EXPLOITING DIVERSITY OF NEURAL NETWORK ENSEMBLES BASED ON EXTREME LEARNING MACHINE

Pedro J. García-Laencina[*], *José-Luis Roca-González*[*], *Andrés Bueno-Crespo*[†] *and José-Luis Sancho-Gómez*[‡]

Abstract: Extreme learning machine (ELM) is an emergent method for training single hidden layer feedforward neural networks (SLFNs) with extremely fast training speed, easy implementation and good generalization performance. This work presents effective ensemble procedures for combining ELMs by exploiting diversity. A large number of ELMs are initially trained in three different scenarios: the original feature input space, the obtained feature subset by forward selection and different random subsets of features. The best combination of ELMs is constructed according to an exact ranking of the trained models and the useless networks are discarded. The experimental results on several regression problems show that robust ensemble approaches that exploit diversity can effectively improve the performance compared with the standard ELM algorithm and other recent ELM extensions.

Key words: *Single layer feedforward neural networks, extreme learning machine, ensemble, regression*

Received: February 17, 2011
Revised and accepted: September 20, 2013

1. Introduction

Single layer feedforward neural networks (SLFNs), as universal approximation models, have shown their usefulness in many research areas over the last decades [1]. Nevertheless, its main drawback is that their traditional training methods do not provide an efficient design and implementation because many parameters have to be properly tuned by slow (often gradient-based) algorithms for achieving a good

[*]P. J. García-Laencina – Corresponding author, J. L. Roca-González
Centro Universitario de la Defensa de San Javier (MDE-UPCT), SPAIN, E-mail: pedroj.garcia@cud.upct.es, jluis.roca@cud.upct.es
[†]A. Bueno-Crespo
Universidad Católica San Antonio de Murcia, SPAIN, E-mail: abueno@pdi.ucam.edu
[‡]J-L. Sancho-Gómez
Universidad Politécnica de Cartagena, SPAIN, E-mail: josel.sancho@upct.es

enough model. Besides, the training stage has to be repeated several times in order to select the model structure, for example the selection of hidden layer size. In order to address it, the extreme learning machine (ELM) has been recently proposed as an emergent method for training single layer feedforward neural networks (SLFNs) [2]. Its main advantages are extremely fast training speed, easy implementation and good generalization performance [2–4]. This method assigns random values to the hidden neuron parameters, i.e., weights and biases for multi-layer perceptron (MLP) networks or the centers and widths for radial basis function (RBF) networks and, then, the output weights are computed using the Moore-Penrose generalized inverse. Nowadays, ELM have attracted many attentions and it has been successfully used in several applications, such as gene classification [5], image watermarking [6] or survey analysis [7]. Although the advantages of the ELM algorithm are clear, it has been found that the obtained networks tend to require more hidden nodes than traditional training methods because the random assignment of parameters may introduce inappropriate values for them. Numerous growing and pruning improvements have been proposed in order to obtain a more compact network and to avoid an extensive search of the optimal hidden layer size. It has been shown that, in general, the growing techniques are more sensitive to initial conditions than pruning procedures and, then, they can be trapped in a sub-optimal solution. From the different pruning approaches, the Optimally Pruned ELM (OP-ELM) methodology stands out as a robust and fast technique for automatic design of ELM networks [8].

Other enhancements have also been obtained by combining the outputs of L different ELMs, i.e, a set of ELM networks jointly solves the problem. An ensemble system tries to exploit the individual different behaviours of the L models to improve the performance of a single model. The simplest way to construct an ensemble is to average the predictions of L individual ELM models (with the same number of hidden neurons and different weight initializations) [9], but other combination schemes are also possible. Liu and Wang [10] present an ensemble ELM method that uses a cross-validation scheme in order to increase the diversity between the individual models, but they use the same hidden layer size for all networks and do not perform a selection of appropriate candidates for the ensemble. Van Heeswijk et al. [11, 12] introduce an adaptive ensemble of random ELM networks where its ensemble weights are iteratively updated according to a predefined learning rate. In order to discard inaccurate individual models, Chen et al. [13] measure the Pearson correlation coefficient between each ELM network (with a predefined hidden layer size) and the ensemble output and, then, compute its product with the mean square error (MSE) of each network. These product values are used to discard networks with large MSE and small diversity.

It is widely known that the success of an ensemble method lies in the diversity among the individual SLFNs [14–16]. Following the basis of the OP-ELM algorithm, this work presents robust combination procedures for ELM networks by exploiting diversity. These approaches firstly train a large set of L neural networks using the OP-ELM algorithm that gives neural architectures with different sizes for each different random initialization. Once the L models are trained, the ensemble is constructed through linear combination. In order to discard useless ELM models from the ensemble system, the L networks are ranked using the MRSR algorithm.

It is important to remark that this ranking is exact for linear problems [17]. Then, following the basis of the OP-ELM algorithm, the L^* models (with $L^* \leq L$) are chosen as those that give a better linear combination for the target data according to the PRESS (PREdiction Sum of Squares) statistic [18]. In this letter, we introduce three different alternatives to construct the L initial models: (1) to train all networks with the same initial input features; (2), to train all networks with a subset of input variables which is obtained by forward feature selection; and (3), to train each ELM network with different random subsets of features. The remaining of this work is organized as follows. Section 2. gives a brief introduction for both the standard ELM and the OP-ELM algorithms. The enhanced ensemble approaches for ELM networks are described in Section 3. and Section 4. gives experimental results on several well-known regression problems. Finally, the main conclusions and future works are reported in Section 5.

