Neural Computing and Applications (2020) 32:13805–13823
https://doi.org/10.1007/s00521-020-04788-9

ORIGINAL ARTICLE

Negative correlation learning in the extreme learning machine framework

Carlos Perales-González[1] · Mariano Carbonero-Ruz[1] · Javier Pérez-Rodríguez[1] · David Becerra-Alonso[1] · Francisco Fernández-Navarro[1]

Received: 9 October 2019 / Accepted: 17 February 2020 / Published online: 2 March 2020

Abstract
Extreme learning machine (ELM) has shown to be a suitable algorithm for classification problems. Several ensemble meta-algorithms have been developed in order to generalize the results of ELM models. Ensemble approaches introduced in the ELM literature mainly come from boosting and bagging frameworks. The generalization of these methods relies on data sampling procedures, under the assumption that training data are heterogeneously enough to set up diverse base learners. The proposed ELM ensemble model overcomes this strong assumption by using the negative correlation learning (NCL) framework. An alternative diversity metric based on the orthogonality of the outputs is proposed. The error function formulation allows us to develop an analytical solution to the parameters of the ELM base learners, which significantly reduce the computational burden of the standard NCL ensemble method. The proposed ensemble method has been validated by an experimental study with a variety of benchmark datasets, comparing it with the existing ensemble methods in ELM. Finally, the proposed method statistically outperforms the comparison ensemble methods in accuracy, also reporting a competitive computational burden (specially if compared to the baseline NCL-inspired method).

Keywords Negative correlation learning · Extreme learning machine · Ensemble · Diversity

1 Introduction

Over the years, extreme learning machine (ELM) [30] has become a competitive algorithm for both multi-classification and regression problems. It has been extensively used not only on traditional supervised machine learning problems, but also on time series prediction [57, 69], image classification [10] and speech recognition [67]. Both then single-hidden layer feedforward network (SLFN) [31] and the kernel trick version [30] are widely used in supervised machine learning problems due to its powerful nonlinear mapping capability [18]. The neural network version of the ELM framework relies on the randomness of the weights between the input and the hidden layer. This allows a speedy calculation and has shown good classification results. In turn, this opened the door to deep learning and ensemble methodologies, in order to solve more recent problems [11, 55].

Deep learning and ensembles methodologies are disputing for performance in main supervised machine learning problems, both in multi-classification and regression [54, 66]. Deep learning predictors focus on decomposing features in multi-level representations through hierarchical architectures for the learning tasks and minimizing errors [48]. Deep learning methodologies are considered to be the natural evolution of artificial neural networks (ANNs). These deep architectures have appeared as powerful representation learning techniques in different ways, such as convolutional neural networks [39, 42], denoising autoencoders [61] or generative adversarial networks [25].

An important disadvantage in deep learning neural networks is that they require excessive parameter tuning. Similar to ANN, back propagation [41] and other gradient-based methods [36] used as solvers are the main responsible for the computational burden. One solution to this problem is found in kernel deep architectures, with less

✉ Carlos Perales-González
cperales@uloyola.es

[1] Universidad Loyola Andalucía, Seville, Spain

hyperparameters to tweak [3, 33]. Some examples are convolutional kernel networks [39] or deep Gaussian processes [9, 15]. Different approaches to deep learning have been studied by the ELM community, using autoencoder ELM networks [56] or network embedding [14]. Although these deep architectures have shown to achieve interesting results, it still requires long training times because of the high number of parameters to tune in a nonlinear optimization problem [35, 48, 54].

Ensemble meta-algorithms focus on generalizing the results of the mixture of classifiers (called base learners in ensemble frameworks), looking for diversity among of them [17]. Easiest ensemble methodologies are achieved by training each base learner separately and then combining through weights, so considerable voting methods have been presented in order to improve performance [4, 8, 22, 53, 58]. In this context, bagging [4] and boosting [22] are the most common approaches [1, 19, 66], mainly because of their easy implementation and their balanced trade-off between diversity among base learners and performance of the ensemble.

Boosting is a family of machine learning meta-algorithms that focuses on iteratively reducing error by combining base learners and generating a weighted majority hypothesis. The AdaBoost meta-algorithm is the better-known example for this approach, with extended use in applications and many variations [51]. Training subsets from the data are selected from the complete training set depending on the performance of the previous iteration. In the ELM community, Riccardi et al. [52] proposed a cost-sensitive adaptation for multi-class AdaBoost [27], using ELM as base learners for ordinal regression problems. From a different perspective, Ran et al. [50] adapted the well-known boosting ridge regression algorithm [58] to the ELM context.

Bagging stands for bootstrap aggregating [4]. It is a learning method that generates several versions of a base learner, selecting some subsets from the training set and using them as new learning sets. Thus, each training subset is used to train a different classifier, making this approach easily parallelizable [60, 70]. In the ELM community, Tian and Meng [57] proposed a bagging approach for day-ahead electricity price prediction, leading to a generalizable ensemble algorithm.

The key point of these ensemble meta-algorithms lies in the training data to generate diversity among the base learners. Diverse solutions are fostered implicitly through sampling data. These subsets are selected under the assumption that different data subsets generate diverse base learners [37]. While this data sampling framework is easily generalizable to several classifiers, it depends on how heterogeneous the data are. If the training data are very homogeneous, the subsets would not be different enough and the base learners would be too similar. There are other useful ways of generating base learners, such as kernel diversity [34, 62] or hybrid systems [24, 65]. Unfortunately, those approaches also fail in quantifying diversity among base learners, as diversity is not promoted explicitly while estimating the parameters of the individuals.

Motivated by this fact, Yao et al. [44–46] and Brown et al. [5–7] proposed a novel ensemble method called negative correlation learning (NCL), which fosters explicitly diversity among ensemble individuals by including it in the error function of the models. Thus, in the concept version of the NCL ensemble method, the error function of each individual in the ensemble is made of two terms:

- A penalty term associated with the diversity among ensemble individuals.
- The mean squared error of the model with respect to the desired output.

NCL is inspired on the research by Perrone et al. [49], describing that error in prediction (bias) and diversity among base learners (variance) are two different ideas that may collide when choosing base learners. Besides taking into account bias and variance of each individual base learner, the novelty of the NCL framework generalization ensemble error also depends on the covariance of base learners [59, 63]. Huanhuan Chen and Xin Yao found that NCL optimization is prone to overfitting the noise in the training set, independent of the hyperparameter tuning of the NCL penalty term. This led them to first propose a regularized negative correlation learning (RNCL) algorithm [32]. In addition, they proposed an evolutionary multi-objective approach [13] to simultaneously optimize the three objectives involved (fitness, NCL diversity and regularization).

From the beginning, the NCL framework was conceived as a regression ensemble approach [45, 59]. Later on, the machine learning community adapted this idea first to binary classification problems [28], then to multi-class problems [63], ordinal regression problems [21] and semi-supervised machine learning problems [12]. Mostly, these algorithms require gradient descent as an iterative optimization of the ensemble. Wang et al. [63] proposed the ambiguity term and AdaBoost to overcome this problem, making that proposal a hybrid between NCL and data sampling approaches.

In this paper, we implement ELM as the base learner of the ensemble model, thus reducing the computational burden of the system. The proposed method avoids the excessive iterations required in traditional NCL algorithms, based on gradient descent [13, 46]. Several ELMs are trained separately and assembled by introducing negative correlation penalty on each base learner. Diversity among

ELM base learners is measured and promoted in this research work by analyzing the angle between the outputs of each individual and the outputs associated with the ensemble model. Ensemble outputs are recalculated after each iteration and orthogonal terms in objective functions are updated. The number of inverses to be computed for each base learner has been drastically reduced by implementing a method inspired on the Sherman–Morrison formula. The computational burden of the proposed method is similar to the resolution of S independent ELM optimization problems, being S the number of base learners composing the ensemble.

The manuscript is organized as follows: an explanation of extreme learning machine for classification problems in Sect. 2. Our proposal is discussed in Sect. 3. The detailed description of the implementation is then explained at the end of this Section. The experimental framework is presented in Sect. 4, and the empirical comparisons are in Sect. 5. Conclusions and discussion are in the final segment of the article, Sect. 6.

2 The extreme learning machine model for classification problems

The ELM paradigm implements the traditional regularized least-squares regression (RLSR) model to address both regression and classification problems. The goal of the model (in its linear version) is to estimate a parameter vector, $\hat{\boldsymbol{\beta}}$, which minimizes the following expression:

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^K}\left(\sum_{n=1}^{N}\left(y_n-\sum_{k=1}^{K}\beta_k x_{kn}\right)^2+\gamma\sum_{k=1}^{K}\Psi_k(\beta_k)\right), \tag{1}$$

where $\mathcal{D}=\{(\mathbf{x}_n,y_n)\}_{n=1}^{N}=\{(x_{1n},\ldots,x_{Kn},y_n)\}_{n=1}^{N}$ is the training set, K is the dimension of the input space (number of attributes of the problem), N is the number of patterns in the training set, $\mathbf{x}_n\in\mathbb{R}^K$ is the vector of attributes of the nth pattern, $y_n\in\mathbb{R}$ is the target value of the nth pattern, $\gamma\in\mathbb{R}>0$ is a user-specified parameter, and Ψ_k are mathematical functions that penalize the increasing value of coefficients β_k.

The parameters of the ELM models are estimated from the previously described RLSR problem in its nonlinear form (as the model predictor is based on a single layer feedforward neural network). Thus, the main modifications introduced by the ELM paradigm (for classification problems) in the RLSR model are:

- The parameters to be estimated (the output matrix of coefficients) are the weights connecting the hidden layer of the model to the output layer. Those parameters are embedded in matrices of dimensions $D\times J$ (D is the number of hidden nodes and J is the number of classes), instead of vectors. For that reason, this matrix is defined as: $\boldsymbol{\beta}=(\boldsymbol{\beta}_1,\ldots,\boldsymbol{\beta}_J)\in\mathbb{R}^{D\times J}$, being $\boldsymbol{\beta}_j\in\mathbb{R}^D$ the weights (coefficients) associated with the jth output node. The first dimension of the matrix, D, is associated with the nonlinear transformation of the input space done by ELM models, whereas the second one, J, is related to the type of problem being addressed (classification problems).

  Below, the dimensions of the problem and the reason for having those dimensions are described in more detail.

  - The ELM model is nonlinear in its input variables, since these are transformed by a nonlinear function $\mathbf{h}:\ \mathbb{R}^K\rightarrow\mathbb{R}^D$. In the neural implementation of the model, $\mathbf{h}(\mathbf{x})$ can be explicitly computed as:

    $$\mathbf{h}(\mathbf{x})=(\phi(\mathbf{x};\mathbf{w}_d,b_d),d=1,\ldots,D), \tag{2}$$

    where D is, as previously described, the dimension of the transformed space, $\phi(\cdot;\mathbf{w}_d,b_d):\mathbb{R}^K\rightarrow\mathbb{R}$ is the mapping function of the dth hidden node, $\mathbf{w}_d\in\mathbb{R}^K$ is the input weight vector associated with the dth hidden node and $b_d\in\mathbb{R}$ is the bias of the dth hidden node. The mapping function chosen is typically sigmoidal, i.e.:

    $$\phi(\mathbf{x};\mathbf{w}_d,b_d)=\frac{1}{1+\exp(-(\mathbf{w}_d'\cdot\mathbf{x}+b_d))}. \tag{3}$$

  - Each target, $\mathbf{y}_n\in\mathbb{R}^J$, is the "1-of-$J$" encoding of the class label of the nth pattern ($y_{nj}=1$ if $\mathbf{x}_n$ is a pattern of the jth class, $y_{nj}=0$ otherwise), being J the number of classes.

- The function that penalizes the increasing value of coefficients is quadratic, $\Psi_k(t)=t^2$.
- The user-specified parameter weighs on the quadratic error instead of the regularized term (which it is equivalent to considering a new parameter $C=1/\gamma$).

Hence, the ELM model in classification problems estimates a coefficient matrix, $\hat{\boldsymbol{\beta}}\in\mathbb{R}^{D\times J}$, that minimizes the following equation (expressed in matrix form):

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^{D\times J}}\left(\|\boldsymbol{\beta}\|^2+C\|\mathbf{H}\boldsymbol{\beta}-\mathbf{Y}\|^2\right), \tag{4}$$

where $\mathbf{H}=(\mathbf{h}'(\mathbf{x}_1),\ldots,\mathbf{h}'(\mathbf{x}_N))\in\mathbb{R}^{N\times D}$ is the output of the hidden layer for the training patterns (nonlinear transformation of the input space), $\mathbf{Y}=(\mathbf{Y}_1,\ldots,\mathbf{Y}_J)=\begin{pmatrix}\mathbf{y}_1'\\ \vdots\\ \mathbf{y}_N'\end{pmatrix}\in\mathbb{R}^{N\times J}$ is the matrix with the desired targets, and

$\mathbf{Y}_j$ is the jth column of the $\mathbf{Y}$ matrix. The solution to that optimization problem is:

$$\hat{\boldsymbol{\beta}} = \left(\frac{\mathbf{I}}{C} + \mathbf{H}'\mathbf{H}\right)^{-1}\mathbf{H}'\mathbf{Y}. \tag{5}$$

This solution can be also expressed as:

$$\hat{\boldsymbol{\beta}} = \mathbf{H}'\left(\frac{\mathbf{I}}{C} + \mathbf{H}'\mathbf{H}\right)^{-1}\mathbf{Y}. \tag{6}$$

The advantages of the ELM model with respect to backpropagation (BP) networks are twofold: on the one hand, ELM networks have the capability of providing better generalization results than their BP networks counterparts. On the other hand, ELM models have a much faster learning speed than BP networks. Those advantages are partially obtained thanks to the novel parameter tuning of the ELM paradigm. Specifically, the training phase of the neural version of the ELM model has three stages: first, the input weights and bias of the hidden nodes are randomly determined ($\mathbf{w}_d$ and b_d); second, the hidden layer output matrix ($\mathbf{H}$) is computed as defined in Eq. (2), and third, the output weight matrix, $\hat{\boldsymbol{\beta}}$, is analytically determined using Eq. (5) or Eq. (6).

