PTR, Prentic Hall Inc.: Upper Saddle River, NJ, USA, 1999.
13. James, G.; Witten, D.; Hastie, T.; Tibshirani, R. *An Introduction to Statistical Learning*; Springer: Berlin/Heidelberg, Germany, 2013; Volume 112.
14. Tikhonov, A.N. On the solution of ill-posed problems and the method of regularization. In *Doklady Akademii Nauk*; Russian Academy of Sciences: Moscow, Russia, 1963; Volume 151, pp. 501–504.
15. Bauer, F.; Lukas, M. Comparing parameter choice methods for regularization of ill-posed problems. *Math. Comput. Simul.* **2011**, *81*, 1795–1841. [CrossRef]
16. Csáji, B.C. Approximation with artificial neural networks. *Fac. Sci. Etvs Lornd Univ. Hung.* **2001**, *24*, 48.
17. 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*, 2413–2421. [CrossRef]
18. Yu, Q.; Miche, Y.; Eirola, E.; van Heeswijk, M.; Séverin, E.; Lendasse, A. Regularized extreme learning machine for regression with missing data. *Neurocomputing* **2013**, *102*, 45–51. [CrossRef]
19. Hastie, T.; Tibshirani, R.; Friedman, J.; Franklin, J., The elements of statistical learning: Data mining, inference and prediction. In *The Elements of Statistical Learning: Data Mining, Inference and Prediction*; Springer: Berlin/Heidelberg, Germany, 2005; Volume. 27, pp. 83–85.
20. Escandell-Montero, P.; Martínez-Martínez, J.M.; Soria-Olivas, E.; Guimerá-Tomás, J.; Martínez-Sober, M.; Serrano-López, A.J. Regularized Committee of Extreme Learning Machine for Regression Problems. In Proceedings of the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2012, Bruges, Belgium, 25–27 April 2012; pp. 251–256.
21. Kuncheva, L. *Combining Pattern Classifiers: Methods and Algorithms*; Wiley-Interscience (John Wiley & Sons): Chichester, West Sussex, UK, 2004.
22. Kassani, P.H.; Teoh, A.B.J.; Kim, E. Sparse pseudoinverse incremental extreme learning machine. *Neurocomputing* **2018**, *287*, 128–142. [CrossRef]
23. Zhao, Y.P.; Pan, Y.T.; Song, F.Q.; Sun, L.; Chen, T.H. Feature selection of generalized extreme learning machine for regression problems. *Neurocomputing* **2018**, *275*, 2810–2823. [CrossRef]

24. Xu, S.; Wang, J. Dynamic extreme learning machine for data stream classification. *Neurocomputing* **2017**, *238*, 433–449. [CrossRef]
25. Peng, Y.; Wang, S.; Long, X.; Lu, B.L. Discriminative graph regularized extreme learning machine and its application to face recognition. *Neurocomputing* **2015**, *149*, 340–353. [CrossRef]
26. Huang, G.; Song, S.; Gupta, J.N.D.; Wu, C. Semi-Supervised and Unsupervised Extreme Learning Machines. *IEEE Trans. Cybern.* **2014**, *44*, 2405–2417. [CrossRef]
27. Belkin, M.; Niyogi, P.; Sindhwani, V. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *J. Mach. Learn. Res.* **2006**, *7*, 2399–2434.
28. Silvestre, L.J.; Lemos, A.P.; Braga, J.P.; Braga, A.P. Dataset structure as prior information for parameter-free regularization of extreme learning machines. *Neurocomputing* **2015**, *169*, 288–294. [CrossRef]
29. Pinto, D.; Lemos, A.P.; Braga, A.P.; Horizonte, B.; Gerais-Brazil, M. An affinity matrix approach for structure selection of extreme learning machines. In *Proceedings*; Presses universitaires de Louvain: Louvain-la-Neuve, Belgium, 2015; p. 343.