2. Preliminaries

Let us assume a learning task defined by N training input vectors $(\mathbf{x}_j, t_j)$, where $\mathbf{x}_j = [x_{j1}, x_{j2}, \ldots, x_{jn}]^T \in \mathbb{R}^n$ and $t_j \in \mathbb{R}$. This dataset is learned using a SLFN with M hidden neurons and activation function $f(\cdot)$, which is mathematically modeled as

$$o_j = \sum_{i=1}^{M} \beta_i f(\mathbf{w}_i^T \mathbf{x}_j + b_i) = \sum_{i=1}^{M} \beta_i h_{ij}, \quad j = 1, ..., N; \tag{1}$$

where $\mathbf{w}_i = [w_{i1}, w_{i2}, \ldots, w_{in}]^T$ is the weight vector connecting the i-th hidden neuron and the input units, $\boldsymbol{\beta} = [\beta_1, \beta_2, \ldots, \beta_M]^T$ is the output weight vector and b_i is the bias parameter of the i-th hidden neuron. Besides, h_{ij} is the i-th hidden output for $\mathbf{x}_j$. Note that these h_{ij} are linearly combined to compute the network outputs, o_j. One of the most popular activation function is the sigmoid and, in this case, the SLFN is widely known as Multi-Layer Perceptron (MLP). The learning objective is that $o_j \approx t_j$ with a good generalization capability.

The standard ELM method is based on the concept that if $\mathbf{w}_i$ and b_i are randomly assigned, then a SLFN can be considered as a linear system and the output weight vector, $\boldsymbol{\beta}$, can be analytically determined through simple generalized inverse operation of the hidden layer output matrices [2, 4]. Thus, given $\mathbf{w}_i$ and b_i, the training of a SLFN is simply equivalent to find a LS (Least-Squares) solution, $\hat{\boldsymbol{\beta}}$, for the linear system

$$\mathbf{H}\boldsymbol{\beta} = \mathbf{T}, \tag{2}$$

where

$$\mathbf{H} = \begin{bmatrix} f(\mathbf{w}_1^T \mathbf{x}_1 + b_1) & \ldots & f(\mathbf{w}_M^T \mathbf{x}_1 + b_M) \\ \vdots & \ldots & \vdots \\ f(\mathbf{w}_1^T \mathbf{x}_N + b_1) & \ldots & f(\mathbf{w}_M^T \mathbf{x}_N + b_M) \end{bmatrix}_{N \times M} \tag{3}$$

From [4], the solution is given by $\hat{\boldsymbol{\beta}} = \mathbf{H}^{\dagger}\mathbf{T}$, where $\mathbf{H}^{\dagger}$ is the Moore-Penrose generalized inverse of $\mathbf{H}$, [19]. ELM is much simpler and faster than traditional learning algorithms for SLFN. Besides, this algorithm can be also used for training Radial Basis Function (RBF) networks [3]. It is should be noted that the standard

ELM algorithm requires a cross-validation (CV) technique (such as 10-fold CV) to choose an appropriate value for M.

2.1 OP-ELM

OP-ELM (Optimally Pruned-ELM) [8] improves ELM by pruning inappropriate hidden neurons using an exact and efficient ranking criterion, which is based on the MRSR (MultiResponse Sparse Regression) algorithm and the PRESS (PREdiction Sum of Squares) statistic. The OP-ELM algorithm provides extremely fast accurate models and achieves roughly the same level of accuracy as that of other well known machine learning methods [8], such as Support Vector Machines (SVM) or Gaussian Processes (GP). In what follows we outline the three main stages of the OP-ELM method.

1. *Random initialization of a large SLFN.* This first step is performed using ELM for a large enough number of neurons, M. From a practical point of view, it is advised to set the hidden layer size clearly higher than the number of features: $M >> n$. While the ELM methodology uses a single type of activation function or kernel (for example, sigmoid functions), in the OP-ELM approach three types of functions (sigmoid, gaussian and linear) can be used in combination for better robustness and generality [8]. For sigmoid functions, the weights are randomly assigned by following a uniform distribution in an interval that covers the input data range (previously whitened: normalized to zero mean and unit variance). Whereas, the gaussian kernels have their centers taken randomly from data points and widths randomly drawn between 20% and 80% percentiles of the distance distribution of the input data [8].

2. *Ranking of hidden units.* In the second stage, MRSR [17] is applied for sorting the hidden neurons according to their accuracy. MRSR is in essence an extension of the well-known Least Angle Regression (LARS) algorithm and, thus, it is a variable ranking method [20], rather than a selection one. It must be remarked that the obtained ranking by MRSR is exact for linear problems [17]. Since the output unit of an ELM network is linear with respect to the randomly initialized hidden units, the MRSR algorithm gives an exact ranking of neurons [8].