In the testing phase, each pattern, $\mathbf{x}$, is projected from the input space to the output space using the following equation:

$$\mathbf{f}(\mathbf{x}) = \mathbf{h}'(\mathbf{x})\hat{\boldsymbol{\beta}}, \tag{7}$$

where $\mathbf{f}(\mathbf{x}) \in \mathbb{R}^J$ is the output function of the ELM classifier. In a classification problem, $\mathbf{f}(\mathbf{x})$ is a vector with J elements, $(\mathbf{f}(\mathbf{x}))_j$ will be used to denote the jth element of that vector. The predicted class corresponds to the vector component with highest value. It is important to stress that the predicted class label for each pattern, $\mathbf{x}$, will be stored in a vector $\hat{y}(\mathbf{x}) \in \mathbb{R}^J$ where all their values are equal to 0 except the element in position

$$\arg\max_{j=1,\ldots,J}(\mathbf{f}(\mathbf{x}))_j,$$

that is equal to 1.

3 Ensemble method proposed: negative correlation learning in the ELM paradigm

The aim of this section is to describe how to build an ensemble model inspired on the negative correlation learning (NCL) framework and made of S ELM classifiers. The ensemble method proposed will be named from now on as Negative Correlation Extreme Learning Machine (NCELM). The NCELM method trains during R iterations S ELM classifiers with two conflicting goals: (i) individuals should provide a competitive mean square error (MSE) and (ii) the outputs generated for each individual should be negatively correlated to the outputs provided by the ensemble. In this research study, the algorithm proposed has two well-defined stages:

- *Initialization* The goal of this stage is to create S standard ELM classifiers using the guidelines described in Sect. 2. On the implementation of the ensemble proposed, each component has its own transformation of the input space ($\mathbf{h}^{(s)}$, $s = 1, \ldots, S$), which is randomly generated as proposed for the ELM philosophy. Specifically, the output of the hidden layer of the sth individual in the ensemble is defined as: $\mathbf{H}^{(s)} = \left(\mathbf{h}^{(s)'}(\mathbf{x}_1), \ldots, \mathbf{h}^{(s)'}(\mathbf{x}_N)\right) \in \mathbb{R}^{N\times D}$. It is important to stress that the hidden layer outputs remain fixed during the all the iterations of the algorithm.

  On the other hand, the coefficients related to the connections between the hidden and the output layer of the sth individual in this first iteration of the algorithm are determined as explained in Eq. (5)[1]:

  $$\hat{\boldsymbol{\beta}}^{(s)}_{(1)} = \left(\mathbf{A}^{(s)}_{(1)}\right)^{-1}\mathbf{H}^{(s)'}\mathbf{Y}, \tag{8}$$

  where $\mathbf{A}^{(s)}_{(1)} = \left(\frac{\mathbf{I}}{C} + \mathbf{H}^{(s)'}\mathbf{H}^{(s)}\right)$.

  Once the parameters of each classifier are estimated, the next step is to obtain the initial set of outputs of the ensemble. The outputs of the ensemble are obtained, on each iteration, by simple averaging of the outputs generated by the individuals. In the first iteration, the output of the ensemble for a test pattern $\mathbf{x}$ is defined as:

  $$\mathbf{f}_{(1)}(\mathbf{x}) = \frac{1}{S}\sum_{s=1}^{S}\mathbf{h}^{(s)'}(\mathbf{x})\hat{\boldsymbol{\beta}}^{(s)}_{(1)}. \tag{9}$$

  The outputs of the ensemble for the training pattern in the first iteration are collected in a matrix $\mathbf{F}_{(1)}$, which is defined as: $\mathbf{F}_{(1)} = (\mathbf{F}_{1,(1)}, \ldots, \mathbf{F}_{J,(1)}) = \begin{pmatrix} \mathbf{f}_{(1)}(\mathbf{x}_1)' \\ \vdots \\ \mathbf{f}_{(1)}(\mathbf{x}_N)' \end{pmatrix} \in \mathbb{R}^{N\times J}$, being $\mathbf{F}_{j,(1)}$ the jth column of the $\mathbf{F}_{(1)}$ matrix.
- *Diversity promotion* On this stage, the diversity measure is introduced in the error function of each ELM classifier. As previously described, the hidden layer outputs associated with each individual are not modified after the initialization of the algorithm. Therefore, the

[1] Subscripts are used to denote the number of the iteration (initialization stage corresponds to the first iteration of the algorithm) and superscripts to index the number of classifiers within the ensemble.

only parameters that need to be estimated from the second iteration of the algorithm (where the diversity promotion phase starts) are those associated with the output coefficient matrices, $\{\hat{\boldsymbol{\beta}}_{(r)}^{(1)},\ldots,\hat{\boldsymbol{\beta}}_{(r)}^{(S)}\}, r=2,\ldots,R$. Similarly to what was done on the first iteration of the algorithm, the outputs of the ensemble are collected in its corresponding matrix after the estimation of the output coefficient matrices, $\{\hat{\boldsymbol{\beta}}_{(r)}^{(1)},\ldots,\hat{\boldsymbol{\beta}}_{(r)}^{(S)}\}, r=2,\ldots,R$. The matrix with the outputs of the ensemble in the rth iteration is defined as:

$$\mathbf{F}_{(r)}=(\mathbf{F}_{1,(r)},\ldots,\mathbf{F}_{J,(r)})=\begin{pmatrix}\mathbf{f}_{(r)}(\mathbf{x}_1)'\\ \vdots\\ \mathbf{f}_{(r)}(\mathbf{x}_N)'\end{pmatrix}\in\mathbb{R}^{N\times J},$$

being $\mathbf{F}_{j,(r)}$ the jth column of the $\mathbf{F}_{(r)}$ matrix. The output of the ensemble for each pattern in the rth iteration is computed as:

$$\mathbf{f}_{(r)}(\mathbf{x})=\frac{1}{S}\sum_{s=1}^{S}\mathbf{h}^{(s)'}(\mathbf{x})\hat{\boldsymbol{\beta}}_{(r)}^{(s)}. \tag{10}$$

Below, the way to estimate the output weight matrices of the individuals of the ensemble during the diversity promotion stage is described. Specifically, the diversity measure adopted in the model, the optimization function, the analytical solution of each individual on each iteration and the algorithmic flow of the ensemble are explained in detail.

3.1 Diversity metric proposed

In the NCL framework, diversity among individuals of the ensemble is promoted explicitly in the error function [28, 44]. Therefore, the error functions of the components of the ensemble include both a penalty term to promote diversity among individuals and the mean square error (MSE) of the model with respect to the desired outputs [21]. Concretely, the error function (in regression problems) for the sth individual of the ensemble is defined as:

$$\min_{\boldsymbol{\beta}^{(s)}\in\mathbb{R}^D,\mathbf{w}^{(s)}\in\mathbb{R}^{K\times D},\mathbf{b}^{(s)}\in\mathbb{R}^D}\Big(1/N\sum\nolimits_{n=1}^{N}\Big(f^{(s)}(\mathbf{x}_n;\boldsymbol{\beta}^{(s)},\mathbf{w}^{(s)},\mathbf{b}^{(s)})-y_n\Big)^2 +1/N\sum\nolimits_{n=1}^{N}\lambda p^{(s)}(\mathbf{x}_n;\boldsymbol{\beta}^{(s)},\mathbf{w}^{(s)},\mathbf{b}^{(s)})\Big)$$

where $f^{(s)}:\mathbb{R}^K\rightarrow\mathbb{R}$ is the output of the sth regressor in the nth pattern, $\boldsymbol{\beta}^{(s)},\mathbf{w}^{(s)},\mathbf{b}^{(s)}$ are the parameters to be tuned in a regression problem, $y_n\in\mathbb{R}$ is the desired target in the nth pattern of the training set, $\lambda\in\mathbb{R}$ is a user-specified hyperparameter that controls the importance of diversity with respect to the MSE of the model and $p^{(s)}:\mathbb{R}^K\rightarrow\mathbb{R}$ is the correlation penalty function associated with the sth individual and the nth pattern. The purpose of minimizing $p^{(s)}$ is to negatively correlate each individual's error with errors of the ensemble, and therefore, the function is defined as:

$$p^{(s)}(\mathbf{x}_n;\boldsymbol{\beta}^{(s)},\mathbf{w}^{(s)},\mathbf{b}^{(s)})=\Big(f^{(s)}(\mathbf{x}_n;\boldsymbol{\beta}^{(s)},\mathbf{w}^{(s)},\mathbf{b}^{(s)}) -\mathbf{f}(\mathbf{x}_n))\sum_{j\neq s}\Big(f^{(j)}(\mathbf{x}_n;\boldsymbol{\beta}^{(j)},\mathbf{w}^{(j)},\mathbf{b}^{(j)})-\mathbf{f}(\mathbf{x}_n)\Big),$$

where $\mathbf{f}(\mathbf{x}_n)$ is the output of the final ensemble model for the nth pattern, which is obtained by simply averaging the corresponding outputs of the individuals in the ensemble.

As shown in [7], this penalty term can be understood as a covariance among the individuals of the ensemble, which is also related to the correlation coefficient (as the second statistic is equal to the first one normalized to 1). Furthermore, the correlation coefficient between two variables $\mathbf{u}$ and $\mathbf{v}$ can be interpreted as the cosine of the angle between them:

$$\rho_{\mathbf{u},\mathbf{v}}=\frac{\sum_l u_l v_l}{\sqrt{\sum_l u_l^2}\sqrt{\sum_l v_l^2}}=\frac{\langle\boldsymbol{u},\boldsymbol{v}\rangle}{\|\boldsymbol{u}\|\|\boldsymbol{v}\|}=\cos(\angle(\boldsymbol{u},\boldsymbol{v})), \tag{11}$$

where u_l and v_l are the lth components of the variables $\mathbf{u}$ and $\mathbf{v}$ after the standardization of the sample.

Motivated by this fact, diversity is measured and promoted in this research work by analyzing the angle between the outputs of each individual and the outputs associated with the ensemble model. These vectors will be most different when, $|\angle(\boldsymbol{u},\boldsymbol{v})|=\pi/2$, and therefore, $\langle\boldsymbol{u},\boldsymbol{v}\rangle=0$. When most similar, $\frac{\langle\boldsymbol{u},\boldsymbol{v}\rangle}{\|\boldsymbol{u}\|\|\boldsymbol{v}\|}=\pm1$. Taking this into account, the metric of diversity among the outputs of the sth classifier with respect to the output of the ensemble (for the jth component of the desired targets in the rth iteration) can be defined as[2]:

$$div\left(\left(\mathbf{H}^{(s)}\hat{\boldsymbol{\beta}}_{(r)}^{(s)}\right)_j,\mathbf{F}_{j,(r-1)}\right)=\left\langle\left(\mathbf{H}^{(s)}\hat{\boldsymbol{\beta}}_{(r)}^{(s)}\right)_j,\mathbf{F}_{j,(r-1)}\right\rangle^2, \tag{12}$$

where $\langle\mathbf{s},\mathbf{t}\rangle=\mathbf{s}'\mathbf{t}$ is the standard dot product and $\left(\mathbf{H}^{(s)}\hat{\boldsymbol{\beta}}_{(r)}^{(s)}\right)_j$ is the jth column of the matrix $\left(\mathbf{H}^{(s)}\hat{\boldsymbol{\beta}}_{(r)}^{(s)}\right)$. As can be seen in the equation, diversity of the individuals in the rth iteration with respect to the ensemble is computed taking into account the outputs of the ensemble in the $r-1$ iteration.

[2] The dot product is squared in order to consider solely the direction of the vector.

3.2 Error function formulation

The error function of the NCELM ensemble method is made of three elements: the regularization term, the errors associated with the individual in the ensemble and the diversity among the outputs of the individual and the final ensemble. Thus, the output weight matrices in the rth iteration ($\boldsymbol{\beta}_{(r)}^{(s)}$, $s = \{2, \ldots, S\}$), for each individual, are obtained from the following optimization problem:

$$\min_{\boldsymbol{\beta} \in \mathbb{R}^{D \times J}} \left(\mathbf{g}^{(s)}(\boldsymbol{\beta}) = \|\boldsymbol{\beta}\|^2 + C\|\mathbf{H}^{(s)}\boldsymbol{\beta} - \mathbf{Y}\|^2 + \lambda \sum_{j=1}^{J} \left\langle \left(\mathbf{H}^{(s)}\boldsymbol{\beta}\right)_j, \mathbf{F}_{j,(r-1)} \right\rangle^2 \right), \tag{13}$$

where $\lambda \in \mathbb{R}$ is a problem-dependent parameter that controls the existing diversity among individuals of the ensemble.

As previously explained, the third term defines the diversity between individuals and the final ensemble model. This component of the error function reaches its minimum value, 0, when all its addends are null. This is equivalent to the orthogonality, one by one, of the J outputs generated by the sth ELM base learner with respect to the outputs of the ensemble. The maximum value in this component is obtained when those outputs (associated with the sth ELM and the ensemble) are proportional. This implies that both models provide the same type of classification (same decisions regarding the class label of each pattern in the training set).

