Genetically Optimized Extreme Learning Machine

Tiago Matias[1,2], Rui Araújo[1,2], Carlos Henggeler Antunes[2,4], and Dulce Gabriel[1,3]
[1]Institute of Systems and Robotics (ISR-UC),
[2]Department of Electrical and Computer Engineering (DEEC-UC), and
[3]Department of Chemistry (DQ-UC),
University of Coimbra, Portugal.
[4]Institute for Systems Engineering and Computers (INESC) Coimbra, Portugal.
tmatias@isr.uc.pt, rui@isr.uc.pt, ch@deec.uc.pt, dracag@isr.uc.pt

Abstract

This paper proposes a learning algorithm for single-hidden layer feedforward neural networks (SLFN) called genetically optimized extreme learning machine (GO-ELM). In the GO-ELM, the structure and the parameters of the SLFN are optimized by a genetic algorithm (GA). The output weights, like in the batch ELM, are obtained by a least squares algorithm, but using Tikhonov's regularization in order to improve the SLFN performance in the presence of noisy data. The GA is used to tune the set of input variables, the hidden-layer configuration and bias, the input weights and the Tikhonov's regularization factor. The proposed method was applied and compared with four other methods over five benchmark problems available in a public repository. Besides it was applied in the estimation of the temperature at the burning zone of a real cement kiln plant.

1 Introduction

Multilayer feedforward neural networks (FFNN) have been used as universal approximators [10, 13] for system identification. However, in industrial applications, linear models are often preferred due to faster training in comparison with multilayer FFNN trained with gradient-descent algorithms [17]. In order to overcome the slow construction of FFNN models, a new method called extreme learning machine (ELM) is proposed in [15]. This method is a new batch learning algorithm for single-hidden layer FFNN (SLFN) where the input weights (weights of connections between the input variables and neurons in the hidden-layer) and the bias of neurons in the hidden-layer are randomly assigned. The output weights (weights of connections between the neurons in the hidden-layer and the output neuron) are obtained using the Moore–Penrose (MP) generalized inverse, considering that the activation function of the output neuron is linear.

Since in ELM the output weights are computed based on the random input weights and bias of the hidden nodes, there may exist a set of non-optimal or unnecessary input weights and bias of the hidden nodes. Furthermore, the ELM tends to require more hidden neurons than conventional tuning-based learning algorithms (based on backpropagation of the error or other learning methods where the output weights are not obtained by least squares method) in some applications, which can negatively affect SLFN performance in unknown testing data [15]. Also, fitting problems can be encountered in the presence of the irrelevant or correlated input variables [17].

The search and optimization properties of genetic algorithms (GAs) make them suitable for model design (architecture and weights) [3], more specifically in SLFNs (for an extended review, see [6, 25]). In [16] an improved GA is used to optimize the structure (connections layout) and the parameters (connection weights and biases) of a SLFN with switches. The switches are unit step functions that make possible the removal of each connection. Using a real encoding scheme, and new crossover and mutation techniques, this improved GA obtains better results in comparison with traditional GAs. The structure and the parameters of the same kind of SLFN with switches are also tuned in [22], in this case using a hybrid Taguchi GA. This approach is similar to a traditional GA but a Taguchi method [23] is used for the crossover process. The use of this method implies the construction of a $(n + 1) \times n$ two-level orthogonal matrix, where n is the number of variables for the optimization process. However, the construction of this orthogonal matrix is not simple. There are some standard orthogonal matrices but they can be only used when n is small. In large networks, n is large and therefore this method is not a good practical approach. In these methodologies, the weights between the hidden-layer and the output layer are optimized by the GA. Using the ELM approach, the output weights (considering an output neuron with linear activation function) could be calculated from the solution of least squares that minimizes the sum of squares of the error between the desired and estimated output of the SLFN. Using the Moore-Penrose generalized inverse to obtain the least squares solution, the output weights could be quickly obtained,

reducing the convergence time of the GA. Furthermore, as the number of variables of the optimization process is lower, the search space to be explored by the GA narrows. This approach was used in [24] where a GA is used to select the connections, and tune the parameters between the input layer and the hidden layer, and a least squares algorithm is applied to tune the parameters between the hidden layer and the output layer. However, in this type of approach it is difficult to deal with the tendency to require more hidden nodes than conventional tuning-based algorithms, as well as the problem caused by the presence of irrelevant variables is difficult to solve. Using a switch for each connection, a hidden-neuron or an input variable can be discarded only if all connection switches associated with this hidden-neuron or input variable are disabled. Furthermore, this methodology does not tune the neurons activation function, and with noisy data the use of least squares without regularization will make the model displaying a poor generalization capability [5]. In this paper, a novel learning algorithm for SLFNs called genetically optimized extreme learning machine (GO-ELM) is proposed. In GO-ELM, a GA is used to optimize the weights of connections between the input layer and the hidden-layer, the bias of neurons of the hidden-layer, the set of input variables, and the hidden-layer configuration (number of neurons and activation function of each neuron). Using auxiliary binary selection variables, the irrelevant variables can be removed, thus decreasing the network size and overcoming the reduction of performance of the ELM in the presence of irrelevant variables. The optimization of the hidden-layer allows the selection of the optimal number of neurons in this layer and the activation function of each neuron, trying to overcome the propensity of ELM for requiring more hidden nodes than conventional tuning-based learning algorithms. Like in original ELM, the output weights are obtained using the least squares algorithm. However, as the use of least squares in noisy data causes model overfitting, the Tikhonov's regularization [12] is used to obtain a robust least squares solution. To prove the effectiveness of the proposed method, it was applied in five benchmark problems available in a public repository and compared with four methodologies. An example of application of GO-ELM in the estimation of the burning zone temperature in a real cement kiln plant is also presented.

