Neural Process Lett (2013) 37:377–392
DOI 10.1007/s11063-012-9253-x

PCA-ELM: A Robust and Pruned Extreme Learning Machine Approach Based on Principal Component Analysis

A. Castaño · F. Fernández-Navarro · C. Hervás-Martínez

Published online: 7 November 2012

Abstract It is well-known that single-hidden-layer feedforward networks (SLFNs) with additive models are universal approximators. However the training of these models was slow until the birth of extreme learning machine (ELM) "Huang et al. Neurocomputing 70(1–3):489–501 (2006)" and its later improvements. Before ELM, the faster algorithms for efficiently training SLFNs were gradient based ones which need to be applied iteratively until a proper model is obtained. This slow convergence implies that SLFNs are not used as widely as they could be, even taking into consideration their overall good performances. The ELM allowed SLFNs to become a suitable option to classify a great number of patterns in a short time. Up to now, the hidden nodes were randomly initiated and tuned (though not in all approaches). This paper proposes a deterministic algorithm to initiate any hidden node with an additive activation function to be trained with ELM. Our algorithm uses the information retrieved from principal components analysis to fit the hidden nodes. This approach considerably decreases computational cost compared to later ELM improvements and overcomes their performance.

Keywords Principal component analysis · Extreme learning machine · Classification

A. Castaño
Department of Informatics, University of Pinar del Río, Pinar del Río, Cuba
e-mail: adiel@info.upr.edu.cu

F. Fernández-Navarro (✉)
Advanced Concepts Team-European Space Research and Technology Centre (ESTEC),
European Space Agency (ESA), Noordwijk, The Netherlands
e-mail: i22fenaf@uco.es; francisco.fernandez.navarro@esa.int

C. Hervás-Martínez
Department of Computer Science and Numerical Analysis, University of Córdoba, Córdoba, Spain
e-mail: chervas@uco.es

1 Introduction

According to the estimation of Jaeger [15], 95 % of the literature on artificial neural networks (ANNs) is mainly about single-hidden-layer feedforward networks (SLFNs). SLFNs have no side or back connections to connect two nodes. Since the SLFNs were designed, a great number of training techniques have been proposed to fit their parameters and structure. A well known architecture is the multilayer perceptron (MLP) which consists of sigmoid nodes. This MLP is generally trained with the back propagation algorithm (BP).

This common architecture can be trained by various algorithms. The most common algorithms can be classified as being based on gradients and heuristics. These algorithms have common characteristics: difficulty in handling large amounts of data and slow convergence under these circumstances. These characteristics imply that the construction of the SLFNs is often quite slow. The slow building of SLFNs comes from the numerous parameters to be adjusted by means of slow algorithms which must be repeated many times to reach a suitable model. For this reason, the SLFNs tend not to be used as widely as they could be, even considering their overall good performance. To tackle this problem Huang et al. [13] propose an algorithm to determine the parameters of the SLFNs, called the extreme learning machine (ELM). This algorithm reduces the traditional computational time required to train a SLFN by gradient-based approaches. ELM divides the training procedure into two steps: a random configuration of the hidden layer and a linear combination fitting using the Moore-Penrose generalized inverse matrix. As can be observed, this algorithm is really fast and its performance is validated [13].

After the introduction of ELM, some approaches have been implemented to improve the performance of the original version. These approaches address the problem of selecting the number of hidden nodes and the fast fitting of hidden node parameters. To select the number of hidden nodes, different growing and/or pruning techniques have been proposed. Huang et al. [10] defines two variants of incremental extreme learning machine (I-ELM) [5,11], where the hidden nodes are added incrementally until they reach a certain error in the residuals. On the other hand, Miche et al. [18] proposes the optimally-pruned extreme learning machine (OP-ELM) which randomly initializes the hidden node weight, and ranks the resultant features from applying hidden node transformations on the training set.

Until now ELM approaches have attempted to avoid the problems introduced by the random 'fitting' of hidden nodes and the fixed number of initial nodes. Guided by these recent approaches we propose a robust principal component analysis ELM (PCA-ELM) algorithm. Our algorithm is suitable to train any SLFN with hidden nodes that presents linear activation functions. Our proposal estimates the hidden node parameters with the information retrieved from PCA on the training dataset. The output node parameters are determined analytically using the Moore-Penrose generalized inverse.

This paper differs from previous ones because:

- The number of hidden nodes and their weights are deterministically determined taking into consideration the information retrieved from a PCA analysis of the training set.
- The proposed algorithm is not an application of original ELM over the covariates obtained from a PCA analysis, as was proposed in [16].

The paper is organized as follows: a background of ELM is given in Sect. 2. The methodology to optimize the ANN parameters based on ELM and PCA is presented in Sect. 3. Section 4 explains the experiments that have been carried out. Finally, Sect. 5 summarises the conclusions of our work.

2 Background of Extreme Learning Machine

Extreme learning machine (ELM) is an efficient algorithm that determines the output weights of a SLFN using an analytical solution instead of the standard gradient descent algorithm [14]. Neural networks have been used to solve classification problems in several domains ranging from computer vision to bioinformatics.

Traditionally, for a SLFN, all the parameters (weights and biases) for the different layers need to be tuned/learned and there is dependency among the different layers. The gradient descent algorithm is slow and has a high chance of converging to a local minima, and to achieve good generalization performance several iterative steps are necessary.

The ELM scheme proposed by Huang et al. [13] overcomes these problems by randomly assigning weights to the input layers and analytically computing the weights for the output layer using a simple generalized inverse operation. The ELM framework has shown comparable classification performance, improved model representation (less complexity) and faster run times in comparison to support vector machines [23] for the microarray classification problem [24].