The NCELM optimization problem (associated with the sth component of the ensemble) can also be formulated as the sum of J separable vector problems, one for each class. Hence, we could rewrite the optimization function for the r-iteration as:

$$\mathbf{g}^{(s)}(\boldsymbol{\beta}) = \sum_{j=1}^{J} \mathbf{g}_j^{(s)}(\boldsymbol{\beta}) = \sum_{j=1}^{J} \left(\|\boldsymbol{\beta}_j\|^2 + C\|\mathbf{H}^{(s)}\boldsymbol{\beta}_j - \mathbf{Y}_j\|^2 + \lambda \left\langle \mathbf{H}^{(s)}\boldsymbol{\beta}_j, \mathbf{F}_{j,(r-1)} \right\rangle^2 \right). \tag{14}$$

As can be seen in Eq. (14), the decision variables of each addend are different, and therefore, the final solution to the coefficient matrix of the sth individual of the ensemble in the rth iteration, $\hat{\boldsymbol{\beta}}_{(r)}^{(s)}$, could be obtained by grouping the $\hat{\boldsymbol{\beta}}_{j,(r)}^{(s)}$ elements by columns. Taking into account that $\|\boldsymbol{s}\|^2 = \boldsymbol{s}'\boldsymbol{s}$ and grouping terms:

$$\min_{(\boldsymbol{\beta})_j \in \mathbb{R}^D} \left(\mathbf{g}_j^{(s)}(\boldsymbol{\beta}) = \boldsymbol{\beta}_j' \left(\mathbf{I} + C\mathbf{H}^{(s)'}\mathbf{H}^{(s)} + \lambda \mathbf{H}^{(s)'}\mathbf{F}_{j,(r-1)}\mathbf{F}_{j,(r-1)}'\mathbf{H}^{(s)} \right) \boldsymbol{\beta}_j - 2C\boldsymbol{\beta}_j'\mathbf{H}^{(s)'}\mathbf{Y}_j + \mathbf{Y}_j'\mathbf{Y}_j \right)$$

where $\mathbf{I}$ is the identity matrix and $\mathbf{Y}_j'\mathbf{Y}_j$ a constant term. Thus, the optimization problem can also be defined as:

$$\min_{\boldsymbol{\beta}_j \in \mathbb{R}^D} \left(\boldsymbol{\beta}_j' \mathbf{A}_{j,(r)}^{(s)} \boldsymbol{\beta}_j - 2\boldsymbol{\beta}_j' \mathbf{H}^{(s)'} \mathbf{Y}_j \right), \tag{15}$$

where $\mathbf{A}_{j,(r)}^{(s)} = \left(\frac{\mathbf{I}}{C} + \mathbf{H}^{(s)'}\mathbf{H}^{(s)} + \frac{\lambda}{C}\mathbf{H}^{(s)'}\mathbf{F}_{j,(r-1)}\mathbf{F}_{j,(r-1)}'\mathbf{H}^{(s)} \right)$. The solution to that optimization problem (for positive definite $\mathbf{A}_{j,(r)}^{(s)}$ matrices) is

$$\hat{\boldsymbol{\beta}}_{j,(r)}^{(s)} = \left(\mathbf{A}_{j,(r)}^{(s)} \right)^{-1} \mathbf{H}^{(s)'} \mathbf{Y}_j, \tag{16}$$

being $\mathbf{A}_{j,(r)}^{(s)}$ a positive definite matrix and therefore invertible. Furthermore, the $\mathbf{g}_j^{(s)}$ function is strictly convex and the critical point obtained is its unique and global minimum.

3.3 Calculation of the inverses via the Sherman–Morrison formula

The main drawback of the NCELM method is its high computational burden, compared to other ensemble methods. To solve the complete optimization problem, it is necessary to compute S inverses during the initialization stage and $S \times J \times R - 1$ inverses in the Diversity promotion stage, for a total of $S + (S \times J \times R - 1)$ inverses during the whole procedure.

In this section, we will describe a method inspired in the Sherman–Morrison formula [26] to reduce the number of inverses to be computed from $S + (S \times J \times R - 1)$ to S. The goal is to estimate all the inverses required in the diversity promotion stage from those computed in the initialization stage. The formula is built from an invertible square matrix ($\boldsymbol{G}$) and two vectors ($\boldsymbol{m}$ and $\boldsymbol{v}$) with the same rank as $\boldsymbol{G}$. The matrix $\boldsymbol{F} = \boldsymbol{G} + \boldsymbol{m}\boldsymbol{v}'$ is invertible if $1 + \boldsymbol{v}'\boldsymbol{G}^{-1}\boldsymbol{m} \neq 0$. If $\boldsymbol{G} + \boldsymbol{m}\boldsymbol{v}'$ is invertible, then its inverse is given by:

$$\boldsymbol{F}^{-1} = \boldsymbol{G}^{-1} - \frac{\boldsymbol{G}^{-1}\boldsymbol{m}\boldsymbol{v}'\boldsymbol{G}^{-1}}{1 + \boldsymbol{v}'\boldsymbol{G}^{-1}\boldsymbol{m}}. \tag{17}$$

We will first rewrite the inverse matrix to be computed on each iteration of the diversity promotion stage, $\mathbf{A}_{j,(r)}^{(s)}$, as:

$$\mathbf{A}_{j,(r)}^{(s)} = \left(\frac{\mathbf{I}}{C} + \mathbf{H}^{(s)\prime}\mathbf{H}^{(s)} + \frac{\lambda}{C}\mathbf{H}^{(s)'}\mathbf{F}_{j,(r-1)}\mathbf{F}_{j,(r-1)}'\mathbf{H}^{(s)} \right) = \mathbf{A}_{(1)}^{(s)} + \mathbf{u}_{j,(r)}^{(s)}\mathbf{u}_{j,(r)}^{(s)\prime},$$

with

$$\mathbf{A}_{(1)}^{(s)} = \frac{\mathbf{I}}{C} + \mathbf{H}^{(s)\prime}\mathbf{H}^{(s)}, \quad \mathbf{u}_{j,(r)}^{(s)} = \sqrt{\frac{\lambda}{C}}\mathbf{H}^{(s)\prime}\mathbf{F}_{j,(r-1)}, \tag{18}$$

and therefore,

$$\left(\mathbf{A}_{j,(r)}^{(s)}\right)^{-1} = \left(\mathbf{A}_{(1)}^{(s)}\right)^{-1} - \frac{\left(\mathbf{A}_{(1)}^{(s)}\right)^{-1}\mathbf{u}_{j,(r)}^{(s)}\mathbf{u}_{j,(r)}^{(s)\prime}\left(\mathbf{A}_{(1)}^{(s)}\right)^{-1}}{1 + \mathbf{u}_{j,(r)}^{(s)\prime}\left(\mathbf{A}_{(1)}^{(s)}\right)^{-1}\mathbf{u}_{j,(r)}^{(s)}}. \tag{19}$$

The $\mathbf{A}_{(1)}^{(s)}$ matrices and their corresponding inverses were those computed in the initialization stage. Thus, the existing inverses in the diversity promotion stage are estimated from the inverses obtained in the first iteration of the algorithm. For that reason, the computational burden of the proposed method will be similar to the resolution of S independent ELM classification problems, achieving a significant improvement in the efficiency of the ensemble method.

3.4 Algorithmic flow of the NCELM method

The algorithmic steps required to estimate the parameters of the proposed method are here briefly described. NCELM has two stages: the initialization stage (Fig. 1, steps 1–8) and the diversity promotion stage (Fig. 1, steps 9–19). On each iteration of the initialization stage, the algorithm starts randomly generating the hidden layer coefficient matrix of the corresponding classifier (Fig. 1, steps 2–4). After that, the inverse of the matrix required for the computation of the coefficients output matrix of each base learner is stored (Fig. 1, step 5). Then, the coefficients are determined in the traditional ELM framework (Fig. 1, step 6). Once the parameters of the initial S ELM classifiers are estimated, the outputs of the ensemble model, for the first iteration, are obtained (Fig. 1, step 8). During the diversity promotion stage, the output matrix of coefficients associated with the individuals of the ensemble is iteratively updated according to the Sherman–Morrison formula (Fig. 1, steps 12–14). After that, the outputs of the ensemble, for that iteration, are obtained (Fig. 1, step 17).

The output of the final ensemble model is:

$$\mathbf{f}_{(R)}(\mathbf{x}) = 1/S\sum_{s=1}^{S}\left(\mathbf{h}^{(s)'}(\mathbf{x})\hat{\boldsymbol{\beta}}_{(R)}^{(s)}\right), \tag{20}$$

where $\mathbf{f}_{(R)}(\mathbf{x})$ is the numerical output of the ensemble model in the last iteration and $\mathbf{f}_{j,(R)}(\mathbf{x})$ is the jth element of the vector in that iteration. Finally, it is important to clarify that the predicted class label for a test pattern $\mathbf{x}$ is included in a vector $\widehat{y}(\mathbf{x}) \in \mathbb{R}^J$ where all values are equal to 0 except the element in position $\arg\max_{j=1,\ldots,J}\mathbf{f}_{j,(R)}(\mathbf{x})$, that is equal to 1.

4 Experimental framework

The experimental framework implemented to illustrate the competitive performance and efficiency of the proposed ensemble is detailed throughout the following section. First, the selected datasets for benchmarking are described in Sect. 4.1. The measures to evaluate the performance of the algorithms are analyzed in Sect. 4.2. The taxonomy of the algorithms used for comparison purposes in Sect. 4.3. Finally, the statistical tests applied to validate the results are specified in Sect. 4.4.

4.1 Datasets

Sixty six datasets have been selected from the UCI repository [20], presenting diversity in size, number of instances and number of labels (binary and multi-class). The features of each selected dataset are summarized in Table 1, with the ascribed ID, the number of patterns (Size), the number of attributes (#Attr.), the number of classes (#Classes) and the number of instances per class (Class distribution).

Although the UCI repository is a widely used source of datasets for benchmarking machine learning models, the format of these datasets is not consistent. Each dataset has been downloaded and processed into a common format, dropping the missing values by rows or by columns depending on how much information is kept after such process.[3]

Features have been standardized and rescaled following a normal distribution $\mathcal{N}(0,1)$. This transformation of the features is extremely important for distance-based classifiers, such as ELM or support vector machines, normalizing the a priori importance among features. The experimental design was conducted using a tenfold cross-validation procedure, with 3 repetitions per fold. A total of 30 error measures are obtained for all the models compared, which assures a proper statistical significance of the results. The partitions are the same for all compared models.

4.2 Measures

In order to evaluate the efficacy of the methods tested, two performance measures are used: accuracy rate and root mean squared error (RMSE).

- *Accuracy rate (Acc)* the proportion of correct predictions from all predictions made. It has been by far the most commonly used metric to assess the performance

[3] It was carried out using a Python repository that has been developed by the authors with this goal in mind and uploaded to Github (https://github.com/cperales/uci-download-process).

Fig. 1 NCELM training algorithm framework

NCELM(S, D, C, λ):

Require: Training set: $\mathcal{D} = \{(\mathbf{x}_n, \boldsymbol{y}_n)\}_{n=1}^{N}$, where $\mathbf{x}_n \in \mathbb{R}^K$ and $\boldsymbol{y}_n \in \mathbb{R}^J$. $\mathbf{Y} = (\mathbf{Y}_j, j = 1, \ldots, J) = \begin{pmatrix} \boldsymbol{y}_1' \\ \vdots \\ \boldsymbol{y}_N' \end{pmatrix}$

Ensure: Optimized ensemble model: $\{\boldsymbol{\beta}_{(R)}^{(1)}, \ldots, \boldsymbol{\beta}_{(R)}^{(S)}\}$.

{Initialization stage}

for $s = 1$ until S **do**

$\mathbf{w}^{(s)} \leftarrow \text{rand}(D, K)$.

$\mathbf{b}^{(s)} \leftarrow \text{rand}(D, 1)$.

$\mathbf{H}^{(s)} \leftarrow \left(\mathbf{h}^{(s)'}(\mathbf{x}_n),\ n = 1, \ldots, N\right)$

$\left(\mathbf{A}_{(1)}^{(s)}\right)^{-1} \leftarrow \left(\frac{\mathbf{I}}{C} + \mathbf{H}^{(s)'}\mathbf{H}^{(s)}\right)^{-1}$.

$\hat{\boldsymbol{\beta}}_{(1)}^{(s)} \leftarrow \left(\mathbf{A}_{(1)}^{(s)}\right)^{-1} \mathbf{H}^{(s)'}\mathbf{Y}$.

end for

$\mathbf{F}_{(1)} \leftarrow (\mathbf{F}_{1,(1)}, \ldots, \mathbf{F}_{J,(1)}) = \begin{pmatrix} \mathbf{f}_{(1)}(\mathbf{x}_1)' \\ \vdots \\ \mathbf{f}_{(1)}(\mathbf{x}_N)' \end{pmatrix}$. {Output of the ensemble in the first iteration}

{Diversity promotion stage}

for $r = 2$ until R **do**

for $s = 1$ until S **do**

for $j = 1$ until J **do**

$\mathbf{u}_{j,(r)}^{(s)} \leftarrow \sqrt{\frac{\lambda}{C}}\mathbf{H}^{(s)'}\mathbf{F}_{j,(r-1)}$

$\left(\mathbf{A}_{j,(r)}^{(s)}\right)^{-1} \leftarrow \left(\mathbf{A}_{(1)}^{(s)}\right)^{-1} - \frac{\left(\mathbf{A}_{(1)}^{(s)}\right)^{-1}\mathbf{u}_{j,(r)}^{(s)}\mathbf{u}_{j,(r)}^{(s)'}\left(\mathbf{A}_{(1)}^{(s)}\right)^{-1}}{1+\mathbf{u}_{j,(r)}^{(s)'}\left(\mathbf{A}_{(1)}^{(s)}\right)^{-1}\mathbf{u}_{j,(r)}^{(s)}}$

$\hat{\boldsymbol{\beta}}_{j,(r)}^{(s)} \leftarrow \left(\mathbf{A}_{j,(r)}^{(s)}\right)^{-1} \mathbf{H}^{(s)'}\mathbf{Y}_j$

end for

end for

$\mathbf{F}_{(r)} \leftarrow (\mathbf{F}_{1,(r)}, \ldots, \mathbf{F}_{J,(r)}) = \begin{pmatrix} \mathbf{f}_{(r)}(\mathbf{x}_1)' \\ \vdots \\ \mathbf{f}_{(r)}(\mathbf{x}_N)' \end{pmatrix}$. {Output of the ensemble in the r-th iteration}

end for

return $\{\boldsymbol{\beta}_{(R)}^{(1)}, \ldots, \boldsymbol{\beta}_{(R)}^{(S)}\}$.

of classifiers for years [64]. The mathematical expression of *Acc* is:

$$Acc = \frac{1}{N}\sum_{n=1}^{N} I(\hat{y}(\boldsymbol{x}_n) = y_n), \tag{21}$$

where $I(\cdot)$ is the zero-one loss function.