The paper is organized as follows. The SLFN architecture is overviewed in Section 2. Section 3 gives a brief review of the batch ELM and GA. The optimization approach for ELM is presented in Section 4. Section 5 presents experimental results. An example of application of the proposed approach is presented in Section 6. Finally, concluding remarks are drawn in Section 7.

2 Adjustable Single Hidden-Layer Feedforward Network Architecture

The neural network considered in this paper is a SLFN with adjustable architecture as shown in Fig. 1, which can

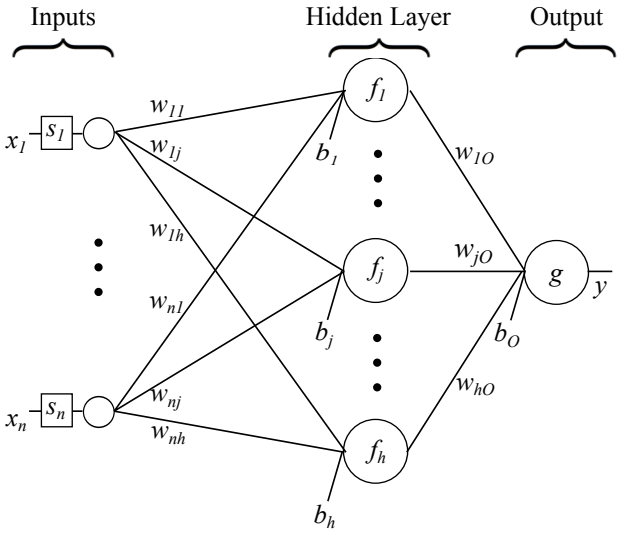

Figure 1: Single hidden-layer feedforward network with adjustable architecture.

be mathematically represented by:

$$y = g\left(b_O + \sum_{j=1}^{h} w_{jO} v_j\right), \quad (1)$$

$$v_j = f_j\left(b_j + \sum_{i=1}^{n} w_{ij} s_i x_i\right). \quad (2)$$

n and h are the number of input variables and the number of the hidden layer neurons, respectively; v_j is the output of the hidden-layer neuron j; x_i, $i = 1, \ldots, n$, are the input variables; w_{ij} is the weight of the connection between the input variable i and the neuron j of the hidden layer; w_{jO} is the weight of the connection between neuron j of the hidden layer and the output neuron; b_j is the bias of the hidden layer neuron j, $j = 1, \ldots, h$, and b_O is the bias of the output neuron; $f_j(\cdot)$ and $g(\cdot)$ represent the activation function of the neuron j of the hidden layer and the activation function of the output neuron, respectively. s_i is a binary variable used in the selection of the input variables during the design of the SLFN.

Using the binary variable s_i, $i = 1, \ldots, n$, each input variable can be considered or not. However, the use of variables s_i is not the single tool to optimize the structure of the SLFN in GO-ELM. The configuration of the hidden layer can be adjusted in order to minimize the output error of the model. The activation function $f_j(\cdot)$, $j = 1, \ldots, h$, of each hidden node can be either zero, if this neuron is unnecessary, or any (predefined) activation function. Unlike in error backpropagation methods, in GO-ELM the activation functions do not have to be differentiable and therefore any activation function can be considered.

3 Preliminaries

In this section, the ELM and the GA are briefly reviewed.

3.1 Extreme Learning Machine

The batch ELM was proposed in [15]. In [14] it is proved that a SLFN with randomly chosen weights between the input layer and the hidden layer and adequately chosen output weights are universal approximators with any bounded non-linear piecewise continuous functions.

Considering that N samples are available, that the output bias is zero, and that the output neuron has a linear activation function, (1) and (2) can be rewritten as:

$$\mathbf{y} = \left(\mathbf{w}_O{}^T \mathbf{V}\right)^T, \tag{3}$$

where $\mathbf{y} = [y(1), \ldots, y(N)]^T$ is the vector of outputs of the SLFN, $\mathbf{w}_O = [w_{1O}, \ldots, w_{hO}]^T$ is the vector of output weights, and $\mathbf{V}$ is the matrix of the outputs of the hidden neurons (1) given by:

$$\mathbf{V} = \begin{bmatrix} v_1(1) & v_1(2) & \ldots & v_1(N) \\ \vdots & \vdots & \ddots & \vdots \\ v_h(1) & v_h(2) & \ldots & v_h(N) \end{bmatrix}, \tag{4}$$

with $s_i = 1, i = 1, \ldots, n$.