3. *Selection of hidden units.* Once the ranking of the hidden neurons has been obtained and $\mathbf{H}$ has been consequently sorted, the best M^* neurons for the ELM model are chosen using the PRESS statistic, which provides an exact estimation of the LOO error for linear models [18]. For the OP-ELM method, it is as follows [21]:

$$E_{LOO} = \frac{1}{N} \sum_{j=1}^{N} \left(\frac{t_j - o_j}{1 - hat_{jj}} \right)^2, \tag{4}$$

where t_j and o_j are respectively the j-th training target and its estimation by the trained machine, and hat_{jj} denotes the j-th value of the HAT-matrix, which is the matrix which transforms $\mathbf{T}$ into $\mathbf{O}$:

$$\mathbf{O} = \mathbf{H}\beta = \mathbf{H}\mathbf{H}^{\dagger}\mathbf{T} = \mathbf{H}(\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{T} = \mathrm{HAT} \cdot \mathbf{T}. \tag{5}$$

Then, HAT-matrix requires to compute $\mathbf{H}^{\dagger}$ during the training stage and its computation can be reused for estimating the LOO error. Note that (4) only needs the diagonal of the HAT-matrix, which can be easily obtained by the the row-wise dot-product between $\mathbf{H}$ and $\mathbf{H}^{\dagger}$. The optimal number of neurons can be found by estimating the LOO error for different numbers of nodes (already ranked by accuracy using MRSR) and selecting the number of neurons (M^*, with $M^* \leq M$) such that minimizes the LOO error:

$$M^* = \underset{k \in \{1,\ldots,M\}}{\arg\min} \; E_{LOO,k}, \tag{6}$$

where $E_{LOO,k}$ denotes the LOO error computed using (4) with a SLFN composed of $k \leq M$ hidden nodes. Note that the hidden nodes are incrementally incorporated to the SLFN according to the order given by MRSR.

On the contrary that the standard ELM algorithm, OP-ELM does not need to split up the learning set into training and validation subsets because it directly determines the optimal hidden layer size by computing the LOO error in a fast way using the PRESS statistic. OP-ELM provides a fast and valuable tool for the architecture design of ELM-based neural networks [8]. Note that OP-ELM provides a different network with different hidden layer size for each initialization of the input weights.

3. Enhanced combination of ELM networks

The use of multiple models may often improve the performance of an individual model [14–16]. Rather than generating a set of different models and using the single 'best' model in isolation, a combination of these networks would exploit, rather than ignore, the information contained in the redundant models. Such combination of networks are sometimes called *committees* or *ensembles*. Two main issues arise when considering the committee based approach: first, the creation of models to be combined in an ensemble; and second, the method by which the outputs of the members are combined [14].

This section introduces robust ensembles of ELM networks. In particular, we consider a linear combination scheme of L OP-ELM networks, $\hat{o} = \sum_{l=1}^{L} \lambda_l o_l$, where $\hat{o}$ is the ensemble output and λ_l is l-th ensemble weight. Thus, given the N input vectors, the following problem must be solved:

$$\sum_{l=1}^{L} \lambda_l o_{jl} = t_j, \quad j = 1, \ldots, N; \tag{7}$$

where o_{jl} is the prediction of the l-th model given $\mathbf{x}_j$ and λ_l is the ensemble weight vector connecting the l-th model and the output units. The ensemble weights can be obtained by solving (7) with the LS method: $\hat{\boldsymbol{\lambda}} = \mathbf{O}^{\dagger}\mathbf{T}$, where $\mathbf{O}^{\dagger}$ is the Moore-Penrose generalized inverse of the network output matrix $\mathbf{O}$, whose l-th column is the l-th network's output vector for the N input patterns. In this work, this ensembling procedure is known as LSC-OPELMs (Least-Squares Combination of OP-ELMs).

Instead of using all the L models, we attempt to choose the L^* networks (with $L^* \leq L$) whose linear combination provides better generalization capability by exploiting diversity of the chosen ELMs. From the basis of OP-ELM, a direct and exact linear combination of the L^* chosen SLFNs is achieved using the MRSR algorithm and the PRESS statistic, and, then, the useless networks are effectively discarded. Moreover, in order to exploit diversity, this work presents different alternatives by varying the input feature space. On the contrary than other ensembling methods, proposed approach has more diversity and independence in the individual ELM networks because they are trained with different hidden layer sizes and subsets of input features. The next two subsections respectively describe the initial construction of the ELM candidates for the ensemble and, then, the ensemble design stage.

3.1 Initial construction of ELM networks

The initial stage is to construct L different SLFNs using the OP-ELM algorithm and we present three different alternatives (see Fig. 1):

1. First, all SLFNs are trained using the n input variables that defines the dataset. In this case, the diversity is given by the different networks (with different hidden layer sizes) obtained with OP-ELM for each input weight initialization. It is known as OC-OPELMs (Optimal Combination of ELMs) and it is an improvement of LSC-OPELMs and OP-ELM.

2. Feature selection is an useful approach where the irrelevant inputs, which can be harmful for modeling the target data, are discarded. Following this, a second alternative is to obtain a subset of input variables through forward feature selection, in which features are sequentially added to an empty candidate set until the addition of further features does not decrease the LOO error. Several repetitions (L) of the OP-ELM algorithm have to be done for averaging the LOO error in each possible feature subset. Once the convergence is reached, we obtain L different SLFNs (with different hidden layer sizes) which have been trained using OP-ELM in the best subset of inputs. It is known as OC-OPELM-FFS (Optimal Combination of OP-ELMs based on Forward Feature Selection).