- *Root mean square error (RMSE)* the standard deviation of the differences between predicted values and target values. This metric is optimized in the ELM loss function and is defined as:

$$RMSE = \frac{1}{N}\sum_{n=1}^{N}\sqrt{\frac{1}{J}\sum_{j=1}^{J}\left(f_j(\boldsymbol{x}_n) - y_{nj}\right)^2}. \tag{22}$$

where $f_j(\boldsymbol{x}_n)$ is the numerical output of the model for the jth class.

Computational time is used for measuring the efficiency. If two algorithms over certain database achieve the same results, the faster one will be the more convenient to use. This happens because, after learning the same information, a less complex algorithm would be available to replicate results from a more complex one. The computational time measured is the sum of the cross-validation, training and testing times.

4.3 Algorithms

The proposed method has been evaluated, comparing the results to the ones of ensemble models both from bagging and boosting approaches, and standard ELM[4]. All of them have been already mentioned in the introduction section.

NCELM Negative correlation extreme learning machine, previously detailed in Sect. 3.

[4] A Python library has been developed by the authors with the algorithms used for these experiments and publicly uploaded to Github (https://github.com/cperales/pyridge)

Table 1 Characteristics of the datasets

ID	Dataset	Size	#Attr.	#Classes	Class distribution
1	Adult	32,561	107	2	(24,720, 7841)
2	Magic-gamma-telescope	19,020	10	2	(12,332, 6688)
3	Crowdsourced-mapping	10,845	28	6	(7509, 1494, 1009, 482, 251, 100)
4	Mushroom	8124	106	2	(4208, 3916)
5	Pen-based-recognition	7494	16	10	(780, 779, 780, 719, 780, 720, 720, 778, 719, 719)
6	Robot-navigation	5456	24	4	(2205, 2097, 328, 826)
7	Spambase	4601	57	2	(2788, 1813)
8	Statlog-project-landsat-satellite	4435	36	6	(1072, 479, 961, 415, 470, 1038)
9	Weight-lifting-exercises	4024	68	5	(1365, 901, 112, 276, 1370)
10	Optical-recognition-digits	3823	64	10	(376, 389, 380, 389, 387, 376, 377, 387, 380, 382)
11	Chess-king-rook-vs-king-pawn	3196	38	2	(1527, 1669)
12	Thyroid-disease-sick-euthyroid	3163	17	2	(2870, 293)
13	Thyroid-disease-allhypo	2800	24	4	(154, 2580, 64, 2)
14	Thyroid-disease-allrep	2800	24	4	(2713, 23, 29, 35)
15	Thyroid-disease-allhyper	2800	24	4	(8, 7, 62, 2723)
16	Seismic-bumps	2584	22	2	(2414, 170)
17	Ozone-level-detection-one	1848	72	2	(1791, 57)
18	Ozone-level-detection-eight	1847	72	2	(1719, 128)
19	Car-evaluation	1728	21	4	(384, 69, 1210, 65)
20	Yeast	1484	8	10	(463, 5, 35, 44, 51, 163, 244, 429, 20, 30)
21	cnae-9	1080	856	9	(120, 120, 120, 120, 120, 120, 120, 120, 120)
22	Qsar-biodegradation	1055	41	2	(699, 356)
23	Statlog-project-german-credit	1000	59	2	(700, 300)
24	Connectionist-bench	990	13	11	(90, 90, 90, 90, 90, 90, 90, 90, 90, 90, 90)
25	Tic-tac-toe-endgame	958	27	2	(332, 626)
26	Mammographic-mass	830	5	2	(427, 403)
27	Blood-transfusion-service-center	748	4	2	(570, 178)
28	Breast-cancer-wisconsin	699	8	2	(458, 241)
29	Credit-approval	690	10	2	(307, 383)
30	Hill-valley-noise	606	100	2	307, 299)
31	Breast-cancer-wisconsin-diagnostic	569	30	2	(357, 212)
32	Climate-model-simulation-crashes	540	18	2	(46, 494)
33	Arrhythmia	452	274	13	(245, 44, 15, 15, 13, 25, 3, 2, 9, 50, 4, 5, 22)
34	Libras-movement	360	90	15	(24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24)
35	Ionosphere	351	34	2	(126, 225)
36	Ecoli	336	7	8	(143, 77, 2, 2, 35, 20, 5, 52)
37	Heart-disease-cleveland	303	11	5	(164, 55, 36, 35, 13)
38	Heart-disease-hungarian	294	4	2	(188, 106)
39	Breast-cancer	286	30	2	(218, 68)
40	Cylinder-bands	277	99	2	(99, 178)
41	Soybean-large	266	35	15	(40, 20, 10, 10, 40, 20, 10, 10, 10, 40, 10, 16, 10, 10, 10)
42	Congressional-voting-records	232	16	2	(124, 108)
43	Glass-identification	214	9	6	(70, 76, 17, 13, 9, 29)
44	Image-segmentation	210	19	7	(30, 30, 30, 30, 30, 30, 30)
45	Seeds	210	7	3	(70, 70, 70)
46	Connectionist-bench-sonar	208	60	2	8111, 97)
47	Breast-cancer-wisconsin-prognostic	198	32	2	(151, 47)
48	Parkinsons	195	22	2	(48, 147)

Table 1 (continued)

ID	Dataset	Size	#Attr.	#Classes	Class distribution
49	FLags	194	48	8	(40, 60, 36, 8, 4, 27, 15, 4)
50	Monks-problems-2	169	6	2	(105, 64)
51	Teaching-assistant-evaluation	151	5	3	(49, 50, 52)
52	Iris	150	4	3	(50, 50, 50)
53	Monks-problems-1	124	6	2	(62, 62)
54	Monks-problems-3	122	6	2	(62, 60)
55	Zoo	101	16	7	(41, 20, 5, 13, 4, 8, 10)
56	Fertility	100	9	2	(88, 12)
57	Post-operative-patient	90	17	4	(63, 1, 2, 24)
58	Hepatitis	80	19	2	(13, 67)
59	Spectf-heart	80	44	2	(40, 40)
60	Spect-heart	80	22	2	(40, 40)
61	Soybean-small	47	35	4	(10, 10, 10, 17)
62	Lenses	24	4	3	(4, 5, 15)
63	Balloons-b	20	4	2	(8, 12)
64	Balloons-a	20	4	2	(8, 12)
65	Balloons-c	20	4	2	(8, 12)
66	Balloons-d	16	4	2	(7, 9)

For each dataset, the number of patterns (*Size*), attributes (*#Attr.*), classes (*#Classes*) and the distribution of instances within classes (*Class distribution*) are shown

ELM Standard extreme learning machine, as described in Sect. 2.

BELM Bagging extreme learning machine [57]. In this implementation, each base learner in the ensemble has the same importance (same weight in the final decision) and was created using a random subset which contains 75% of the training set.

BRELM Boosting ridge extreme learning machine [50]. The result of applying each classifier to the training dataset, without renormalizing using 1-of-J encoding, is added to the next classifier. Prediction of each $\boldsymbol{\beta}^{(s)}$ is adjusted to $\mu^{(s)} = \mathbf{Y} - \sum_{l=1}^{s-1} \mathbf{H}\boldsymbol{\beta}^{(l)}$.

AELM AdaBoost extreme learning machine [52]. It is based on the classical idea of multi-class AdaBoost [27], but training instances are weighted and not completely removed from one base learner to the next. Using ELM as base learner implies that from a given $\mathbf{H}$, the different $\boldsymbol{\beta}$ are estimated, making patterns that were wrongly classified on previous iterations more important than the rest. During all iterations, cost-sensitive weights remain normalized avoiding overfitting.

ANCELM AdaBoost negative correlation extreme learning machine [63]. Apart from multi-class AdaBoost implementation from SAMME loss [27], diversity among the outputs of the base learners is introduced explicitly through an ambiguity penalty, making this algorithm a mixed approach from boosting and NCL frameworks.

Multi-class AdaBoost algorithms (AELM and ANCELM) relying on SAMME loss function [27], so they are not able to compute the RMSE metric as their outputs are categorical. Consequently, only accuracy is reported on those methods. Except for ANCELM, the rest of the ensembles have been already tested for ELM base learners [50, 52, 57]. This ensemble was tested by Wang et al. [63] for both binary and multi-class classification problems using neural networks and decision trees as base learners. ANCELM is not only an AdaBoost approach but also a combination of NCL ideas previously discussed in Sect. 1. The ensemble is computed sequentially while encouraging diversity through an ambiguity term, which makes this algorithm more flexible and simpler than other NCL algorithms. It is applicable to both binary and multi-class problems using SAMME modification Hastie et al. [27].

Hyperparameters for each algorithm are selected by a grid search in a fivefold nested cross-validation. This grid is defined in Table 2. Every ELM base learner uses the sigmoid activation function. The ensemble size has been set as $S = 25$ for all the methods, since Brown et al. study

[7] supports ensembles of this size as a competitive trade-off between ensemble diversity and performance. Grid values for the regularization parameter C are chosen from other ELM articles [30]. The selected number of neurons in the hidden layer D follows the criteria from neural ELM ensemble articles [50, 57] and for λ values for ANCELM are assigned according to Wang et al. [63]. Comparing Eqs. (4) and (13), the NCL term with λ is a perturbation from an individual ELM error function, so only small values are considered, $\lambda \in \{10^{-4}, 10^{-3} \ldots, 1\}$.

4.4 Statistical tests

The decision about selecting the best method according to performances is not a trivial task. It is necessary to provide statistical support in order to compare NCELM with the rest of the algorithms presented in Sect. 4.3. The nature of our benchmark datasets does not assure normality [16], so assumptions to apply parametric tests will not be made. It will be necessary to proceed with nonparametric tests for multiple comparisons.

Since six different algorithms are handled, a pre-hoc test is needed in order to determine whether the output results are statistically similar or different as a group. The Friedman test [23] can be used for these comparisons. It detects differences taking into account the global set of algorithms. The procedure involves ranking the result of each algorithm over a dataset, then considering the values of ranks by methods. Once the null hypothesis (all classifiers perform equally well) is rejected by the Friedman's test, it is possible to continue with a post hoc test for finding the pairwise comparisons.

Then, the post hoc test is needed to ensure that NCELM performs better than the algorithms described in Sect. 4.3. The Holm test [29] performs sequentially pairwise comparisons against the control method (NCELM) with a step-down procedure that starts with the most significant p value. If this p value is low according to the significance level, the corresponding hypothesis is rejected. Then, the second significant p value needs to be compared. If the second hypothesis is also rejected, the test proceeds with the third, and so on. As soon as a certain null hypothesis cannot be rejected, all remaining hypotheses are retained as well. For these statistical tests, two significance levels are considered, $\alpha = 0.05$ and $\alpha = 0.10$.

5 Results and discussion

As explained in Sect. 4, comparative analysis among the algorithms previously detailed is carried out following the experimental design. Results for accuracy (*Acc*) are given in Table 3, for *RMSE* in Table 4 and for time in Fig. 4. ID of each dataset referenced in these Tables comes from Table 1 in Sect. 4.1. Both performance measures, results and discussion are in Sect. 5.1, while computational time is reported in Sect. 5.2.

5.1 Performance measures

Table 3 reports the average generalization results for the *Acc* metric for all the datasets considered and the methods used for comparison purposes. It also includes the standard deviation per dataset and method in subscript. As can be seen in this table, NCELM achieves best results in forty of the sixty six datasets, followed by ELM, which is best in fifteen datasets.

Figure 2 compares the performance in *Acc* between the proposed and the comparative ensemble methods in pairs. Each bar is the difference between the *Acc* result of NCELM minus another method. Dataset IDs are specified on the horizontal axis. Positive values indicate that NCELM outperforms the alternative method. Datasets are ordered by how much they are outperformed by NCELM. The figures show that NCELM is an interesting proposal for different types of dataset. Positive differences of *Acc* are not biased toward large or small datasets, since dataset numbers are ordered by size according to Table 1. It can also be deduced that when NCELM outperforms other methods, and it does so with more significance than when it loses.

Table 2 Hyperparameters for each model

Algorithm	References	Hyperparameters
NCELM		$S = 25$, $C \in \{10^{-2}, \ldots, 10^{2}\}$, $D \in \{10, \ldots, 50\}$, $\lambda \in \{10^{-4}, 10^{-3} \ldots, 1\}$
ELM	[30]	$C \in \{10^{-2}, 1, 10, 10^{2}\}$, $D \in \{10, 20, 30, 40, 50\}$
BELM	[57]	$S = 25$, $C \in \{10^{-2}, \ldots, 10^{2}\}$, $D \in \{10, \ldots, 50\}$
BRELM	[50]	$S = 25$, $C \in \{10^{-2}, \ldots, 10^{2}\}$, $D \in \{10, \ldots, 50\}$
AELM	[52]	$S = 25$, $C \in \{10^{-2}, \ldots, 10^{2}\}$, $D \in \{10, \ldots, 50\}$
ANCELM	[63]	$S = 25$, $C \in \{10^{-2}, \ldots, 10^{2}\}$, $D \in \{10, \ldots, 50\}$, $\lambda \in \{0.25, 0.5, 1, 5, 10\}$