Suppose that there are n training patterns $(\mathbf{x}_i, \mathbf{t}_i)$, $i = 1, 2, \ldots, n$, where $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ik})^T$ and $\mathbf{t}_i = (t_{i1}, t_{i2}, \ldots, t_{iJ})^T$ are the i-th input pattern and its target, respectively. Let us denote that $\mathbf{w}_l = (w_{l1}, w_{l2}, \ldots, w_{lk})^T$ is the weight vector connecting the input units to the l-th hidden unit, $l = 1, \ldots, m$, and $\boldsymbol{\beta}^j = (\beta_1^j, \beta_2^j, \ldots, \beta_m^j)^T$ is the weight vector connecting the hidden nodes to the j-th output node. The main goal of the training process is to determine the optimized parameters: $\mathbf{w}_l$, and $\boldsymbol{\beta}^j$, so that they minimize the squared error (SE) function defined by:

$$SE = \sum_{i=1}^{n} \sum_{f=1}^{J} (o_{if} - t_{if})^2, \tag{1}$$

where o_{if} is the estimated output corresponding to the i-th input pattern and the f-th class, which is defined as:

$$o_{if} = \sum_{l=1}^{m} \beta_l^f \phi(\mathbf{x}_i; \mathbf{w}_l), \quad i = 1, 2, \ldots, n \tag{2}$$

where $\phi(\mathbf{x})$ is the activation function.

The minimization process of squared error function in the ELM is performed by using a linear system:

$$\mathbf{H}\boldsymbol{\beta} = \mathbf{T}, \tag{3}$$

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

$$\begin{aligned} \mathbf{H} &= (\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_m) \\ &= \begin{pmatrix} \phi_1(\mathbf{x}_1; \mathbf{w}_1) & \ldots & \phi_m(\mathbf{x}_1; \mathbf{w}_m) \\ \cdots & \cdots & \cdots \\ \phi_1(\mathbf{x}_n; \mathbf{w}_1) & \ldots & \phi_m(\mathbf{x}_n; \mathbf{w}_m) \end{pmatrix}_{n \times m}, \end{aligned} \tag{4}$$

$$\mathbf{T} = (\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_n)_{n \times J}^T, \tag{5}$$

and

$$\boldsymbol{\beta} = (\boldsymbol{\beta}^1, \boldsymbol{\beta}^2, \ldots, \boldsymbol{\beta}^J)_{m \times J}. \tag{6}$$

The ELM learning algorithm proceeds by choosing an activation function $\phi(\mathbf{x})$ and the number of hidden nodes/neurons m. In the first step, arbitrary weights are assigned to the

input weight vectors $\mathbf{w}_l$. The matrix $\mathbf{H}$ is then computed and the output weights $\boldsymbol{\beta}$ are determined as $\hat{\boldsymbol{\beta}} = \mathbf{H}^{\dagger}\mathbf{T}$ where $\mathbf{H}^{\dagger}$ is the Moore-Penrose generalized inverse of the hidden layer output matrix $\mathbf{H}$.

The Moore-Penrose based solution for $\boldsymbol{\beta}$ is shown to be one of the least-square solutions of the general linear system $\mathbf{H}\boldsymbol{\beta} = \mathbf{T}$, and thus can achieve the smallest training error (and not get stuck in local minima as the gradient descent algorithms). It was also shown that the solution is unique, has the smallest norm of weights and hence, good generalization performance [13,3].

Hence, the ELM algorithm for SLFNs can be summarized in these steps: Given a training set $(\mathbf{x}_i, \mathbf{t}_i)$, $i = 1, 2, \ldots, n$, activation function $\phi(\mathbf{x})$ and m hidden neurons:

1. Assign arbitrary input weights $\mathbf{w}_l$, $l = 1, \ldots, m$
2. Calculate the hidden layer output matrix $\mathbf{H}$.
3. Calculate the output weights $\boldsymbol{\beta}$:$\hat{\boldsymbol{\beta}} = \mathbf{H}^{\dagger}\mathbf{T}$.

where $\mathbf{H}$, $\boldsymbol{\beta}$ and $\mathbf{T}$ are as defined before.

The universal approximation capability of ELM has been showed in [10,8,9]. Furthermore, Huang et al. [12] has proved the classification capability of ELM. In fact, Huang et al. [12] shows that ELM generally outperforms SVM in different type of problems.

Recently, pruning-based ELM approaches have been proposed to address the architectural design of the ELM classifier network, since too few/many hidden nodes employed would lead to underfitting/overfitting issues in pattern classification. In [19], the pruned-ELM (P-ELM) algorithm is proposed for designing the ELM classifier network. P-ELM uses statistical methods to measure the relevance of hidden nodes. Beginning from an initially large number of hidden nodes, irrelevant nodes are then pruned by considering their relevance to the class labels.

Another pruning-based ELM approach is the optimally pruned extreme learning machine (OP-ELM) methodology, proposed by Mich et al. [17]. In the OP-ELM methodology, the network is first created using the ELM method, and then, the most relevant nodes are selected using least angle regression (LARS) ranking of the nodes. Finally, the selection of the final model structure is achieved through leave-one-out validation in the last step of OP-ELM that selects the number of neurons.

3 Description of the Methodology

The PCA-ELM is made up of the four main steps summarized in Fig. 1.

3.1 Principal Component Analysis

In the context of high dimensionality (curse of dimensionality), the aim of PCA is to reduce the dimensionality of the data while retaining as much as possible of the variation present in the original dataset. PCA allows us to compute a linear transformation that maps data from a high dimensional space to a lower dimensional space. In order to preserve as much information as possible, the following expression has to be minimized:

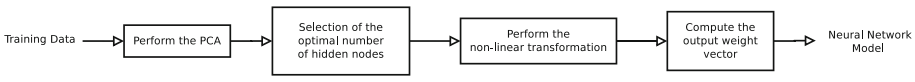

Fig. 1 The four steps of the PCA-ELM algorithm

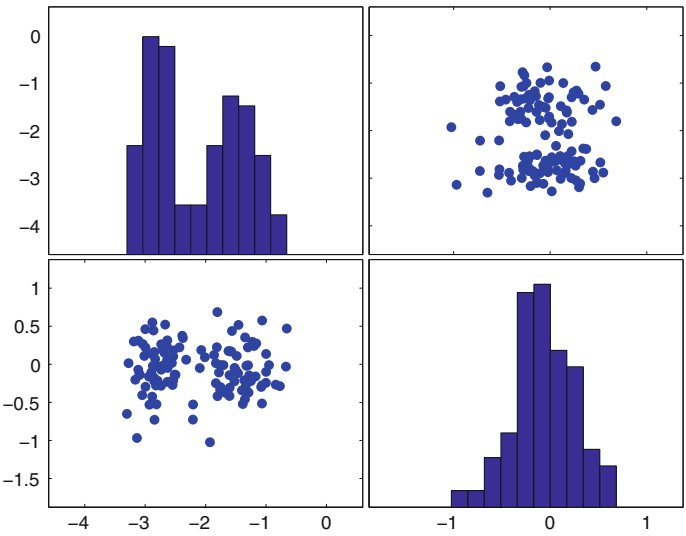

Fig. 2 Data representation of two clusters which boundary points overlaps when they projects in the two axis.