3. In order to obtain an ensemble system that outperforms the individual networks, it is critical that there should be enough diversity among the L models. For increasing the diversity, the third alternative of this work is to consider different random subsets of features during the OP-ELM training of each network. Thus, we obtain L different SLFNs which are composed of different number of hidden units and input features. It is known as OC-OPELMs-RFS (Optimal Combination of OP-ELMs based on Random Feature Subsets). In this case, it may be required a enough large set of SLFN candidates with random input subsets because inaccurate models can be obtained from certain feature subsets. Due to this, high-dimensional datasets require a larger number of random networks. Note that these inappropriate models (i.e., inappropriate random feature subsets) will be automatically omitted in the next pruning stage.

3.2 Ensemble design

Once the L different ELMs have been trained, their network outputs are previously ranked using the MRSR method [17] and the network output matrix is consequently ordered. Then, according to this work, the better L^* models are selected to minimize the LOO error, which can be exactly measured with the PRESS statistic [18]. The inappropriate and useless models are discarded from the ensemble. This ensemble pruning reduces the storage needs, speeds up the operation stage and has the potential of improving the performance of the single neural architecture. We would like to remark that the obtained ensemble system is optimal. Note that this stage is the same for the above three alternatives.

4. Experimental results

First of all, a toy example is used to illustrate the performance of the standard ELM algorithm on a simple problem that can be plotted. In this case, the ELM method is compared with the standard back-propagation (BP) algorithm for training SLFNs. In particular, MLP schemes are trained using the scaled conjugate gradient (SCG) method that is a computationally efficient and widely used algorithm to train artificial neural networks. Besides, we also evaluate the training of FFNN with ELM and BP in a real-world regression problem. These first simulations are done for a predefined hidden layer size. Once the advantages of ELM are shown, in the next subsection, we evaluate the different ensemble procedures based on ELM, described in Section 3.

4.1 First evaluation of ELM: Toy problem

First, a set of 1000 cases are generated following a sum of sines. This gives a one-dimensional dataset which is known as *Sinusoidal*, and it is divided into two subsets with equal number of samples (500 cases) for training and test stages. In order to compare BP and ELM training approaches, both of them are assigned the same number of hidden neurons (from 1 to 30 neurons) with sigmoid activation function. For each hidden layer size, 20 simulations (i.e., 20 different weight initializations) of the standard MLP training (BP with SCG optimization, BP-SCG) and the ELM algorithm are performed. In the case of MLP training, the maximum number of training iterations is established to 500 and, in order to avoid overfitting, the MLP learning is early stopped when the training error gets saturated [1]. Fig. 2 shows the evolution of the training time (in seconds) with respect to the hidden layer size with both training procedures: MLP -line with squares- and ELM -line with circles-. Note that this plot uses logarithmic scaling on the vertical axis. As it can be observed in this figure, the ELM algorithm runs around 100 times faster than the BP-SCG training for MLP, without considering that C executable environment may run much faster than MATLAB environment. It is straight-forward to see that the training time increases as the hidden layer size is larger.

Fig. 3 shows the Root Mean Square Error (RMSE) in the test set with different number of neurons in the gradient-based MLP training and the ELM method. Results are average of 20 simulations. From this figure, we can observe that the