30. Mohammed, A.A.; Minhas, R.; Wu, Q.J.; Sid-Ahmed, M.A. Human face recognition based on multidimensional PCA and extreme learning machine. *Pattern Recognit.* **2011**, *44*, 2588–2597. [CrossRef]
31. Cao, J.; Zhang, K.; Luo, M.; Yin, C.; Lai, X. Extreme learning machine and adaptive sparse representation for image classification. *Neural Netw.* **2016**, *81*, 91–102. [CrossRef] [PubMed]
32. Iosifidis, A.; Tefas, A.; Pitas, I. On the kernel extreme learning machine classifier. *Pattern Recognit. Lett.* **2015**, *54*, 11–17. [CrossRef]
33. Xin, J.; Wang, Z.; Qu, L.; Wang, G. Elastic extreme learning machine for big data classification. *Neurocomputing* **2015**, *149*, 464–471. [CrossRef]
34. Musikawan, P.; Sunat, K.; Kongsorot, Y.; Horata, P.; Chiewchanwattana, S. Parallelized Metaheuristic-Ensemble of Heterogeneous Feedforward Neural Networks for Regression Problems. *IEEE Access* **2019**, *7*, 26909–26932. [CrossRef]
35. Liangjun, C.; Honeine, P.; Hua, Q.; Jihong, Z.; Xia, S. Correntropy-based robust multilayer extreme learning machines. *Pattern Recognit.* **2018**, *84*, 357–370. [CrossRef]
36. Chen, B.; Wang, X.; Lu, N.; Wang, S.; Cao, J.; Qin, J. Mixture correntropy for robust learning. *Pattern Recognit.* **2018**, *79*, 318–327. [CrossRef]
37. Gao, J.; Chai, S.; Zhang, B.; Xia, Y. Research on Network Intrusion Detection Based on Incremental Extreme Learning Machine and Adaptive Principal Component Analysis. *Energies* **2019**, *12*, 1223. [CrossRef]
38. de Campos Souza, P.V.; Araujo, V.J.S.; Araujo, V.S.; Batista, L.O.; Guimaraes, A.J. Pruning Extreme Wavelets Learning Machine by Automatic Relevance Determination. In *Engineering Applications of Neural Networks*; Macintyre, J., Iliadis, L., Maglogiannis, I., Jayne, C., Eds.; Springer International Publishing: Cham, Switzerland, 2019; pp. 208–220.
39. de Campos Souza, P.V. Pruning method in the architecture of extreme learning machines based on partial least squares regression. *IEEE Lat. Am. Trans.* **2018**, *16*, 2864–2871. [CrossRef]
40. He, B.; Sun, T.; Yan, T.; Shen, Y.; Nian, R. A pruning ensemble model of extreme learning machine with *L_{1/2}* regularizer. *Multidimens. Syst. Signal Process.* **2017**, *28*, 1051–1069. [CrossRef]
41. Fan, Y.T.; Wu, W.; Yang, W.Y.; Fan, Q.W.; Wang, J. A pruning algorithm with L 1/2 regularizer for extreme learning machine. *J. Zhejiang Univ. Sci. C* **2014**, *15*, 119–125. [CrossRef]
42. Chang, J.; Sha, J. Prune Deep Neural Networks With the Modified *L_{1/2}* Penalty. *IEEE Access* **2018**, *7*, 2273–2280. [CrossRef]
43. Alemu, H.Z.; Zhao, J.; Li, F.; Wu, W. Group *L_{1/2}* regularization for pruning hidden layer nodes of feedforward neural networks. *IEEE Access* **2019**, *7*, 9540–9557. [CrossRef]
44. Xie, X.; Zhang, H.; Wang, J.; Chang, Q.; Wang, J.; Pal, N.R. Learning Optimized Structure of Neural Networks by Hidden Node Pruning With L_1 Regularization. *IEEE Trans. Cybern.* **2019**. [CrossRef]
45. Schaffer, C. Overfitting Avoidance as Bias. *Mach. Learn.* **1993**, *10*, 153–178. [CrossRef]
46. Islam, M.; Yao, X.; Nirjon, S.; Islam, M.; Murase, K. Bagging and Boosting Negatively Correlated Neural Networks. *IEEE Trans. Syst. Man Cybern. Part B Cybern.* **2008**, *38*, 771–784. [CrossRef] [PubMed]
47. Murphy, K.P. *Machine Learning: A Probabilistic Perspective*; The MIT Press: Cambridge, MA, USA, 2012.
48. Girosi, F.; Jones, M.; Poggio, T. Regularization Theory and Neural Networks Architectures. *Neural Comput.* **1995**, *7*, 219–269. [CrossRef]
49. Tibshirani, R. Regression shrinkage and selection via the lasso. *J. R. Stat. Soc.* **1996**, *58B*, 267–288. [CrossRef]

50. Efron, B.; Tibshirani, R.J. *An Introduction to the Bootstrap*; CRC Press: Boca Raton, FL, USA, 1994.
51. Lichman, M. *UCI Machine Learning Repository*; University of California: Irvine, CA, USA, 2013.
52. Ho, T.K.; Kleinberg, E.M. Building projectable classifiers of arbitrary complexity. In Proceedings of the 13th International Conference on Pattern Recognition, Vienna, Austria, 25–29 August 1996;Volume 2, pp. 880–885.