$$\textbf{minimize} \|x - \overline{x}\| \tag{7}$$

The PCA method has a geometrical interpretation since PCA projects the data along the directions where the data varies the most. These directions are determined by the eigenvectors of the covariance matrix corresponding to the largest eigenvalues, and the magnitude of the eigenvalues corresponds to the variance of the data along the eigenvector directions. The main properties of PCA are:

- The new variables are uncorrelated.
- The covariance matrix represents only second order statistics among the vector values.
- Since the new variables are linear combinations of the original variables, it is usually difficult to interpret their meaning.

It is important to note that PCA also has some assumptions:

- Linearity: patterns are assumed to be linear combinations of some basis. There exist non-linear methods such as kernel PCA that solve that assumption [21].
- Principal components with larger associated variances represent interesting structures, while those with lower variances represent noise.
- The principal components are orthogonal. It makes PCA soluble using linear algebra decomposition techniques (like the singular value decomposition (SVD) technique).

The original ELM algorithm indicates and validates the use of neural networks with sigmoid nodes in a unique and randomly fitted hidden layer. This layer transforms the feature space into a new one. PCA is an orthogonal transformation of initial axis into new ones maximizing the variance. To analyze the PCA, two well defined clusters are presented in a 2D space Figure (Fig. 2). As can be observed, the two clusters represent two classes in a classification problem, where the patterns located near the boundaries of the two clusters overlap in their projections over the two axis.

This initial axis may be rotated by means of PCA, obtaining the two new clusters represented in figure (Fig. 3). The new clusters obtained after the axis rotation do not overlap their

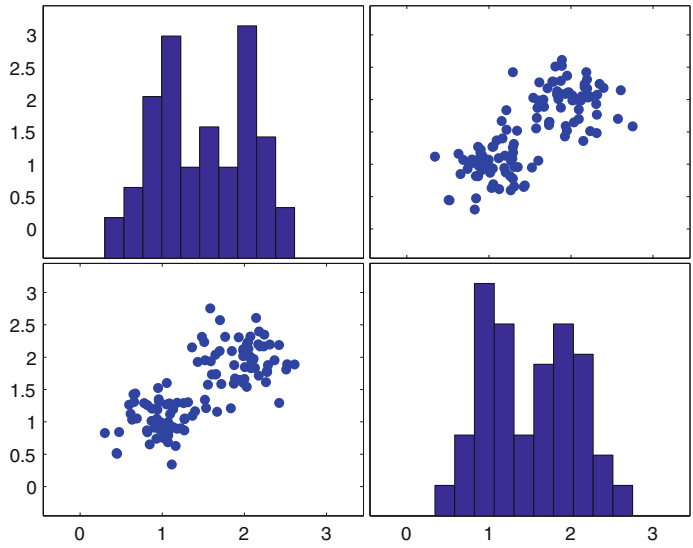

Fig. 3 Data representation after axis rotation produced by PCA over the data of the two clusters

projections in the X-axis. Considering this, the sigmoid node weights may be initialized with the coefficients of the principal components. An important aspect that must be taken into consideration is that variables are initially scaled due to the difficulty that PCA experiences when handling variables with different scales.

Finally, the PCA algorithm can be summarized in these general steps:

1. Pre-treatment of data: scaling.
2. Determination of covariance matrix.
3. Determination of eigenvalues and eigenvectors of covariance matrix (eigenvalue decomposition).
4. Determination of scores: the scores are the data formed by transforming the original data into the space of the principal components..

3.2 Selection of the Number of Hidden Nodes

Hidden nodes in a SLFN transform feature space into another feature space. The original ELM defines the number of nodes as a parameter to be defined. Recent improvements such as OP-ELM suggest the use of methods that rank features (LARS) to define the order of importance of nodes and to remove (pruning) nodes in order of importance. Others increase the number of hidden nodes until reaching a stop criteria (ex. residual error reduction). Our algorithm sets the maximum number of hidden nodes by determining the amount of principal components (orthogonal vectors) necessary to explain the 90 % of the variance in the data.

A comparison with ELM state-of-the-art algorithms indicates that the complexity of the models we obtain is significantly simpler than all models obtained by other algorithms. This deterministic technique to set the number of hidden nodes causes the complexity of the model built on a given dataset to remain constant. This feature is not present in the rest of the ELM algorithms because these algorithms consider the random initialization of hidden nodes.

