Neural Comput & Applic
DOI 10.1007/s00521-015-1974-0

ORIGINAL ARTICLE

Enforcement of the principal component analysis–extreme learning machine algorithm by linear discriminant analysis

A. Castaño[1] · F. Fernández-Navarro[2] · Annalisa Riccardi[3] · C. Hervás-Martínez[4]

Received: 26 June 2014 / Accepted: 5 June 2015

Abstract In the majority of traditional extreme learning machine (ELM) approaches, the parameters of the basis functions are randomly generated and do not need to be tuned, while the weights connecting the hidden layer to the output layer are analytically estimated. The determination of the optimal number of basis functions to be included in the hidden layer is still an open problem. Cross-validation and heuristic approaches (constructive and destructive) are some of the methodologies used to perform this task. Recently, a deterministic algorithm based on the principal component analysis (PCA) and ELM has been proposed to assess the number of basis functions according to the number of principal components necessary to explain the 90 % of the variance in the data. In this work, the PCA part of the PCA–ELM algorithm is joined to the linear discriminant analysis (LDA) as a hybrid means to perform the pruning of the hidden nodes. This is justified by the fact that the LDA approach is outperforming the PCA one on a set of problems. Hence, the idea of combining the two approaches in a LDA–PCA–ELM algorithm is shown to be in average better than its PCA–ELM and LDA–ELM counterparts. Moreover, the performance in classification and the number of basis functions selected by the algorithm, on a set of benchmark problems, have been compared and validated in the experimental section using nonparametric tests against a set of existing ELM techniques.

Keywords Principal component analysis · Linear discriminant analysis · Extreme learning machine · Neural networks

1 Introduction

The extreme learning machine (ELM) framework has received much attention in the machine learning community since Huang et al. [17, 18] proposed it in the International Joint Conference on Neural Networks. The ELM framework provides a way to estimate the parameters of a single hidden layer feedforward network (SLFN) analytically. The main idea of the framework is very intuitive and proceeds as follows: The parameters of the basis functions (also called hidden nodes) are randomly generated, while the weights connecting the hidden layer to the output layer are analytically computed by inverting the hidden layer output matrix. The ELM does not require any iterations to determine the network parameters, reducing dramatically the computational time. Thus, ELM is an highly computationally efficient learning algorithm that provides good generalization performance, even comparable to the

✉ F. Fernández-Navarro
i22fenaf@uco.es; fafernandez@uloyola.es

A. Castaño
adiel2008@gmail.com

Annalisa Riccardi
nina1983@gmail.com

C. Hervás-Martínez
chervas@uco.es

[1] Department of Computer Science, Universidad Politécnica Salesiana, Quito, Ecuador

[2] Deparment of Mathematics and Engineering, Universidad Loyola Andalucia, Seville, Spain

[3] Advanced Concepts Team, European Space Research and Technology Centre (ESTEC), European Space Agency (ESA), Noordwijk, The Netherlands

[4] Department of Computer Science and Numerical Analysis, University of Córdoba, Córdoba, Spain

Published online: 25 June 2015

generalization performance of the support vector machine (SVM) [15]. The ELM framework has been successfully applied in facial expression recognition [28], engine air-ratio control [29] or electricity price classification [25] to name just a few of real-world applications.

In the ELM framework, the determination of the optimal number of basis functions becomes an interesting and critical problem to exploit at most the ELM advantages. However, the original ELM [16] does not provide any effective solution to this problem. In most cases, the number of hidden nodes is part of a pre-processing analysis based on heuristics (trial and error or cross-validation procedures), a very tedious task in many real-world applications.

To avoid the aforementioned problems, some improvements of the original ELM have been proposed to optimize the network architecture. The available methodologies can be divided into two groups: destructive and constructive methods. For the former approach, Rong et al. [23] proposed the pruned ELM (P-ELM) for classification problems. The algorithm is initialized with a large network and then removes the basis functions having low relevance to the output, in the meaning of small output layer weights. Miche et al. [22] proposed the optimally pruned ELM (OP-ELM) algorithm which randomly initializes the hidden node weights and ranks the resultant basis functions. The OP-ELM algorithm applies the pruning strategy by the multi-response sparse regression (MRSR) algorithm and the leave-one-out (LOO) validation method. The main drawback of these destructive methods is that the algorithm starts with a large network that increases the computational complexity of the methodology. For the latter approach, the incremental ELM (I-ELM) [15] and its variants [13, 14] are based on the idea of adding basis functions one by one to the hidden layer and incrementally update the output weights. The main problems of these algorithms are that those methods cannot lead to an optimal network architecture, and the output weights of the SLFN are required to be recursively updated. On the other hand, Feng et al. [7] proposed the error minimized ELM (EM-ELM). The EM-ELM method can add random basis functions one by one or group by group. Unfortunately, the basis functions added are randomly generated and might deteriorate the performance with a large number of basis functions because no generalization performance is guaranteed. In a similar direction, Lan et al. [19] proposed the constructive selection of hidden nodes in ELM (CS-ELM). In this approach, the basis functions are selected by the MRSR algorithm. However, the CS-ELM is suitable just for regression problems and the selection of the basis functions is carried out on normalized regressors (increasing the computational burden).

Furthermore, all constructive and destructive methods for model selection of the ELM directly work on the basis function (hidden node) output matrix $\mathbf{H} \in \mathbb{R}^N \times \mathbb{R}^S$, where N is the number of patterns in the training set and S the number of basis functions. Generally, the number of training patterns, N, is significantly higher than the number basis functions, S, making the $\mathbf{H}$ matrix potentially rank deficient and leading to unstable model structures. Finally, constructive and destructive methods limit the number of available architectures, thus introducing constraints in the search space of possible structures that may not be suitable to the problem. Although these methods have been proved useful in simulated data [27], their application to real problems has been rather unsuccessful [11].

Recently, Castaño et al. [4] proposed the principal component analysis–ELM (PCA–ELM) where the basis functions are fitted taking into account the information retrieved from principal components analysis. The PCA–ELM algorithm sets the number of basis functions by determining the amount of principal components (orthogonal vectors) necessary to explain the 90 % of the variance in the training set. The output node parameters are determined analytically using the Moore–Penrose generalized inverse as in the base ELM algorithm. This approach considerably decreases the computational cost compared to later ELM improvements, since no cross-validation is needed and statistically outperform them.

