Applied Soft Computing 109 (2021) 107482

Contents lists available at ScienceDirect

Applied Soft Computing

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

Negative Correlation Hidden Layer for the Extreme Learning Machine

Carlos Perales-González, Francisco Fernández-Navarro, Javier Pérez-Rodríguez *, Mariano Carbonero-Ruz

Department of Quantitative Methods, Universidad Loyola Andalucía, Spain

ARTICLE INFO

Article history:
Received 31 August 2020
Received in revised form 7 April 2021
Accepted 3 May 2021
Available online 15 May 2021

Keywords:
Negative Correlation Learning
Extreme Learning Machine
Feature mapping
Hidden layer
Diversity

ABSTRACT

Extreme Learning Machine (ELM) algorithms have achieved unprecedented performance in supervised machine learning tasks. However, the preconfiguration of the nodes in the hidden layer in ELM models through randomness does not always lead to a suitable transformation of the original features. Consequently, the performance of these models relies on broad exploration of these feature mappings, generally using a large number of nodes in the hidden layer. In this paper, a novel ELM architecture is presented, called Negative Correlation Hidden Layer ELM (NCHL-ELM), based on the Negative Correlation Learning (NCL) framework. This model incorporates a parameter into each node in the original ELM hidden layer, and these parameters are optimized by reducing the error in the training set and promoting the diversity among them in order to improve the generalization results. Mathematically, the ELM minimization problem is perturbed by a penalty term, which represents a measure of diversity among the parameters. A variety of regression and classification benchmark datasets have been selected in order to compare NCHL-ELM with other state-of-the-art ELM models. Statistical tests indicate the superiority of our method in both regression and classification problems.

1. Introduction

The Extreme Learning Machine (ELM) framework was originally presented to the scientific community by Huang et al. [1] as a single hidden layer feedforward neural network, whose connections between input and hidden layers are randomly assigned. The only parameters necessary to be tuned are the weights between the hidden and the output layers: they are analytically estimated by solving the standard least-squares minimization problem [2,3]. This methodology significantly improves the computational burden of neural networks and allows them to process heavy datasets within a reasonable computational time [4]. ELM models have reported competitive performances with respect to both standard machine learning datasets (regression and classification) [5] and other more challenging problems such as image detection [6], time series [7], wind turbine evaluation [8], topological information [9,10] air quality prediction [11] or real-time river flow prediction [12].

As previously explained, the main difference between ELM and other gradient-based neural networks models lies in the feature mapping of their models. For the ELM framework, the parameters that define the transformation of the feature space are not tuned, but are instead randomly generated. As pointed out in the literature, this random configuration of weights and bias of the nodes in the hidden layer can lead to a poor exploration of the transformed space if the number of nodes in the hidden layer is not high enough [13,14], thus giving rise to poor generalization performance. The computational complexity of the ELM model critically depends on the number of hidden nodes, and so different researchers have proposed a variety of methods that address the problem of massive and inefficient feature mapping.

The first group of methods, the so-called incremental and decremental methodologies, are focused on the problem of achieving the optimal number of nodes. As a decremental method, Miche et al. [15] proposed a node pruning method that reduces the sensitivity of the ELM to irrelevant variables, obtaining a more parsimonious ELM model. Later, two regularization penalties were included in the error function of the previously described model: the L_1 regularization term to rank the nodes of the hidden layer, and the L_2 regularization for numerical stability of the method [16]. On the other hand, Huang et al. [17] proposed an incremental methodology in which hidden nodes are added to the layer step by step until a stopping criterion is reached. After that, this approach was improved by including several hidden nodes in the layer in each step and selecting the most important one in the last part of each step [18]. In this way, it is also worth mentioning that Wang et al. [19] improved this incremental method by introducing an enhanced random search.

The second group of methods that overcome the inefficient feature mapping problem do not attempt to reduce the number of nodes in the hidden layer, but attempt to control how the

* Corresponding author.
E-mail address: jperez@uloyola.es (J. Pérez-Rodríguez).

https://doi.org/10.1016/j.asoc.2021.107482
1568-4946/

weights that connect input and hidden layers are assigned. While standard ELM relies on random distributions such as uniform distribution to obtain these weights, Castaño et al. [20] discussed how the principal component analysis method could be used to retrieve information from the training data and later use it for the hidden layer configuration. They proposed PCA-ELM, which uses the coefficients from PCA applied to the training data as weights of the nodes in the hidden layer. Cervellera et al. [21] used quasirandom distributions from the quasi-Monte Carlo method to replace the random assignments of weight and bias, called low-discrepancy sequences (LDSs). The performance results are not subject to a probabilistic coincidence during the training stage because weights in the hidden layer are not random. However, the quasi-Monte Carlo method presents limitations when it is used in multidimensional spaces [22]. In those cases, the random quasi-Monte Carlo method achieves relevancy because it combines low-discrepancy sequences with random distribution. Henríquez et al. [23] followed this reasoning and proposed two parallel layers in the hidden layer: one with weights from LDSs, and another from a random distribution. They are joined by a mathematical convolution, and output weights are analytically obtained as in standard ELM.

The search for efficient feature mapping by assigning diverse weights in the hidden layer is conceptually similar to the promotion of diversity in an ensemble of predictors, also called base learners in these methodologies [24,25]. An ensemble methodology generates base learners whose outputs are different, aiming to achieve a final output prediction that reduces the generalized error [24,26]. The final architecture works as a predictor which is stronger than the base learners, through the combination of their outputs, and thus the searched base learner outputs are diverse. For example, the Negative Correlation Learning (NCL) framework explicitly promotes diversity among base learners by introducing a correlation penalty term into the error function of each base learner so that each model minimizes both mean square error and the correlation of the ensemble [27,28]. Although this ensemble method was thought of as a neural network framework [29–31], it was later adapted to others, such as deep learning [32,33] and ELM [34].

Motivated by the previously mentioned facts, this paper proposes an alternative ELM method that differs from the base model on the following points:

- The proposed method introduces a penalty term in the error function that is inspired by the NCL framework.
- In the proposed method, the basis functions are treated as base learners that produce negatively correlated outputs concerning the outputs of the model. Thus, the final model simultaneously minimizes both the mean square error and the correlation of the outputs of the hidden nodes for the output of the proposed method.
- A parameter optimization procedure has been also proposed based on the functional decomposition of the model.
- The importance of each component of the error function is controlled by a user-specified parameter and the optimal ranges of values for the new parameters associated with the models have been determined by a genetic hyperparameter optimization procedure.

The manuscript is organized as follows: an explanation of the proposed method, along with its comparison to the ELM paradigm in terms of architecture and error function, is provided in Section 2. The experimental framework is presented in Section 3, and the empirical comparisons are in Section 4. Finally, conclusions and discussion are provided in the final segment of the manuscript, Section 5.

2. The proposed method

The goal of this section is to mathematically describe the architecture of the proposed model, the formulation of its error function formulation and the way in which its parameters are obtained. Additionally, the algorithmic steps required to estimate the parameters are also fully described, along with the adaption of the algorithm to classification problems and an analysis of its computational complexity.

2.1. Model architecture

The goal of this section is to describe the main differences between the ELM architecture (in its neural network version) and the architecture of the proposed method (hereinafter referred to as the Negative Correlation Hidden Layer Extreme Learning Machine, NCHL-ELM). In both cases, the architectures of the two models will be described for the regression scenario, and consequently, the training set from which the parameters are estimated will have the form $\mathcal{D} = \{(\mathbf{x}_n, t_n)\}_{n=1}^{N}$, where $\mathbf{x}_n \in \mathbb{R}^K$ is the vector of attributes of the nth pattern, K is the dimension of the input space and $t_n \in \mathbb{R}$ is the desired output.

The ELM output for a pattern $\mathbf{x}$, $f(\mathbf{x}) : \mathbb{R}^K \rightarrow \mathbb{R}$, is obtained by linearly combining the outputs of the hidden nodes, $\phi_1(\mathbf{x}), \ldots, \phi_D(\mathbf{x})$, as:

$$f(\mathbf{x}) = \sum_{d=1}^{D} \beta_d \phi_d(\mathbf{x}; \mathbf{w}_d),$$

where β_d is the connection between the output and the dth hidden node, $\phi_d(\mathbf{x}; \mathbf{w}_d)$ is the sigmoidal transformation of the pattern $\mathbf{x}$, ϕ_d: $\mathbb{R}^K \rightarrow \mathbb{R}$ and $\mathbf{w}_d \in \mathbb{R}^K$ are the weights (parameters) associated with the dth hidden node (randomly generated in the ELM paradigm). Alternatively, the model is typically presented in its matricial form as:

$$\mathbf{f} = \mathbf{H}\boldsymbol{\beta} \in \mathbb{R}^N,$$

where $\mathbf{f} \in \mathbb{R}^N$ are the outputs of the ELM models for the training set, $\boldsymbol{\beta} = (\beta_1 \ldots \beta_D)' \in \mathbb{R}^D$, $\mathbf{H} = \begin{pmatrix} \mathbf{h}(\mathbf{x}_1) & \ldots & \mathbf{h}(\mathbf{x}_N) \end{pmatrix}' \in \mathbb{R}^{N \times D}$ are the outputs of the hidden layer and $\mathbf{h}(\mathbf{x}) = (\phi(\mathbf{x}; \mathbf{w}_d), d = 1, \ldots, D)$ is the output of the hidden layer for a specific pattern $\mathbf{x}$, $\mathbf{h} : \mathbb{R}^K \rightarrow \mathbb{R}^D$.