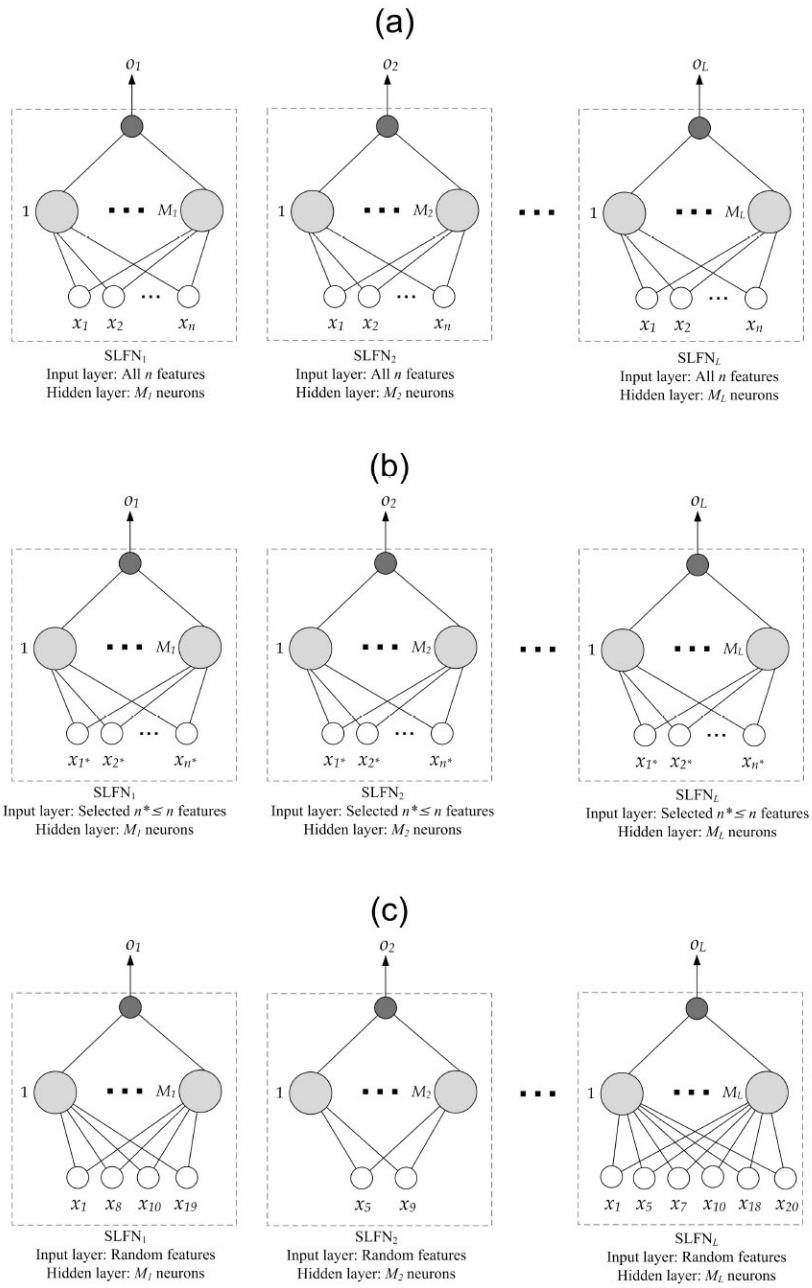

Fig. 1 *Initial construction of L SFLNs for the ensemble using: (a) OC-OPELM; (b) OC-OPELM-FFS; and (c) OC-OPELM-RFS. Diversity in the ensemble is exploited by using different input data spaces: all input attributes in (a); the sequential forward-selected features in (b); and random features for each SLFN in (c). Note that each SLFN is trained using OP-ELM with different weight initializations and, then, different hidden layer is achieved for each individual network.*

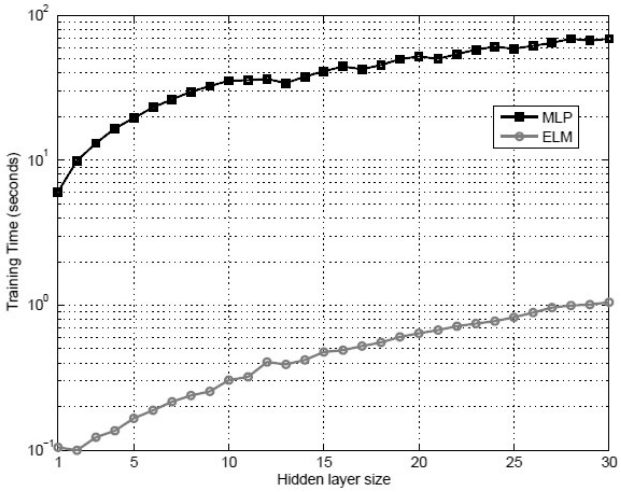

Fig. 2 *Training time (in seconds) vs. Number of hidden neurons in a SLFN when the Sinusoidal problem is learned using the gradient-based MLP training approach and the ELM methodology.*

standard MLP training performs better than the ELM algorithm when the hidden layer size is small. It is due to the fact that the input weights are not updated in the ELM method and, thus, this procedure is quite sensitive to input weights initialization with fewer hidden nodes. It means that ELM usually needs higher number of hidden neurons due to the random determination of the input weights and hidden biases. Note that it can be a limitation in real applications with hardware implementations, but it can be solved using an efficient pruning design. We would like to remark that, in general, pruning approaches work better than growing procedures for ELM network because the ELM algorithm requires an enough large number of neurons and, hence, small ELM models lead to underfit the target data. The generalization performance obtained by the ELM algorithm is very close (even better) to the generalization performance of the standard MLP training when the number of hidden neurons is higher than 15, and of course with less training time.

In order to give an illustrative overview, Fig. 4 plots several examples of the obtained models (MLP in blue dashed line and ELM in red continuous line) fitting the input data points for different number of neurons (2, 8, 15 and 25). It can be seen from this figure that the smaller networks (2 and 8 hidden neurons) trained with the standard MLP approach fit clearly better than the ELM schemes. Indeed, the global fitting of the ELM model with 2 hidden neurons is very inaccurate. Whereas, when the hidden layer size is large enough (15 and 25 neurons), the ELM models, and the MLP networks, fit very well to the input data.

4.2 Evaluation of neural network ensembles based on ELM

Now, we show the experimental results on six well-known regression problems, which are summarized in Tab. I. The first four datasets (*Friedman*, *Elevators*,

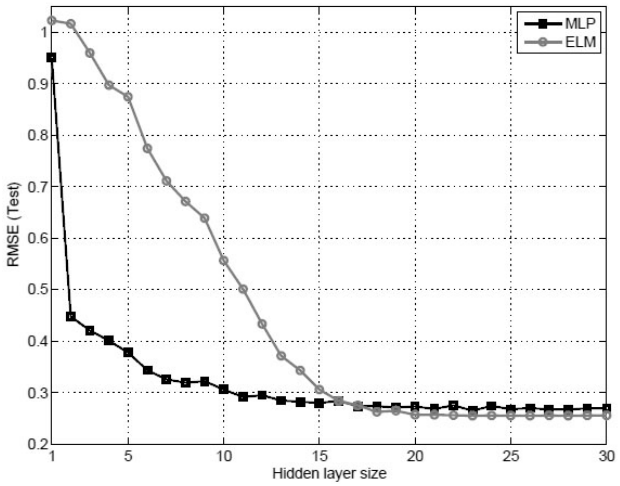

Fig. 3 *Evolution of the Root Mean Square Error (RMSE) in the test set of the Sinusoidal problem with respect to the hidden layer size of FFNN which is designed using the gradient-based MLP training approach and the ELM methodology. Results are averages of 20 different trials.*