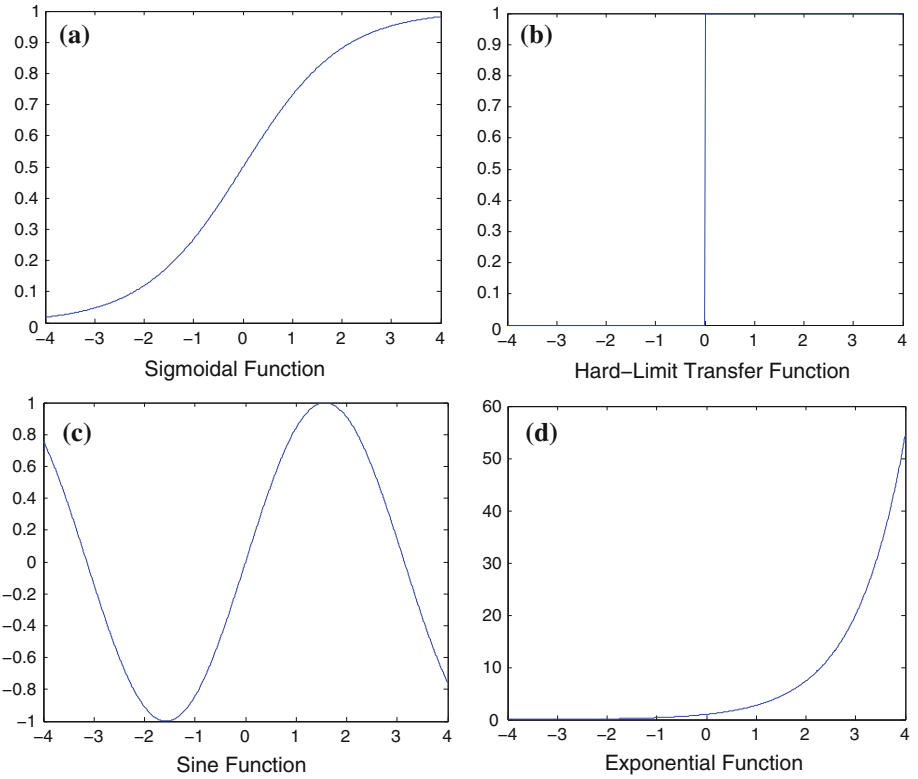

Fig. 4 Graph of the different activation functions used for comparison purposes

3.3 Non-Linear Transformation

Well-known sigmoid nodes are composed of a linear activation function and a sigmoid transfer function. However there are different node types that share the same linear activation function which can be initiated using the PCA approach described in Subsect. 3. Because of this, any 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 SLFNs trained with ELM methodology. Some of the most common nodes with a linear activation function are mentioned in continuation:

– Sigmoidal function (Sig) (Fig. 4a). In this context, Sigmoidal function refers to the special case of the logistic function, defined by the formula:

$$sig(n) = \frac{1}{1 + \exp(-n)} \tag{8}$$

– Hard-limit transfer function (Hardlim) (Fig. 4b). This transfer function returns zero if the argument of the function is less than zero and returns one if the argument is greater than or equal to zero. Hardlim is defined as follows:

$$hardlim(n) = 1 \text{ if } n \geq 0; = 0, \text{ otherwise} \tag{9}$$

– Sine (Sine) (Fig. 4c). This transfer function returns zero if the argument is near the $2K\pi$

– Product unit (PU) (Fig. 4d). This transfer function is a multiplicative model represented as:

$$PU(n) = x_1^{w_1} \times x_2^{w_2} \times \cdots \times x_n^{w_n} = e^{w_1 \times \log(x_1) + w_2 \times \log(x_2) + \cdots + w_n \times \log(x_n)} \quad (10)$$

As can be observed an a priori modification of input space, applying log function to the entire training and testing datasets, makes it possible to define an equivalent node formulated as a exponential function.

3.4 Estimation of the Output Weight Vector

Finally, the estimation of output layer weights is done as proposed by Huang et al. [13], by solving the linear system $\mathbf{H}\boldsymbol{\beta} = \mathbf{T}$ using the the Moore-Penrose generalized inverse. The characteristics of the procedure were previously detailed in Sect. 2.

3.5 Discussion about the Advantages of the PCA-ELM

An analysis of the algorithm being proposed highlights some advantages with respect to previous approaches, beginning with the elimination of the random initiation of hidden nodes. Indeed, up to now, the ELM algorithms initiate the weights of hidden nodes with random values (tuned in certain approaches), but the proposed methodology fits the coefficients using a deterministic and fast method (PCA). This implies that the algorithm does not present a respective high standard deviation of its performance and model complexity as OP-ELM and other algorithms do.

Another critical aspect of ELM algorithms is the selection of the number of hidden nodes. Our approach determines the amount of nodes needed using the information retrieved from the explained variance of the principal components calculated. On the other hand, the algorithms that prune the initial model present a difficulty: that the techniques to select the nodes to be pruned (ex. leave-one out) produce an excessive computational cost with respect to our algorithm, although some pruning algorithms need to rank the nodes with feature rankers. These algorithms produce an ordered list of nodes, the same as our algorithm, but the computational cost of good rankers is much lower than the calculation of explained variance (used by our algorithm).

4 Experiments

The methodology proposed was applied to fifteen datasets taken from the UCI repository [1]. The datasets selected include binary problems and multi-class problems and present different numbers of instances, features and classes. The datasets with their corresponding partitions have been included on a public website[1].

In the first subsection, a description of the datasets and the experimental configuration is given. Then, the proposed method is compared to different basis functions, in order to determine the best performing basis function for the PCA-ELM methodology. Finally, there is a presentation of the proposed model compared to other ELM approaches with different basis functions in the hidden layer.

[1] http://www.uco.es/grupos/ayrna/index.php?lang=en ("Datasets" section)

4.1 Experimental Design

The proposed method (PCA-ELM) is compared to other ELM algorithms using different ANN models. In particular, our proposal is compared to:

- The original extreme learning machine (ELM) [13]. Two different basis functions have been employed for the ELM algorithm: Sigmoidal (ELM (Sig)) and the radial basis function (ELM (RBF)). In the ELM (RBF) algorithm, the centers have been taken randomly from the data points and the widths randomly drawn between percertile 20 % and percentile 80 % of the distance distribution of the input space, as suggested in [17].
  The main difference between ELM (RBF) and ELM with the Radbas basis function is that Radbas applies a standard lineal combination of input variables and the connections between the input and hidden layers, and ELM (RBF) measures the distance of each pattern to its centroid, weighting the final output by its radius.
  The only remaining parameter in this process is the number of nodes (m) in the hidden layer. In the experiments, the m value for ELM (Sig) and ELM (RBF) was experimentally determined by a cross validation procedure applied to the training set, using the values $\{10, 20, \ldots, 20\}$.
- The optimally pruned extreme learning machine (OP-ELM) [17]. Two different basis functions have been employed for the OP-ELM algorithm: the sigmoidal (OP-ELM (Sig)) and the radial basis function (OP-ELM (RBF)). In the OP-ELM (RBF), the values of the centers and widths were initialized in the same way as in the ELM (RBF) algorithm. The number of nodes (m) in the hidden layer in the OP-ELM algorithm is set at 100, since this algorithm prunes the useless neurons from the hidden layer.
- The evolutionary extreme learning machine (E-ELM) [25,2,20] improves the original ELM by using a differential evolution (DE) algorithm. Differential evolution was proposed by Storn and Price [22] and it is known as one of the most efficient evolutionary algorithms. The E-ELM uses DE to select the input weights between input and hidden layers and the Moore-Penrose generalized inverse to analytically determine the output weights between hidden and output layers.
  In the same way as in the ELM algorithm, the most critical parameter in the E-ELM algorithm is the number of the hidden nodes, m. The number of hidden nodes was also adjusted in a similar way, gradually increasing its value by an interval of 10 ($\{10, 20, \ldots, 100\}$) and then selecting the nearly optimal number of nodes based on a cross-validation method. In order to achieve good performance results, the population of the E-ELM is set at 100 individuals to obtain better diversity in the population. Similarly, the maximum number of generations is set at 50. The E-ELM algorithm has been implemented using the Sigmoidal unit as the basis function in the hidden layer.