Considering that the input weights and bias matrix $\mathbf{W}$,

$$\mathbf{W} = \begin{bmatrix} b_1 & b_2 & \ldots & b_h \\ w_{11} & w_{12} & \ldots & w_{1h} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n1} & w_{n2} & \ldots & w_{nh} \end{bmatrix}, \tag{5}$$

is randomly assigned, the output weights vector $\mathbf{w}_O$ is estimated as:

$$\hat{\mathbf{w}}_O = \mathbf{V}^\dagger \mathbf{y}_d, \tag{6}$$

where $\mathbf{V}^\dagger$ is the Moore-Penrose generalized inverse of the hidden-layer output matrix $\mathbf{V}$, and $\mathbf{y}_d = [y_d(1), \ldots, y_d(N)]^T$ is the desired output.

Considering that $\mathbf{V} \in \mathbb{R}^{N \times h}$ with $N \geq h$ and $rank(\mathbf{V}) = h$, the Moore-Penrose generalized inverse of $\mathbf{V}$ can be given by:

$$\mathbf{V}^\dagger = (\mathbf{V}^T \mathbf{V})^{-1} \mathbf{V}^T. \tag{7}$$

Substituting (7) into (6), the estimation of $\mathbf{w}_O$ can be obtained by the following least-squares solution:

$$\hat{\mathbf{w}}_O = (\mathbf{V}^T \mathbf{V})^{-1} \mathbf{V}^T \mathbf{y}_d. \tag{8}$$

3.2 Genetic Algorithms

The basic principles of GA were introduced by Holland [11]. GA are search and optimization methods that have been used to solve complex problems effectively [18, 21, 2]. The principle of GA is the simulation of the natural processes of evolution applying the Darwin's theory of natural selection. In a GA the possible solutions are encoded into chromosomes (individuals) and the fittest ones are more susceptible to be selected for reproduction, producing offspring with characteristics of both parents.

The GA used in this paper is devoted to mixed integer optimization problems [8]. In [8] the authors proposed a methodology that allows solving optimization problems where the decision variables can be a combination of real, integer, and binary variables.

The detailed description of this method is given in this section.

3.2.1 Variable Coding and Initial Population

In order to minimize cost functions involving real, integer, and binary variables, all variables are mapped into continuous variables and a real encoding technique is applied. In the real encoding representation, each chromosome has the same length as the vector of decision variables and each element of the chromosome is encoded as a floating point number within the interval [0,1].

The initial population $\mathbf{P}$ is usually chosen randomly with uniform distribution and can be represented by:

$$\mathbf{P} = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_m\}, \tag{9}$$

where

$$\mathbf{p}_k = \left[p_{k_1}, p_{k_2}, \ldots, p_{k_q}\right]^T. \tag{10}$$

p_{k_l} is the variable l, $l = 1, 2, \ldots, q$, of chromosome k, $k = 1, 2, \ldots, m$, with $0 \leq p_{k_l} \leq 1$. m and q are the population size and the number of variables to be tuned, respectively.

3.2.2 Evaluation

Each chromosome represents one possible solution to the optimization problem. Therefore, each chromosome can be evaluated using a fitness function that is specific to the problem being solved. As all variables are mapped into continuous values between 0 and 1, before obtaining the fitness of each individual these values need to be converted into the actual variable values according to the domain of the problem and the corresponding true variable types. If the true value of the l-th variable ($l = 1, 2, \ldots, v$) of individual k ($k = 1, 2, \ldots, m$) is real, it is given by:

$$z_{k_l} = (z_l^{max} - z_l^{min}) p_{k_l} + z_l^{min}, \tag{11}$$

where z_l^{min} and z_l^{max} represent the true variable bounds ($z_l^{min} \leq z_{k_l} \leq z_l^{max}$). If it is a integer value:

$$z_{k_l} = rounddown\left((z_l^{max} - z_l^{min} + 1) p_{k_l}\right) + z_l^{min}, \tag{12}$$

where $rounddown(\cdot)$ is a function that rounds to the greatest integer than is lower than or equal to its argument. If the true value is binary, it is given by:

$$z_{k_l} = round\left(p_{k_l}\right), \tag{13}$$

where $round(\cdot)$ is a function that rounds to the nearest integer.

After the conversion of variables, the fitness of each individual can be obtained. The fitness function to evaluate a chromosome in the population can be written as:

$$fitness = \psi(\mathbf{p}_k) \in \mathbb{R}, \tag{14}$$

where the fitness function $\psi(\cdot)$ is specific to the problem to be solved.

3.2.3 Genetic Operators

Based upon their fitness values, a set of individuals is selected to survive to the next generation while the remaining are discarded. The surviving individuals become the

mating pool and the discarded chromosomes are replaced by new offspring. To select the parents from the mating pool, tournament selection was used [9]. For each parent, five individuals from the mating pool are randomly picked and the individual with best fitness is selected to be the parent. For each pair of parents two new individuals (offspring) are generated by crossover and mutation.