On the other hand, linear discriminant analysis (LDA) has been used to reduce the dimensionality of the problem while maintaining the discriminability between pre-defined classes [20]. The mathematical model of the LDA is a linear combination of the input patterns and the projections generated in the LDA. In the LDA algorithm, $\mathbf{W} \in \mathbb{R}^S \times \mathbb{R}^K$ is the matrix corresponding to the S largest eigenvectors of the matrix $\mathbf{S}_w{}^{-1}\mathbf{S}_b$, where $\mathbf{S}_w$ is the within-class scatter matrix and $\mathbf{S}_b$ is the between-class scatter matrix. $\mathbf{S}_b$ is computed as the covariance matrix of the classes mean values, while $\mathbf{S}_w$ is computed as the average of the classes covariance matrices. Thus, it is assumed that the scatter of the data within class is the same for each class, unrealistic assumption in practical problems, which still is able to lead to good performances in practical classification tests where the reduced LDA model is as effective or even more effective than the original one. Figure 1 reports a 2-dimensional, 2-classes problem and the corresponding first PCA and LDA basis vectors. Clearly, as seen in this example, the two basis vectors define different projections, and in particular, the projection of patterns onto the first LDA basis vector is, in general, more effective in discriminating between classes than the one provided by the first PCA vector.

This can be explained by the fact that the LDA algorithm makes direct use of the between-classes covariance information while the PCA algorithm deals with the data as a whole, without including information on the underlying

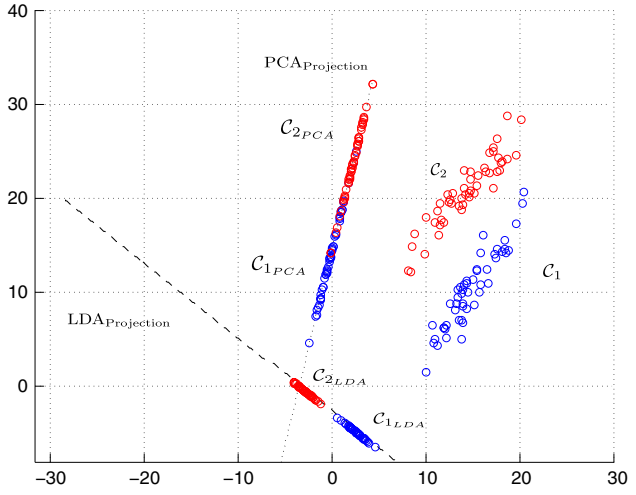

Fig. 1 Comparison of projections of PCA and LDA in a two-class classification problem: the classes are better separated by the projection onto the first LDA basis vector ($\mathbf{w}$) than the projection onto the first PCA eigenvector

class structure, into its analysis. Motivated by this fact, the LDA–PCA–ELM algorithm has been proposed, where the main difference with respect to the PCA–ELM method is that the hidden layer has been enlarged to include additional nodes for the LDA. It will be shown that the hybrid LDA–PCA approach in the ELM framework can improve the generalization performance of the model with respect to the two single analysis (PCA–ELM and LDA–ELM). This is due to the collaborative transformations applied by the two different analysis in the hidden nodes: while the LDA algorithm operates toward a better discrimination between classes, the PCA algorithm prefers a wide coverage of the data space.

The paper is organized as follows: A background of LDA and PCA is given in Sect. 2. The methodology to optimize the SLFN parameters based on ELM and the joined LDA–PCA is presented in Sect. 3. Section 4 describes the experimental framework adopted to evaluate the effectiveness of the method. Section 5 explains the results obtained. Finally, Sect. 6 summarizes the conclusions of the presented work.

2 Linear discriminant analysis and principal component analysis

As previously stated, the main contribution of this work is the extension of the previously proposed PCA–ELM algorithm [4] by the linear discriminant analysis (LDA) technique. The goal of this section is to establish the main differences of these two techniques. First, the LDA algorithm will be introduced, and afterward, the PCA method will be described.

Note that the LDA technique uses the information of the targets in a classification problem. Because of that, the classification context is described. In a classification problem, given a single pattern $\mathbf{x} = (x_1, \ldots, x_K) \in \mathbb{R}^K$, its corresponding class label $y \in \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_J\}$ needs to be assessed, according to the learning performed on the available dataset. The training dataset $D = \{\mathcal{X}, \mathcal{Y}\} = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^N$ is considered, where $\mathbf{x}_n = (x_n^{(1)}, \ldots, x_n^{(K)}) \in \mathbb{R}^K$ is the vector of measurements, and $\mathbf{y}_n \in \mathbb{R}^J$ is the known class level of the n-th pattern. In this work, the common technique to represent class levels using the "1-of-J" encoding vector is assumed; hence, $\mathbf{y}_n = (y_n^{(1)}, y_n^{(2)}, \ldots, y_n^{(J)})$ with $y_n^{(j)} = 1$ in case $\mathbf{x}_n$ is a pattern classified as class $\mathcal{C}_j$, 0 otherwise.

2.1 Linear discriminant analysis

LDA is a method used in pattern recognition and machine learning to define a linear combination of features able to discriminate between two or more classes of patterns. LDA is also closely related to PCA which also seeks for linear combinations of features with the purpose of better separating the patterns. However, the main difference between the two approaches is that LDA explicitly attempts to model the difference between classes, while PCA does not use targets information to perform the transformations.

By applying LDA, the projections that maximize the distance between patterns of different classes and minimize the distance between patterns of the same class are found. In another words, the maximization of the between-class scatter matrix $\mathbf{S}_\text{b}$, together with the minimization of the within-class scatter matrix $\mathbf{S}_\text{w}$ in the projective subspace, is performed. The within-class scatter matrix $\mathbf{S}_\text{w} \in \mathbb{R}^K \times \mathbb{R}^K$ and the between-class scatter matrix $\mathbf{S}_\text{b} \in \mathbb{R}^K \times \mathbb{R}^K$ are defined as:

$$\mathbf{S}_\text{w} = \sum_{j=1}^{J} \sum_{\mathbf{x} \in \mathcal{C}_\text{j}} (\mathbf{x} - \overline{\mathbf{x}}_{\mathcal{C}_\text{j}})(\mathbf{x} - \overline{\mathbf{x}}_{\mathcal{C}_\text{j}})^\mathbf{T}, \tag{1}$$

$$\mathbf{S}_\text{b} = \sum_{j=1}^{J} (\overline{\mathbf{x}}_{\mathcal{C}_j} - \overline{\mathbf{x}})(\overline{\mathbf{x}}_{\mathcal{C}_j} - \overline{\mathbf{x}})^T, \tag{2}$$

where $\overline{\mathbf{x}}_{\mathcal{C}_j} = (\frac{1}{N_j}) \sum_{\mathbf{x} \in \mathcal{C}_\text{j}} \mathbf{x}$ is the mean of the j-th class with N_j the number of patterns belonging to it, $\overline{\mathbf{x}} = (\frac{1}{N}) \sum_{i=1}^{N} \mathbf{x_i}$ is the mean of all patterns, and J is the number of classes. The goal is to maximize the variance between classes while minimizing the variance within class

$$J(\mathbf{W}) = \underset{\mathbf{W}}{\text{argmax}} \left(\frac{\mathbf{W}^T \mathbf{S}_\text{b} \mathbf{W}}{\mathbf{W}^\mathbf{T} \mathbf{S}_\text{w} \mathbf{W}} \right). \tag{3}$$

It has been proved already in [9] that, if $\mathbf{S}_\text{w}$ is a non-singular matrix, and then, the maximum is attained by the

matrix $\mathbf{W}$ that has column vectors as the nonzero eigenvectors of the matrix $\mathbf{S}_{\mathrm{w}}^{-1}\mathbf{S}_{\mathrm{b}}$, and they are at most $J-1$. Hence, the subspace for LDA is spanned by the set of vectors

$$\mathbf{W} = \{\mathbf{w_1}, \mathbf{w_2}, \ldots, \mathbf{w_{J-1}}\} \in \mathbb{R}^{\mathbf{K}} \times \mathbb{R}^{\mathbf{J-1}}.$$

The main characteristics of LDA are:

- The new variables generally present high separability between patterns of different classes and great union between same class ones.
- At most produces $J-1$ feature projections.