The experimental design was conducted using a holdout cross validation procedure with $3/4 \cdot n$ instances for the training dataset and $n/4$ instances for the generalization set. To evaluate the stability of the methods, the ELM approaches were run 30 times for each problem. The performance of each model was evaluated using the correct classification rate (C) in the generalisation set, the number of hidden nodes (NHN), and the average time needed to train the model at each iteration, measured in seconds (T).

Furthermore, a simple linear rescaling of the input variables was carried out over the interval $[-1, 1]$ for PCA-ELM (Sig), PCA-ELM (Sine), PCA-ELM (Hardlim) and in the interval $[1, 2]$ for PCA-ELM (PU), with X_i^* being the transformed variables. Finally, all the simulations were carried out in the MATLAB 2009 (R2009a) environment running in an Intel Core

i5, 2.27 GHZ CPU. The source code in MATLAB of the PCA-ELM methodology is freely available upon request to the authors.

4.2 Selection of the Best Performing Basis Function

The aim of this subsection is to determine the best performing non-linear transformation of the principal components. As discussed above, four non-linear transformations have been considered in this study: sigmoidal transformation (PCA-ELM (Sig), product unit transformation (PCA-ELM (PU)), sine transformation (PCA-ELM (Sine)) and a sub-function transformation (PCA-ELM (Hardlim)). Table 1 summarizes the results obtained by the PCA-ELM algorithm using different basis functions. Based on the mean C_{G}, the ranking of each method in each dataset ($R = 1$ for the best performing method and $R = 4$ for the worst one) is obtained and the mean accuracy ($\overline{C}_{\mathrm{G}}$) and the mean ranking ($\overline{R}_{C_{\mathrm{G}}}$) are also included in Table 1. From a quantitative point of view, it can be concluded that the PCA-ELM (Sig) method obtained the best results for eleven datasets in C_{G}. Furthermore, the PCA-ELM (Sig) method yields the best mean ($\overline{C}_{\mathrm{G}} = 80.632\%$) and ranking ($\overline{R}_{C_{\mathrm{G}}} = 1.333$) in C_{G}. The results of T and NHN

Table 1 Statistical results of the PCA-ELM algorithm using different basis functions

	Method (C_{G} (%))			
	Sig	PU	Sine	Hardlim
Hepatitis	79.487	*79.487*	**82.051**	76.923
Heart	77.941	55.882	*77.941*	**79.411**
Haberman	**76.315**	42.105	73.684	*73.684*
Card	**90.017**	43.930	*83.237*	80.346
German	**76.000**	58.400	*72.400*	70.800
Gene	**86.976**	51.702	50.189	*68.600*
Lymph	**81.081**	32.432	*78.378*	75.675
E. coli	*87.882*	42.352	**89.411**	74.117
Yeast	*51.482*	7.277	**52.830**	36.927
PostOp	**81.818**	*72.737*	68.181	72.727
BreastW	**93.714**	65.71	*91.428*	89.714
Ionos	**86.363**	36.363	77.272	*79.545*
Vote	**92.885**	61.467	62.385	*89.908*
Diabetes	**72.875**	67.708	*71.354*	67.187
Breast	**74.647**	*70.422*	64.788	64.788
$\overline{C}_{\mathrm{G}}(\%)$	**80.632**	52.530	73.035	*73.356*
$\overline{R}_{C_{\mathrm{G}}}$	**1.333**	3.533	*2.366*	2.766
	Bonferroni–Dunn test			
	$CD_{\alpha=0.1} = 1.003$, $CD_{\alpha=0.05} = 1.128$			
CD	–	$2.200_{\bullet}$	$1.033_{\circ}$	$1.433_{\bullet}$

The best result is in bold face
Mean and standard deviation (SD) of the accuracy in the generalization set (C_{G} (%)), mean accuracy ($\overline{C}_{\mathrm{G}}$ (%)), mean accuracy ranking ($\overline{R}_{C_{\mathrm{G}}}$), critical difference (CD) values and differences of rankings of the Bonferroni–Dunn tests in C_G with $\alpha = 0.05$ (PCA-ELM (Sig) is the control method)
•, ○: Statistically difference with $\alpha = 0.05$ (•) and $\alpha = 0.1$ (○)

for each basis function and each dataset have not been included in Table 1 since the results of the different basis functions for one dataset are the same in T and NHN.

To determine the statistical significance of the rank differences observed for each method in different datasets, a non-parametric Friedman test [6] has been carried out with the ranking of C_G of the best models as the test variables. The test shows that the effect of the method used for classification is statistically significant at a significance level of 5 %, as the confidence interval is $C_0 = (0, F_{0.05} = 2.80)$ and the F-distribution statistical values are $F^* = 14.20 \notin C_0$ for C_G. Consequently, we reject the null-hypothesis stating that all algorithms perform equally in mean ranking.

Based on this rejection, the Bonferroni–Dunn post-hoc test is used to compare all the classifiers to a given classifier, the control methods. This test considers that the performance of any two classifiers is deemed significantly different if their mean ranks differ by at least the critical difference (CD):

$$CD = q\sqrt{\frac{K(K+1)}{6D}}, \tag{11}$$

where K and D are the number of classifiers and datasets, and the q value is derived from the studentized range statistic divided by $\sqrt{2}$ [4,7]. This test can be computed using Eq. (11) with appropriate adjusted values of q [7].