The crossover operation consists in producing offspring from the selected parents. Uniform crossover is used because it generally provides a larger exploration of the cost surface than other crossover operators [9]. In uniform crossover, first a random binary mask with the same length of the individuals is created. Then, each offspring receives values of variables from the first or second parent depending on whether the value of the mask bit is zero or one: the offspring 1/(2), receives the values from parent 1/(2) if the respective mask bit is one and receives the values from parent 2/(1) if the respective mask bit is zero. Consider the following example:

$$\begin{array}{rcccc} \text{Parent 1} & = & p_{1_1} & p_{1_2} & p_{1_3} & p_{1_4}, \\ \text{Parent 2} & = & p_{2_1} & p_{2_2} & p_{2_3} & p_{2_4}, \\ \text{Mask} & = & 1 & 0 & 0 & 1 \\ \text{Offspring 1} & = & p_{1_1} & p_{2_2} & p_{2_3} & p_{1_4}, \\ \text{Offspring 2} & = & p_{2_1} & p_{1_2} & p_{1_3} & p_{2_4}. \end{array}$$

After crossover, the mutation operator is used to maintain the diversity of the population and to prevent the algorithm from being trapped in local minima.

For each new offspring a random number r is generated and if $r < r_m$, where r_m is the mutation probability, this offspring is mutated. The mutation used is a two step operator. First, a random element of the individual is replaced by an uniform random value within the interval [0,1]. Being $\mathbf{p}_k = \left[p_{k_1}, p_{k_2}, p_{k_3}, p_{k_4}\right]^T$ the offspring, if the second element was selected to be replaced, the mutated chromosome is given by:

$$\mathbf{p}_k^1 = \left[p_{k_1}, p'_{k_2}, p_{k_3}, p_{k_4}\right]^T, \tag{15}$$

where p'_{k_2} is a new random value within the interval [0,1].

In a second step, a random adjustment factor is added to the chromosome. The adjustment factor comes from multiplying each element l within the previously mutated chromosome $\mathbf{p}_k^1$ by a random number ($-1 \leq \beta_{k_l} \leq 1$) and multiplying the resulting chromosome by a mutation factor ($0 \leq \eta_k \leq 1$) so that:

$$\mathbf{p}_k^c = \eta_k \left[\beta_{k_1} p_{k_1}, \beta_{k_2} p'_{k_2}, \beta_{k_3} p_{k_3}, \beta_{k_4} p_{k_4}\right]^T. \tag{16}$$

Finally, the mutated chromosome is given by:

$$\mathbf{p}_k^2 = rem\left(\mathbf{p}_k^1 + \mathbf{p}_k^c\right), \tag{17}$$

where rem is the remainder of each variable after the division by one.

4 Genetically Optimized Extreme Learning Machine

In GO-ELM, the weights of the output connections are obtained using the same ELM methodology presented in Section 3.1, however with a change.

The objective of the least squares method is to obtain the best output weights by solving the following problem:

$$\min(\|\mathbf{y} - \mathbf{y}_d\|_2), \tag{18}$$

where $||\cdot||_2$ is the Euclidean norm. The minimum-norm solution to this problem is given by (8).

The use of least squares can be considered a two-stage minimization problem involving: the determination of the solutions to (18), and the determination of the solution with minimum norm among solutions obtained in the previous stage.

The use of Tikhonov's regularization [12] allows the transformation of this two-stage problem into a single-stage minimization problem defined by:

$$\min(\|\mathbf{y} - \mathbf{y}_d\|_2 + \alpha\|\mathbf{w}_O\|_2), \tag{19}$$

where $\alpha > 0$ is a regularization parameter.

The solution to this problem is given by [12]:

$$\mathbf{w}_O = (\mathbf{V}^T\mathbf{V} + \alpha\mathbf{I})^{-1}\mathbf{V}^T\mathbf{y}_d, \tag{20}$$

where $\mathbf{I}$ is the $h \times h$ identity matrix.

If $\mathbf{V}$ is ill-conditioned, problem (19) should be preferred over problem (18) because the solution is numerically more stable [1]. Furthermore, using the Tikhonov's regularization, the robustness of the least squares solution against noise is improved.

As previously mentioned, the ELM tends to require more hidden nodes than conventional tuning-based algorithms. Furthermore, the presence of irrelevant variables in the training data set causes a decrease in the performance. To overcome these problems, in the proposed methodology the optimization of the set of input variables, the number and activation function of the neurons in the hidden layer, the connections weights between the inputs and the neurons of the hidden layer, the bias of the hidden layer neurons, and the regularization parameter α is made by GO-ELM. To perform this optimization, the approach presented in Section 3.2 is used.