In the case of the NCHL-ELM model, the output of each hidden node (randomly generated as in the ELM paradigm) is corrected through an extra parameter, denoted as βs. The outputs of the hidden nodes are corrected to be negatively correlated with the final output of the model. Furthermore, the outputs of each hidden node are all assumed to be equally important. This assumption is inherited from the NCL framework (in which all base learners are equally important) [35,36]. Hence, no additional parameters are included in the output layer of the model, allowing us to compare the efficiency and performance of the proposed method with respect to the conventional ELM in the same scenario (same number of parameters and same computational burden). Thus, the parameters to be estimated are included in the hidden layer of the model instead of the output layer (as in the ELM framework). Taking all of these points into account, the output for a pattern, $\mathbf{x}$, $f(\mathbf{x}) : \mathbb{R}^K \rightarrow \mathbb{R}$, is obtained in the NCHL-ELM model by averaging the corrected outputs of the hidden nodes, $b_1(\mathbf{x}), \ldots, b_D(\mathbf{x})$ (assuming that all are equally important), as:

$$f(\mathbf{x}) = \sum_{d=1}^{D} (1/D) b_d(\mathbf{x}; \beta_d, \mathbf{w}_d),$$

where $b_d : \mathbb{R}^K \rightarrow \mathbb{R}$ is the output of the dth hidden node corrected with a parameter β_d to promote diversity in the hidden layer space, and is therefore mathematically defined as:

$$b_d(\mathbf{x}; \beta_d, \mathbf{w}_d) = \beta_d \phi_d(\mathbf{x}; \mathbf{w}_d),$$

where $\phi_d : \mathbb{R}^K \rightarrow \mathbb{R}$, the output of the dth hidden node for the pattern $\mathbf{x}$ and $\mathbf{w}_d \in \mathbb{R}^K$ are the connections between the dth hidden node and the input layer (randomly generated).

It is worth mentioning that the parameter estimation of the model is divided into two stages. In the first stage, the parameters associated with the first part of the hidden layer ($\mathbf{w}_1, \ldots, \mathbf{w}_D$) are randomly generated through uniform distributions. In the second stage, the parameters $\boldsymbol{\beta}$ are estimated to promote diversity between the outputs of the hidden nodes (in its corrected version) and the outputs of the model.

The NCHL-ELM model can also be represented in matricial form as:

$$\mathbf{f} = \mathbf{B}(1/D \ldots 1/D)'$$

where $(1/D \ldots 1/D)' \in \mathbb{R}^D$ and $\mathbf{B} \in \mathbb{R}^{N \times D}$ are the outputs of the hidden nodes corrected by their corresponding parameters, which are defined as $\mathbf{B} = (\mathbf{b}_1 \ldots \mathbf{b}_D) \in \mathbb{R}^{N \times D}$, where $\mathbf{b}_d \in \mathbb{R}^N$ are the outputs of the dth hidden node in its corrected form:

$$\mathbf{b}_d = \boldsymbol{\phi}_d \beta_d,$$

where $\boldsymbol{\phi}_d = (\phi_d(\mathbf{x}_1; \mathbf{w}_d) \ldots \phi_d(\mathbf{x}_N; \mathbf{w}_d))' \in \mathbb{R}^N$. Finally, Fig. 1 graphically represents the main differences in architecture between the two models presented.

2.2. Error function formulation

In this section, the main differences in error function formulations between the base ELM model and the proposed method (NCHL-ELM) will be stressed. In both cases, the methods estimate the same parameters, $\boldsymbol{\beta} \in \mathbb{R}^D$. The ELM framework optimizes its parameters according to regularized least squares regression:

$$\min_{\boldsymbol{\beta} \in \mathbb{R}^D} \|\boldsymbol{\beta}\|^2 + C\|\mathbf{f} - \mathbf{t}\|^2, \tag{1}$$

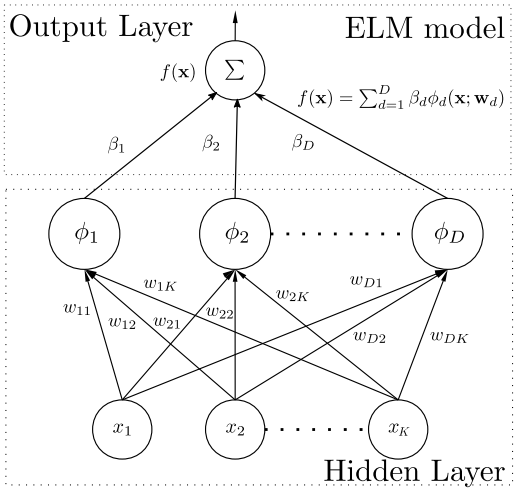
where $\mathbf{t} = (t_1 \ldots t_N)' \in \mathbb{R}^N$. The error function of the NCHL-ELM is very similar to that of ELM, but includes an additional penalty to the error function to promote diversity in the outputs of the hidden nodes involved in the model. As previously mentioned, the penalty included in the error function is inspired by the NCL framework. Taking into account all of these points, the error function of NCHL-ELM is mathematically defined as:

$$\min_{\boldsymbol{\beta} \in \mathbb{R}^D} 1/D\|\boldsymbol{\beta}\|^2 + C\|\mathbf{f} - \mathbf{t}\|^2 - \lambda/D \sum_{d=1}^{D} \|\mathbf{b}_d - \mathbf{f}\|^2, \tag{2}$$

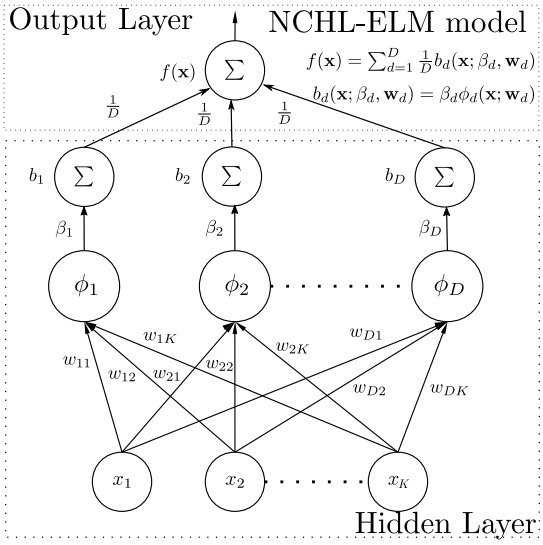
where the third addend represents the previously mentioned penalty term, which promotes diversity between the hidden node outputs (in its corrected form) and the final output of the model. The behavior of this component is very similar to that of the penalty term included in the NCL framework [37].

2.3. Functional decomposition of the error function

In this section, the error function of the NCHL-ELM model is decomposed $\forall k,\ k = 1, \ldots, D$, in two parts, isolating the parameters of the kth hidden node in one of those parts. Thus, the goal of the section is to express the kth component in a mathematical form that can be analytically determined with standard methods (following the guidelines provided in Section 2.4). For ease of reference, the function $f_{(k)}$ will refer to the output of the model



(a) ELM architecture



(b) NCHL-ELM architecture

Fig. 1. Graphical representation of the main differences between ELM and NCHL-ELM architectures.

without considering the kth hidden node (b_k). Thus, the output of the function $f_{(k)}$ for a pattern $\mathbf{x}$ is defined as:

$$f_{(k)}(\mathbf{x}) = (1/D) \sum_{d \neq k} b_d(\mathbf{x}; \beta_d, \mathbf{w}_d).$$

Consequently, $\mathbf{f}_{(k)} \in \mathbb{R}^N$ is the vector of outputs associated with the function $f_{(k)}$ for the training set,

$$\mathbf{f}_{(k)} = (f_{(k)}(\mathbf{x}_1) \ldots f_{(k)}(\mathbf{x}_N))' \in \mathbb{R}^N.$$

The regularization function can be easily decomposed as:

$$\|\boldsymbol{\beta}\|^2 = \|\boldsymbol{\beta}_k\|^2 + \sum_{d \neq k} \|\boldsymbol{\beta}_d\|^2,$$

where the second of these addends does not depend on b_k. For the fitness function, we first rewrite the expression $\mathbf{f} - \mathbf{t}$ as:

$$\mathbf{f} - \mathbf{t} = (1/D) \sum_{d=1}^{D} \mathbf{b}_d - \mathbf{t} = (1/D)\mathbf{b}_k + (\mathbf{f}_{(k)} - \mathbf{t}),$$

where the second addend, which is denoted as $\mathbf{a}_{(k)} = \mathbf{f}_{(k)} - \mathbf{t}$, does not depend on b_k. Hence, the function can be reformulated as follows:

$$\|\mathbf{f}-\mathbf{t}\|^2 = \left\|(1/D)\mathbf{b}_k + \mathbf{a}_{(k)}\right\|^2 = 1/D^2\|\mathbf{b}_k\|^2 + 2/D\mathbf{b}_k'\mathbf{a}_{(k)} + \|\mathbf{a}_{(k)}\|^2.$$

Finally, regarding the diversity term, we will first disaggregate the function as:

$$-\sum_{d=1}^{D}\|\mathbf{b}_d-\mathbf{f}\|^2 = -\|\mathbf{b}_k-\mathbf{f}\|^2 - \sum_{d\neq k}^{D}\|\mathbf{b}_d-\mathbf{f}\|^2.$$

The first part can be reformulated as:

$$\mathbf{b}_k-\mathbf{f} = \mathbf{b}_k - (1/D)\mathbf{b}_k - 1/D\sum_{d\neq k}\mathbf{b}_d = ((D-1)/D)\mathbf{b}_k - \mathbf{f}_{(k)}.$$

Analogously, the second term:

$$\mathbf{b}_d-\mathbf{f} = \mathbf{b}_d - (1/D)\sum_{j\neq k}\mathbf{b}_j - (1/D)\mathbf{b}_k = (\mathbf{b}_d-\mathbf{f}_{(k)}) - (1/D)\mathbf{b}_k,$$

where the first addend, which is denoted as $\mathbf{c}_{d,(k)}$, does not depend on b_k. Hence, it is straightforward to show that the diversity term can be formulated as:

$$-\frac{D-1}{D}\|\mathbf{b}_k\|^2 + 2\mathbf{b}_k'\mathbf{f}_{(k)} - \|\mathbf{f}_{(k)}\|^2 - \sum_{d\neq k}\|\mathbf{c}_{d,(k)}\|^2,$$

Finally, the error function is obtained by including all of the abovementioned terms with their corresponding weights, rewriting the existing norms as dot products, substituting $\mathbf{a}_{(k)}$ for its expression and $\mathbf{b}_k$ for $\mathbf{b}_k = \boldsymbol{\phi}_k\beta_k \in \mathbb{R}^N$ in the previous equations:

$$\frac{1}{D}\left(\beta_k'\left(\mathbf{I} + \frac{C-\lambda(D-1)}{D}\boldsymbol{\phi}_k'\boldsymbol{\phi}_k\right)\beta_k - 2\beta_k'\boldsymbol{\phi}_k'(C\mathbf{t} - (C+\lambda)\mathbf{f}_{(k)})\right) + \mathbf{d}_{(k)}.$$

where $\mathbf{d}_{(k)}$ is constant with respect to b_k and is defined as:

$$\mathbf{d}_{(k)} = \frac{1}{D}\sum_{d\neq k}\|\boldsymbol{\beta}_d\|^2 + C\|\mathbf{a}_{(k)}\|^2 - \frac{\lambda}{D}\left(\|\mathbf{f}_{(k)}\|^2 + \sum_{d\neq k}\|\mathbf{c}_{d,(k)}\|^2\right)$$

2.4. Parameter estimation

The goal of this section is to provide an analytical solution for the parameters of the NCHL-ELM method, $\beta_k, k = 1,\ldots,D$. As in Section 2.3, the error function is in quadratic form with respect to β_k. Thus, the error function will have a minimum if its associated matrix is positive definite, a condition that is met for $C - \lambda(S-1) > 0$.