The results of the Bonferroni–Dunn and Nemenyi tests for $\alpha = 0.10$ and $\alpha = 0.05$ using C_G as the test variable can be seen in Table 1. From the results of these tests, it can be concluded that PCA-ELM (Sig) obtains a significantly better C_G ranking than all the remaining method for $\alpha = 0.10$. Therefore, the Sigmoidal basis function (PCA-ELM (Sig)) is the most suitable approach for the classification task (considering the datasets selected).

As can be observed, the performance of PU models is clearly inferior when compared to others. The most intuitive cause is that PU behaves well for correlated variables and the PCA produce uncorrelated covariates.

4.3 Comparison to Other Related Extreme Learning Approaches

This section focuses on the PCA-ELM (Sig) method, as it is the best in terms of C_G among the different PCA-ELM approaches proposed in this paper. In order to complete the experimental section, this method has been compared to other ELM techniques for classification given in Sect. 4.1.

Tables 2, 3 and 4 present the results obtained with the different ELM techniques, and the result obtained by the PCA-ELM (Sig) method for C_G, NHN and T. The mean accuracy, number of hidden nodes and training time ($\overline{C}_G$, $\overline{NHN}$ and $\overline{T}$) and the mean ranking ($\overline{R}_{C_G}$, $\overline{R}_{NHN}$ and $\overline{R}_T$) are also included in Tables 2, 3 and 4.

From the analysis of the results, it can be concluded (taking C_G into account) from a descriptive point of view that the PCA-ELM (Sig) method obtains the best result for nine datasets and the E-ELM method yields the highest performance for three datasets. Furthermore, the PCA-ELM (Sig) method obtains the best mean ranking ($\overline{R}_{C_G} = 2.533$), followed by the OP-ELM (RBF) method ($\overline{R}_{C_G} = 3.060$), and reports the highest mean accuracy ($\overline{C}_G = 80.632\,\%$), followed by the OP-ELM (RBF) method ($\overline{C}_G = 80.365\,\%$).

Using NHN as the variable test, a descriptive analysis of the results leads to the following remarks: the PCA-ELM (Sig) method obtains the best result for nine out of fifteen datasets, the second best results for two other datasets, and the best mean NHN and mean ranking ($\overline{NHN} = 16.800$ and $\overline{R}_{NHN} = 1.666$).

Table 2 Statistical results of the ELM algorithm using different basis functions and methodologies

	Method (C_G (%))					
	ELM (Sig)	ELM (RBF)	OP-ELM (Sig)	OP-ELM (RBF)	E-ELM	PCA-ELM (Sig)
Hepatitis	76.666 ± 3.770	77.435 ± 3.393	76.923 ± 5.039	76.726 ± 1.872	77.450 ± 5.099	**79.487**
Heart	76.960 ± 2.897	77.156 ± 1.881	*77.402 ± 3.486*	77.039 ± 0.857	77.247 ± 2.812	**77.941**
Haberman	72.587 ± 1.014	69.122 ± 2.461	69.517 ± 2.809	72.894 ± 2.506	*73.315 ± 1.732*	**76.345**
Card	87.341 ± 2.301	87.477 ± 0.892	84.720 ± 1.844	86.897 ± 3.429	*87.527 ± 1.634*	**90.017**
German	72.373 ± 1.559	*74.960 ± 0.947*	72.380 ± 1.955	74.613 ± 1.404	71.320 ± 2.729	**76.000**
Gene	73.892 ± 1.345	85.439 ± 1.000	71.929 ± 1.762	85.569 ± 1.103	*85.851 ± 1.325*	**86.976**
Lymph	77.927 ± 5.412	76.234 ± 1.435	80.000 ± 3.728	**88.108 ± 3.142**	76.316 ± 6.717	*81.081*
E. coli	*86.470 ± 1.478*	85.333 ± 1.189	83.725 ± 2.074	85.803 ± 2.371	84.470 ± 2.130	**87.882**
Yeast	57.124 ± 0.816	57.203 ± 0.776	*55.714 ± 1.628*	57.556 ± 0.803	**57.996 ± 1.715**	51.482
PostOp	77.272 ± 9.399	75.454 ± 5.973	68.030 ± 9.878	*78.939 ± 4.217*	73.272 ± 8.243	**81.818**
BreastW	95.638 ± 0.842	95.676 ± 0.387	**96.019 ± 0.627**	*95.714 ± 0.666*	94.514 ± 1.117	93.714
Ionos	87.575 ± 2.766	89.162 ± 2.214	88.712 ± 2.998	*89.386 ± 2.430*	**89.863 ± 3.160**	86.363
Vote	93.853 ± 0.687	93.984 ± 0.751	92.823 ± 1.326	*94.159 ± 0.701*	**94.220 ± 1.621**	92.885
Diabetes	74.148 ± 1.366	**74.631 ± 1.301**	72.725 ± 1.532	*74.288 ± 1.234*	72.645 ± 1.756	72.875
Breast	68.028 ± 1.883	68.014 ± 2.816	67.840 ± 1.924	67.793 ± 1.873	*68.915 ± 2.554*	**74.647**
$\overline{C}_G$ (%)	78.523	79.152	77.230	*80.365*	78.994	**80.632**
$\overline{R}_{C_G}$	4.133	3.533	4.533	*3.066*	3.200	**2.533**
	Bonferroni–Dunn test					
	$CD_{\alpha=0.1}$ = 1.588, $CD_{\alpha=0.05}$ = 1.759					
CD	$1.600_{\circ}$	1.000	$2.000_{\bullet}$	0.533	0.666	–

The best result is in bold face and the second best result in italics

•, ○: Statistically difference with $\alpha = 0.05$ (•) and $\alpha = 0.1$ (○)

Mean and standard deviation (SD) of the accuracy in the generalisation set (C_G (%)), mean accuracy ($\overline{C}_G$(%)), mean accuracy ranking ($\overline{R}_{C_G}$)

Table 3 Statistical results of the ELM algorithm using different basis functions and methodologies