The optimization of the SLFN is performed so as to maximize the following fitness function:

$$fitness = \frac{1}{1 + E_{rmse}(\mathbf{y}, \mathbf{y}_d)}, \tag{21}$$

where

$$E_{rmse}(\mathbf{y}, \mathbf{y}_d) = \sqrt{\frac{1}{N}\sum_{k=1}^{N}\left[y(k) - y_d(k)\right]^2} \tag{22}$$

is the root mean square error (RMSE) between the desired (real) output and the estimated values of the output. To improve the generalization performance, the estimation error $E_{rmse}(\mathbf{y}, \mathbf{y}_d)$ is obtained in a validation data set that has no overlap with the training data set.

In the optimization process, each chromosome $\mathbf{p}_k$, $k = 1, \ldots, m$ of the population $\mathbf{P}$ will be constituted by:

$$\begin{array}{rcl} \mathbf{p}_k & = & [w_{11}, \ldots, w_{nh}, b_1, \ldots, b_h, \\ & & s_1, \ldots, s_n, s_1^\lambda, \ldots, s_h^\lambda, \alpha]^T; \\ k & = & 1, \ldots, m, \end{array} \tag{23}$$

Table 1: Publicly available benchmark data sets description.

Data set	N	n
Automobile MPG	392	6
Cancer	194	32
Servo	167	4
Boston Housing	506	13
Price	160	15

where $s_j^\lambda \in \{0,1,2\}$, $j = 1,\ldots,h$, is an integer variable that defines the activation function f_j of each neuron j of the hidden-layer as follows:

$$f_j(\upsilon) = \begin{cases} 0, & \text{if } s_j^\lambda = 0, \\ 1/(1+\exp(-\upsilon)), & \text{if } s_j^\lambda = 1, \\ \upsilon, & \text{if } s_j^\lambda = 2. \end{cases} \quad (24)$$

The use of parameters s_j^λ makes it possible the adjustment of the number of neurons (if $s_j^\lambda = 0$ the neuron is not considered), and the activation function of each neuron (sigmoid or linear function). In this work only these two types of activation function have been used; however, any type of activation function can be considered. In applications where the set of input variables is known (without irrelevant or redundant variables) the elements $s_1,\ldots,s_n$ can be removed from the chromosome or can be forced to one during the optimization process.

As previously mentioned, all decision variables are mapped into real variables within the interval [0,1]. So before evaluating the fitness of each individual, all variables need to be converted into their true value. The variables s_i, $i = 1,\ldots,n$, are binary variables and thus are converted using (13). The variables s_j^λ, $j = 1,\ldots,h$, are integer variables and thus are converted using (12), considering that the lower and upper bounds are 0 and 2, respectively. The input weights w_{ij} and bias b_j are converted using (11), considering that the lower and upper bounds are -1 and 1. Finally, the regularization parameter is also converted using (11), considering the lower and upper bounds are 0 and 100.

5 Results

This section presents experimental results in five benchmark data sets available in [7]. Table 1 presents the number of samples N and the number of input variables of these benchmark data sets.

The data was divided as follows: the first half was used for training, and the second half was used for testing. All the input and output variables have been normalized to the range [-1,1].

The proposed method is compared with (i) the original batch ELM [15], (ii) the method proposed in [16] (IGA-SLFN), (iii) the self-adaptive evolutionary ELM (SaE-ELM) [4], and (iv) SLFN trained using the Levenberg-Marquardt algorithm (LM-SLFN). All these simulations have been made in Matlab environment running on a PC with 3.40GHz CPU with 4 cores and 8GB RAM. In ELM, IGA-SLFN, SaE-ELM, and LN-SLFN the number of neurons in the hidden-layer was gradually increased and the one with the best results in testing set is presented. In GO-ELM, as it has the capability to optimize the structure, it was considered that the initial number of neurons in the hidden-layer was 30. The performance of the methods is evaluated using mean and standard deviation of RMSE between the estimated and desired outputs in 20 trials. In the proposed method GO-ELM, SaE-ELM, and LM-SLFN, 30 % of the training data set was randomly picked and was used as validation data set. In GO-ELM and SaE-ELM, the population size was 80 individuals and the maximum number of generations was 100. In IGA-SLFN, it was considered that the population size was 80 individuals and the maximum number of generations was 500. In GO-ELM, the mutation probability was 10% and the mating pool was constituted by the best 40% of the individuals. These parameters were tuned by means of experimentation.

Table 2 presents the average of the 20 trials of the five methods in all data sets and the number of neurons in the hidden-layer used to obtain these results. In the proposed method GO-ELM, as the hidden-layer configuration is optimized, the mean of the number of hidden neurons is presented. For all data sets, the best RMSE is shown in bold face in Table 2. Statistical paired t-test using RMSE is also conducted for all data sets. Specifically, paired t-test between GO-ELM and each one of the other methods is conducted. In this test it is considered that the null hypothesis is that the mean RMSE of the two tested methods is the same, and that the significance level is 0.05 for all experiments. The symbols "(+)" and "(−)" are used to indicate the win or loss situation of GO-ELM over the other tested method.