Thus, the minimum of the error function is obtained by deriving the error function with respect to its parameters ($\beta_k, k = 1,\ldots,D$) and solving the following system of equations ($k = 1,\ldots,S$):

$$\left(\mathbf{I} + \frac{C-\lambda(D-1)}{D}\boldsymbol{\phi}_k'\boldsymbol{\phi}_k\right)\beta_k - \boldsymbol{\phi}_k'(C\mathbf{t} - (C+\lambda)\mathbf{f}_{(k)}) = 0.$$

For the sake of clarity, we rewrite the system of equations using two novel parameters, $\gamma = \frac{C-\lambda(D-1)}{D}$ and $\delta = \frac{\lambda}{\gamma}$ as ($k = 1,\ldots,S$):

$$\left(\frac{\mathbf{I}}{\gamma} + \boldsymbol{\phi}_k'\boldsymbol{\phi}_k\right)\beta_k + (1+\delta)\sum_{d\neq k}(\boldsymbol{\phi}_k'\boldsymbol{\phi}_d)\beta_d = (D+\delta(D-1))\boldsymbol{\phi}_k'\mathbf{t},$$

which is linear. The matrix of the system is:

$$\mathbf{A} = \begin{pmatrix} \frac{\mathbf{I}}{\gamma} + \boldsymbol{\phi}_1'\boldsymbol{\phi}_1 & \ldots & (1+\delta)\boldsymbol{\phi}_1'\boldsymbol{\phi}_d & \ldots & (1+\delta)\boldsymbol{\phi}_1'\boldsymbol{\phi}_D \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ (1+\delta)\boldsymbol{\phi}_d'\boldsymbol{\phi}_1 & \ldots & \frac{\mathbf{I}}{\gamma} + \boldsymbol{\phi}_d'\boldsymbol{\phi}_d & \ldots & (1+\delta)\boldsymbol{\phi}_d'\boldsymbol{\phi}_D \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ (1+\delta)\boldsymbol{\phi}_D'\boldsymbol{\phi}_1 & \ldots & (1+\delta)\boldsymbol{\phi}_D'\boldsymbol{\phi}_d & \ldots & \frac{\mathbf{I}}{\gamma} + \boldsymbol{\phi}_D'\boldsymbol{\phi}_D \end{pmatrix} \in \mathbb{R}^{D\times D}$$

The matrix $\mathbf{A}$ allows us to obtain the solution of the optimization problem:

$$\boldsymbol{\beta} = \mathbf{A}^{-1}(D+\delta(D-1))\boldsymbol{\phi}'\mathbf{t}.$$

where $\boldsymbol{\phi} = (\boldsymbol{\phi}_1 \ldots \boldsymbol{\phi}_D) \in \mathbb{R}^{N\times D}$ is the output of the hidden nodes of the model.

2.5. Algorithmic flow of the NCHL-ELM method

The algorithmic steps required to estimate the parameters of the NCHL-ELM method are analyzed in this section. As previously described, NCHL-ELM is composed of two stages: the initialization stage (Fig. 2, steps 1–4) and the parameter estimation stage (Fig. 2, steps 5–6). In the initialization stage, the algorithm generates the hidden layer nodes output matrix (Fig. 2, steps 1–4). After that, the parameters of the hidden nodes are estimated as described in Section 2.4 (parameter estimation stage, Fig. 2, steps 5–6).

2.6. Adaptation to classification problems

The GNCL method addresses both classification and regression problems in a unified form. In the case of the classification scenario, each target is encoded using "1-of-J", where J is the number of labels of the classification problem. Mathematically, each target is defined as $\mathbf{t}_n = (t_{n1}\ldots t_{nJ}) \in \{0,1\}^J$ (a vector with J components in which each element is equal to either 0 or 1), $n = 1,\ldots,N$, where

$$t_{nj} = \begin{cases} 1 & \mathbf{x}_n \text{ is a pattern of the } j\text{th class,} \\ 0 & \text{otherwise.} \end{cases}$$

Accordingly, the output function of the ensemble transforms the inputs from the input space of dimension K to the J dimension space defined by the labels, $f : \mathbb{R}^K \rightarrow \mathbb{R}^J$, and each class label is approximated via the previously defined regression function. Thus, the classification problem is split into J regression problems that are addressed by the algorithm detailed in Fig. 2. The predicted class label for a test pattern $\boldsymbol{x}$ is stored in a vector $\widehat{y}(\boldsymbol{x}) \in \{0,1\}^J$, in which all values are equal to 0 except for the element in the position with the maximum output value that is equal to 1.

2.7. Analysis of the computational burden

In this section, the computational complexity of the NCHL-ELM method is analyzed and primarily compared with the base ELM neural network model. The computational complexity of the NCHL-ELM method is mainly determined by the number of hidden nodes, D, and the number of patterns of the training set, N. The most time-consuming part of the model is the inversion of the matrix $\mathbf{A}$ of dimension $D \times D$ and its multiplication with a matrix of dimension $D \times N$ (the $\boldsymbol{\phi}'$ matrix). The computational complexity of the multiplication of two matrices $D \times N$ and $D \times D$ is $O(D \cdot N)$, while the complexity of inverting the matrix of dimension D is $O(D^3)$. Therefore, the computational complexity of the proposed method is $O(D^3 + (D \cdot N))$. The computational complexity of the ELM model (in its neural network version) is

NCHL-ELM (D, γ, δ):

Require: Training set: $\mathcal{D} = \{(\mathbf{x}_n, t_n)\}_{n=1}^{N}$, where $\mathbf{x}_n \in \mathbb{R}^K$ and $t_n \in \mathbb{R}$.
Ensure: Optimized neural network model: $\boldsymbol{\beta} = (\beta_1 \dots \beta_D)' \in \mathbb{R}^D$.

1: **for** $d = 1$ until D **do**
2: $\quad \mathbf{w}_d \leftarrow \text{rand}(K)$
3: $\quad \boldsymbol{\phi}_d \leftarrow \text{computeHiddenNodesOutputs}(\mathbf{x}_1, \dots, \mathbf{x}_N)$
4: **end for**
5: $$\mathbf{A} \leftarrow \begin{pmatrix} \frac{\mathbf{I}}{\gamma} + \boldsymbol{\phi}_1'\boldsymbol{\phi}_1 & \dots & (1+\delta)\boldsymbol{\phi}_1'\boldsymbol{\phi}_d & \dots & (1+\delta)\boldsymbol{\phi}_1'\boldsymbol{\phi}_D \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ (1+\delta)\boldsymbol{\phi}_d'\boldsymbol{\phi}_1 & \dots & \frac{\mathbf{I}}{\gamma} + \boldsymbol{\phi}_d'\boldsymbol{\phi}_d & \dots & (1+\delta)\boldsymbol{\phi}_d'\boldsymbol{\phi}_D \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ (1+\delta)\boldsymbol{\phi}_D'\boldsymbol{\phi}_1 & \dots & (1+\delta)\boldsymbol{\phi}_D'\boldsymbol{\phi}_d & \dots & \frac{\mathbf{I}}{\gamma} + \boldsymbol{\phi}_D'\boldsymbol{\phi}_D \end{pmatrix} \in \mathbb{R}^{D \times D}$$
6: $\boldsymbol{\beta} \leftarrow \mathbf{A}^{-1}(D + \delta(D-1))\boldsymbol{\phi}'\mathbf{t}$
7: **return** $\boldsymbol{\beta}$.

Fig. 2. GNCL training algorithm framework.

also conditioned by the number of hidden nodes, D, and the size of the training set, N, and it is exactly the same as the NCHL-ELM method in that it must invert a matrix of dimension $D \times D$ and multiply the result with a matrix of dimension $D \times N$.

3. Experimental framework

An extensive experimental study has been carried out in order to show the community the competitive performance of the negative correlated output layer ELM (NCHL-ELM) method. The description of the datasets employed is available in Section 3.1. The measures used to evaluate the performance are detailed in Section 3.2, whereas brief descriptions of the algorithms, along with the configuration of their parameters, are given in Section 3.3. Finally, the statistical tests implemented to validate the obtained results are specified in Section 3.4.