2.2 Principal component analysis

PCA technique is an orthogonal transformation of the feature space that aims to find a set of linearly uncorrelated components that better describes the variance between the data. In particular, the principal component is the vector that accounts for the greatest variance, while the following components are chosen with the same criteria but with the orthogonality constraint. Mathematically, those vectors are the eigenvectors corresponding to the largest eigenvalues of the dataset covariance matrix. The data are further projected onto a subset of those directions for dimensionality reduction.

If the matrix whose columns are the eigenvectors sorted according to the ascending order of the corresponding eigenvalues is denoted by $\mathbf{U} \in \mathbb{R}^{\mathbf{K}} \times \mathbb{R}^{\mathbf{K}}$, the PCA transformation of the data is

$$\mathbf{X} = \mathbf{U^T X},$$

where $\mathbf{X} = (\mathbf{x_1}, \ldots, \mathbf{x_N}) \in \mathbf{R}^{\mathbf{K}} \times \mathbf{R}^{\mathbf{N}}$ denotes the patterns matrix. By selecting only the first d rows of $\tilde{\mathbf{X}}$ (with $d \leq K$), the data have been projected from K down to d dimensions.

The PCA technique has been traditionally implemented by the application of the singular value decomposition (SVD) of the patterns matrix $\mathbf{X}$, defined as

$$\mathbf{X} = \mathbf{Z\Sigma V^T}$$

where $\mathbf{V} \in \mathbb{R}^{\mathbf{N}} \times \mathbb{R}^{\mathbf{N}}$ and $\mathbf{Z} \in \mathbb{R}^{\mathbf{K}} \times \mathbb{R}^{\mathbf{K}}$ are orthonormal matrix, $\mathbf{\Sigma} \in \mathbb{R}^{K} \times \mathbb{R}^{N}$ is a pseudo-diagonal matrix whose diagonal entries are ordered in a descending order, and they correspond to the eigenvalues of the $\mathbf{X}$ matrix and the values outside the diagonal are zero. The eigenvectors of the covariance matrix $C = (\frac{1}{N-1})\mathbf{X^T X}$ are computed from the $\mathbf{V}$ matrix since

$$\mathbf{X^T X} = \mathbf{V\Sigma Z^T Z\Sigma V^T} = \mathbf{V\Sigma^2 V^T}$$

and then

$$C = \left(\frac{1}{N-1}\right)\mathbf{V\Sigma^2 V^T}.$$

Being C symmetric, it stands that

$$C = \mathbf{V\Lambda V^T}$$

where Λ is the diagonal matrix with eigenvalues of C as entries values on the diagonal and the column of V as the corresponding eigenvectors. The eigenvalues and eigenvectors of the covariance matrix of the pattern matrix can then be easily computed by the elements of the SVD decomposition.

It is important also to stress the attention on the necessity of performing the mean subtraction (also called "mean centering") to ensure that during the PCA analysis, the first principal component is not too close to the mean of the data, but describes the direction of maximum variance.

The main characteristics of PCA can be summarized as:

- The features in the projected space are uncorrelated.
- The covariance matrix represents only second-order statistics among the vector values.
- Maximize variance of patterns in the projected space.

2.3 Why LDA can outperform PCA for classification tasks?

First of all, it is worth mentioning that in a classification problem of dimension K and J classes, the PCA methods estimate K projections, while the LDA method finds $J-1$ projections. As previously noted, LDA uses targets information to improve the performance of the algorithms that applies LDA with respect to the same algorithms that instead make use of PCA. To justify the proposed combination of LDA and PCA for classification, the example in Fig. 1 is discussed. As can be seen, the patterns are distributed in a particular and not uncommon way. The PCA tries to maximize data variance, while LDA finds the best projection that separates the classes. A joined combination of the two approaches can help in preserving distinction between classes and a good spread of the patterns within each class.

Due to the intuitive characteristics of LDA for classification and the results obtained in previous works [4], where PCA transformations were included in the ELM framework, the integration of LDA into the PCA–ELM model is here proposed and discussed.

3 The proposed method: linear discriminant analysis–principal component analysis–extreme learning machine (LDA–PCA–ELM)

The goal of this section is to describe the proposed method, named linear discriminant analysis–principal component analysis–extreme learning machine (LDA–PCA–ELM). As

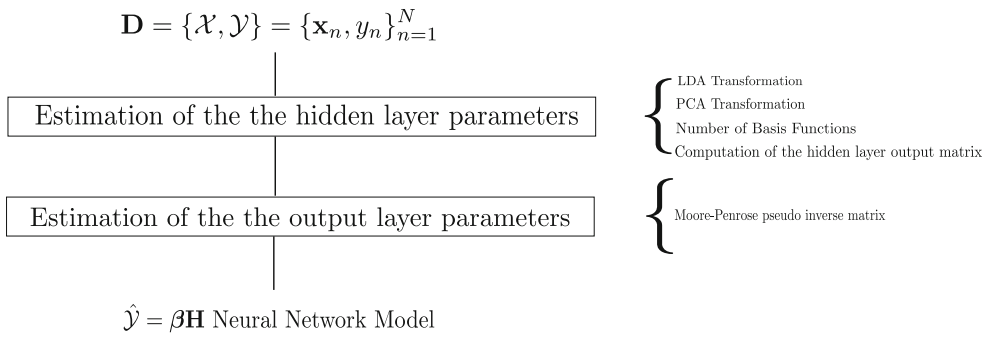

Fig. 2 LDA–PCA–ELM framework ($\hat{\mathcal{Y}}$ is the estimated outputs of the model)

shown in the experimental section, the algorithm is a suitable option to estimate the parameters of SLFN. A SLFN is a type of neural network composed by three layers: a input, a hidden and an output layer. Let us denote with $\mathbf{a_s} = (a_1^s, a_2^s, \ldots, a_K^s)^T$ the weight vector connecting the input units to the s-th hidden unit, $s = 1, \ldots, S$, and $\boldsymbol{\beta}^j = (\beta_1^j, \beta_2^j, \ldots, \beta_S^j)^T$ is the weight vector connecting the hidden nodes to the j-th output node $\forall j = 1, 2, \ldots, J$. During the training process, the optimal parameters, $\mathbf{a^s}$ and $\boldsymbol{\beta}^j$, are determined so that they minimize the squared error (SE) function defined by:

$$\mathrm{SE} = \sum_{n=1}^{N}\sum_{j=1}^{J}(f_j(\mathbf{x_n}) - y_n^{(j)})^2, \tag{4}$$

where $f_j(\mathbf{x_n})$ is the estimated output of the j-th class for the n-th input pattern, which is defined as:

$$f_j(\mathbf{x_n}) = \sum_{\mathbf{s=1}}^{\mathbf{S}} \beta_{\mathbf{s}}^{\mathbf{j}} \phi_{\mathbf{s}}(\mathbf{x_n; a^s}) \tag{5}$$

where $\phi_s(\mathbf{x_n; a^s})$ is the sigmoidal activation function in this work. The minimization of the squared error function in the ELM is achieved solving the following linear system:

$$\mathbf{H}\boldsymbol{\beta} = \mathcal{Y}, \tag{6}$$

where $\mathbf{H}$ is known as the hidden layer output matrix of the SLFN and defined as:

$$\begin{aligned}\mathbf{H} &= (\mathbf{h_1}, \mathbf{h_2}, \ldots, \mathbf{h_S}) \\ &= \begin{pmatrix} \phi_1(\mathbf{x_1; a^1}) & \ldots & \phi_S(\mathbf{x_1; a^S}) \\ \ldots & \ldots & \ldots \\ \phi_1(\mathbf{x_n; a^1}) & \ldots & \phi_S(\mathbf{x_n; a^S}) \end{pmatrix}_{\mathbb{R}^N \times \mathbb{R}^S},\end{aligned} \tag{7}$$

$$\mathcal{Y} = (\mathbf{y_1}, \mathbf{y_2}, \ldots, \mathbf{y_n})^{\mathbf{T}}_{\mathbb{R}^{\mathbf{N}} \times \mathbb{R}^{\mathbf{J}}}, \tag{8}$$

and

$$\boldsymbol{\beta} = (\boldsymbol{\beta}^1, \boldsymbol{\beta}^2, \ldots, \boldsymbol{\beta}^J)_{\mathbb{R}^S \times \mathbb{R}^J}. \tag{9}$$

In the case of the standard ELM framework proposed by Huang et. al. [17], the parameters connecting the input layer to the hidden layer ($\mathbf{a^s}, \forall \mathbf{s} = \mathbf{1}, \ldots, \mathbf{S}$) are randomly determined, while the weights for the output layer are analytically computed using the Moore–Penrose pseudo-inverse matrix.

The main contribution of the proposed algorithm (LDA–PCA–ELM) is the initialization of the weights connecting the input layer to the hidden layer. In the LDA–PCA–ELM algorithm, these weights are initialized according to the combined LDA and PCA methods fitting the biases to zero. The LDA–PCA–ELM algorithm can be viewed as a two-stage algorithm as shown in Fig. 2.

3.1 Estimation of the hidden layer parameters

The first step of the LDA–PCA–ELM algorithm is the execution of the LDA and PCA algorithm over the data in the training set. Then, the parameters $\mathbf{a^s}$, $s = 1, \ldots, S$ are initialized according to the basis projection vectors generated by both the LDA and PCA approaches. Taking into consideration that the maximum number of projections generated by the LDA are $J - 1$ and the maximum number of projections generated by the PCA are K, the maximum number of basis functions to be included in the SLFN is $J - 1 + K$. Hence, $S \in [1, J - 1 + K]$.

The next step is the pruning step: In this work, it is considered that the basis functions to be included in the SLFN must explain the 90 % of the variance of the data in both approaches. This method prunes the model using the same mechanism used by PCA–ELM. Therefore, only those S_1 basis functions that explain 90 % or more of the variance of the data in the training set out of the $J - 1$ possible basis functions are included in the model for the LDA case, and S_2 basis functions that explain 90 % or more of the variance of the data in the training set out of the K possible basis functions are included in the model for the PCA case. Hence,

Table 1 Main differences of the ELM algorithm with respect to the LDA–PCA–ELM algorithm

ELM	LDA–PCA–ELM
1. Assign arbitrary input weights $\mathbf{a}^s$, $s = 1, \ldots, S$	1. Execute LDA and PCA over the training set
2. Calculate the hidden layer output matrix $\mathbf{H}$	2. Select the optimal number of transformations
3. Calculate the output weights $\boldsymbol{\beta}$: $\boldsymbol{\beta} = \mathbf{H}^{\dagger}\mathcal{Y}$	3. Calculate the hidden layer output matrix $\mathbf{H}$
	4. Calculate the output weights $\boldsymbol{\beta}$: $\boldsymbol{\beta} = \mathbf{H}^{\dagger}\mathcal{Y}$

where $\mathbf{H}$, $\boldsymbol{\beta}$ and $\mathcal{Y}$ are as defined before

$$S = S_1 + S_2.$$

This deterministic technique to set the number of hidden nodes rises the complexity of the model with respect to random-based initialization of the ELM algorithms.

After that, the LDA–PCA–ELM proceeds by choosing an activation function $\phi(\mathbf{x})$ to generate the hidden layer output matrix $\mathbf{H}$. Among the existing hidden node types, the one composed by a linear activation function and a transfer function are considered in this work. Because of this, any hidden node that is compounded by a linear combination as the activation function may be taken into consideration as a suitable choice to be used in the hidden layer of the SLFN. Some of the most common hidden nodes with a linear activation function are mentioned hereafter:

- Sigmoidal function (Sig): In particular, the special case of the logistic function is defined as:

$$\mathrm{sig}(n) = \frac{1}{1 + \exp(-n)}. \tag{10}$$

- Hard-limit transfer function (hardlim): It returns one for nonzero negative values, zero otherwise:

$$\mathrm{hardlim}(n) = \begin{cases} 1 & \text{if } n \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

- Sine (Sin): It returns zero for values close to $2K\pi$.

In order to validate our algorithm and for the sake of simplicity, the experiments are only done using sigmoidal nodes in the hidden layer.

3.2 Estimation of the output layer parameters

Finally, the estimation of the output layer weights is done as proposed by Huang [17], solving the linear system $\mathbf{H}\boldsymbol{\beta} = \mathcal{Y}$ using the Moore–Penrose generalized inverse. In this way, the values of the $\boldsymbol{\beta}$ parameters are estimated as:

$$\boldsymbol{\beta} = \mathbf{H}^{\dagger}\mathcal{Y} \tag{12}$$

where $\mathbf{H}^{\dagger}$ is the Moore–Penrose generalized inverse of the hidden layer output matrix. It has also been shown that this solution is the least square solutions of the general linear system $\mathbf{H}\boldsymbol{\beta} = \mathcal{Y}$; therefore, it minimizes the training error. Moreover, it is unique and has the smallest norm of weights, hence having also good generalization performance [5, 17].