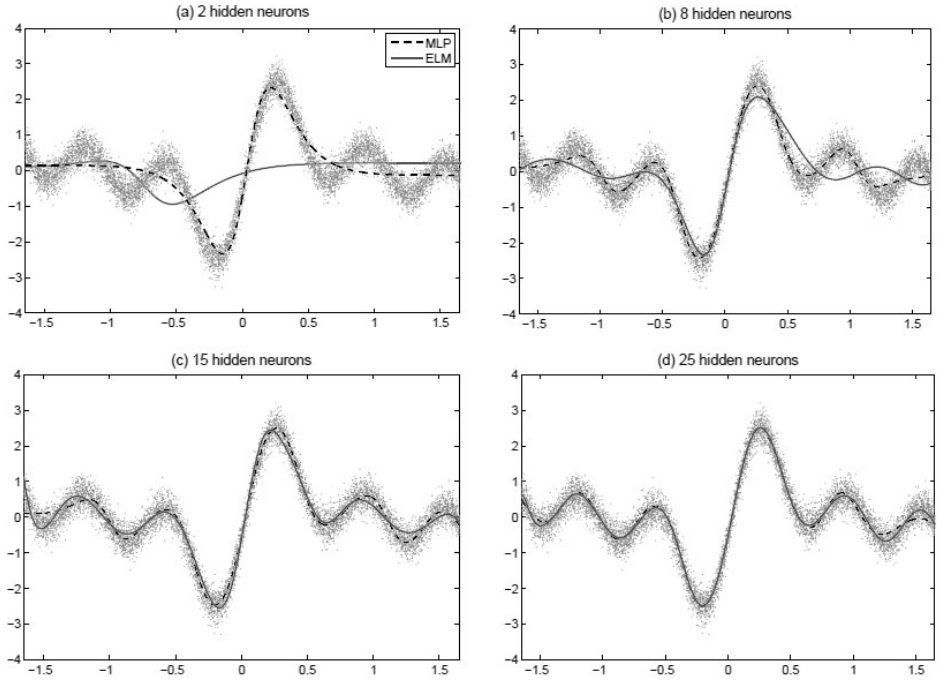

Fig. 4 *Examples of training results using MLP and ELM on the Sinusoidal problem for different hidden layer size.*

Bank, *Stocks*) are from L. Torgo's website [22], the *Parkinsons* dataset is obtained from UCI Machine Learning Repository [23] and the *Tecator* dataset is from the StatLib archive [24]. These problems belong to several scientific areas (in particular, aeronautics, economics, biomedical engineering and chemometrics), except for *Friedman* that is an artificial problem. Besides, these regression problems have been selected trying to cover different complexities, such as large datasets or high dimensional data.

Dataset	**Training Samples**	**Testing Samples**	n
Friedman	27179	13589	10
Elevators	8752	7847	18
Stocks	634	316	9
Bank	4499	3693	8
Parkinsons	3525	2350	16
Tecator	172	44	100

Tab. I *Regression datasets used in the experiments.*

Initially, we have evaluated the standard ELM algorithm and the OP-ELM method for training individual SLFNs by considering that the hidden layer size (M) is between 1 and 150. In order to achieve L different networks, the learning and testing phases are repeated $L = 50$ times (50 different weight initializations) in which the sigmoid function is selected as the activation function. It should be noted that this previous stage of construction of L different networks has been repeated 10 trials in order to measure the consistence of the obtained predictions by the ensemble procedures. After that, the ensemble based approaches are validated. In particular, for each trial, the L trained OP-ELM networks are firstly combined by the least-squares solution of (7), i.e., the LSC-OPELM method. Next, the three ensemble procedures (OC-OPELMs, OC-OPELMs-FFS and OC-OPELMs-RFS) are evaluated. Note that the OC-OPELM method makes use of the same previously L trained models with OP-ELM for each trial. In the experiments, all approaches use the original learning and testing sets, except for the standard ELM algorithm that uses 10-fold CV for selecting the hidden layer size. Thus, the training time of ELM includes the 10-fold CV procedure. As in the first evaluation experiment, all simulations have been carried out in MATLAB 7.11(R2010b) environment running in a computer with 4 GB of memory and 2.67 GHz processor.

Tab. II shows the obtained results using the five aforementioned ELM-based approaches. For each approach and dataset, this table shows the computational training time, the test RMSE results and the model size. In particular, for each trial, $L = 50$ different models have been trained in all problems, except for *Tecator* because this high-dimensional dataset requires to train more networks ($L = 100$) in order to OC-OPELMs-FFS obtains different random networks with enough input feature diversity, as we have already mentioned in the previous section. Note that the third column of Tab. II for ELM and OP-ELM shows its obtained test RMSE results (mean and standard deviation) for the best SLFN according to the CV procedure. Meanwhile, the test RSME results (mean and standard deviation) of the remaining approaches are given by the different ensemble schemes of the