3.1. Datasets

The experimental framework has been designed to validate the method for both regression and classification problems. Therefore, 35 regression datasets and 39 classification datasets have been selected. This selection was carried out with the aim of including in the benchmark datasets diverse types of classification/regression problems regarding the number of classes, number of training patterns and number of attributes. Tables 1 and 2 show a summary of the main characteristics of the selected datasets: identification number (*ID*), name (*Dataset*), number of patterns (*Size*) and attributes (*Attr.*) for both regression and classification. The number of classes (*Classes*) and their distribution (*Class distribution*) have also been included in the dataset characterization of classification problems. IDs are assigned by sorting the datasets from the largest number of patterns to the lowest. From here until the end, the datasets are noted according to the IDs in this table. While classification datasets are extracted from the UCI Machine Learning Repository [38], regression benchmark problems come from different machine learning repositories: UCI [38], Department of Statistics of the University of Florida[1] and LIACC.[2]

The experimental results were obtained through a 10-fold cross-validation procedure, with 3 repetitions per fold. Thus, a total of 30 error measures were obtained for all of the compared methods, which assures a proper statistical significance of the results. The partitions were the same for all compared models. Input values were standarized following a normal distribution $\mathcal{N}(0, 1)$, regression labels are simple linear scaled to [0, 1] and class labels were binarized, following the "1-of-J" encoding. Additionally, the authors of this study have developed a Python repository (publicly available in Github, https://github.com/cperales/uci-download-process) to automatically download and process (dropping the missing values and binarizing the categorical attributes) all the selected datasets.

3.2. Measures

The metrics used for the performance validation were all standard metrics in their environments.[3] In this sense, the simplicity and successful application of the accuracy rate (*Acc*) has allowed it to be widely used as a performance measure for classification problems. Following the same reasoning, root mean square error (*RMSE*) is the main measure in the validation of an algorithm for regression problems. Those performance metrics are mathematically defined as follows:

- *Acc* is the number of successful hits (correct classifications) relative to the total number of predictions. The mathematical expression of *Acc* is:

$$Acc = (1/N) \sum_{n=1}^{N} I\left(\hat{\mathbf{y}}\left(\mathbf{x}_n\right) = \mathbf{t}_n\right), \tag{3}$$

  where $I(\cdot)$ is the zero–one loss function.
- *RMSE* is the standard deviation of the differences between predicted and target values, and it is defined as:

$$RMSE = (1/N) \sqrt{\sum_{n=1}^{N} (f(\mathbf{x}_n) - t_n)^2}. \tag{4}$$

3.3. Algorithms

The proposed method has been evaluated by comparing its results to those of other recent state-of-the-art ELM proposals. Below, the comparison methods will be briefly described:

- **Negative Correlated Hidden Layer Extreme Learning Machine (NCHL-ELM)** (described in Section 2).

[1] This repository is managed by Larry Winner, url: http://users.stat.ufl.edu/~winner/datasets.html.

[2] LIACC url: http://www.ncc.up.pt/liacc/ML/statlog/datasets.html.

[3] Well-known and standard metrics in the validation of machine learning problems for classification/regression problems.

Table 1
Characteristics of the selected classification datasets, sorted by size (number of instances).

Classification datasets					
ID	Dataset	Size	#Attr.	#Classes	Class distribution
1	statlog-shuttle	43 500	9	7	(34108, 6748, 2458, 132, 37, 11, 6)
2	letter-recognition	20 000	16	26	(813, 805, 803, 796, 792, 789, 787, 786, 783, 783, 775, 773, 768, 766, 764, 761, 758, 755, 753, 752, 748, 747, 739, 736, 734, 734)
3	electrical-grid	10 000	12	2	(6380, 3620)
4	pen-based-recognition-handwritten-digits	7 494	16	10	(780, 780, 780, 779, 778, 720, 720, 719, 719, 719)
5	wall-following-robot-navigation-24	5 456	24	4	(2205, 2097, 826, 328)
6	spambase	4 601	57	2	(2788, 1813)
7	optical-recognition-handwritten-digits	3 823	64	10	(389, 389, 387, 387, 382, 380, 380, 377, 376, 376)
8	thyroid-disease-sick-euthyroid	3 163	20	2	(2870, 293)
9	thyroid-disease-allhyper	2 800	27	4	(2723, 62, 8, 7)
10	thyroid-disease-allbp	2 028	23	5	(936, 716, 265, 82, 29)
11	ozone-level-detection-one	1 848	72	2	(1791, 57)
12	ozone-level-detection-eight	1 847	72	2	(1719, 128)
13	car-evaluation	1 728	21	4	(1210, 384, 69, 65)
14	contraceptive-method-choice	1 473	12	3	(629, 511, 333)
15	cnae-9	1 080	856	9	(120, 120, 120, 120, 120, 120, 120, 120, 120)
16	connectionist-bench	990	10	11	(90, 90, 90, 90, 90, 90, 90, 90, 90, 90, 90)
17	tic-tac-toe-endgame	958	27	2	(626, 332)
18	mammographic-mass	878	14	2	(461, 417)
19	hill-valley-noise	606	100	2	(307, 299)
20	breast-cancer-wisconsin-diagnostic	569	30	2	(357, 212)
21	climate-model-simulation-crashes	540	18	2	(494, 46)
22	congressional-voting-records	435	48	2	(267, 168)
23	monks-problems-2	432	6	2	(290, 142)
24	ionosphere	351	34	2	(225, 126)
25	heart-disease-cleveland	299	23	5	(161, 54, 36, 35, 13)
26	heart-disease-hungarian	294	19	2	(188, 106)
27	breast-cancer	286	39	2	(218, 68)
28	image-segmentation	210	19	7	(30, 30, 30, 30, 30, 30, 30)
29	wine	178	13	3	(71, 59, 48)
30	teaching-assistant-evaluation	151	54	3	(52, 50, 49)
31	monks-problems-1	124	6	2	(62, 62)
32	monks-problems-3	122	6	2	(62, 60)
33	fertility	100	9	2	(88, 12)
34	spectf-heart	80	44	2	(40, 40)
35	soybean-small	47	45	4	(17, 10, 10, 10)
36	balloons-a	20	4	2	(12, 8)
37	balloons-b	20	4	2	(12, 8)
38	balloons-c	20	4	2	(12, 8)
39	balloons-d	16	4	2	(9, 7)

- **Extreme Learning Machine (ELM)** [5]. The ELM neural network model with the L_2 regularization [5]. In this implementation of the model, weights and bias in the hidden layer were randomly generated by following a uniform distribution, whereas output weights were optimized according to the ELM minimization problem.
- **Principal Component Analysis Extreme Learning Machine (PCA-ELM)** [20]. Principal component analysis (PCA) is employed as a deterministic approach to estimate the values of the parameters associated to the hidden layer. The only coefficients stored are those which explain the 90% variability of the training data. Then, the activation function is applied, and output weights are obtained through the minimization ELM problem.
- **Sobol Extreme Learning Machine (S-ELM)** [21]. In this method, the parameters of the hidden layer are randomly generated through low-discrepancy frequency distributions, unlike the standard ELM framework which relies on the uniform distribution. Specifically, the Sobol distribution was employed in the implementation of the method to randomly compute the weights of the hidden nodes, as suggested in [21].
- **Parallel Layer Extreme Learning Machine (PL-ELM)** [23]. In this method, the hidden layer in the standard ELM is replaced by a convolution of two hidden layers. In the first layer, the weights of the hidden nodes are computed with the pseudo-random Sobol distribution, whereas the parameters in the second layer are randomly generated through a uniform distribution. Thus, the method relies on two parallel hidden layers of the neural network architecture for performance improvement.

The performance of the comparison methods critically depends on the configuration of two hyperparameters: the regularization parameter, C, and the number of hidden nodes, D. Those hyperparameters were determined by a grid search in a 5-fold nested cross-validation. The optimal value for the regularization parameter for all comparison methods was determined with the following grid: $C \in \{10^{-3}, \ldots, 10^{3}\}$. The number of hidden nodes, D, in all models except in the PCA-ELM method was determined using the grid: $D \in \{10, 20, 30, 40, 50\}$. In the PCA-ELM method, the number of hidden nodes is dynamically determined in each regression/classification problem according to the amount of principal components necessary to explain 90% of the variance in the training data, as suggested in [20].

The proposed method substitutes its two main hyperparameters (the regularization, C, and diversity, λ) by two hyperparameters, $\gamma > 0$ and $\delta > 0$, which ensures that the matrix $\boldsymbol{A}$ is invertible. Unfortunately, this transformation of the variables forced us to test the model with a set of hyperparameters about which no knowledge for the optimal grid range is available in the specialized literature. Thus, the grid for these two hyperparameters was determined by genetic hyperparameter optimization based on the performance with respect to several selected datasets. As a result of this experimental procedure, the optimal range for these hyperparameters was established as

Table 2
Characteristics of the selected regression, sorted by size (number of instances).

Regression datasets				
ID	Dataset	Size	#Attr.	Repository
1	friedman	40 768	9	LIACC (University of Porto)
2	nwp-min	7 590	21	UCI ML repository
3	parkinsons-motor	5 875	16	UCI ML repository
4	parkinsons-total	5 875	16	UCI ML repository
5	winequality-white	4 898	11	UCI ML repository
6	abalone	4 177	10	UCI ML repository
7	skillcraft	3 338	18	UCI ML repository
8	winequality-red	1 599	11	UCI ML repository
9	airfoil-self-noise	1 503	5	LIACC (University of Porto)
10	forestfires	517	28	UCI ML repository
11	housing	506	13	UCI ML repository
12	yatch	308	6	UCI ML repository
13	servo	167	12	UCI ML repository
14	hybrid	153	11	Department of statistics (University of Florida)
15	usopen-men-2013a	126	168	UCI ML repository
16	usopen-men-2013b	126	168	UCI ML repository
17	frenchopen-men-2013a	123	170	UCI ML repository
18	frenchopen-men-2013b	123	170	UCI ML repository
19	wimbledon-women-2013a	118	170	UCI ML repository
20	wimbledon-women-2013b	118	170	UCI ML repository
21	wimbledon-men-2013a	113	163	UCI ML repository
22	wimbledon-men-2013b	113	163	UCI ML repository
23	frenchopen-women-2013a	111	155	UCI ML repository
24	frenchopen-women-2013b	111	155	UCI ML repository
25	ausopen-men-2013a	103	138	UCI ML repository
26	ausopen-men-2013b	103	138	UCI ML repository
27	slump-flow	103	7	UCI ML repository
28	ausopen-women-2013a	99	141	UCI ML repository
29	ausopen-women-2013b	99	141	UCI ML repository
30	pyrim	74	27	LIACC (University of Porto)
31	usopen-women-2013a	74	106	UCI ML repository
32	brazilian-logistic	60	20	UCI ML repository
33	japanemg	45	5	Department of statistics (University of Florida)
34	diabetes	43	2	LIACC (University of Porto)
35	beer	23	7	Department of statistics (University of Florida)

$\gamma \in \{10^{-7}, 10^{-2}, 10^{-1}, 1, 10^{5}\}$, $\delta \in \{10^{-7}, 10^{-2}, 10^{3}\}$. A detailed explanation of the hyperparameter estimation, along with some illustrative graphics, are provided in Section 4.2.