Table 1 summarizes the algorithmic steps of the base ELM algorithm and the LDA–PCA–ELM method and highlights their main differences.

3.3 Discussion about the advantages of the LDA–PCA–ELM

The LDA–PCA–ELM proposed algorithm is an extension of the previously proposed PCA–ELM algorithm, and it inherits some of the most important advantages of the PCA–ELM and the good properties of LDA with respect to the state-of-the-art ELM approaches:

1. The algorithm is deterministic instead of stochastic. This implies that the algorithm does not present a high standard deviation of its performance and model complexity as OP-ELM and other algorithms do.
2. Previously proposed approaches determine the optimal number of basis functions by growing or pruning techniques. These techniques are generally hard time-consuming. The proposed approach determines the amount of hidden nodes taking into consideration the cumulative variance expressed in the LDA and PCA algorithms.

The main drawback of the proposed method with respect to the PCA–ELM is that PCA–ELM can be applied to regression and classification problems, while LDA–PCA–ELM can be applied only to classification problems because LDA uses information of the targets, trying to maximize the distance between classes and minimize the distance within class.

4 Experimental framework

In this section, the setting of the experimental study performed is presented. Section 4.1 lists the datasets tested; Sect. 4.2 provides a description of the metrics used to evaluate the performance of the algorithms; Sect. 4.3 presents the list of algorithms implemented for comparison together with their parameters; finally, Sect. 4.4 describes

Table 2 Characteristics of the datasets used for the experiments: number of instances (size), number of Real (R), binary (B) and nominal (N) input variables, total number of inputs (In.), number of classes (Out.), and per-class distribution of the instances (distribution)

Dataset	Size	R	B	N	In	Out	Distribution
Hepatitis (HEP)	155	6	13	–	19	2	(32, 123)
Heart (HEA)	270	13	–	–	13	2	(150, 120)
Breast (BRE)	286	15	–	–	15	2	(215, 71)
Haberman (HAB)	306	3	–	–	3	2	(225, 81)
Ionos (ION)	351	33	1	–	34	2	(126, 225)
Vote (VOT)	435	–	16	–	16	2	(267, 168)
Card (CAR)	690	6	4	5	51	2	(307, 383)
BreastW (BRW)	699	9	–	–	9	2	(524, 175)
Diabetes (DIA)	822	8	–	–	8	2	(576, 246)
German (GER)	1000	6	3	11	61	2	(700, 300)
Post-Op (POP)	90	1	–	7	20	3	(2, 24, 64)
Gene (GEN)	3175	–	–	60	120	3	(765, 765, 1648)
Lymph (LYM)	148	3	9	6	38	4	(2, 81, 61, 4)
Ecoli (ECO)	336	7	–	–	7	8	(143, 77, 52, 35, 20, 5, 2, 2)
Yeast (YEA)	1484	8	–	–	8	10	(463, 429, 30, 163, 51, 44, 35, 244, 20, 5)

All nominal variables are transformed to binary variables

the statistical tests performed to validate the obtained results.

4.1 Datasets considered

The proposed algorithm was tested on fifteen datasets taken from the UCI repository [1]. The datasets selected range from binary problems to multi-class problems, and they present a variety of size, number of features and number of classes. Table 2 summarizes the characteristics of the datasets used. In particular, the size of the datasets vary from 90 to 3175 patterns, and the number of features ranges from 3 to 120. They have been partitioned using a hold-out cross-validation procedure with $3/4 \cdot n$ instances for the training dataset and n / 4 instances for the generalization set. To account for the aleatory nature of the ELM stochastic approaches,[1] those algorithms were run 30 times for each problem. The ELM deterministic approaches (PCA–ELM, LDA–ELM and LDA–PCA–ELM algorithms) were run just one time per dataset. In addition, instances with missing values have been discarded before the execution of the methods over the datasets, and the whole set of values have been normalized in the interval [0, 1] to uniform the sensitivity of attributes with different range domains.

[1] By ELM stochastic approaches, we are naming to those ELM-based methodologies where the hidden layer parameters are randomly initialized. In this study, the only deterministic methodologies considered are those that initialize the hidden layer parameters according to the LDA or PCA projections (or a combination of both).

4.2 Evaluation metrics

Two metrics were used to assess the performance of each method:

- **Accuracy rate** (Acc) It is the number of successful hits (correct classifications) relative to the total number of classifications. It is the most widely used metric.
- **Number of hidden nodes** (NHN) In this work, we use the NHN to measure the complexity of the neural networks models. Taking into account that all the methods considered work with fully connected neural networks, the NHN is a robust metric to estimate the complexity of the model.

Additionally, the time required to estimate the parameters of each method has been also considered. The time (T) is the simplest way to estimate the computational efficiency of the methods. The average time elapsed (in seconds) is considered, and this includes cross-validation, training and test time.

4.3 ELM algorithms selected

The proposed method (LDA–PCA–ELM) is compared to the following ELM approaches:

- The cross-validated original extreme learning machine (ELM_{CV}) [17]. In this method, the number of hidden nodes in the base ELM algorithm has been selected by a nested fivefold cross-validation procedure over the training set, i.e., once the lowest cross-validation error alternative was obtained, it was applied to the complete training set and test results were extracted. The criteria

for selecting the best configuration were the Acc performance. The values considered in the cross-validation procedure were $\{10, 20, \ldots, 100\}$. The sigmoidal nonlinear transformation was the basis function considered for the hidden layer.