53. Hsu, C.W.; Chang, C.C.; Lin, C.J. A Practical Guide to Support Vector Classification. 2003. Available online: http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide (accessed on 15 April 2010).
54. Montgomery, D.C. *Design and Analysis of Experiments*; John Wiley & Sons: Hoboken, NJ, USA, 2017.
55. Blake, C. *UCI Repository of Machine Learning Databases*; University of California, Irvine, CA, USA, 1998.
56. Ferreira, R.P.; Martiniano, A.; Ferreira, A.; Romero, M.; Sassi, R.J. Container crane controller with the use of a NeuroFuzzy Network. In *IFIP International Conference on Advances in Production Management Systems*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 122–129.
57. Acı, M.; Avcı, M. Artificial neural network approach for atomic coordinate prediction of carbon nanotubes. *Appl. Phys. A* **2016**, *122*, 631. [CrossRef]
58. Mike, M. *Statistical Datasets*; Carnegie Mellon University Department of Statistics and Data Science: Pittsburgh, PA, USA, 1989.
59. Martiniano, A.; Ferreira, R.; Sassi, R.; Affonso, C. Application of a neuro fuzzy network in prediction of absenteeism at work. In Proceedings of the 7th Iberian Conference on Information Systems and Technologies (CISTI 2012), Madrid, Spain, 20–23 June 2012; pp. 1–4.
60. De Vito, S.; Massera, E.; Piga, M.; Martinotto, L.; Di Francia, G. On field calibration of an electronic nose for benzene estimation in an urban pollution monitoring scenario. *Sens. Actuators B Chem.* **2008**, *129*, 750–757. [CrossRef]
61. Tüfekci, P. Prediction of full load electrical power output of a base load operated combined cycle power plant using machine learning methods. *Int. J. Electr. Power Energy Syst.* **2014**, *60*, 126–140. [CrossRef]
62. de Campos Souza, P.V.; Araujo, V.S.; Guimaraes, A.J.; Araujo, V.J.S.; Rezende, T.S. Method of pruning the hidden layer of the extreme learning machine based on correlation coefficient. In Proceedings of the 2018 IEEE Latin American Conference on Computational Intelligence (LA-CCI), Gudalajara, Mexico, 7–9 November 2018; pp. 1–6. [CrossRef]
63. Hajj, N.; Rizk, Y.; Awad, M. A subjectivity classification framework for sports articles using improved cortical algorithms. *Neural Comput. Appl.* **2018**, *31*, 8069–8085. [CrossRef]
64. Hooda, N.; Bawa, S.; Rana, P.S. Fraudulent Firm Classification: A Case Study of an External Audit. *Appl. Artif. Intell.* **2018**, *32*, 48–64. [CrossRef]
65. Hagiwara, K.; Fukumizu, K. Relation between weight size and degree of over-fitting in neural network regression. *Neural Netw.* **2008**, *21*, 48–58. [CrossRef] [PubMed]
66. Livieris, I.E.; Iliadis, L.; Pintelas, P. On ensemble techniques of weight-constrained neural networks. *Evol. Syst.* **2020**. [CrossRef]
67. Livieris, I.E.; Pintelas, P. An improved weight-constrained neural network training algorithm. *Neural Comput. Appl.* **2019**, *32*, 4177–4185. [CrossRef]
68. Livieris, I.E.; Pintelas, P. An adaptive nonmonotone active set–weight constrained–neural network training algorithm. *Neurocomputing* **2019**, *360*, 294–303. [CrossRef]
69. Livieris, I.E.; Pintelas, E.; Kotsilieris, T.; Stavroyiannis, S.; Pintelas, P. Weight-constrained neural networks in forecasting tourist volumes: A case study. *Electronics* **2019**, *8*, 1005. [CrossRef]

© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).