Finally, all the ELM-based methods were implemented one-by-one by the authors in the Python language programming. Additionally, the library developed by the authors with the algorithms used for the experiments has been publicly uploaded to Github (https://github.com/cperales/).

3.4. Statistical tests

In this paper, the NCHL-ELM method is presented as a novel approach in the ELM paradigm which implements the NCL framework ideas within the configuration of parameters of the hidden layers of an ELM-based neural network. In order to show that the NCHL-ELM model is a promising method in this area, it is crucial to validate its performance with respect to the performance of the comparison methods with statistical tests. In the designed experimental framework, a pre-hoc test was applied with the rankings of the regression/classification methods in the different datasets aiming to evaluate the statistical significance of the rank differences. In the evaluations in which the test detected statistical differences in the rankings of the methods, a post-hoc test was carried out to ascertain which models are distinctive among the multiple comparisons performed using the best performing method as a control method.

Parametric statistical procedures are based on assumptions such as homoscedasticity, independence and normality of data. Since these characteristics cannot be fulfilled for the rankings of the results and the experimental design adopted in the study, non-parametric tests were implemented. First, two nonparametric Friedman tests (with the ranking of *Acc* and *RMSE* of the models as the test variables) were carried out for $\alpha = 0.05$ and $\alpha = 0.10$. As a consequence of the rejection of the two null hypotheses associated with the two pre-hoc tests (the output results of the set of algorithms were statistically similar), the nonparametric Holm post-hoc test was implemented to ascertain whether the control method, NCHL-ELM, statistically outperforms the comparison methods when considering $\alpha = 0.05$ and $\alpha = 0.10$.

4. Empirical results

In this section, the results obtained by the different methods implemented in the selected datasets are analyzed. Thus, the performance results are discussed and statistically compared in Section 4.1. Additionally, the sensitivity of the performance of the proposed method to the hyperparameter configuration, along with the estimation of its optimal grid, are deeply explained in Section 4.2.

4.1. Performance analysis

Tables 3 and 4 show the generalization performances of the methods implemented in the classification/regression problems used in the experimental framework. In Tables 3 and 4, the best result for each dataset is highlighted in bold-face, with the second-best result shown in italics. Finally, the standard deviation for dataset and method is indicated as a subscript.

As can be observed in Tables 3 and 4, the proposed method is competitive for classification problems, as it obtains the first or second best results in most of the datasets considered. For regression problems, the PCA-ELM method is the one which typically achieves the best generalization results. Despite this, it

Table 3
Comparison of the NCHL-ELM method with other recent ELM approaches (Classification analysis): mean and standard deviation of the accuracy (*Acc*) in the generalization sets. The best average result for each dataset is highlighted in bold-face, and the second best result is indicated by italics.

Classification datasets (*Acc* results)					
ID	NCHL-ELM	PCA-ELM	ELM	S-ELM	PL-ELM
1	$0.98930_{0.003}$	$\mathit{0.98963}_{0.003}$	$0.88071_{0.002}$	$0.98101_{0.002}$	$\mathbf{0.98975}_{0.002}$
2	$\mathbf{0.64399}_{0.015}$	$\mathit{0.64364}_{0.015}$	$0.45811_{0.012}$	$0.54739_{0.012}$	$0.54516_{0.020}$
3	$\mathit{0.82153}_{0.012}$	$0.81937_{0.013}$	$0.78457_{0.035}$	$0.82013_{0.014}$	$\mathbf{0.83247}_{0.013}$
4	$\mathbf{0.93502}_{0.010}$	$\mathit{0.93195}_{0.006}$	$0.79211_{0.014}$	$0.90814_{0.007}$	$0.92994_{0.011}$
5	$\mathbf{0.66448}_{0.042}$	$\mathit{0.66067}_{0.039}$	$0.64183_{0.034}$	$0.64091_{0.037}$	$0.65725_{0.043}$
6	$\mathit{0.88580}_{0.031}$	$0.88030_{0.031}$	$\mathbf{0.89248}_{0.035}$	$0.81094_{0.028}$	$0.79781_{0.038}$
7	$\mathit{0.89015}_{0.020}$	$0.88780_{0.015}$	$\mathbf{0.91491}_{0.015}$	$0.81439_{0.026}$	$0.81215_{0.023}$
8	$\mathbf{0.90737}_{0.001}$	$\mathbf{0.90737}_{0.001}$	$0.90705_{0.002}$	$\mathbf{0.90737}_{0.001}$	$\mathbf{0.90737}_{0.001}$
9	$\mathbf{0.97252}_{0.003}$	$\mathbf{0.97252}_{0.003}$	$\mathbf{0.97252}_{0.003}$	$0.97204_{0.004}$	$0.97144_{0.004}$
10	$\mathit{0.62643}_{0.035}$	$0.62282_{0.035}$	$\mathbf{0.62706}_{0.034}$	$0.62198_{0.033}$	$0.62074_{0.028}$
11	$\mathbf{0.96916}_{0.002}$	$\mathbf{0.96916}_{0.002}$	$\mathbf{0.96916}_{0.002}$	$\mathbf{0.96916}_{0.002}$	$\mathbf{0.96916}_{0.002}$
12	$\mathit{0.93070}_{0.002}$	$\mathit{0.93070}_{0.002}$	$\mathit{0.93070}_{0.002}$	$\mathbf{0.93143}_{0.003}$	$\mathit{0.93070}_{0.002}$
13	$0.76701_{0.055}$	$\mathit{0.76951}_{0.054}$	$0.73927_{0.089}$	$\mathbf{0.76969}_{0.054}$	$0.76527_{0.060}$
14	$\mathit{0.51707}_{0.030}$	$0.51643_{0.027}$	$0.47767_{0.033}$	$0.49758_{0.030}$	$\mathbf{0.51819}_{0.038}$
15	$\mathit{0.52315}_{0.064}$	$0.51914_{0.048}$	$\mathbf{0.94537}_{0.029}$	$0.32418_{0.039}$	$0.26996_{0.054}$
16	$\mathit{0.52020}_{0.080}$	$0.49865_{0.071}$	$0.35185_{0.069}$	$0.40438_{0.078}$	$\mathbf{0.52559}_{0.073}$
17	$\mathit{0.77477}_{0.068}$	$0.75195_{0.080}$	$0.62670_{0.054}$	$\mathbf{0.88020}_{0.054}$	$0.72715_{0.085}$
18	$\mathbf{0.81056}_{0.048}$	$0.80525_{0.045}$	$\mathit{0.80985}_{0.047}$	$0.80790_{0.043}$	$0.80481_{0.044}$
19	$\mathbf{0.63113}_{0.062}$	$\mathit{0.62716}_{0.047}$	$0.52068_{0.041}$	$0.50560_{0.031}$	$0.54919_{0.036}$
20	$\mathit{0.95962}_{0.024}$	$\mathbf{0.96089}_{0.024}$	$0.92578_{0.028}$	$0.93277_{0.022}$	$0.94104_{0.030}$
21	$\mathbf{0.92106}_{0.011}$	$\mathit{0.91921}_{0.011}$	$0.91119_{0.010}$	$0.82039_{0.029}$	$0.72454_{0.060}$
22	$\mathit{0.93362}_{0.049}$	$\mathbf{0.93444}_{0.049}$	$0.91829_{0.056}$	$0.92691_{0.050}$	$0.93051_{0.049}$
23	$0.63158_{0.116}$	$0.62290_{0.108}$	$\mathbf{0.66868}_{0.013}$	$\mathit{0.66756}_{0.082}$	$0.65081_{0.107}$
24	$\mathit{0.86545}_{0.064}$	$0.85194_{0.068}$	$\mathbf{0.87201}_{0.058}$	$0.80666_{0.065}$	$0.74754_{0.058}$
25	$\mathbf{0.57533}_{0.058}$	$\mathit{0.56352}_{0.053}$	$0.54163_{0.037}$	$0.56301_{0.050}$	$0.55584_{0.054}$
26	$0.81400_{0.051}$	$0.80587_{0.055}$	$\mathbf{0.81921}_{0.069}$	$\mathit{0.81796}_{0.064}$	$0.81261_{0.062}$
27	$0.73312_{0.113}$	$0.72644_{0.079}$	$\mathbf{0.75194}_{0.115}$	$0.72852_{0.125}$	$\mathit{0.73363}_{0.097}$
28	$\mathbf{0.86667}_{0.070}$	$\mathbf{0.86667}_{0.059}$	$0.66508_{0.072}$	$0.86190_{0.067}$	$0.84921_{0.062}$
29	$\mathit{0.96880}_{0.042}$	$\mathbf{0.97156}_{0.041}$	$0.93733_{0.053}$	$0.93666_{0.068}$	$0.93563_{0.060}$
30	$\mathit{0.56823}_{0.178}$	$0.51510_{0.182}$	$0.49562_{0.160}$	$\mathbf{0.59115}_{0.165}$	$0.54575_{0.201}$
31	$0.63770_{0.123}$	$0.63532_{0.134}$	$0.63929_{0.116}$	$\mathbf{0.67897}_{0.133}$	$\mathit{0.65754}_{0.140}$
32	$0.75685_{0.152}$	$\mathit{0.77625}_{0.124}$	$0.70277_{0.122}$	$0.69336_{0.185}$	$\mathbf{0.82822}_{0.132}$
33	$\mathbf{0.88172}_{0.046}$	$\mathit{0.88141}_{0.032}$	$\mathit{0.88141}_{0.032}$	$0.86869_{0.051}$	$0.85990_{0.093}$
34	$\mathit{0.73333}_{0.140}$	$\mathit{0.73333}_{0.132}$	$\mathbf{0.79167}_{0.113}$	$0.49583_{0.094}$	$0.47917_{0.125}$
35	$0.96333_{0.105}$	$0.96000_{0.095}$	$\mathbf{1.00000}_{0.000}$	$\mathit{0.99333}_{0.036}$	$0.98000_{0.060}$
36	$\mathit{0.97222}_{0.106}$	$\mathit{0.97222}_{0.106}$	$\mathbf{1.00000}_{0.000}$	$0.90000_{0.300}$	$0.87778_{0.304}$
37	$\mathbf{0.98333}_{0.090}$	$\mathbf{0.98333}_{0.090}$	$0.95556_{0.113}$	$0.96667_{0.100}$	$0.95556_{0.136}$
38	$\mathbf{1.00000}_{0.000}$	$0.93333_{0.200}$	$0.71667_{0.236}$	$0.88889_{0.189}$	$\mathit{0.95556}_{0.142}$
39	$0.95000_{0.156}$	$\mathbf{1.00000}_{0.000}$	$0.74167_{0.209}$	$0.71389_{0.277}$	$\mathbf{1.00000}_{0.000}$