	Method (NHN)					
	ELM (Sig)	ELM (RBF)	OP-ELM (Sig)	OP-ELM (RBF)	E-ELM	PCA-ELM (Sig)
Hepatitis	20	*10*	28.000 ± 19.546	**6.500 ± 2.330**	20	12
Heart	20	10	23.833 ± 11.867	**6.666 ± 4.011**	20	*8*
Haberman	20	30	17.000 ± 6.512	*9.333 ± 5.832*	20	**3**
Card	60	40	40.000 ± 7.543	**33.166 ± 16.533**	50	**17**
German	70	50	32.833 ± 6.908	**27.166 ± 9.161**	70	*28*
Gene	100	100	**86.500 ± 3.256**	*93.666 ± 8.995*	100	98
Lymph	50	60	*34.666 ± 9.553*	42.833 ± 19.550	40	**17**
E. coli	20	40	76.500 ± 13.655	71.000 ± 15.887	*20*	**5**
Yeast	*50*	60	92.833 ± 6.390	62.166 ± 10.392	60	**7**
PostOp	10	10	*7.666 ± 5.280*	**5.000 ± 0.000**	10	9
BreastW	20	10	22.166 ± 9.067	*11.833 ± 8.952*	20	**5**
Ionos	50	50	44.000 ± 9.040	44.833 ± 8.575	*30*	**17**
Vote	40	50	42.833 ± 9.067	*27.500 ± 8.173*	40	**11**
Diabetes	20	*20*	30.000 ± 11.596	23.666 ± 7.062	30	**6**
Breast	10	10	**6.000 ± 5.477**	*6.500 ± 5.746*	20	9
$\overline{NHN}$	37.333	36.666	38.988	*31.455*	36.666	**16.800**
$\overline{R}_{NHN}$	4.333	4.166	3.933	*2.600*	4.300	**1.666**
	Bonferroni–Dunn test					
	$CD_{\alpha=0.1}$= 1.588, $CD_{\alpha=0.05}$= 1.759					
CD	2.666•	2.500•	2.266•	0.9333	2.633•	–

The best result is in bold face and the second best result in italics
Mean and standard deviation (SD) of the number of hidden nodes (NHN), mean number of hidden nodes ($\overline{NHN}$), mean number of hidden nodes ranking ($\overline{R}_{NHN}$)
•, ○: Statistically difference with $\alpha = 0.05$ (•) and $\alpha = 0.1$ (○)

Taking T into account, the PCA-ELM (Sig) method obtains the best result for seven out of fifteen datasets, and the second best results for eight other datasets. The reason for this is that the PCA-ELM (Sig), in general, needs much fewer hidden nodes than the traditional ELM approach. Thanks to this issue, the PCA-ELM (Sig) could even outperform the original ELM method in efficiency.

Another aspect that is important to point out is that the PCA-ELM method (deterministic method) is far more robust than other ELM approaches, which can be observed on analyzing the values of standard deviation that different ELM approaches generated for each dataset.

It is necessary again to ascertain if there are differences in the mean ranking of C_G, NHN and T, so a procedure similar to that used in the previous subsection has been applied. The non-parametric Friedman test shows that the effect of the method used for classification is statistically significant at a significance level of 5 %, as the confidence interval is $C_0 = (0, F_{0.05} = 2.34)$ and the F-distribution statistical values are $F^* = 2.534 \notin C_0$ for C_G, $F^* = 7.562 \notin C_0$ for NHN, and $F^* = 160.721 \notin C_0$ for T. Consequently, we reject the null-hypothesis stating that all algorithms perform equally in mean ranking.

On the basis of this rejection, the Bonferroni–Dunn post-hoc test is used to compare all the methods to a given control method. The differences in rankings between the different

Table 4 Statistical results of the ELM algorithm using different basis functions and methodologies

	Method (T)					
	ELM (Sig)	ELM (RBF)	OP-ELM (Sig)	OP-ELM (RBF)	E-ELM	PCA-ELM (Sig)
Hepatitis	**0.002 ± 0.001**	0.005 ± 0.003	0.244 ± 0.013	0.250 ± 0.015	6.346 ± 1.241	*0.002*
Heart	*0.002 ± 0.001*	0.011 ± 0.004	0.276 ± 0.028	0.272 ± 0.021	9.316 ± 1.060	**0.001**
Haberman	*0.002 ± 0.002*	0.013 ± 0.002	0.276 ± 0.035	0.261 ± 0.018	7.295 ± 1.653	**0.001**
Card	**0.011 ± 0.001**	0.074 ± 0.004	0.452 ± 0.062	0.458 ± 0.074	56.007 ± 8.327	*0.014*
German	**0.018 ± 0.003**	0.151 ± 0.007	0.570 ± 0.051	0.761 ± 0.067	84.483 ± 16.040	*0.020*
Gene	**0.078 ± 0.002**	1.203 ± 0.032	2.797 ± 0.078	3.493 ± 0.021	578.122 ± 66.898	*0.168*
Lymph	**0.005 ± 0.002**	0.007 ± 0.001	0.114 ± 0.080	0.126 ± 0.062	12.300 ± 1.892	*0.009*
E. coli	**0.002 ± 0.002**	0.018 ± 0.003	0.404 ± 0.042	0.402 ± 0.053	7.086 ± 0.308	*0.002*
Yeast	*0.015 ± 0.004*	0.234 ± 0.007	1.438 ± 0.063	1.692 ± 0.188	86.748 ± 18.812	**0.003**
PostOp	**0.001 ± 0.001**	0.001 ± 0.001	0.051 ± 0.002	0.056 ± 0.003	2.187 ± 0.334	*0.002*
BreastW	*0.003 ± 0.003*	0.039 ± 0.001	0.110 ± 0.010	0.174 ± 0.014	8.471 ± 1.412	**0.002**
Ionos	*0.006 ± 0.001*	0.025 ± 0.002	0.313 ± 0.042	0.303 ± 0.023	14.681 ± 3.124	**0.005**
Vote	*0.005 ± 0.003*	0.024 ± 0.008	0.337 ± 0.012	0.353 ± 0.047	23.046 ± 2.774	**0.003**
Diabetes	*0.003 ± 0.002*	0.062 ± 0.062	0.442 ± 0.015	0.496 ± 0.036	35.213 ± 7.984	**0.002**
Breast	**0.002 ± 0.001**	0.010 ± 0.004	0.290 ± 0.038	0.271 ± 0.023	7.794 ± 0.291	*0.006*
$\overline{T}$	**0.010**	0.125	0.540	0.624	62.606	*0.016*
$\overline{R}_T$	**1.566**	2.833	4.333	4.666	6.000	*1.600*
	Bonferroni–Dunn test					
	$CD_{\alpha=0.1}$= 1.588, $CD_{\alpha=0.05}$= 1.759					
CD	0.033	1.233	2.733$_\bullet$	3.066$_\bullet$	4.400$_\bullet$	-