Table 3 Accuracy generalized results

ID	NCELM	ELM	BELM	BRELM	AELM	ANCELM
1	**0.82222**$_{0.00482}$	0.82098$_{0.00561}$	*0.82101*$_{0.00807}$	0.82037$_{0.00555}$	0.75868$_{0.01725}$	0.75099$_{0.01387}$
2	0.78009$_{0.00838}$	*0.82529*$_{0.00976}$	**0.82589**$_{0.009388}$	0.82198$_{0.00946}$	0.75468$_{0.01253}$	0.73304$_{0.02623}$
3	0.79755$_{0.02732}$	**0.82438**$_{0.03300}$	*0.82432*$_{0.03645}$	0.82112$_{0.03304}$	0.78392$_{0.04000}$	0.76772$_{0.03693}$
1	**0.94332**$_{0.09188}$	0.91891$_{0.08431}$	0.91412$_{0.09441}$	*0.93862*$_{0.05902}$	0.85478$_{0.10042}$	0.84344$_{0.10354}$
2	0.94583$_{0.00797}$	0.93435$_{0.00879}$	0.93266$_{0.00903}$	0.93453$_{0.01045}$	**0.95067**$_{0.01052}$	*0.94783*$_{0.01478}$
3	0.65897$_{0.02478}$	0.66718$_{0.03124}$	*0.67058*$_{0.02790}$	**0.67131**$_{0.02395}$	0.64183$_{0.03667}$	0.63810$_{0.03098}$
4	**0.91304**$_{0.02036}$	0.88508$_{0.03185}$	0.88428$_{0.03158}$	*0.88646*$_{0.02592}$	0.79218$_{0.05341}$	0.79906$_{0.06159}$
5	**0.84084**$_{0.03971}$	0.82706$_{0.05009}$	0.83879$_{0.04277}$	*0.84021*$_{0.04458}$	0.75429$_{0.08071}$	0.75781$_{0.06959}$
6	**0.87240**$_{0.07520}$	0.81128$_{0.10877}$	0.83615$_{0.07259}$	0.83595$_{0.09319}$	0.82252$_{0.08361}$	*0.84030*$_{0.07309}$
7	**0.93549**$_{0.01245}$	0.89097$_{0.01750}$	0.88978$_{0.01789}$	0.88665$_{0.01768}$	0.89970$_{0.02195}$	*0.90154*$_{0.01877}$
8	**0.88235**$_{0.06417}$	0.81745$_{0.08440}$	0.81505$_{0.07225}$	0.81215$_{0.07683}$	*0.82438*$_{0.08247}$	0.79923$_{0.08095}$
9	**0.90737**$_{0.00131}$	**0.90737**$_{0.00131}$	**0.90737**$_{0.00131}$	**0.90737**$_{0.00131}$	**0.90737**$_{0.00131}$	**0.90737**$_{0.00131}$
10	*0.92133*$_{0.00438}$	**0.92145**$_{0.00415}$	**0.92145**$_{0.00415}$	**0.92145**$_{0.00415}$	**0.92145**$_{0.00415}$	**0.92145**$_{0.00415}$
11	0.96835$_{0.00521}$	**0.96895**$_{0.00343}$	**0.96895**$_{0.00343}$	**0.96895**$_{0.00343}$	**0.96895**$_{0.00343}$	*0.96871*$_{0.00349}$
12	**0.97252**$_{0.00347}$	**0.97252**$_{0.00347}$	**0.97252**$_{0.00347}$	**0.97252**$_{0.00347}$	0.96967$_{0.00925}$	*0.97216*$_{0.00415}$
13	*0.93370*$_{0.00274}$	0.93202$_{0.00759}$	0.93318$_{0.00345}$	0.93279$_{0.00547}$	**0.93421**$_{0.00012}$	**0.93421**$_{0.00012}$
14	**0.96916**$_{0.00240}$	**0.96916**$_{0.00240}$	**0.96916**$_{0.00240}$	**0.96916**$_{0.00240}$	**0.96916**$_{0.00240}$	**0.96916**$_{0.00240}$
15	**0.93070**$_{0.00195}$	**0.93070**$_{0.00195}$	**0.93070**$_{0.00195}$	**0.93070**$_{0.00195}$	**0.93070**$_{0.00195}$	**0.93070**$_{0.00195}$
16	0.76308$_{0.05765}$	0.75912$_{0.06501}$	0.75722$_{0.06418}$	0.76628$_{0.06466}$	*0.78359*$_{0.07564}$	**0.79397**$_{0.07024}$
17	**0.58888**$_{0.02689}$	*0.58305*$_{0.03219}$	0.57790$_{0.03539}$	0.57218$_{0.03452}$	0.51638$_{0.03920}$	0.51331$_{0.03380}$
18	**0.91481**$_{0.03736}$	0.55922$_{0.04416}$	*0.57301*$_{0.03988}$	0.55648$_{0.05004}$	0.48292$_{0.05199}$	0.47654$_{0.04048}$
19	**0.85511**$_{0.06266}$	0.84691$_{0.05813}$	*0.84914*$_{0.05675}$	0.84244$_{0.06650}$	0.79018$_{0.05654}$	0.79519$_{0.06564}$
20	**0.74767**$_{0.04609}$	0.72233$_{0.04835}$	0.71833$_{0.03541}$	0.72033$_{0.05148}$	0.73267$_{0.04049}$	*0.73400*$_{0.04484}$
21	**0.57980**$_{0.07905}$	0.44040$_{0.07160}$	0.41111$_{0.06970}$	0.42929$_{0.07141}$	0.45825$_{0.06590}$	*0.49798*$_{0.08705}$
22	**0.81392**$_{0.05585}$	*0.79128*$_{0.07569}$	0.77876$_{0.06055}$	0.79039$_{0.07282}$	0.76689$_{0.08336}$	0.77981$_{0.05749}$
23	**0.81967**$_{0.04980}$	0.81352$_{0.04719}$	*0.81597*$_{0.04683}$	0.80709$_{0.05542}$	0.74458$_{0.09915}$	0.76729$_{0.08133}$
24	0.76028$_{0.02617}$	0.75450$_{0.10637}$	**0.76250**$_{0.10234}$	0.75184$_{0.10285}$	*0.76205*$_{0.00411}$	*0.76205*$_{0.00411}$
25	*0.95862*$_{0.02675}$	**0.96004**$_{0.02521}$	0.95625$_{0.02858}$	0.95814$_{0.02534}$	0.90572$_{0.11721}$	0.90289$_{0.10450}$
26	**0.85109**$_{0.16735}$	0.84489$_{0.17257}$	0.84333$_{0.16533}$	*0.85067*$_{0.16933}$	0.76192$_{0.12412}$	0.75905$_{0.13577}$
27	0.52758$_{0.05442}$	0.61062$_{0.04524}$	**0.63502**$_{0.05824}$	*0.63258*$_{0.05682}$	0.52476$_{0.03953}$	0.54392$_{0.05324}$
28	**0.97550**$_{0.02193}$	*0.96431*$_{0.02660}$	0.96274$_{0.02784}$	0.95849$_{0.02227}$	0.86264$_{0.07521}$	0.86125$_{0.07502}$
29	0.91555$_{0.00797}$	*0.91861*$_{0.00940}$	0.91611$_{0.00860}$	**0.92113**$_{0.01432}$	0.91493$_{0.00786}$	0.91118$_{0.01264}$
30	**0.65227**$_{0.05971}$	0.59097$_{0.06174}$	0.59386$_{0.06260}$	*0.59526*$_{0.05403}$	0.51307$_{0.07208}$	0.51835$_{0.06688}$
31	**0.79407**$_{0.14197}$	0.63778$_{0.11938}$	0.66778$_{0.15002}$	0.68815$_{0.13253}$	*0.71963*$_{0.14826}$	0.71111$_{0.11877}$
32	**0.87005**$_{0.05551}$	*0.86932*$_{0.06922}$	0.84993$_{0.05599}$	0.85856$_{0.06372}$	0.77913$_{0.07638}$	0.80761$_{0.07016}$
33	**0.86980**$_{0.03206}$	0.86633$_{0.04570}$	0.86549$_{0.04117}$	*0.86793*$_{0.03303}$	0.77472$_{0.05757}$	0.77613$_{0.05598}$
34	**0.56422**$_{0.03974}$	*0.56364*$_{0.03389}$	0.56250$_{0.05053}$	0.55816$_{0.05616}$	0.55460$_{0.06380}$	0.54530$_{0.06247}$
35	0.79273$_{0.06200}$	**0.79666**$_{0.07307}$	*0.79444*$_{0.06833}$	0.78405$_{0.07886}$	0.72670$_{0.10745}$	0.71914$_{0.12535}$
36	0.73389$_{0.09669}$	0.73159$_{0.06720}$	0.73282$_{0.08753}$	0.72478$_{0.09597}$	**0.76360**$_{0.00939}$	0.72469$_{0.09228}$
37	0.62916$_{0.07248}$	*0.63697*$_{0.10027}$	**0.64478**$_{0.07227}$	0.62630$_{0.11132}$	0.63041$_{0.07946}$	0.60979$_{0.11015}$
38	**0.89981**$_{0.04876}$	*0.86377*$_{0.05943}$	0.85361$_{0.06504}$	0.85983$_{0.06044}$	0.82711$_{0.07563}$	0.81429$_{0.05628}$
39	**0.96795**$_{0.03370}$	0.95936$_{0.04218}$	0.95677$_{0.04110}$	*0.96656*$_{0.03683}$	0.84857$_{0.08942}$	0.84431$_{0.10536}$
40	**0.66333**$_{0.10045}$	*0.65236*$_{0.11055}$	0.64836$_{0.08019}$	0.62035$_{0.09212}$	0.58434$_{0.12809}$	0.63278$_{0.11236}$
41	**0.88730**$_{0.05830}$	*0.87302*$_{0.05119}$	0.86667$_{0.06890}$	0.86508$_{0.05510}$	0.84286$_{0.07693}$	0.83175$_{0.07034}$
42	0.93651$_{0.05406}$	*0.94603*$_{0.05732}$	0.94127$_{0.06109}$	**0.94762**$_{0.05813}$	0.93016$_{0.06704}$	0.94127$_{0.05314}$
43	**0.69366**$_{0.18093}$	*0.67852*$_{0.13544}$	0.66125$_{0.18618}$	0.67089$_{0.16390}$	0.64398$_{0.18190}$	0.64688$_{0.16414}$
44	**0.82093**$_{0.05721}$	*0.78503*$_{0.07635}$	0.78499$_{0.05326}$	0.78129$_{0.05497}$	0.75794$_{0.02656}$	0.75970$_{0.02685}$
45	**0.82276**$_{0.13341}$	0.80128$_{0.11837}$	*0.82146*$_{0.08873}$	0.78693$_{0.10410}$	0.79404$_{0.10728}$	0.79517$_{0.12504}$

Table 3 (continued)

ID	NCELM	ELM	BELM	BRELM	AELM	ANCELM
46	$\mathbf{0.50429}_{0.10415}$	$0.45789_{0.13582}$	$\mathit{0.47028}_{0.10917}$	$0.46568_{0.11506}$	$0.42958_{0.12279}$	$0.39437_{0.10111}$
47	$0.58442_{0.11019}$	$\mathbf{0.63799}_{0.12321}$	$0.56431_{0.13823}$	$\mathit{0.61526}_{0.13944}$	$0.59804_{0.06248}$	$0.60512_{0.04608}$
48	$\mathit{0.52502}_{0.13150}$	$0.51881_{0.13907}$	$\mathbf{0.55375}_{0.13574}$	$0.51863_{0.12846}$	$0.52046_{0.14048}$	$0.51214_{0.12873}$
49	$\mathbf{0.97556}_{0.04383}$	$0.96222_{0.06367}$	$0.96222_{0.05072}$	$\mathit{0.96889}_{0.04787}$	$0.94889_{0.04771}$	$0.94222_{0.05370}$
50	$\mathbf{0.70893}_{0.12390}$	$0.67024_{0.15432}$	$\mathit{0.70337}_{0.14711}$	$0.65873_{0.13641}$	$0.66190_{0.13520}$	$0.61845_{0.12136}$
51	$\mathit{0.79721}_{0.12227}$	$\mathbf{0.79798}_{0.12883}$	$0.79101_{0.10910}$	$0.78553_{0.11020}$	$0.69837_{0.12392}$	$0.65514_{0.14081}$
52	$\mathit{0.95832}_{0.04825}$	$\mathbf{0.95853}_{0.04807}$	$0.94094_{0.05818}$	$0.95276_{0.05253}$	$0.94156_{0.05330}$	$0.94517_{0.05914}$
53	$0.87475_{0.03732}$	$0.85323_{0.08761}$	$\mathit{0.87808}_{0.03488}$	$0.87444_{0.04644}$	$\mathbf{0.88141}_{0.03191}$	$\mathbf{0.88141}_{0.03191}$
54	$0.63669_{0.11880}$	$\mathit{0.69243}_{0.06550}$	$\mathbf{0.70864}_{0.06591}$	$0.67530_{0.06813}$	$0.58352_{0.15347}$	$0.58378_{0.16658}$
55	$0.82612_{0.11535}$	$0.80397_{0.14498}$	$0.79292_{0.17873}$	$0.78743_{0.15609}$	$\mathit{0.84372}_{0.08354}$	$\mathbf{0.84993}_{0.06819}$
56	$\mathbf{0.76250}_{0.10383}$	$0.74167_{0.12472}$	$\mathit{0.74583}_{0.12700}$	$0.70000_{0.13919}$	$0.65000_{0.13844}$	$0.59583_{0.16035}$
57	$\mathbf{0.75000}_{0.18257}$	$0.70833_{0.13044}$	$\mathit{0.72083}_{0.14678}$	$0.67083_{0.13495}$	$\mathbf{0.75000}_{0.15811}$	$0.70000_{0.18708}$
58	$0.98333_{0.06236}$	$\mathit{0.98500}_{0.05649}$	$0.96167_{0.08630}$	$\mathbf{1.00000}_{0.00000}$	$0.98000_{0.06000}$	$0.91667_{0.14568}$
59	$0.78889_{0.34543}$	$0.78889_{0.34543}$	$0.78333_{0.36742}$	$\mathit{0.79722}_{0.34740}$	$\mathbf{0.80556}_{0.31131}$	$0.72500_{0.36524}$
60	$\mathbf{1.00000}_{0.00000}$	$\mathbf{1.00000}_{0.00000}$	$\mathbf{1.00000}_{0.00000}$	$\mathbf{1.00000}_{0.00000}$	$\mathit{0.94444}_{0.21228}$	$0.85556_{0.24242}$
61	$\mathbf{1.00000}_{0.00000}$	$\mathbf{1.00000}_{0.00000}$	$\mathit{0.98889}_{0.05984}$	$0.92222_{0.21830}$	$0.96667_{0.12472}$	$0.96667_{0.12472}$
62	$\mathbf{1.00000}_{0.00000}$	$\mathbf{1.00000}_{0.00000}$	$0.96111_{0.11928}$	$\mathit{0.97778}_{0.08315}$	$0.94444_{0.16851}$	$0.94444_{0.17392}$
63	$\mathit{0.99167}_{0.04488}$	$\mathit{0.99167}_{0.04488}$	$0.90833_{0.14649}$	$\mathbf{1.00000}_{0.00000}$	$0.92500_{0.16576}$	$0.96944_{0.12635}$

From all the tests, the average value is presented together with the standard deviation of the results. The best average result for each dataset is highlighted in bold face and the second one in italics

Similarly, Table 4 reports the average values provided by the comparison methods in the datasets considered for the *RMSE* metric. Columns represent methods, while rows represent datasets, with standard deviation as a subscript. As explained in Sect. 4.3, AdaBoost methods (AELM and ANCELM) are not able to compute *RMSE* values, and therefore, these algorithms do not appear in Table 4. The table shows how the NCELM method achieved the best results in thirty eight of the sixty six datasets (providing competitive results in both large and small datasets), followed by BRELM outperforming others in ten datasets. Both BELM and ELM obtained the best results in nine datasets.