- The optimally pruned extreme learning machine (OP-ELM) [21]. The OP-ELM extends the original ELM algorithm wrapping it around with a methodology that prunes the hidden neurons, leading to a more robust overall algorithm. Again, the sigmoidal nonlinear transformation was the one considered in this study. The number of nodes (S) in the hidden layer in the OP-ELM algorithm is set at the beginning to 100, since this algorithm prunes the useless neurons from the hidden layer.
- The cross-validated evolutionary extreme learning machine E-ELM$_{\mathrm{CV}}$ [3, 24, 30] improves the original ELM by using a differential evolution (DE) algorithm [26] and determines the optimal number of hidden nodes by a cross-validation procedure. The E-ELM$_{\mathrm{CV}}$ uses the evolutionary technique to optimize the input weights and the Moore–Penrose generalized inverse to analytically determine the output weights.
  In the same way as in the ELM algorithm, the most critical parameter in the E-ELM$_{\mathrm{CV}}$ algorithm is the number of the hidden nodes, S. The number of hidden nodes was selected with a cross-validation procedure based on a set of nodes multiple of 10 ($\{10, 20, \ldots, 100\}$). To widely explore the search space, a population of 100 individuals is used in the evolutionary procedure, for a maximum number of generations equal to 50. The E-ELM$_{\mathrm{CV}}$ algorithm has been implemented using the sigmoidal unit as the basis function in the hidden layer.
- The incremental extreme learning machine (I-ELM) [15]. This algorithm proposes a procedure that increases the network architecture adding random nodes till the residual error is bigger than a threshold. This threshold must be set in advance, and its value is very sensible to the number of patterns and classes. The algorithm was initially proposed for regression but can be extended for classification.
- The pruned extreme learning machine (P-ELM) [23] uses statistical methods to measure the relevance of hidden nodes. Hidden nodes are ranked using statistical techniques such as χ^2 and information gain. Irrelevant nodes are then pruned by considering their relevance to the class labels. The optimal number of hidden nodes is estimated considering the performance of the classifier on a (stratified) validation set constructed with the 25 % of the training set patterns as suggested by the authors.
- The error minimized extreme learning machine (EM-ELM) [7]. The main difference of this algorithm with respect to the I-ELM method resides in the computation of the output layer weights which are estimated with the goal of improving the computational burden of the previously mentioned algorithm. The algorithm is strongly sensitive to an error threshold parameter which depends on number of patterns, classes and hidden nodes. A hyper-parameter was introduced in the model to implement a fair experimental comparison with respect to the remaining methods. The new hyper-parameter was defined as the relative error with respect to the number of patterns and classes. The hyper-parameter space was explored by considering a cross-validation procedure based on a set of hidden nodes multiple of 10 ($\{10, 20, \ldots, 100\}$).
- The principal component analysis–extreme learning machine (PCA–ELM) [4]. This algorithm does not have any hyper-parameter to be estimated by cross-validation. As explained before, the number of hidden nodes, S, is estimated according to the accumulated variance expressed into the eigenvectors.
- The linear discriminant analysis–extreme learning machine (LDA–ELM). This algorithm has been also included to prove that the combination of LDA and PCA is able to outperform its single counterparts.

4.4 Statistical tests

Hypothesis testing techniques is a useful tool to statistically interpret the results obtained in the experimental study and discriminate between methods according to their performance [6]. Nonparametric tests have been chosen over parametric ones, for this specific case, because the initial conditions that guarantee the reliability of the second ones may not be satisfied, entailing the statistical analysis to lose plausibility.

In particular, the Friedman pre hoc test, is used. Its application can outline significant differences between the methods tested. Moreover, the Holm post hoc procedures will detect which algorithms are significant to be used for comparisons.

5 Experiments

5.1 Analysis of generalization performance

For the sake of simplicity, only the graphical and the summary of the statistical results achieved are included, whereas the complete results can be found at the url www.esa.int/act/LDA-PCA-ELM.zip.

In the scatterplot of Fig. 3, each point compares LDA–PCA–ELM to another methodology on a single dataset. LDA–PCA–ELM was considered as the base method because it reported the best mean results. The x-axis position of the point is the mean performance of LDA–PCA–ELM of one dataset, and the y-axis position is the performance of the compared algorithm (using Acc). Therefore, points below the $y = x$ line correspond to datasets for which LDA–PCA–ELM performs better than the other algorithm when the metric should be maximized (Acc), and points above the $y = x$ line represent to datasets for which LDA–PCA–ELM achieves a better performance than the compared method when the metric should be minimized (NHN metric). As can be seen in Fig. 3, the LDA–PCA–ELM is a competitive methodology if it is compared to the most promising approach (the PCA–ELM method). From the analysis of the results (Table 3), it can be concluded that the LDA–PCA–ELM model produces the best mean ranking in Acc ($\overline{R_{\text{Acc}}} = 2.10$), reporting also the highest mean accuracy ($\overline{\text{Acc}} = 82.91\,\%$).

The nonparametric Friedman test [10], using as test variable the Acc value computed in the generalization set of the best models, has been performed to determine the statistical significance of the rank differences reported for each method in the different datasets. The test shows that the effect of the method used for classification is statistically significant at a significance level of 10 %.

Based on this rejection, the Holm post hoc test was used to compare all classifiers with a control method [12]. For the experiments carried out, the control method selected is the one reporting the best mean ranking in Acc, the LDA–PCA–ELM method. The results of the Holm test for $\alpha = 0.10$ can be seen in Table 3. By using a level of significance $\alpha = 0.10$, LDA–PCA–ELM is significantly better than the rest of the algorithms considered except PCA–ELM.

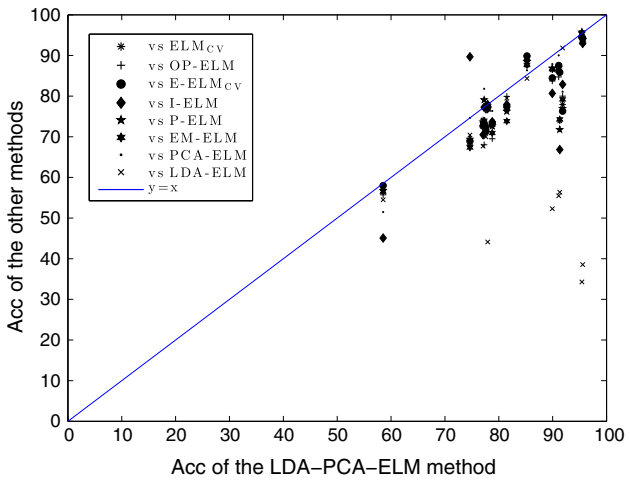

Fig. 3 Graphical results of the generalization performance

Table 3 Statistical results using Acc as the variable test: mean Acc in the generalization set ($\overline{\text{Acc}}$), mean ranking ($\overline{R}_{\text{Acc}}$), z-statistic for the Holm post hoc test, p value and corrected alpha (α'_{Holm}) for the same test

Method	$\overline{\text{Acc}}$	$\overline{R}_{\text{Acc}}$	z-statistic	p value	α'_{Holm}
LDA–ELM	63.35	7.00	4.90	0.000	0.012
EM-ELM	77.85	6.06	3.96	7.0E−5	0.014
OP-ELM	77.43	5.73	3.63	2.8E−4	0.016
ELM$_{CV}$	78.51	5.13	3.10	0.001	0.020
I-ELM	78.50	5.06	3.03	0.024	0.025
E-ELM$_{CV}$	78.99	5.20	2.96	0.030	0.033
P-ELM	78.49	4.86	2.76	0.005	0.050
PCA–ELM	*80.63*	*3.70*	1.73	0.080	0.100
LDA–PCA–ELM	**82.91**	**2.10**	–	–	

Best results in bold face; second best in italics

5.2 Analysis of model complexity

In this section, a graphical plot of the number of hidden nodes (NHN) is shown. Figure 4 is the scatter plot representation of the NHN included in each model for the methodologies considered. As can be seen in Fig. 4, the LDA–PCA–ELM is a competitive method when compared to the state-of-the-art ELM methods. As can be seen, the most algorithms are located over the diagonal line indicating that are more complex models. From a purely descriptive point of view, we can see how the LDA–PCA–ELM methods obtain the third best mean ranking and overall NHN (Table 4). The additional performance gains for including the LDA transformations in the hidden layer