The best result is in bold face and the second best result in italics
Mean and standard deviation (SD) of the training time (T), mean training time ($\overline{T}$), mean training time ranking ($\overline{R}_T$)
•, ○: Statistically difference with $\alpha = 0.05$ (•) and $\alpha = 0.1$ (○)

algorithms and the results of the Bonferroni–Dunn test for $\alpha = 0.1$ and $\alpha = 0.05$ can be seen in Tables 2, 3 and 4, using the corresponding critical values. By using this test, it can be seen that the PCA-ELM (Sig) method: significantly outperforms the ELM (Sig) and OP-ELM (Sig) methods for $\alpha = 0.10$ using C_G as the test variable; outperforms the ELM (Sig), ELM (RBF), OP-ELM (Sig), and E-ELM methods for $\alpha = 0.05$, using NHN as the test variable; and, finally, outperforms OP-ELM (Sig), OP-ELM (RBF) and E-ELM methods using T as the test variable for $\alpha = 0.05$, which justifies the proposal.

5 Conclusions

The proposed PCA-ELM algorithm is a fast and robust ELM-based algorithm. Our proposal estimates the hidden node parameters with the information retrieved from PCA on the training set while the output node parameters are determined using the Moore-Penrose generalized inverse. Our algorithm was validated by experimentation using fifteen well-known datasets. The results obtained were statistically compared using the Bonferroni–Dunn, Nemenyi and

Friedman tests. This statistical analysis indicates that our approach improves on previous ones. The most important aspect introduced is the elimination of the random initiation of hidden neurons.

Acknowledgements The authors would like to thank Dr. Huang GB who generously provided us with the source code of the ELM–RBF. This work has been partially subsidized by the TIN2011-22794 project of the Spanish Inter-Ministerial Commission of Science and Technology (MICYT), FEDER funds and the P2011-TIC-7508 project of the "Junta de Andalucía" (Spain). This work has been partially subsidized with the "Doctoral Training on Softcomputing" project subsidized by the Junta de Andalucía, the Ibero-American University Postgraduate Association (AUIP) and the Ministry of Higher Education of the Republic of Cuba.

References

1. Asuncion A, Newman D (2007) UCI machine learning repository. http://www.ics.uci.edu/~mlearn/MLRepository.html. Accessed 8 Sept 2007
2. 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
3. Chen L, Zhou L, Pung HK (2008) Universal approximation and qos violation application of extreme learning machine. Neural Process Lett 28(2):81–95
4. Dunn OJ (1961) Multiple comparisons among means. J Am Stat Assoc 56:52–56
5. 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
6. Friedman M (1940) A comparison of alternative tests of significance for the problem of *m* rankings. Ann Math Stat 11(1):86–92
7. Hochberg Y, Tamhane A (1987) Multiple comparison procedures. Wiley, New York
8. Huang GB, Chen L (2007) Convex incremental extreme learning machine. Neurocomputing 70(16–18): 3056–3062
9. Huang GB, Chen L (2008) Enhanced random search based incremental extreme learning machine. Neurocomputing 71(16–18):3460–3468
10. Huang GB, Chen L, Siew CK (2006) Universal approximation using incremental constructive feedforward networks with random hidden nodes. IEEE Trans Neural Netw 17:4
11. Huang GB, Li MB, Chen L, Siew CK (2008) Incremental extreme learning machine with fully complex hidden nodes. Neurocomputing 71(4–6):576–583
12. Huang GB, Zhou H, Ding X, Zhang R (2012) Extreme learning machine for regression and multiclass classification. IEEE Trans Syst Man Cybern B 42(2):513–529
13. Huang GB, Zhu Q, Siew C (2006) Extreme learning machine: theory and applications. Neurocomputing 70(1–3):489–501
14. Huang GB, Zhu QY, Siew CK (2004) Extreme learning machine: a new learning scheme of feedforward neural networks. IEEE Int Conf Neural Netw Conf Proc 2:985–990
15. Jaeger H (2001) The "echo state" approach to analysing and training recurrent neural networks. GMD Rep 148:1435–2702
16. Kim J, Shin H, Lee Y, Lee M (2007) Algorithm for classifying arrhythmia using extreme learning machine and principal component analysis. In: 29th Annual international conference of the IEEE, engineering in medicine and biology society, 2007. EMBS, New York, pp 3257–3260
17. Mich 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
18. 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
19. Rong HJ, Ong YS, Tan AH, Zhu Z (2008) A fast pruned-extreme learning machine for classification problem. Neurocomputing 72(1–3):359–366
20. 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
21. Schlkopf B, Smola AJ, Müller KR (1999) Kernel principal component analysis. In: Advances in kernel methods: support vector learning. MIT Press, Cambridge, pp 327–352

22. Storn R, Price K. (1997) Differential evolution: a fast and efficient heuristic for global optimization over continuous spaces. J Glob Optim 11:341–359
23. Vapnik VN (1999) The nature of statistical learning theory. Springer, Berlin
24. Zhang R, Huang GB, Sundararajan N, Saratchandran P (2007) Multicategory classification using an extreme learning machine for microarray gene expression cancer diagnosis. Comput Biol Bioinform IEEE/ACM Trans 4(3):485–495
25. Zhu QY, Qin A, Suganthan P, Huang GB (2005) Evolutionary extreme learning machine. Pattern Recognit 38(10):1759–1763