Table 4
Comparison of the NCHL-ELM method with other recent ELM approaches (Regression analysis): mean and standard deviation of the root mean square error (*RMSE*) results in the generalization sets. The best average result for each dataset is highlighted in bold-face, and the second best result is indicated by italics.

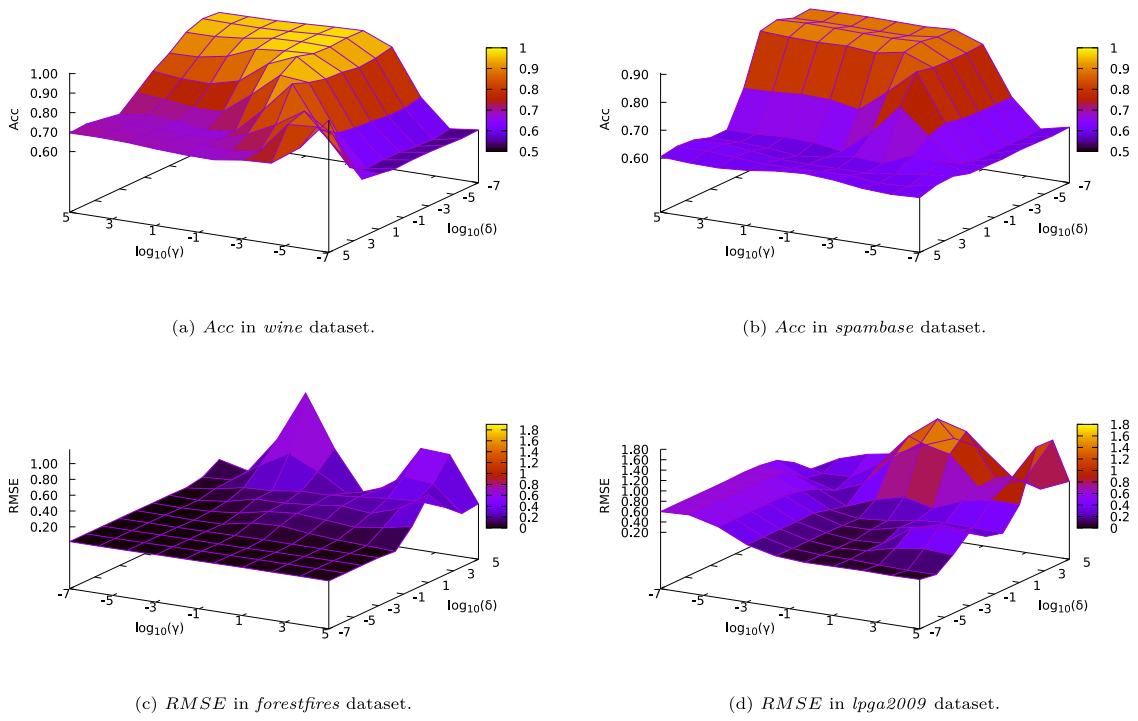
Regression datasets (*RMSE* results)					
ID	NCHL-ELM	PCA-ELM	ELM	S-ELM	PL-ELM
1	$\mathbf{0.06297}_{0.001}$	$0.09223_{0.017}$	$\mathit{0.06316}_{0.001}$	$0.06851_{0.001}$	$0.07840_{0.004}$
2	$\mathbf{0.05731}_{0.002}$	$0.08391_{0.004}$	$\mathit{0.05765}_{0.002}$	$0.20910_{0.005}$	$0.23532_{0.008}$
3	$\mathbf{0.17730}_{0.005}$	$0.20674_{0.005}$	$\mathit{0.17802}_{0.005}$	$0.21704_{0.003}$	$0.22470_{0.009}$
4	$\mathit{0.16486}_{0.006}$	$0.18946_{0.007}$	$\mathbf{0.16441}_{0.005}$	$0.20346_{0.007}$	$0.21228_{0.007}$
5	$\mathbf{0.09567}_{0.002}$	$0.12425_{0.006}$	$\mathit{0.09579}_{0.002}$	$0.12095_{0.003}$	$0.12727_{0.006}$
6	$\mathbf{0.05499}_{0.003}$	$0.08574_{0.004}$	$\mathit{0.05502}_{0.003}$	$0.06450_{0.004}$	$0.07038_{0.005}$
7	$\mathbf{0.13067}_{0.006}$	$0.13439_{0.007}$	$\mathit{0.13140}_{0.007}$	$0.16512_{0.006}$	$0.19016_{0.013}$
8	$\mathbf{0.09950}_{0.004}$	$0.11472_{0.006}$	$\mathit{0.10015}_{0.004}$	$0.11917_{0.009}$	$0.13158_{0.011}$
9	$\mathit{0.07568}_{0.006}$	$0.20479_{0.008}$	$0.07579_{0.006}$	$0.08570_{0.005}$	$\mathbf{0.07526}_{0.005}$
10	$\mathbf{0.01324}_{0.013}$	$0.01834_{0.011}$	$0.01805_{0.012}$	$0.01806_{0.012}$	$\mathit{0.01695}_{0.012}$
11	$\mathbf{0.07148}_{0.009}$	$0.09444_{0.013}$	$\mathit{0.07237}_{0.009}$	$0.10449_{0.015}$	$0.10422_{0.014}$
12	$\mathbf{0.06582}_{0.016}$	$0.12110_{0.014}$	$0.07757_{0.016}$	$0.11455_{0.023}$	$\mathit{0.06857}_{0.014}$
13	$\mathit{0.13049}_{0.023}$	$0.14745_{0.042}$	$0.13549_{0.026}$	$\mathbf{0.12967}_{0.023}$	$0.13269_{0.024}$
14	$\mathit{0.08808}_{0.023}$	$0.11229_{0.030}$	$0.09351_{0.031}$	$0.08830_{0.028}$	$\mathbf{0.08313}_{0.027}$
15	$\mathit{0.34554}_{0.076}$	$\mathbf{0.22704}_{0.089}$	$0.36971_{0.081}$	$0.41658_{0.116}$	$0.43727_{0.122}$
16	$\mathit{0.15763}_{0.037}$	$\mathbf{0.08905}_{0.049}$	$0.16098_{0.039}$	$0.20166_{0.055}$	$0.19453_{0.047}$
17	$\mathit{0.37826}_{0.115}$	$\mathbf{0.16423}_{0.092}$	$0.38705_{0.105}$	$0.40402_{0.130}$	$0.44107_{0.148}$
18	$\mathit{0.15356}_{0.035}$	$\mathbf{0.09309}_{0.039}$	$0.15909_{0.046}$	$0.21520_{0.087}$	$0.21951_{0.081}$
19	$\mathit{0.36010}_{0.098}$	$\mathbf{0.16982}_{0.114}$	$0.36108_{0.087}$	$0.38463_{0.147}$	$0.40252_{0.120}$
20	$\mathit{0.17007}_{0.045}$	$\mathbf{0.10071}_{0.034}$	$0.17211_{0.042}$	$0.24645_{0.106}$	$0.25373_{0.097}$
21	$0.39557_{0.114}$	$\mathbf{0.21589}_{0.117}$	$\mathit{0.38360}_{0.106}$	$0.39027_{0.176}$	$0.40535_{0.187}$
22	$0.14888_{0.035}$	$\mathbf{0.08712}_{0.043}$	$\mathit{0.14812}_{0.048}$	$0.23150_{0.109}$	$0.20609_{0.084}$
23	$0.38240_{0.092}$	$\mathbf{0.22468}_{0.094}$	$0.39245_{0.089}$	$\mathit{0.37558}_{0.084}$	$0.40278_{0.120}$
24	$\mathit{0.19541}_{0.052}$	$\mathbf{0.10562}_{0.056}$	$0.20063_{0.055}$	$0.27307_{0.126}$	$0.26666_{0.109}$
25	$\mathit{0.31043}_{0.074}$	$\mathbf{0.18517}_{0.101}$	$0.31490_{0.078}$	$0.34934_{0.131}$	$0.36877_{0.106}$
26	$\mathit{0.12080}_{0.040}$	$\mathbf{0.06088}_{0.017}$	$0.12184_{0.032}$	$0.19384_{0.073}$	$0.17767_{0.063}$
27	$\mathit{0.18751}_{0.035}$	$0.21085_{0.048}$	$\mathbf{0.18610}_{0.036}$	$0.19568_{0.030}$	$0.19021_{0.041}$
28	$0.39307_{0.124}$	$\mathbf{0.21416}_{0.108}$	$\mathit{0.37812}_{0.089}$	$0.46085_{0.201}$	$0.48816_{0.176}$
29	$0.16123_{0.033}$	$\mathbf{0.08445}_{0.044}$	$\mathit{0.15462}_{0.041}$	$0.22862_{0.114}$	$0.21014_{0.095}$
30	$\mathbf{0.08822}_{0.063}$	$0.11810_{0.079}$	$\mathit{0.09089}_{0.075}$	$0.30741_{0.089}$	$0.31374_{0.076}$
31	$0.34991_{0.138}$	$\mathbf{0.15430}_{0.073}$	$0.37601_{0.146}$	$\mathit{0.33128}_{0.210}$	$0.35592_{0.156}$
32	$\mathbf{0.04276}_{0.020}$	$0.06395_{0.021}$	$\mathit{0.04885}_{0.023}$	$0.07207_{0.026}$	$0.09739_{0.024}$
33	$\mathbf{0.11450}_{0.051}$	$0.12473_{0.091}$	$\mathit{0.11834}_{0.057}$	$0.12887_{0.067}$	$0.13530_{0.077}$
34	$\mathit{0.13893}_{0.050}$	$0.24635_{0.079}$	$0.14335_{0.046}$	$\mathbf{0.13834}_{0.045}$	$0.14551_{0.048}$
35	$\mathit{0.29752}_{0.131}$	$\mathbf{0.27166}_{0.189}$	$0.31650_{0.130}$	$0.35272_{0.170}$	$0.32166_{0.151}$