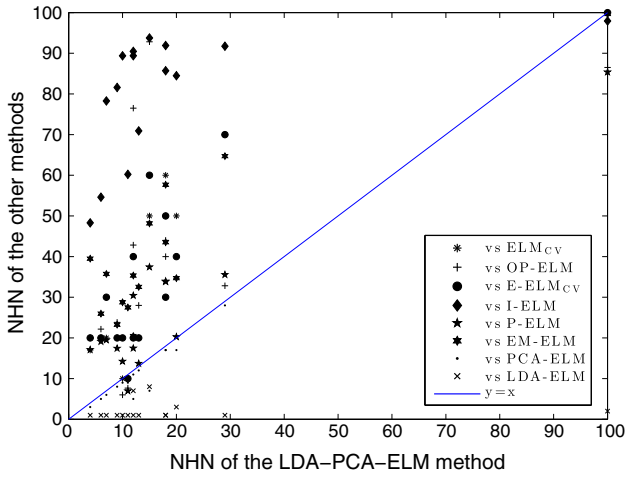

Fig. 4 Graphical results of the complexity of the models considered (according to the number of hidden nodes, NHN)

Table 4 Statistical results using NHN as the variable test: mean NHN selected in the training step ($\overline{\text{NHN}}$), mean ranking ($\overline{R}_{\text{NHN}}$), z-statistic for the Holm post-hoc test, p value and corrected alpha (α'_{Holm}) for the same test

Method	$\overline{\text{NHN}}$	$\overline{R}_{\text{NHN}}$	z-statistic	p value	α'_{Holm}
I-ELM	80.57	1.33	7.53	0.000	0.012
EM-ELM	41.16	3.13	5.73	0.000	0.014
ELM_{CV}	37.33	3.63	5.23	0.000	0.016
E-ELM_{CV}	36.67	3.76	5.1	0.000	0.020
OP-ELM	38.99	4.30	4.56	2.0E−5	0.025
P-ELM	26.82	5.93	2.93	0.003	0.033
LDA–PCA–ELM	18.93	6.30	2.56	0.102	0.050
PCA–ELM	*16.80*	*7.73*	1.13	0.257	0.100
LDA–ELM	**2.06**	**8.86**	–	–	–

Best results in bold face; second best in italics

justifies the inclusion of these transformations in the final model (despite on increase in the model complexity).

It is necessary though to determine whether there are differences in the mean ranking of NHN. Hence, another nonparametric Friedman test has been performed, showing that the effect of the method used for classification is statistically significant at a significance level of 10 %, as the confidence interval is $C_0 = (0, F_{0.05} = 2.02)$ and the F-distribution statistical values are $F^* = 45.12 \notin C_0$ for NHN. Accordingly, the null hypothesis stating that all algorithms perform equally in mean ranking is rejected.

On the basis of this rejection, the Holm post hoc test is used to compare all the methods to a given control method. The differences in rankings between the different algorithms and the results of the Holm test for $\alpha = 0.100$ are reported in Table 4, using the corresponding p values. By using this test, it can be seen that the LDA–ELM method significantly outperforms the remaining methods (except the PCA–ELM method).

5.3 Time complexity analysis

The time complexity of the proposed algorithm is analyzed in this subsection. The LDA–PCA–ELM algorithm is composed of three main time-consuming tasks:

- Computation of the LDA vectors.
- Computation of the PCA eigenvectors.
- Determination of the Moore–Penrose pseudo-inverse matrix.

The LDA has $O(nkt + t^3)$ time complexity and requires $O(nk + nt + kt)$ memory, where n is the number of training patterns, k is the number of features and $t = \min(n, k)$ [2]. When both n and k are large, it is difficult to apply LDA. On the other hand, the PCA has $O(nk^2)$ time complexity.

Table 5 Mean training time over all the datasets ($\overline{T}$) of the different methods considered

Computation time			
ELM_{CV}	0.010	P-ELM	0.066
OP-ELM	0.540	EM-ELM	0.239
E-ELM_{CV}	62.606	PCA–ELM	0.016
I-ELM	0.155	LDA–ELM	0.038

Control method LDA–PCA–ELM†: 0.061

Obviously, the proposed approach (LDA–PCA–ELM) is more time-consuming than the baseline algorithm considered (PCA–ELM), taking into account that $n \gg k$ in most of the real-world applications and that it requires the computation also of the LDA vectors. Finally, the inversion of a size $N \times N$ matrix has a complexity of $O(N^2 \log(N))$.

Table 5 reports the average running time (considering also the cross-validation and the test time) of the algorithms considered. All the experiments were run using a common MATLAB framework proposed in [8, 24]. The proposed algorithm was developed and included in the above-mentioned framework. In general, the most efficient algorithm is the base ELM. Despite this, the proposed method achieved a competitive computational time, specially if it is compared to EM-ELM,[2] OP-ELM and ELM_{CV} algorithms.

5.4 Sensitivity analysis

The proposed algorithm relies mainly on only one hyper-parameter: the percentage of variance explained (PVE) by the LDA and PCA projections, used to determine the number of hidden nodes (NHN). In the experimental study, the number of projections considered are those that explain the 90 % of PVE. A study has been performed to analyze the sensitivity of the model, in terms of accuracy (Acc), with respect to this hyper-parameter (percentage of variance explained) to justify the setup decision. The datasets selected for the study are those with the maximum number of possible projections (those with the maximum value of $J - 1 + K$) and they are: the *Card* (52 possible projections), *Ionos* (35 possible projections), *Gene* (122 possible projections) and *German* (62 possible projections) datasets. Several runs of the LDA–PCA–ELM algorithm have been performed for values of the hyper-parameter ranging in the set:

$$\text{PVE} \in \{10, 20, 40, 60, 90, 100\} \tag{13}$$

[2] The computational burden of EM-ELM was seen severely affected by the estimation of the threshold parameter.

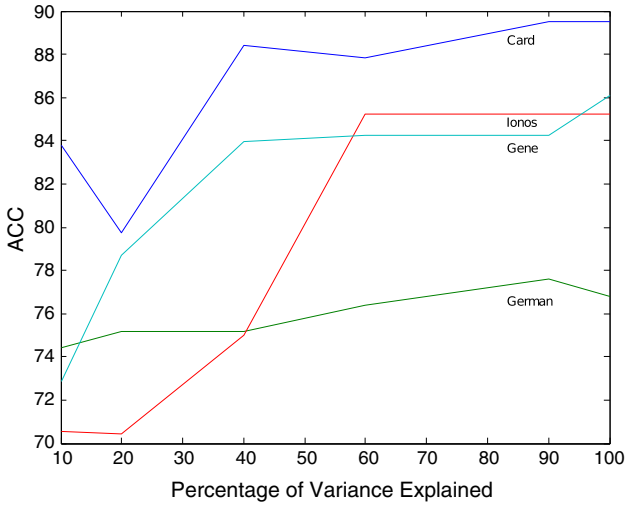

Fig. 5 Hyper-parameter study on Acc for the LDA–PCA–ELM algorithm and the PVE parameter

Results are reported in Fig. 5. As expected, the generalization performance of LDA–PCA–ELM tends to monotonically increase with the PVE. In general, as observed from Fig. 5, the best performance values are obtained with a PVE of 90 % (which is also true for the other datasets). There is only few exceptions to this rule, and all of them would lead to slightly increase the generalization performance by increasing significantly the NHN. For example, for the *Gene* dataset, the model gets a 84.23 % of generalization performance for PVE equal to 90 % and a 86.12 % of generalization performance for PVE equal to 100 %. However, in the first case, NHN = 100 while in the second case, NHN = 122. Therefore, in order to reduce the complexity of the algorithm, the PVE parameter can be directly set to 90 %.