Data set	Learning Method	RMSE results	Training Time (s)	Model size
Friedman	ELM	2.69 ± 3.0e-2	1.61e+3	139
	OP-ELM	2.63 ± 9.0e-3	4.30e+2	98
	LSC-OPELM	2.38 ± 6.2e-3	2.26e+3	50
	OC-OPELM	2.38 ± 5.9e-3	2.21e+3	30
	OC-OPELM-FFS	1.45 ± 3.1e-3	5.46e+4	16
	OC-OPELM-RFS	1.28 ± 4.0e-3	2.05e+3	34
Elevators	ELM	3.59e-3 ± 8.5e-5	7.54e+4	141
	OP-ELM	3.65e-3 ± 3.6e-5	3.94e+1	112
	LSC-OPELM	2.93e-3 ± 2.3e-5	2.00e+3	50
	OC-OPELM	2.78e-3 ± 2.4e-5	2.02e+3	31
	OC-OPELM-FFS	2.29e-3 ± 2.5e-5	8.99e+4	24
	OC-OPELM-RFS	2.74e-3 ± 2.7e-5	1.80e+3	40
Stocks	ELM	1.12 ± 5.8e-2	4.94e+1	138
	OP-ELM	1.11 ± 2.5e-2	2.36e+0	100
	LSC-OPELM	0.80 ± 1.6e-2	1.19e+2	50
	OC-OPELM	0.78 ± 1.2e-2	1.21e+2	24
	OC-OPELM-FFS	0.73 ± 1.0e-2	2.34e+3	17
	OC-OPELM-RFS	0.72 ± 1.4e-2	1.00e+2	17
Bank	ELM	4.61e-2 ± 1.2e-3	3.82e+2	130
	OP-ELM	4.49e-2 ± 5.5e-4	3.40e+1	106
	LSC-OPELM	3.62e-2 ± 3.3e-4	1.71e+3	50
	OC-OPELM	3.60e-2 ± 3.1e-4	1.69e+3	33
	OC-OPELM-FFS	3.29e-2 ± 3.3e-4	2.15e+4	20
	OC-OPELM-RFS	3.17e-2 ± 3.8e-4	1.53e+3	33
Parkinsons	ELM	9.06 ± 9.7e-2	2.12e+2	143
	OP-ELM	9.66 ± 1.6e-2	2.44e+1	82
	LSC-OPELM	8.98 ± 1.6e-2	1.25e+3	50
	OC-OPELM	8.91 ± 1.4e-2	1.25e+3	21
	OC-OPELM-FFS	9.00 ± 1.0e-2	6.40e+4	10
	OC-OPELM-RFS	8.80 ± 1.7e-2	1.17e+3	38
Tecator	ELM	4.89 ± 8.2e-1	8.27e+2	76
	OP-ELM	3.51 ± 1.2e-1	5.30e+2	66
	LSC-OPELM	2.45 ± 1.2e-1	5.40e+2	100
	OC-OPELM	2.29 ± 9.0e-2	5.40e+2	44
	OC-OPELM-FFS	1.31 ± 7.9e-2	1.05e+4	15
	OC-OPELM-RFS	2.00 ± 9.2e-2	5.29e+2	30

Tab. II *Experimental results on six regression datasets.*

$L^* \leq L$ networks. With respect to the fourth column, it should be noted that it shows the average training time (in seconds) for building a single model using ELM and OP-ELM. Whereas, for the different ensemble methods, we show the average training time for building L networks with OP-ELM and, then, ensembling them using the approaches introduced in Section 3.. Finally, in the last column of Tab. II, we show the average hidden layer size for the ELM and OP-ELM methods and, meanwhile, the average number of chosen networks is shown for the remaining ensemble-based approaches.

According to these results (see Tab. II), and as it has been also shown in [8], OP-ELM outperforms ELM by using smaller models (more compact hidden layer sizes) and less training time, except for *Parkinsons* dataset. With respect to the ensemble-based approaches, all of them provide better generalization capabilities than ELM and OP-ELM in all problems. Besides, in general, ensemble procedures increase stability (i.e. lower standard deviation) with respect to ELM and OP-ELM. It is worthy of remark that these advantages are obtained with a negligible increment of the total computational time for training L models using OP-ELM, except for the OC-OPELMs-FFS procedure which is based on incremental forward feature selection and, thus, it needs more computational efforts. In all simulations, the OC-OPELMs-FFS and OC-OPELMs-RFS provide better results than the ensemble procedures based on the original input feature space. Moreover, the use of random feature subsets is beneficial to the ensemble construction. This approach may be better than performing a previous forward feature selection, except for *Tecator* and *Elevators* datasets. For example, in *Tecator*, the input data space is successfully reduced from one hundred to only five features. Whereas, in *Friedman* dataset, although OC-OPELMs-FFS chooses the relevant features (which are previously known by the definition of this artificial dataset), this method achieves worst performance results than OC-OPELMs-RFS, which uses random feature subsets (including relevant and irrelevant features). As it is expected, OC-OPELMs-RFS requires more networks to construct the ensemble system than OC-OPELMs-FFS. Thus, in general, it is recommended to exploit diversity in the ensemble of ELM networks by varying the input data space and, also, a feature selection (prior to the ensemble construction) is clearly useful for high-dimensional datasets.

5. Conclusions

The ELM algorithm and its recent extensions, such as the OP-ELM method, provide simple, fast and robust learning algorithms for SLFNs with random hidden nodes. The training process of an ELM model can be several order of magnitude faster than traditional learning algorithms for FFNN, while attaining comparable or even better approximation and generalization capabilities.