is important to mention that NCHL-ELM is a more consistent method, since it yields either the best or the second best generalization result for almost every regression dataset. Unfortunately, the PCA-ELM method does not exhibit this competitive behavior: therefore, it could only be considered an interesting proposal for some specific regression datasets.

As previously mentioned, the main difference between the standard ELM and the proposed method lies in the incorporation of an additional term in the error function to promote diversity between the outputs of the hidden nodes. The proposed method converges to the standard ELM model when $\lambda = 0$ (see Eqs. (1) and (2)). For this particular reason, the standard ELM model was included in the experimental study, and, as can be seen in Tables 3 and 4, the model proposed tends to outperform the ELM model in most of the datasets considered. According to those results, we can claim that the incorporation of the diversity term in the error function of the ELM model helps that its generalization capability improves.

Table 5 includes the statistical treatment of the generalization results previously analyzed for the different methods in the considered datasets. Thus, Table 5 shows the results of the Holm test for classification and regression problems along with the mean accuracy achieved by the ensemble models in the generalization sets of the considered classification problems ($\overline{Acc}$), the corresponding mean ranking of each method associated with the *Acc* metric ($\overline{R}_{Acc}$), mean *RMSE* ($\overline{RMSE}$) and mean RMSE ranking ($\overline{R}_{RMSE}$), z-statistic for regression and classification problems, p-values for the ensemble models used for comparison purposes, and adjusted α values ($\alpha_{0.10}$ and $\alpha_{0.05}$). As can be observed in Table 5, the NCHL-ELM method obtains both the best mean *Acc* generalization results, $\overline{Acc} = 0.80760$, and the best mean ranking, $\overline{R}_{Acc} = 2.15385$, in classification problems. The ELM method achieves the second best classification performance results for the considered datasets ($\overline{Acc} = 0.80302$, $\overline{R}_{Acc} = 2.76923$). Although the differences in classification performance with respect to mean *Acc* are not noticeable, they are definitely remarkable with respect to mean ranking, which reinforces the idea that the NCHL-ELM method offers consistent performance. In regression problems, the method with the best mean *RMSE* is not the one which reports the best mean *RMSE* ranking. Specifically, the PCA-ELM method achieves the best results for mean *RMSE* $\overline{RMSE} = 0.14119$, and the NCHL-ELM method is the one with the best mean *RMSE* ranking, $\overline{R}_{RMSE} = 1.82857$. The second best performing generalization results are yielded by NCHL-ELM and ELM, respectively: $\overline{RMSE} = 0.17658$ and $\overline{R}_{RMSE} = 2.60000$.

To determine the statistical significance of the previously reported results, two nonparametric Friedman tests were carried out with the rankings of *Acc* and *RMSE* of the method implemented as the test variables. The acceptance region associated with the null hypothesis for classification datasets is $C_0 = (0, F_{0.05}) = (0, 2.43116)$, and rank differences established the F-distribution statistical value as $F^* = 5.01563 \notin C_0$. For



(a) *Acc* in *wine* dataset.

(b) *Acc* in *spambase* dataset.

(c) *RMSE* in *forestfires* dataset.

(d) *RMSE* in *lpga2009* dataset.

Fig. 3. Hyperparameters study on *Acc* and *RMSE* for the NCHL-ELM method and the parameters γ and δ.

Table 5
Mean *Acc* and *RMSE* generalization performances ($\overline{Acc}$ and $\overline{RMSE}$) and mean generalization rankings ($\overline{R}_{Acc}$ and $\overline{R}_{RMSE}$). Results from the Holm test for $\alpha = 0.05$ and $\alpha = 0.10$ are also shown. For mean performances and rankings, the best results are in bold face, and the second best ones are in italics.

Classification						
Method	$\overline{Acc}$	$\overline{R}_{Acc}$	z-statistic	p-value	$\alpha_{0.05}$	$\alpha_{0.10}$
PL-ELM$_\bullet$	0.77295	3.43590	3.58057	0.00034	0.01250	0.02500
S-ELM$_\bullet$	0.76840	3.34615	3.32991	0.00087	0.01667	0.03333
PCA-ELM$_\bullet$	0.77380	3.29487	3.18670	0.00144	0.02500	0.05000
ELM$_\circ$	*0.80302*	*2.76923*	1.71866	0.08568	0.05000	0.10000
NCHL-ELM	**0.80760**	**2.15385**	–	–	–	–
Regression						
Method	$\overline{RMSE}$	$\overline{R}_{RMSE}$	z-statistic	p-value	$\alpha_{0.05}$	$\alpha_{0.10}$
PL-ELM$_\bullet$	0.22189	4.17143	6.19862	0.00000	0.01250	0.02500
S-ELM$_\bullet$	0.21733	3.77143	5.14032	0.00000	0.01667	0.03333
PCA-ELM$_\circ$	**0.14119**	2.62857	2.11660	0.03429	0.02500	0.05000
ELM$_\circ$	0.18008	*2.60000*	2.04101	0.04125	0.05000	0.10000
NCHL-ELM	*0.17658*	**1.82857**	–	–	–	–

•: Statistical difference with $\alpha = 0.05$
○: Statistical difference with $\alpha = 0.10$.

regression datasets, the Friedman test also shows the significance of the applied method, since the acceptance region is $C_0 = (0, F_{0.05}) = (0, 2.43824)$ and the F-distribution statistical value is $F^* = 19.43863 \notin C_0$. Therefore, the null hypothesis (all algorithms perform equally with respect to average *Acc* and RMSE rankings) is rejected in both cases.

Based on this rejection, the nonparametric post-hoc Holm test is used to compare all methods with the proposed NCHL-ELM method (the control method). As can be found in Table 5, the Holm's tests indicate that the control method (NCHL-ELM) outperforms all remaining ELM-inspired models with respect to *Acc* and *RMSE* for $\alpha = 0.10$. Based on those statistical results, we could hypothesize that the proposed method is an appealing method for classification and regression problems.

4.2. Sensitivity analysis

The proposed method mainly relies on two hyperparameters: γ and δ. These hyperparameters were created to ensure the inversion of the matrix $\boldsymbol{A}$, which is designed assuming that both γ and δ are positive. This assumption is trivially fulfilled by implementing a hyperparameter optimization with positive values of γ and δ. Unfortunately, there is no prior knowledge available in the literature about the optimal grid for these hyperparameters. Hence, in order to present a reproducible experimental framework, the grid with the optimal values for (γ, δ) is required. Below, the procedure followed to determine the values for this grid is fully described.

The optimal grid was determined through a genetic hyperparameter optimization procedure [39,40] in which different hyperparameter configurations were tested with different datasets. The initial population of the genetic algorithm consisted of 100 pairs of γ and δ which were generated by a log-normal distribution of $\mu = 1.0$ and $\sigma = 100.0$. The mutation operator was implemented by considering a normal distribution of $\sigma = 0.01$ over 200 generations. The final set of values for the grid was determined after analyzing different histograms of the optimal pairs (γ, δ) for all the datasets considered in the experimental study. Hence, the optimal grid for the two hyperparameters was: $\gamma \in \{10^{-7}, 10^{-2}, 10^{-1}, 1, 10^{5}\}$, $\delta \in \{10^{-7}, 10^{-2}, 10^{3}\}$.

Fig. 3 shows the average *Acc* and *RMSE* of the proposed method over 10 repetitions per fold for two regression datasets (*forestfires* and *lpga2009*) and two classification datasets (binary *spambase* and multi-class *wine* datasets), respectively, considering a slight expansion of the previously defined optimal grid (with the axis represented by a logarithmic scale). The number of hidden nodes in the hyperparameters study is set to 50 (the maximum value considered in the nested cross-validation). Some additional δ values are included in the plot in order to show the same number of values in the axis for the two hyperparameters considered. Hence, the values of the hyperparameters range in the sets as follows: $\gamma, \delta \in \{10^{-7}, 10^{-5}, 10^{-3}, 10^{-1}, 10^{1}, 10^{3}, 10^{5}\}$. The grid is wide enough to include the optimal values for γ and δ. As

can be seen in Fig. 3, the optimal values of the hyperparameters do not appear along the edges of the grid, but are typically closer to the center.