From the analysis of the results it can be verified that GO-ELM consistently performs at least as well as, if not better than, all benchmark methods in four of the five benchmark data sets (Automobile MPG, Cancer, Housing and Price data sets) and in two of them (Cancer and Housing) shows also the lowest standard deviation of RMSE. In the Servo data set, the fitting performance of SaE-ELM and batch ELM is statistically better that GO-ELM. The training time used by GO-ELM is longer than in IGA-SLFN, LM-SLFN, and ELM due to the optimization of SLFN. However, as in GO-ELM the number of neurons is not defined by trial-and-error, being only necessary to initialize the methodology with a big number of neurons, the long training time can compensated. The lowest training time was obtained by the ELM, as expected.

Fig. 2 presents the GA convergence curves of GO-ELM in the Price data set. From the analysis of the figure, it can be seen that the convergence of the GA is fast and therefore the training time of the GO-ELM can be shortened if necessary; however, this can cause a reduction of the GO-ELM performance.

Table 3 shows the relative frequency of the presence of each input variable in the input variable set after each

Table 2: Results of the application of the five methods in the benchmark data sets using the RMSE as the performance measure.

Data set	Method	Mean Testing RMSE	Training Time (s)	Hidden Neurons	p-value
Automobile MPG	GO-ELM	**0.2603**	3.4492	13.45	-
	IGA-SLFN	0.3691	1.0937	30	**0.00**(+)
	SaE-ELM	0.2660	35.9230	21	0.72
	LM-SLFN	0.3571	0.7075	16	**0.00**(+)
	ELM	0.2624	0.0015	16	0.88
Cancer	GO-ELM	**0.5635**	5.0675	15.25	-
	IGA-SLFN	0.6027	1.6171	15	**0.00**(+)
	SaE-ELM	0.5903	7.7140	16	**0.00**(+)
	LM-SLFN	0.7582	1.1675	16	**0.00**(+)
	ELM	0.5980	0.0020	15	**0.00**(+)
Servo	GO-ELM	0.2671	4.2768	19.45	-
	IGA-SLFN	0.3464	0.8364	18	**0.00**(+)
	SaE-ELM	**0.2315**	29.7725	17	**0.00**(−)
	LM-SLFN	0.2830	0.5545	15	0.28
	ELM	0.2397	0.0050	26	**0.00**(−)
Housing	GO-ELM	**0.3576**	3.8974	18.70	-
	IGA-SLFN	0.3962	1.0993	19	**0.00**(+)
	SaE-ELM	0.5297	8.1410	15	**0.00**(+)
	LM-SLFN	0.5383	1.1540	16	**0.00**(+)
	ELM	0.4434	0.0015	11	**0.00**(+)
Price	GO-ELM	**0.1838**	3.4219	10.40	-
	IGA-SLFN	0.2213	1.1082	19	**0.01**(+)
	SaE-ELM	0.2970	7.3430	15	**0.00**(+)
	LM-SLFN	0.4858	1.0280	16	**0.00**(+)
	ELM	0.2115	0.0005	13	**0.01**(+)

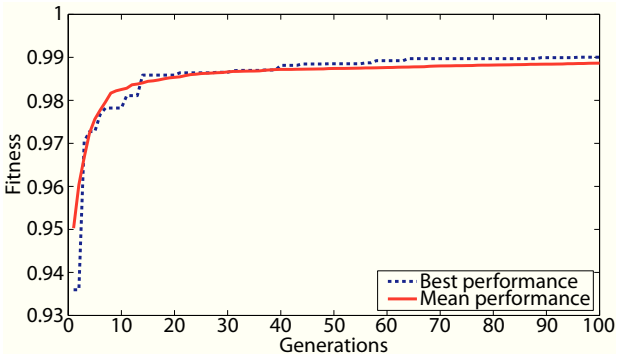

Figure 2: Convergence of GA in Price data set (trial with best final fitness and mean performances of the 20 trials).

trial by GO-ELM in the Automobile MPG, Servo, Boston Housing, and Price data sets. From the analysis of the table it is clear that in these data sets some input variables were selected with much more frequency than others and therefore it can be concluded that some irrelevant input variables exist. Due to the large number of input variables, the results of GO-ELM with respect to the most selected input variables is not presented, but in average a set with 11.4 input variables was obtained.

6 Application Example: Estimation of the Burning Zone Temperature in a Cement Kiln Plant

Inside a rotary cement kiln, temperatures in the range of 1200-1700°C heat a mixture of limestone, shale, clay, sand, and smaller quantities of other substances, resulting in small black nodules called clinkers. Outside the kiln these clinkers are cooled and grounded to produce cement [20]. The control of the temperature inside the kiln is crucial: insufficiently high maximum temperatures in the kiln result in incompletely reacted products and poor-quality cement, while excessive maximum temperatures waste energy and propitiate the formation of NO_x pollutant compounds that have several environmental impacts [19].

As the contact temperature measurement is impossible, this is made using a pyrometer. However, due to the flying dust within the kiln system that blocks the sensor after some time in operation, it has to be removed and cleaned by an operator, which can take a long time. It is therefore desirable to develop a model that is able to replace the pyrometer that measures the burning zone temperature.