Figure 3 compares the *RMSE* performance between NCELM and the comparative ensemble methods in pairs by plotting the difference in average *RMSE* for a dataset. Since a low value of *RMSE* is searched and difference is calculated as in Fig. 3, the lower the bars, the better the comparative performance of our method. As with *Acc*, datasets are ordered on each figure depending on this outperformance of NCELM, not showing a dependence on dataset number.

In order to determine the statistical significance of NCELM, nonparametric Friedman tests [23] are carried out with the rankings of *Acc* and *RMSE*. Statistical significance of the *Acc* rank differences with $\alpha = 0.05$, with a confidence interval of $C_0 = (0, F_{0.05}) = (0, 2.24177)$ and the F-distribution statistical values being $F^* = 19.25809 \notin C_0$. For *RMSE*, $C_0 = (0, F_{0.05}) = (0, 2.65091)$ and $F^* = 3.00365 \notin C_0$. Based on the rejection of the null hypothesis, the Holm post hoc test [29] is used to compare all classifiers to NCELM both in *Acc* and *RMSE*. Table 5 summarizes the ranks and the output of Holm post hoc test for *Acc* and *RMSE*, respectively. From a purely descriptive point of view, NCELM achieves the best ranking in both performance measures ($R_{Acc_{NCELM}} = 2.21212$ and $R_{RMSE_{NCELM}} = 2.10606$), followed by ELM in *Acc* ($R_{Acc_{ELM}} = 2.96212$) and in *RMSE* ($R_{RMSE_{ELM}} = 2.57576$). Considering the results in Table 5, it can be concluded that the proposed method is significantly better in both *Acc* and *RMSE* than the compared methods.

5.2 Execution time

The computational time required to perform the nested cross-validation, training and testing for the experimental design adopted for the classification problems (ordered from larger to smaller datasets) and the ensemble methods considered is shown in Fig. 4. The NCELM method is computationally more efficient than the baseline NCL-inspired method (ANCELM).

Figure 4 shows that there is an offset for small datasets due to random matrix construction. NCELM does not reduce time for small datasets, since it requires the initial

Table 4 *RMSE* tenfold results

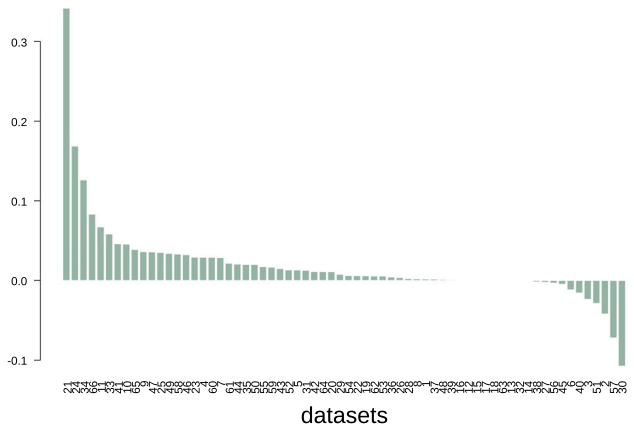
ID	NCELM	ELM	BELM	BRELM
1	$\mathbf{0.27508}_{0.00194}$	$0.28367_{0.00347}$	$\mathit{0.28366}_{0.00612}$	$0.28611_{0.00347}$
2	$0.34787_{0.00638}$	$\mathit{0.28346}_{0.00568}$	$\mathbf{0.28315}_{0.00462}$	$0.28508_{0.00550}$
3	$0.46884_{0.02447}$	$\mathbf{0.44584}_{0.03350}$	$\mathit{0.44594}_{0.03445}$	$0.44906_{0.03356}$
1	$\mathbf{0.16863}_{0.07216}$	$0.20331_{0.06780}$	$0.20831_{0.07387}$	$\mathit{0.19145}_{0.04937}$
2	$\mathbf{0.42164}_{0.00681}$	$0.46910_{0.00988}$	$0.46890_{0.01184}$	$\mathit{0.46626}_{0.01198}$
3	$0.63182_{0.01504}$	$0.62498_{0.02035}$	$\mathit{0.62436}_{0.01633}$	$\mathbf{0.62129}_{0.02087}$
4	$\mathbf{0.22692}_{0.02916}$	$\mathit{0.24741}_{0.02961}$	$0.24764_{0.02838}$	$0.24861_{0.02299}$
5	$\mathbf{0.40482}_{0.06160}$	$0.43361_{0.06213}$	$0.42952_{0.06273}$	$\mathit{0.42530}_{0.06132}$
6	$\mathbf{0.41758}_{0.07543}$	$0.51477_{0.08580}$	$\mathit{0.49724}_{0.06835}$	$0.50714_{0.07206}$
7	$\mathbf{0.53417}_{0.01000}$	$\mathit{0.60023}_{0.01082}$	$0.60277_{0.01156}$	$0.60119_{0.01052}$
8	$\mathbf{0.32053}_{0.04578}$	$\mathit{0.32589}_{0.04623}$	$0.32803_{0.04019}$	$0.32650_{0.04345}$
9	$0.18503_{0.00608}$	$\mathit{0.16235}_{0.00288}$	$\mathbf{0.16226}_{0.00278}$	$0.16512_{0.00428}$
10	$0.22759_{0.01944}$	$\mathit{0.21245}_{0.01733}$	$0.21794_{0.02343}$	$\mathbf{0.20872}_{0.01924}$
11	$\mathbf{0.12765}_{0.01531}$	$0.14784_{0.02311}$	$0.14437_{0.01502}$	$\mathit{0.13734}_{0.02480}$
12	$\mathbf{0.12110}_{0.00674}$	$0.14565_{0.02248}$	$0.14984_{0.01944}$	$\mathit{0.12736}_{0.01919}$
13	$0.14026_{0.04144}$	$0.13630_{0.04744}$	$\mathbf{0.13356}_{0.04396}$	$\mathit{0.13518}_{0.04670}$
14	$\mathbf{0.05892}_{0.00870}$	$0.06310_{0.00831}$	$\mathit{0.06232}_{0.00785}$	$0.06766_{0.00833}$
15	$\mathbf{0.11904}_{0.01439}$	$0.12970_{0.01853}$	$\mathit{0.12880}_{0.01819}$	$0.13578_{0.01773}$
16	$\mathbf{0.50208}_{0.03677}$	$0.52106_{0.05220}$	$0.51903_{0.04665}$	$\mathit{0.51859}_{0.04046}$
17	$0.72987_{0.01664}$	$\mathbf{0.72273}_{0.01890}$	$\mathit{0.72382}_{0.02059}$	$0.72878_{0.02074}$
18	$\mathbf{0.71781}_{0.01248}$	$0.80500_{0.01680}$	$0.80606_{0.02032}$	$\mathit{0.80355}_{0.01810}$
19	$\mathbf{0.26635}_{0.03965}$	$\mathit{0.28017}_{0.03796}$	$0.28093_{0.03549}$	$0.28058_{0.04158}$
20	$\mathbf{0.34727}_{0.01790}$	$\mathit{0.35876}_{0.02200}$	$0.35899_{0.01861}$	$0.36178_{0.02185}$
21	$\mathbf{0.80199}_{0.01313}$	$\mathit{0.90045}_{0.03615}$	$0.91273_{0.03566}$	$0.91489_{0.03757}$
22	$\mathit{0.32957}_{0.04274}$	$0.33138_{0.04647}$	$0.33001_{0.03872}$	$\mathbf{0.31715}_{0.04014}$
23	$0.27879_{0.03467}$	$0.27044_{0.03432}$	$\mathbf{0.26413}_{0.03451}$	$\mathit{0.27029}_{0.03636}$
24	$0.32894_{0.06701}$	$\mathit{0.32385}_{0.06339}$	$\mathbf{0.32221}_{0.06353}$	$0.32651_{0.06352}$
25	$0.11772_{0.03840}$	$\mathbf{0.09919}_{0.02562}$	$0.10933_{0.03429}$	$\mathit{0.10725}_{0.02564}$
26	$0.24618_{0.10108}$	$\mathit{0.24593}_{0.11155}$	$0.24854_{0.11104}$	$\mathbf{0.22727}_{0.11338}$
27	$0.51383_{0.05412}$	$0.47073_{0.01804}$	$\mathbf{0.45608}_{0.02631}$	$\mathit{0.46279}_{0.02216}$
28	$\mathbf{0.14205}_{0.01580}$	$0.15458_{0.01905}$	$0.15243_{0.01967}$	$\mathit{0.14850}_{0.01796}$
29	$0.20968_{0.01464}$	$\mathbf{0.17810}_{0.01503}$	$\mathit{0.17878}_{0.01167}$	$0.17955_{0.01457}$
30	$\mathbf{0.65201}_{0.03163}$	$0.71822_{0.03744}$	$\mathit{0.71547}_{0.03936}$	$0.72092_{0.03756}$
31	$\mathbf{0.69751}_{0.05853}$	$0.82200_{0.06559}$	$0.81343_{0.07571}$	$\mathit{0.80453}_{0.07147}$
32	$\mathbf{0.23731}_{0.04064}$	$0.25088_{0.04366}$	$\mathit{0.24844}_{0.04082}$	$0.25038_{0.04763}$
33	$0.44421_{0.03942}$	$\mathbf{0.42299}_{0.04185}$	$\mathit{0.42533}_{0.03134}$	$0.43145_{0.04408}$
34	$\mathbf{0.64935}_{0.03892}$	$0.65610_{0.04420}$	$\mathit{0.65338}_{0.04443}$	$0.66241_{0.04418}$
35	$0.31817_{0.03669}$	$0.30291_{0.03063}$	$\mathit{0.29714}_{0.03784}$	$\mathbf{0.27764}_{0.03080}$
36	$\mathbf{0.32291}_{0.06391}$	$0.34088_{0.05571}$	$\mathit{0.33965}_{0.05552}$	$0.34099_{0.06392}$
37	$\mathbf{0.41343}_{0.04197}$	$\mathit{0.41353}_{0.05348}$	$0.42335_{0.04185}$	$0.44060_{0.06730}$
38	$\mathbf{0.46978}_{0.04955}$	$\mathit{0.57295}_{0.04735}$	$0.58499_{0.05015}$	$0.58779_{0.04513}$
39	$\mathbf{0.10900}_{0.03955}$	$0.12737_{0.03825}$	$0.12520_{0.03841}$	$\mathit{0.12363}_{0.03410}$
40	$\mathbf{0.64807}_{0.06373}$	$\mathit{0.68065}_{0.08625}$	$0.68723_{0.07967}$	$0.71831_{0.13050}$
41	$\mathbf{0.43365}_{0.04176}$	$\mathit{0.47212}_{0.03891}$	$0.49127_{0.05368}$	$0.48404_{0.05501}$
42	$0.30704_{0.09642}$	$\mathit{0.28036}_{0.07670}$	$\mathbf{0.27391}_{0.08978}$	$0.28339_{0.08468}$
43	$\mathbf{0.38394}_{0.08566}$	$0.41636_{0.07562}$	$0.39783_{0.09818}$	$\mathit{0.39603}_{0.10339}$
44	$\mathbf{0.32281}_{0.02174}$	$0.33301_{0.03788}$	$\mathit{0.33091}_{0.03044}$	$0.33617_{0.03042}$
45	$\mathbf{0.28342}_{0.07258}$	$0.32133_{0.08458}$	$\mathit{0.31277}_{0.06916}$	$0.33433_{0.07731}$
46	$\mathbf{0.77951}_{0.03885}$	$0.82029_{0.07266}$	$\mathit{0.80771}_{0.05722}$	$0.83822_{0.07337}$
47	$0.44845_{0.03713}$	$\mathbf{0.42682}_{0.07346}$	$0.46158_{0.05711}$	$\mathit{0.44669}_{0.08615}$
48	$\mathbf{0.73869}_{0.06084}$	$\mathit{0.74013}_{0.07696}$	$0.75030_{0.07184}$	$0.76670_{0.11323}$
49	$0.24668_{0.03841}$	$0.23466_{0.04915}$	$\mathit{0.22877}_{0.04944}$	$\mathbf{0.22431}_{0.05278}$
50	$\mathbf{0.36622}_{0.06962}$	$0.40276_{0.09297}$	$\mathit{0.38850}_{0.09387}$	$0.41776_{0.10369}$
51	$\mathbf{0.30752}_{0.06948}$	$0.31544_{0.08166}$	$\mathit{0.30808}_{0.06254}$	$0.32190_{0.06578}$
52	$\mathbf{0.21922}_{0.06972}$	$0.27919_{0.04691}$	$\mathit{0.26785}_{0.05133}$	$0.28846_{0.05354}$
53	$\mathit{0.21486}_{0.04621}$	$0.22895_{0.08370}$	$\mathbf{0.21198}_{0.05531}$	$0.23373_{0.06336}$
54	$0.71800_{0.08524}$	$\mathbf{0.60846}_{0.05331}$	$\mathit{0.61348}_{0.04709}$	$0.61462_{0.05618}$
55	$\mathbf{0.23786}_{0.07820}$	$0.27067_{0.09808}$	$\mathit{0.26412}_{0.09056}$	$0.28442_{0.10751}$
56	$\mathit{0.36364}_{0.04643}$	$\mathbf{0.35162}_{0.05458}$	$0.37098_{0.06666}$	$0.38449_{0.09224}$
57	$\mathit{0.38099}_{0.05283}$	$\mathbf{0.38045}_{0.06207}$	$0.38766_{0.06981}$	$0.39245_{0.09859}$
58	$0.42170_{0.08404}$	$\mathit{0.34798}_{0.09185}$	$0.36754_{0.10179}$	$\mathbf{0.22877}_{0.10450}$
59	$\mathbf{0.48202}_{0.28442}$	$0.53245_{0.33152}$	$\mathit{0.49155}_{0.25384}$	$0.56918_{0.34581}$
60	$0.29719_{0.09674}$	$0.29524_{0.09978}$	$\mathbf{0.24919}_{0.09557}$	$\mathit{0.26165}_{0.06701}$
61	$0.40428_{0.04849}$	$\mathit{0.32292}_{0.07965}$	$0.32735_{0.07938}$	$\mathbf{0.28212}_{0.07528}$
62	$0.29062_{0.13384}$	$\mathit{0.27460}_{0.11330}$	$0.28578_{0.10068}$	$\mathbf{0.26618}_{0.10223}$
63	$0.14932_{0.07715}$	$\mathit{0.13059}_{0.06630}$	$0.22901_{0.11126}$	$\mathbf{0.07989}_{0.07213}$