6 Conclusions

In this work, the linear discriminant analysis–principal component analysis–extreme learning machine (LDA–PCA–ELM) algorithm is proposed. As shown in the experiments, the proposal is a fast and robust ELM-based algorithm. The LDA–PCA–ELM algorithm estimates the parameters of the hidden layer according to the eigenvector and the projections generated by the PCA and LDA algorithms, respectively. As in the base ELM method, the weights connecting the hidden and the output layer are estimated analytically according to the Moore–Penrose pseudo-inverse matrix. The proposed algorithm is validated using fifteen well-known classification datasets. The results obtained are statistically compared using the Holm and Friedman tests. This statistical analysis indicates that the proposed approach is a competitive method in time and in number of hidden nodes, improving the previous methods especially in overall accuracy. The main limitation of the methodology proposed with respect to traditional ELM approaches is that ELM algorithms can be applied to several problems including classification and regression problems or even to unsupervised learning problems, while LDA–PCA–ELM algorithm can be applied only to classification problems because LDA needs information about the classification labels.

References

1. Asuncion A, Newman D (2007) UCI machine learning repository. http://www.ics.uci.edu/~mlearn/MLRepository.html
2. Cai D, He X, Han J (2008) Training linear discriminant analysis in linear time. In: IEEE 24th international conference on data engineering (ICDE 2008), pp 209–217
3. Cao J, Lin Z, Huang G (2011) Composite function wavelet neural networks with differential evolution and extreme learning machine. Neural Process Lett 33(3):251–265
4. Castaño A, Fernández-Navarro F, Hervás-Martínez C (2013) PCA-ELM: a robust and pruned extreme learning machine approach based on principal component analysis. Neural Process Lett 37(3):377–392
5. Chen L, Zhou L, Pung HK (2008) Universal approximation and QoS violation application of extreme learning machine. Neural Process Lett 28(2):81–95
6. Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7:1–30
7. Feng G, Huang GB, Lin Q, Gay R (2009) Error minimized extreme learning machine with growth of hidden nodes and incremental learning. IEEE Trans Neural Netw 20(8):1352–1357
8. Fernández-Navarro F, Hervás-Martínez C, Sánchez-Monedero J, Gutierrez PA (2011) MELM-GRBF: a modified version of the extreme learning machine for generalized radial basis function neural networks. Neurocomputing 74(16):2502–2510
9. Fisher R (1938) The statistical utilization of multiple measurements. Ann Eugen 8:376–386
10. Friedman M (1940) A comparison of alternative tests of significance for the problem of *m* rankings. Ann Math Stat 11(1):86–92
11. Hassibi B, Stork DG (1993) Second order derivatives for network pruning: optimal brain surgeon. In: Advances in neural information processing systems 5 (NIPS 1992). Morgan Kaufmann, San Mateo, CA, pp 164-171. ISBN: 1558602747
12. Hochberg Y, Tamhane A (1987) Multiple comparison procedures. Wiley, Hoboken
13. Huang GB, Chen L (2007) Convex incremental extreme learning machine. Neurocomputing 70(16–18):3056–3062
14. Huang GB, Chen L (2008) Enhanced random search based incremental extreme learning machine. Neurocomputing 71(16–18):3460–3468
15. Huang GB, Chen L, Siew CK (2006) Universal approximation using incremental constructive feedforward networks with random hidden nodes. IEEE Trans Neural Netw 17(4):879–892
16. Huang GB, Zhou H, Ding X, Zhang R (2012) Extreme learning machine for regression and multiclass classification. IEEE Trans Syst Man Cybern Part B 42(2):513–529
17. Huang GB, Zhu Q, Siew C (2006) Extreme learning machine: theory and applications. Neurocomputing 70(1–3):489–501

18. Huang GB, Zhu QY, Siew CK (2004) Extreme learning machine: a new learning scheme of feedforward neural networks. In: IEEE international conference on neural networks—conference proceedings, vol 2, pp 985–990
19. Lan Y, Soh YC, Huang GB (2010) Constructive hidden nodes selection of extreme learning machine for regression. Neurocomputing 73(16–18):3191–3199
20. Martínez AM, Kak AC (2001) PCA versus lDA. IEEE Trans Pattern Anal Mach Intell 23(2):228–233
21. Miche Y, Sorjamaa A, Bas P, Simula O, Jutten C, Lendasse A (2010) OP-ELM: optimally pruned extreme learning machine. IEEE Trans Neural Netw 21(1):158–162
22. Miche Y, Sorjamaa A, Lendasse A (2008) OP-ELM: theory, experiments and a toolbox. In: Artificial neural networks—ICANN 2008, lecture notes in computer science, vol 5163, Springer, Berlin, pp 145–154
23. Rong HJ, Ong YS, Tan AH, Zhu Z (2008) A fast pruned-extreme learning machine for classification problem. Neurocomputing 72(1–3):359–366
24. Sánchez-Monedero J, Gutiérrez PA, Fernández-Navarro F, Hervás-Martínez C (2011) Weighting efficient accuracy and minimum sensitivity for evolving multi-class classifiers. Neural Process Lett 34(2):101–116
25. Shrivastava NA, Panigrahi BK, Lim M-H (2014) Electricity price classification using extreme learning machines. Neural Comput Appl 1–10. doi:10.1007/s00521-013-1537-1
26. Storn R, Price K (1997) Differential evolution. A fast and efficient heuristic for global optimization over continuous spaces. J Global Optim 11:341–359
27. Thodberg HH (1991) Improving generalization of neural networks through pruning. Int J Neural Syst 1(04):317–326
28. Uçar A, Demir Y, Güzeliş C (2014) A new facial expression recognition based on curvelet transform and online sequential extreme learning machine initialized with spherical clustering. Neural Comput Appl 1–12. doi:10.1007/s00521-014-1569-1
29. Wong PK, Wong HC, Vong CM, Xie Z, Huang S (2014) Model predictive engine air-ratio control using online sequential extreme learning machine. Neural Comput Appl 1–14. doi:10.1007/s00521-014-1555-7
30. Zhu QY, Qin A, Suganthan P, Huang GB (2005) Evolutionary extreme learning machine. Pattern Recogn 38(10):1759–1763