In this section, a data set[1] from a real cement kiln plant was used. The data set refers to 194 monitorized vari-

[1]Provided by "Acontrol - Automação e Controle Industrial, Lda",

Table 3: Relative frequency of selection of input variables by GO-ELM over the 20 trials of each of the Automobile MPG, Servo, Boston Housing, and Price data sets. A value of 1 indicates the variable was selected 20 times.

Data set	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}
Automobile MPG	0.05	1.00	0.15	0.95	0.55	1.00	-	-	-	-	-	-	-	-	-
Servo	0.30	1.00	1.00	1.00	-	-	-	-	-	-	-	-	-	-	-
Boston Housing	0.40	0.55	0.20	0.05	0.90	1.00	0.65	1.00	0.30	0.95	1.00	0.45	1.00	-	-
Price	0.15	0.90	1.00	0.15	0.15	0.30	0.95	0.05	0	0.05	0.75	0.80	0.05	0.30	0.35

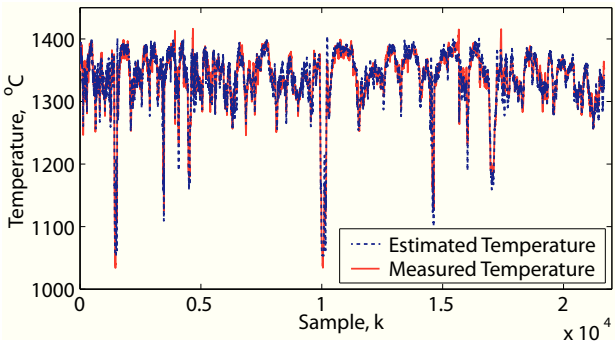

Figure 3: Measured and estimated burning zone temperatures using the GO-ELM in training set.

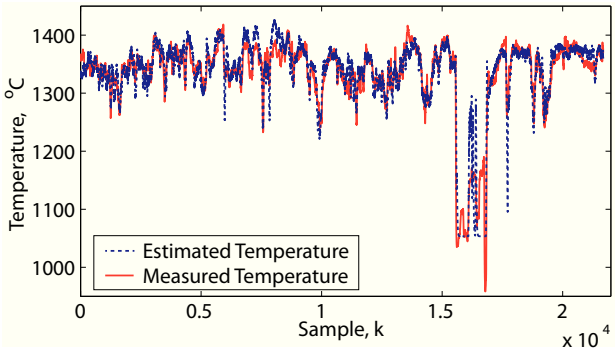

Figure 4: Measured and estimated burning zone temperatures using the GO-ELM in testing set.

ables, recorded with a sampling interval of $T = 1$ [min], during over a month period leading to 43469 entries. The monitored variables refer to several system variables, regarding temperatures, pressures, concentrations, manual and laboratorial entries, and several other variables associated with the cement production process, from the preheater (cyclone) tower until the chimney and cement mill. From the initial set, the variables that are online measured and which contain pertinent process information were selected, resulting in a data set with 79 variables (78 inputs and one output). As the dataset was obtained from a real process, the data was passed through a first order Butterworth low-pass filter with 0.02 [Hz] band width in order to eliminate the noise from the signal.

The first half of the data set was used to obtain the model using the GO-ELM method and the second half was used for testing. 30% of the training data set was randomly picked and was used as validation data set. All the input and output variables have been normalized (scaled) to the range of [-1,1]. It was considered that the maximum number of neurons in the hidden-layer was 50, the population size was 80 individuals, the maximum number of generations was 100, and the mutation probability was 10%.

Figures 3 and 4 present the burning zone temperatures measured by the pyrometer and the unscaled temperatures estimated by the SLFN trained using the GO-ELM method in the training and testing sets, respectively. As can be seen, the model learned the behavior of the process from the training set and has the capability to estimate well the temperature in the burning zone for over more than two weeks. The SLFN obtained by the GO-ELM method was composed by 40 input variables and a hidden-layer with 35 neurons (19 with sigmoidal activation function and 16 with linear activation function). From the obtained results it can be concluded that an SLFN trained using the GO-ELM method can be used to replace the pyrometer sensor in the measurement of the burning zone temperatures.

Coimbra, Portugal.

7 Conclusion

A novel learning algorithm for SLFNs called genetically optimized extreme learning machine is presented. In order to solve the tendency of ELM to require more neurons in the hidden-layer than conventional tuning-based learning algorithms and the reduction of performance exhibited by ELM in the presence of irrelevant input variables, the proposed method uses a GA to optimize the structure and the parameters of the SLFN. Like in the original ELM, the output weights are obtained using the least squares algorithm, but with Tikhonov's regularization. The regularization penalizes the solution with larger norms and allows an improvement in the SLFN generalization capability, improving the performance in the test data.

To validate and demonstrate the performance and effectiveness of the proposed method, it was applied on five benchmark data sets available in a public repository. The performance of the proposed method was at least statistically equal, if not better, than the performance of IGA-SLFN, SaE-ELM, LM-SLFN, and ELM in four of the five data sets. The results also show that in the proposed

method the number of neurons in the hidden-layer does not need to be selected by trial-and-error and the relevant input variables can be automatically selected, reducing the network size and improving the generalization capability.