From all the tests, the average value is presented together with the variance of the results. The best average result for each dataset is highlighted in bold face and the second best in italics

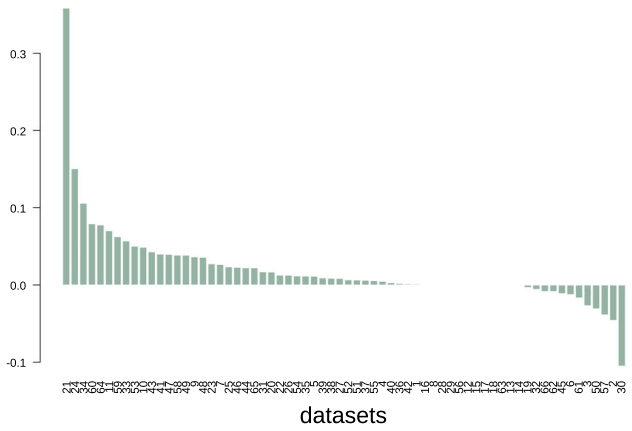
generation of 25 random matrices, while other methods simply generate a singular **H** that all base learners share. For larger datasets, while other comparative methods such as AELM increase highly with the size of the datasets, NCELM's computational time increase is slower. Thus, NCELM seems to be an appealing model for medium and large datasets. The Sherman–Morrison theorem avoids the calculation of several matrix inverses for NCELM, which is definitely an improvement in computational terms.

5.3 Sensitivity analysis

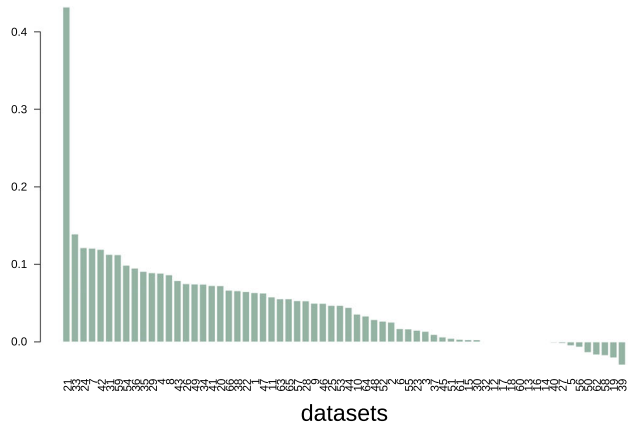
The performance of the proposed ensemble method depends on the configuration of two user-specified hyperparameters: C and λ. The way in which the performance of the ensemble method changes with respect to different values of the hyperparameters has been analyzed on two datasets, breast cancer (binary) and seeds (multi-class). In this analysis, the values of the hyperparameters are represented in the X and Y axes, and the accuracy of the ensemble in the Z-axis. The number of hidden nodes in the hyperparameters study was set to 50, (as it was the maximum value considered in the nested cross-validation). The ensemble size was of 25 ($S = 25$) for the two classification



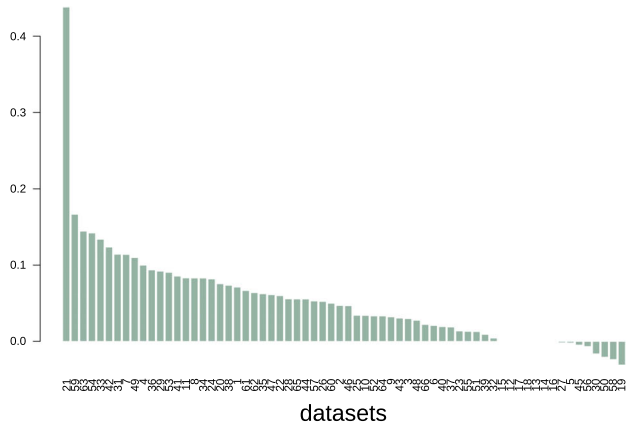
(a) NCELM vs BELM.



(b) NCELM vs BRELM.

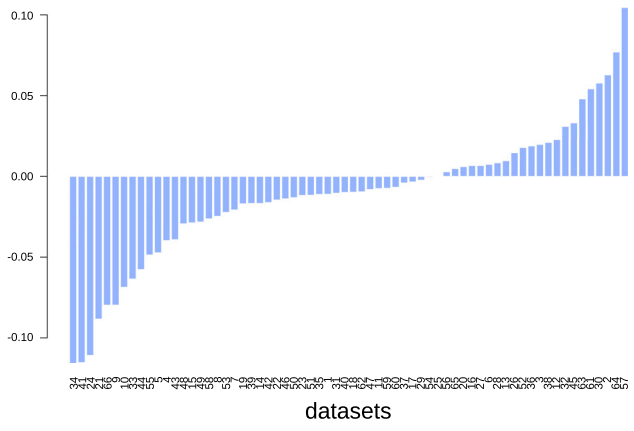


(c) NCELM vs AELM.

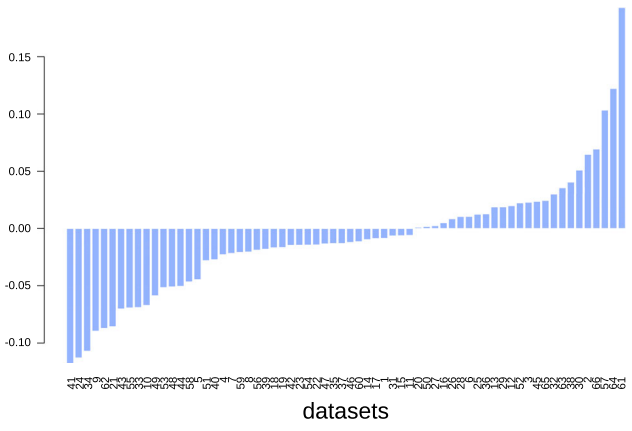


(d) NCELM vs ANCELM.

Fig. 2 Plots of comparison of the *Acc* generalization results



(a) NCELM vs BELM.



(b) NCELM vs BRELM.

Fig. 3 Plots of comparison of the *RMSE* generalization results

problems considered. The method was run 3 times in a tenfold for hyperparameter ranges: $C \in \{10^{-3}, 10^{-2}, \ldots, 10^{3}\}$, $\lambda \in \{10^{-4}, \ldots, 1\}$.

Figure 5 reports the average performance of the method over the 3 repetitions per fold for the selected classification datasets. The axis that represented the hyperparameters is

Table 5 *Acc* and *RMSE* performances and rankings with Holm test

Method	$\overline{Acc}$	$\overline{R}_{Acc}$	z-statistic	p value	$\alpha_{0.05}$	$\alpha_{0.10}$
Acc statistical analysis						
ANCELM$_\bullet$	0.77001	4.65909	7.51366	0.00000	0.01000	0.01000
AELM$_\bullet$	0.77592	4.33333	6.51338	0.00000	0.01250	0.01250
BELM$_\bullet$	0.80435	3.42424	3.72193	0.00020	0.01667	0.01667
BRELM$_\bullet$	0.80318	3.40909	3.67541	0.00024	0.02500	0.02500
ELM$_\bullet$	*0.80629*	*2.96212*	2.30295	0.02128	0.05000	0.0500
NCELM	*0.82323*	*2.21212*	–	–	–	–
Method	$\overline{RMSE}$	$\overline{R}_{RMSE}$	z-statistic	p value	$\alpha_{0.05}$	$\alpha_{0.10}$
RMSE statistical analysis						
BRELM$_\bullet$	*0.38060*	2.72727	2.76421	0.00571	0.01667	0.03333
BELM$_\circ$	0.38213	2.59091	2.15745	0.03097	0.02500	0.05000
ELM$_\circ$	0.38162	2.57576	2.09003	0.03661	0.05000	0.10000
NCELM	**0.37105**	**2.10606**	–	–	–	–

Best results are in bold face; second best in italics. • for significance with $\alpha = 0.05$; ○ for significance with $\alpha = 0.10$; $\overline{Acc}$ is the average *Acc* in the generalization set, and $\overline{R}_{Acc}$ is the average ranking; $\overline{RMSE}$ is the average RMSE in the generalization set, and $\overline{R}_{RMSE}$ is the average ranking

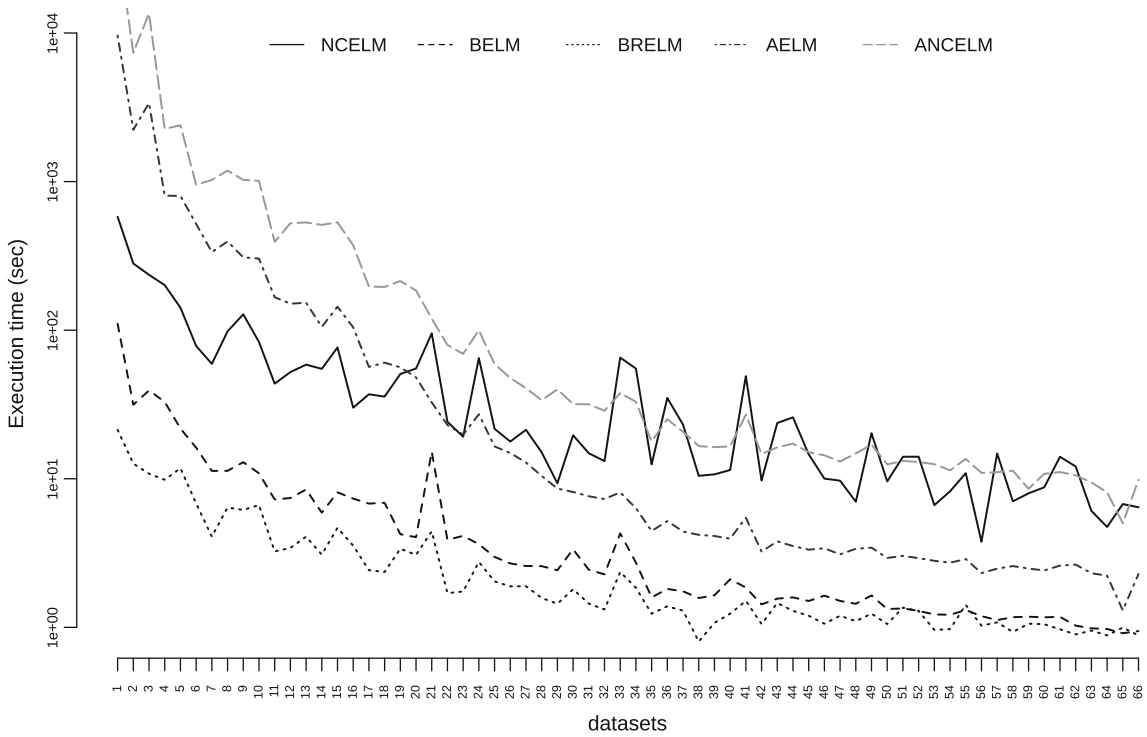
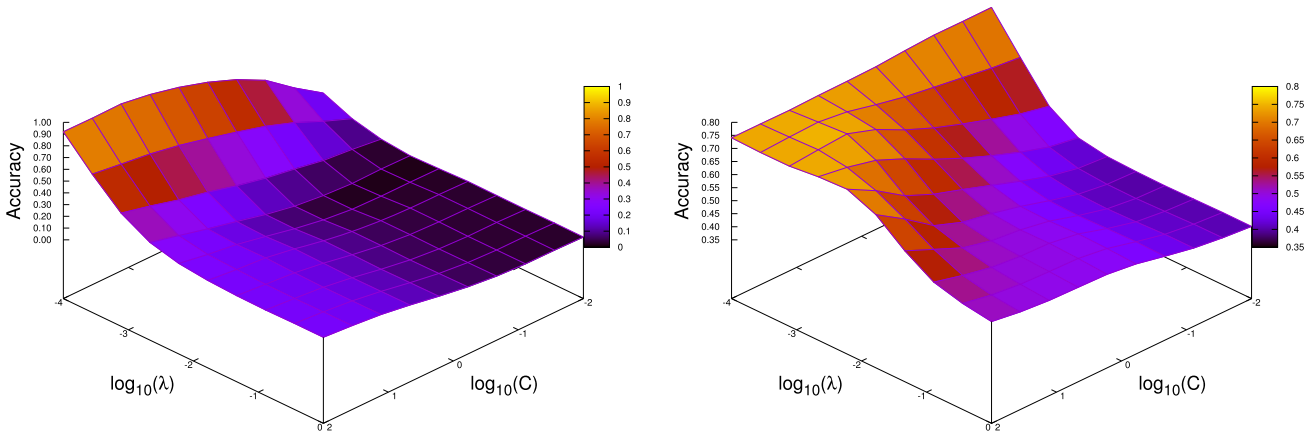


Fig. 4 Execution time in seconds needed for each dataset

on logarithmic scale for a better understanding of the figures. Figure 5a shows surface slopes from high values of accuracy near 1 (100% of correct classification rate) to 0 (0% of correct classification rate), while in Fig. 5b accuracy generalization results are within 0.85 and 0.35 (as indicated in the side legends). Figure 5 shows how the differences in accuracy with respect to the different values of the hyperparameters are considerable. For this reason, the ensemble method requires a proper hyperparameter selection in order to achieve a competitive performance.