This paper presents effective ensemble procedures based on ELM in order to exploit the diversity among the different SLFNs in the ensemble system. The ensemble procedures discard inaccurate individual models from the ensemble system using the basis of the OP-ELM methodology and, thus, an optimal combination of networks is achieved. In contrast to other ELM ensembling methods, note that each network is automatically constructed without fixing a predefined hidden layer size. Besides, ELM networks have been constructed with different input feature

spaces: the original input feature space, the input subset obtained by forward feature selection and the random feature subsets. The experimental results show that an ensemble approach of ELM models outperforms an individual ELM model in terms of generalization capability. The ensemble diversity is increased by varying the input feature space and it produces a performance improvement. As future work, it is intended to explore data editing techniques in order to generate different learning subsets for training the individual ELM networks.

References

[1] C. M. Bishop: Pattern Recognition and Machine Learning. Springer, August 2006.

[2] G. Huang, Q. Zhu, and C. Siew: Extreme learning machine: a new learning scheme of feedforward neural networks. In IEEE International Joint Conference on Neural Networks, volume 2, pp. 985–990, 2004.

[3] G.-B. Huang and C.-K. Siew: Extreme learning machine with randomly assigned RBF kernels. International Journal of Information Technology, 11(1), pp. 16–24, 2005.

[4] G. Huang, Q. Zhu, and C. Siew: Extreme learning machine: Theory and applications. Neurocomputing, 70(1-3), pp. 489–501, 2006.

[5] J. Sánchez-Monedero, M. Cruz-Ramírez, F. Fernández-Navarro, P. A. Gutiérrez, and C. Hervás-Martínez: On the suitability of extreme learning machine for gene classification using feature selection. In 10th International Conference on Intelligent Systems Design and Applications (ISDA2010), pp. 507–512, 2010.

[6] W. Deng and L. Chen: Color image watermarking using regularized extreme learning machine. Neural Network World, 20(3), pp. 317–330, 2010.

[7] J. Sánchez-Monedero, C. Hervás-Martínez, P. A. Gutiérrez, M. Carbonero Ruíz, Ramírez Moreno M. C., and Cruz-Ramírez M. Assessing the evolution of learning capabilities and disorders with a graphical exploratory analysis of surveys containing missing and conflicting answers. Neural Network World, 20(7), pp. 899–912, 2010.

[8] Y. Miche, A. Sorjamaa, P. Bas, O. Simula, C. Jutten, and A. Lendasse: OP-ELM: Optimally Pruned Extreme Learning Machine. IEEE Transactions on Neural Networks, 21(1), pp. 158–162, December 2009.

[9] Y. Lan, Y.C. Soh, and G.-B. Huang: Ensemble of online sequential extreme learning machine. Neurocomputing, 72(13-15), pp. 3391–3395, 2009.

[10] N. Liu and H. Wang: Ensemble based extreme learning machine. IEEE Signal Processing Letters, 17(8), pp. 754–757, 2010.

[11] M. Van Heeswijk, Y. Miche, T. Lindh-Knuutila, P. Hilbers, T. Honkela, E. Oja, and A. Lendasse: Adaptive ensemble models of extreme learning machines for time series prediction. In International Conference on Artificial Neural Networks, volume 5769, pp. 305–314, 2009.

[12] M. van Heeswijk, Y. Miche, E. Oja, and A. Lendasse: GPU-accelerated and parallelized ELM ensembles for large-scale regression. Neurocomputing, 74(16), pp. 2430–2437, September 2011.

[13] H. Chen, H. Chen, X. Nian, and P. Liu: Ensembling extreme learning machines. In 4th International Symposium on Neural Networks, ISNN, pp. 1069–1076, 2007.

[14] L. K. Hansen and P. Salamon: Neural network ensembles. IEEE Transactions on Pattern Analysis and Machine Intelligence, 12(10), pp. 993–1001, 1990.

[15] Ch. J. Merz and M. J. Pazzani: Combining neural network regression estimates with regularized linear weights. In Advances in Neural Information Processing Systems 9 (NIPS), pp. 564–570, 1996.

[16] Z. Zhou. Ensembling neural networks: Many could be better than all. Artificial Intelligence, 137(1-2), pp. 239–263, May 2002.

[17] T. Similä and J. Tikka: Multiresponse sparse regression with application to multidimensional scaling. In Proceedings of the 15th International Conference on Artificial Neural Networks: Formal Models and Their Applications - ICANN 2005, LNCS 3697, pp. 97–102, 2005.

[18] R. Myers: Classical and Modern Regression with Applications. Duxbury Press, 2 edition, 1990.

[19] C. R. Rao and S. K. Mitra: Generalized Inverse of Matrices and its Applications. Wiley, New York, 1971.

[20] B. Efron, T. Hastie, L. Johnstone, and R. Tibshirani: Least angle regression. Annals of Statistics, 32, pp. 407–499, 2004.

[21] G. Bontempi: Statistical foundations of machine learning. Université Libre de Bruxelles, 2011.

[22] L. Torgo: Regression datasets. `http://www.liaad.up.pt/∼ltorgo/regression/datasets.html`.

[23] A. Asuncion and D. J. Newman: UCI machine learning repository. `http://www.ics.uci.edu/∼mlearn/mlrepository.html`.

[24] P. Vlachos: Statlib archive. `http://lib.stat.cmu.edu/`.