5. Conclusions

This paper presents a novel Extreme Learning Machine (ELM) model for both regression and classification problems, called the Negative Correlation Hidden Layer Extreme Learning Machine (NCHL-ELM). This predictor minimizes the residuals between the outputs of the model and the expected results while simultaneously explicitly maximizing the diversity among the optimizable parameters of the model. The mathematical formulation of the diversity is inspired by the Negative Correlation Learning (NCL) framework, which is an ensemble regression learning methodology. In the NCHL-ELM method, the diversity is promoted using a penalty term added to the ELM objective function. This addend represents the sum of the differences between each hidden node output and the final output of the model. The parameters of the NCHL-ELM algorithm are found by minimizing this objective function, and the importance of the diversity term is managed through a user-specified parameter, as in the NCL framework. Additionally, in this paper, the optimal range of values for the user-specified parameters of the model is given by genetic hyperparameter optimization.

The performance of NCHL-ELM has been compared with four ELM models over 39 classification datasets and 35 regression datasets. In the experiments, NCHL-ELM models show how robust the performance is when compared with other state-of-the-art methods, as competitive results are consistently provided for most of the datasets considered. Independently of the characteristics of the dataset, NCHL-ELM usually achieves the best or second best result for each case. Based on these results, it can be concluded that the performance is improved when diversity is explicitly promoted in ELM models.

In future work, the generation of the parameters in the first part of the hidden layer could be replaced through random distribution. As discussed in the Introduction, comparison methods seek diversity in the transformed feature space by modifying the weights that connect the input layer with the hidden layer through low-discrepancy sequences, principal component analysis or other techniques. However, in the NCHL-ELM model, the diversity is promoted not in the first part of the hidden layer, but in the final outputs, as the weights that connect the first part of the hidden layer with the final one are estimated to maximize the diversity between the outputs of the hidden layer and the final output of the model. Hence, the study of the double diversity promotion, with respect to both the outputs and the parameters of the feature transformed space, merit further exploration.

CRediT authorship contribution statement

Carlos Perales-González: Acquisition of data, Analysis and/or interpretation of data, Writing - review & editing. **Francisco Fernández-Navarro:** Conception and design of study, Analysis and/or interpretation of data, Writing - original draft. **Javier Pérez-Rodríguez:** Analysis and/or interpretation of data, Writing - original draft. **Mariano Carbonero-Ruz:** Conception and design of study, Writing - review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

All authors approved the version of the manuscript to be published.

References

[1] G.-B. Huang, L. Chen, C.-K. Siew, Universal approximation using incremental constructive feedforward networks with random hidden nodes, IEEE Trans. Neural Netw. 17 (4) (2006) 879–892.
[2] Ö.F. Ertuğrul, A novel randomized machine learning approach: Reservoir computing extreme learning machine, Appl. Soft Comput. 94 (2020) 106433.
[3] Y. Chen, C. Tong, Y. Ge, T. Lan, Fault detection based on auto-regressive extreme learning machine for nonlinear dynamic processes, Appl. Soft Comput. (2021) 107319.
[4] L. Kasun, H. Zhou, G.-B. Huang, C.-M. Vong, Representational learning with extreme learning machine for big data, IEEE Intell. Syst. 28 (6) (2013) 31–34.
[5] G.-B. Huang, H. Zhou, X. Ding, R. Zhang, Extreme learning machine for regression and multiclass classification, IEEE Trans. Syst. Man Cybern. B 42 (2) (2012) 513–529.
[6] A.A. Mohammed, R. Minhas, Q.J. Wu, M.A. Sid-Ahmed, Human face recognition based on multidimensional PCA and extreme learning machine, Pattern Recognit. 44 (10–11) (2011) 2588–2597.
[7] H. Tian, B. Meng, A new modeling method based on bagging ELM for day-ahead electricity price prediction, in: IEEE International Conference on Bio-Inspired Computing: Theories and Applications, IEEE, 2010, pp. 1076–1079.
[8] X. Wen, Modeling and performance evaluation of wind turbine based on ant colony optimization-extreme learning machine, Appl. Soft Comput. 94 (2020) 106476.
[9] Y. Chen, W. Wu, Mapping mineral prospectivity using an extreme learning machine regression, Ore Geol. Rev. 80 (2017) 200–213.
[10] N.L. da Costa, L.A.G. Llobodanin, M.D. de Lima, I.A. Castro, R. Barbosa, Geographical recognition of syrah wines by combining feature selection with extreme learning machine, Measurement 120 (2018) 92–99.
[11] F. Jiang, J. He, T. Tian, A clustering-based ensemble approach with improved pigeon-inspired optimization and extreme learning machine for air quality prediction, Appl. Soft Comput. 85 (2019) 105827.
[12] Z.M. Yaseen, S.O. Sulaiman, R.C. Deo, K.-W. Chau, An enhanced extreme learning machine model for river flow forecasting: State-of-the-art, practical applications in water resource engineering area and future research direction, J. Hydrol. 569 (2019) 387–408.
[13] P. Gastaldo, R. Zunino, E. Cambria, S. Decherchi, Combining ELM with random projections, IEEE Intell. Syst. 28 (6) (2013) 46–48.
[14] G.B. Huang, D.H. Wang, Y. Lan, Extreme learning machines: A survey, Int. J. Mach. Learn. Cybern. 2 (2) (2011) 107–122.
[15] Y. Miche, A. Sorjamaa, P. Bas, O. Simula, C. Jutten, A. Lendasse, OP-ELM: optimally pruned extreme learning machine, IEEE Trans. Neural Netw. 21 (1) (2010) 158–162.
[16] Y. Miche, M. Van Heeswijk, P. Bas, O. Simula, A. Lendasse, TROP-ELM: a double-regularized ELM using LARS and Tikhonov regularization, Neurocomputing 74 (16) (2011) 2413–2421.
[17] G.-B. Huang, L. Chen, Convex incremental extreme learning machine, Neurocomputing 70 (16) (2007) 3056–3062.
[18] G.-B. Huang, L. Chen, Enhanced random search based incremental extreme learning machine, Neurocomputing 71 (16) (2008) 3460–3468.
[19] W. Wang, R. Zhang, Improved convex incremental extreme learning machine based on enhanced random search, in: Unifying Electrical Engineering and Electronics Engineering, Springer, 2014, pp. 2033–2040.
[20] A. Castaño, F. Fernández-Navarro, C. Hervás-Martínez, PCA-ELM: A robust and pruned extreme learning machine approach based on principal component analysis, Neural Process. Lett. 37 (3) (2013) 377–392.
[21] C. Cervellera, D. Macciò, Low-discrepancy points for deterministic assignment of hidden weights in extreme learning machines, IEEE Trans. Neural Netw. Learn. Syst. 27 (4) (2016) 891–896.
[22] C. Lemieux, Monte Carlo and Quasi-Monte Carlo Sampling, 2009, arXiv: arXiv:1011.1669v3.
[23] P.A. Henríquez, G.A. Ruz, Extreme learning machine with a deterministic assignment of hidden weights in two parallel layers, Neurocomputing 226 (2017) 109–116.
[24] L.I. Kuncheva, C.J. Whitaker, Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy, Mach. Learn. 51 (2) (2003) 181–207.
[25] T. Zhou, H. Ishibuchi, S. Wang, Stacked blockwise combination of interpretable TSK fuzzy classifiers by negative correlation learning, IEEE Trans. Fuzzy Syst. 26 (6) (2018) 3327–3341.

[26] L. Zhang, Z. Shi, M.M. Cheng, Y. Liu, J.W. Bian, J.T. Zhou, G. Zheng, Z. Zeng, Nonlinear regression via deep negative correlation learning, IEEE Trans. Pattern Anal. Mach. Intell. 43 (3) (2021) 982–998.
[27] G. Brown, J.L. Wyatt, P. Tino, Managing diversity in regression ensembles, J. Mach. Learn. Res. 6 (Sep) (2005) 1621–1650.
[28] J. Yu, A selective deep stacked denoising autoencoders ensemble with negative correlation learning for gearbox fault diagnosis, Comput. Ind. 108 (2019) 62–72.
[29] Y. Liu, X. Yao, Ensemble learning via negative correlation, Neural Netw. 12 (10) (1999) 1399–1404.
[30] T. Higuchi, X. Yao, Y. Liu, Evolutionary ensembles with negative correlation learning, IEEE Trans. Evol. Comput. 4 (4) (2002) 380–387.
[31] Z.S.H. Chan, N. Kasabov, A preliminary study on negative correlation learning via correlation-corrected data, Neural Process. Lett. 21 (3) (2005) 207–214.
[32] Z. Shi, L. Zhang, Y. Liu, X. Cao, Y. Ye, M.-M. Cheng, G. Zheng, Crowd counting with deep negative correlation learning, in: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 5382–5390.
[33] H.W. Reeve, G. Brown, Diversity and degrees of freedom in regression ensembles, Neurocomputing 298 (2018) 55–68.
[34] C. Perales-Gonzalez, M. Carbonero-Ruz, J. Perez-Rodriguez, D. Becerra-Alonso, F. Fernandez-Navarro, Negative correlation learning in the extreme learning machine framework, Neural Comput. Appl. 32 (17) (2020) 13805–13823.
[35] Y. Liu, X. Yao, Negatively correlated neural networks can produce best ensembles, Aust. J. Intell. Inf. Process. Syst. 4 (3/4) (1997) 176–185.
[36] H. Chen, X. Yao, Regularized negative correlation learning for neural network ensembles, IEEE Trans. Neural Netw. 20 (12) (2009) 1962–1979.
[37] G. Brown, J. Wyatt, Negative correlation learning and the ambiguity family of ensemble methods, in: International Workshop on Multiple Classifier Systems, Springer, 2003, pp. 266–275.
[38] D. Dheeru, E. Karra Taniskidou, UCI Machine Learning Repository, University of California, Irvine, School of Information and Computer Sciences, 2019, URL: http://archive.ics.uci.edu/ml.
[39] A.H. Wright, Genetic algorithms for real parameter optimization, in: Foundations of Genetic Algorithms, Vol. 1, 1991, pp. 205–218.
[40] I. Syarif, A. Prugel-Bennett, G. Wills, SVM parameter optimization using grid search and genetic algorithm to improve classification performance, Telecommun. Comput. Electron. Control 14 (4) (2016) 1502–1509.