The study of hyperparameter optimization (HPO) has a long history in machine learning [2, 38, 47], and each methodology has its own advantages and disadvantages. Thus, there is a family of methods that addresses HPO with



(a) Seeds dataset. **(b)** Breast-cancer dataset

Fig. 5 Hyperparameters study on accuracy for the NCLELM method and the parameters C and λ

computationally expensive approaches. The main advantage of this family of methods is its high performance (if compared to approaches with lesser computational burden). Evolutionary HPO is based on the idea of hyperparameter systematic evaluation, beginning with a random point and optimizing gradually toward a proper solution [68]. Reinforcement learning ideas have also been applied to HPO. For instance, Li et al. [43] define a hyperparameter exploratory as a non-stochastic infinite-armed bandit problem.

Although the importance of a proper choice of hyperparameters is graphically exposed in Fig. 5, adding computational time to complex machine learning algorithms significantly extends the total execution time, making it unfeasible in some cases. Less costly HPO methodologies could sometimes lead to machine learning models being accurate enough. For instance, Krueger et al. [40] propose an improved cross-validation procedure by selecting training subsets and sequentially choosing the best hyperparameter set. In this context, the hyperparameters of the proposed ensemble method are determined through a grid search as it achieves a competitive compromise between performance and computational burden. Additionally, the definition of a grid for hyperparameters allows experiments to be easily reproduced by other researchers, thus standardizing experimental frameworks in the research field.

6 Conclusions

This paper presents a new ensemble approach that introduces the negative correlation learning (NCL) framework into the extreme learning machine (ELM) community. The proposed ensemble method, named negative correlation extreme learning machine (NCELM), generates S initial ELM base classifiers and then incorporates into their error functions a penalty term inspired in the NCL framework. The proposed penalty term promotes explicitly diversity among the base classifiers and the final ensemble by analyzing the angle between the outputs of each individual and the outputs of the ensemble. Additionally, the computational burden of the proposed NCELM method is similar to the resolution of S independent ELM optimization problems, as the inverses are estimated through the Sherman–Morrison formula. The experiments show that: (1) fostering explicitly diversity among base classifiers generates ensembles with significantly better performances than those that promote diversity by data sampling and (2) the proposed NCELM method is more efficient than the baseline NCL-inspired method used for comparison purposes. The main limitation of the proposed method is that the outputs of the ensemble are assumed to be constant with respect to each base classifier in the iterations of the optimization procedure. Moreover, the parameters of the models are determined in a iterative way which undoubtedly increase the computational burden of the method (partially reduced by the implementation of the Sherman–Morrison formula). For these reasons, a highly desirable future work would be the global optimization of the parameters. This would address the two previously mentioned limitations of the proposed method.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Bauer E, Kohavi R (1999) An empirical comparison of voting classification algorithms: bagging, boosting, and variants. Mach Learn 36(1–2):105–139
2. Bengio Y (2000) Gradient-based optimization of hyperparameters. Neural Comput 12(8):1889–1900
3. Bergstra J, Bengio Y (2012) Random search for hyper-parameter optimization. J Mach Learn Res 13:281–305
4. Breiman L (1996) Bagging predictors. Mach Learn 24(2):123–140
5. Brown G, Wyatt J (2003) Negative correlation learning and the ambiguity family of ensemble methods. In: 4th international workshop on multiple classifier systems, vol 2709. Springer, pp 266–275
6. Brown G, Wyatt J, Harris R, Yao X (2005) Diversity creation methods: a survey and categorisation. Inf Fusion 6(1):5–20
7. Brown G, Wyatt J, Tino P (2005) Managing diversity in regression ensembles. J Mach Learn Res 6:1621–1650
8. Bühlmann P, Yu B (2003) Boosting with the L2 loss: regression and classification. J Am Stat Assoc 98(462):324–339
9. Bui T, Hernández-Lobato D, Hernandez-Lobato J, Li Y, Turner R (2016) Deep Gaussian processes for regression using approximate expectation propagation. In: 33rd international conference on machine learning, vol 48. ICML, pp 1472–1481
10. Cao F, Yang Z, Ren J, Chen W, Han G, Shen Y (2019) Local block multilayer sparse extreme learning machine for effective feature extraction and classification of hyperspectral images. IEEE Trans Geosci Remote Sens 57(8):5580–5594
11. Chaturvedi I, Ragusa E, Gastaldo P, Zunino R, Cambria E (2018) Bayesian network based extreme learning machine for subjectivity detection. J Frankl Inst 355(4):1780–1797
12. Chen H, Jiang B, Yao X (2018) Semisupervised negative correlation learning. IEEE Trans Neural Netw Learn Syst 29(11):5366–5379
13. Chen H, Yao X (2010) Multiobjective neural network ensembles based on regularized negative correlation learning. IEEE Trans Knowl Data Eng 22(12):1738–1751
14. Chu Y, Feng C, Guo C, Wang Y (2018) Network embedding based on deep extreme learning machine. Int J Mach Learn Cybern 10(10):2709–2724
15. Damianou A, Lawrence N (2013) Deep Gaussian processes. In: Artificial intelligence and statistics. AISTATS, pp 207–215
16. Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7:1–30
17. Dietterich TG (2000) Ensemble methods in machine learning. In: International workshop on multiple classifier systems. Springer, Berlin, pp 1–15
18. Ding S, Zhao H, Zhang Y, Xu X, Nie R (2015) Extreme learning machine: algorithm, theory and applications. Artif Intell Rev 44(1):103–115
19. Domingos P (1997) Why does bagging work? A Bayesian account and its implications. In: 3rd international conference on knowledge discovery and data mining. KDD, pp 155–158
20. Dua D, Graff C (2019) UCI machine learning repository. School of Information and Computer Sciences, University of California, Irvine. http://archive.ics.uci.edu/ml
21. Fernández-Navarro F, Gutiérrez PA, Hervás-Martánez C, Yao X (2013) Negative correlation ensemble learning for ordinal regression. IEEE Trans Neural Netw Learn Syst 24(11):1836–1849
22. Freund Y, Schapire RE (1997) A decision-theoretic generalization of on-line learning and an application to boosting. J Comput Syst Sci 55(1):119–139
23. Friedman M (1940) A comparison of alternative tests of significance for the problem of m rankings. Ann Math Stat 11(1):86–92
24. Göçken M, Özçalıcı M, Boru A, Dosdoğru AT (2019) Stock price prediction using hybrid soft computing models incorporating parameter tuning and input variable selection. Neural Comput Appl 31(2):577–592
25. Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2014) Generative adversarial nets. In: Advances in neural information processing systems, vol 27. NIPS, pp 2672–2680
26. Hager WW (1989) Updating the inverse of a matrix. SIAM Rev 31(2):221–239
27. Hastie T, Rosset S, Zhu J, Zou H (2009) Multi-class AdaBoost. Stat Interface 2(3):349–360
28. Higuchi T, Yao X, Liu Y (2002) Evolutionary ensembles with negative correlation learning. IEEE Trans Evol Comput 4(4):380–387
29. Holm S (1979) A simple sequentially rejective multiple test procedure. Scand J Stat 6(2):65–70
30. Huang GB, Zhou H, Ding X, Zhang R (2012) Extreme learning machine for regression and multiclass classification. IEEE Trans Syst Man Cybern B Cybern 42(2):513–29
31. Huang GB, Zhu QY, Siew CK (2006) Extreme learning machine: theory and applications. Neurocomputing 70(1–3):489–501
32. Chen H, Yao X (2009) Regularized negative correlation learning for neural network ensembles. IEEE Trans Neural Netw 20(12):1962–1979
33. Ibrahim W, Abadeh M (2019) Protein fold recognition using deep kernelized extreme learning machine and linear discriminant analysis. Neural Comput Appl 31(8):4201–4214
34. Islam MA, Anderson DT, Ball JE, Younan NH: Fusion of diverse features and kernels using LP norm based multiple kernel learning in hyperspectral image processing. In: 8th workshop on hyperspectral image and signal processing: evolution in remote sensing. IEEE, pp 1–5 (2016)
35. Jia X, Li X, Jin Y, Miao J (2019) Region-enhanced multi-layer extreme learning machine. Cognit Comput 11(1):101–109
36. Kingma DP, Ba J (2015) Adam: a method for stochastic optimization. In: 3rd international conference on learning representations. ICLR
37. Ko AHR, Sabourin R, De Oliveira LE, De Souza Britto A (2008) The implication of data diversity for a classifier-free ensemble selection in random subspaces. In: 19th international conference on pattern recognition. ICPR, pp 2251–2255
38. Kohavi R, John GH (1995) Automatic parameter selection by minimizing estimated error. In: Machine Learning Proceedings. Elsevier, pp 304–312
39. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, vol 25. NIPS, pp 1097–1105
40. Krueger T, Panknin D, Braun M (2015) Fast cross-validation via sequential testing. J Mach Learn Res 16:1103–1155
41. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521(7553):436–444
42. LeCun Y, Bengio Y et al (1998) Convolutional networks for images, speech, and time series. MIT Press, Cambridge, pp 255–258
43. Li L, Jamieson K, DeSalvo G, Rostamizadeh A, Talwalkar A (2018) Hyperband: a novel bandit-based approach to hyperparameter optimization. J Mach Learn Res 18(1):1–52
44. Liu Y, Yao X (1999) Ensemble learning via negative correlation. Neural Netw 12(10):1399–1404
45. Liu Y, Yao X (1999) Negatively correlated neural networks for classification. Artif Life Robot 3(4):255–259

46. Liu Y, Yao X (1999) Simultaneous training of negatively correlated neural networks in an ensemble. IEEE Trans Syst Man Cybern Part B (Cybern) 29(6):716–725
47. MacKay DJ (1996) Hyperparameters: optimize, or integrate out? In: 13th international workshop on maximum entropy and Bayesian methods, vol 62. Springer, pp 43–59
48. Mehrkanoon S (2019) Deep neural-kernel blocks. Neural Netw 116:46–55
49. Perrone M, Cooper L (1992) When networks disagree: ensemble methods for hybrid neural networks. Tech. rep., Brown University Providence, Institute for Brain and Neural Systems
50. Ran Y, Sun X, Sun H, Sun L, Wang X (2012) Boosting ridge extreme learning machine. In: IEEE symposium on robotics and applications. IEEE, pp 881–884
51. Rátsch G, Onoda T, Müller KR (2001) Soft margins for adaboost. Mach Learn 42(3):287–320
52. Riccardi A, Fernández-Navarro F, Carloni S (2014) Cost-sensitive AdaBoost algorithm for ordinal regression based on extreme learning machine. IEEE Trans Cybern 44(10):1898–1909
53. Schaal S, Atkeson CG (1996) From isolation to cooperation: an alternative view of a system of experts. In: Advances in neural information processing systems, vol 8. NIPS, pp 605–611
54. Schmidhuber J (2015) Deep learning in neural networks: an overview. Neural Netw 61:85–117
55. Shan P, Zhao Y, Sha X, Wang Q, Lv X, Peng S, Ying Y (2018) Interval lasso regression based extreme learning machine for nonlinear multivariate calibration of near infrared spectroscopic datasets. Anal Methods 10(25):3011–3022
56. Tang J, Deng C, Huang GB (2016) Extreme learning machine for multilayer perceptron. IEEE Trans Neural Netw Learn Syst 27(4):809–821
57. Tian H, Meng B (2010) A new modeling method based on bagging elm for day-ahead electricity price prediction. In: 5th international conference on bio-inspired computing: theories and applications. IEEE, pp 1076–1079
58. Tutz G, Binder H (2007) Boosting ridge regression. Comput Stat Data Anal 51(12):6044–6059
59. Ueda N, Nakano R (1996) Generalization error of ensemble estimators. In: International conference on neural networks. IEEE, pp 90–95
60. Van Heeswijk M, Miche Y, Oja E, Lendasse A (2011) Gpu-accelerated and parallelized elm ensembles for large-scale regression. Neurocomputing 74(16):2430–2437
61. Vincent P, Larochelle H, Lajoie I, Bengio Y, Manzagol PA (2010) Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion. J Mach Learn Res 11:3371–3408
62. Wang H, Zheng B, Yoon SW, Ko HS (2018) A support vector machine-based ensemble algorithm for breast cancer diagnosis. Eur J Oper Res 267(2):687–699
63. Wang S, Chen H, Yao X (2010) Negative correlation learning for classification ensembles. In: International joint conference on neural networks. IEEE, pp 1–8
64. Witten IH, Frank E (2005) 2nd data mining: practical machine learning tools and techniques. Data management systems. Elsevier, Amsterdam
65. Woźniak M, Graña M, Corchado E (2014) A survey of multiple classifier systems as hybrid systems. Inf Fusion 16:3–17
66. Wyner AJ, Olson M, Bleich J, Mease D (2017) Explaining the success of adaboost and random forests as interpolating classifiers. J Mach Learn Res 18(1):1558–1590
67. Xu X, Deng J, Coutinho E, Wu C, Zhao L, Schuller BW (2019) Connecting subspace learning and extreme learning machine in speech emotion recognition. IEEE Trans Multimed 21(3):795–808
68. Young SR, Rose DC, Karnowski TP, Lim SH, Patton RM (2015) Optimizing deep learning hyper-parameters through an evolutionary algorithm. In: Proceedings of the workshop on machine learning in high-performance computing environments. Association for Computing Machinery, pp 1–5
69. Zhang W, Xu A, Ping D, Gao M (2019) An improved kernel-based incremental extreme learning machine with fixed budget for nonstationary time series prediction. Neural Comput Appl 31(3):637–652
70. Zhao J, Liang Z, Yang Y (2012) Parallelized incremental support vector machines based on mapreduce and bagging technique. In: International conference on information science and technology. IEEE, pp 297–301