The GO-ELM method was also successfully applied to the estimation of the burning zone temperature using a data set from a real cement kiln plant. The SLFN with GO-ELM had the capability to replace with accuracy the pyrometer sensor over more than two weeks data.

In future work different methods like differential evolution or simulated annealing as well as their hybridization with GA will be tested in the optimization of the structure and parameters of the SLFN.

Acknowledgment

This work was supported by Project FAir-Control "Factory Air Pollution Control" (reference: E!6498), supported by the Eurostars Programme of the EUREKA network, financed by "Fundação para a Ciência e a Tecnologia" (FCT), do Ministério da Educação e Ciência, "Agência de Inovação" (AdI), and the Seventh Framework Programme for Research and Technological Development (FP7) of the European Union.

Tiago Matias and Rui Araújo acknowledge the support of FCT project PEst-C/EEI/UI0048/2011.

C. H. Antunes acknowledges the support of FCT project PEst-C/EEI/UI0308/2011 and QREN Mais Centro Program iCIS project (CENTRO-07-ST24-FEDER-002003).

References

[1] A. Ben-Israel and T. N. Greville. *Generalized Inverses: Theory and Applications*. Springer-Verlag, New York, USA, second edition, 2003.

[2] C. Bi. Deterministic local alignment methods improved by a simple genetic algorithm. *Neurocomputing*, 73(13-15):2394–2406, 2010.

[3] E. Cantú-Paz and C. Kamath. Evolving neural networks to identify bent-double galaxies in the first survey. *Neural Networks*, 16(3-4):507–517, 2003.

[4] J. Cao, Z. Lin, and G.-B. Huang. Self-adaptive evolutionary extreme learning machine. *Neural Processing Letters*, 36(3):285–305, 2012.

[5] S. Chen, E. S. Chng, and K. Alkadhimi. Regularized orthogonal least squares algorithm for constructing radial basis function networks. *International Journal of Control*, 64(5):829–837, 1996.

[6] S. Ding, H. Li, C. Su, J. Yu, and F. Jin. Evolutionary artificial neural networks: a review. *Artificial Intelligence Review*, 39(3):251–260, 2013.

[7] A. Frank and A. Asuncion. UCI machine learning repository, 2010.

[8] R. L. Haupt. Antenna design with a mixed integer genetic algorithm. *IEEE Transactions on Antennas and Propagation*, 55(3):577–582, 2007.

[9] R. L. Haupt and S. E. Haupt. *Practical Genetic Algorithms, 2nd edn with CD-ROM*. John Wiley & Sons, New York, NY, 2004.

[10] R. Hecht-Nielsen. Theory of the back propagation neural network. In *International Joint Conference on Neural Networks*, pages 593–605, June 1989.

[11] J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, USA, 1975.

[12] J. Honerkamp and J. Weese. Tikhonov's regularization method for ill-posed problems. *Continuum Mechanics and Thermodynamics*, 2:17–30, 1990. 10.1007/BF01170953.

[13] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[14] G.-B. Huang, L. Chen, and C.-K. Siew. Universal approximation using incremental constructive feedforward networks with random hidden nodes. *IEEE Transactions on Neural Networks*, 17(4):879–892, July 2006.

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

[16] F. H. F. Leung, H. K. Lam, S. H. Ling, and P. K. S. Tam. Tuning of the structure and parameters of neural network using an improved genetic algorithm. *IEEE Transactions on Neural Networks*, 14(1):79–88, January 2003.

[17] 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):158–162, 2010.

[18] M. H. Mohamed. Rules extraction from constructively trained neural networks based on genetic algorithms. *Neurocomputing*, 74(17):3180–3192, 2011.

[19] M. Sadeghian and A. Fatehi. Identification of nonlinear predictor and simulator models of a cement rotary kiln by locally linear neuro-fuzzy technique. In *Proc. 2nd IEEE International Conference on Computer and Electrical Engineering (ICCEE 2009)*, pages 168–173, December 2009.

[20] M. Shoaib, M. Balahab, and A. Abdel-Rahman. Influence of cement kiln dust substitution on the mechanical properties of concrete. *Cement and Concrete Research*, 30(3):371–377, March 2000.

[21] F. Souza, T. Matias, and R. Araújo. Co-evolutionary genetic multilayer perceptron for feature selection and model design. In *IEEE 16th Conference on Emerging Technologies Factory Automation (ETFA 2011)*, pages 1–7, September 2011.

[22] J.-T. Tsai, J.-H. Chou, and T.-K. Liu. Tuning the structure and parameters of a neural network by using hybrid taguchi-genetic algorithm. *IEEE Transactions on Neural Networks*, 17(1):69–80, January 2006.

[23] J.-T. Tsai, T.-K. Liu, and J.-H. Chou. Hybrid taguchi-genetic algorithm for global numerical optimization. *IEEE Transactions on Evolutionary Computation*, 8(4):365–377, August 2004.

[24] J. Xu, Y. Lu, and D. W. C. Ho. A combined genetic algorithm and orthogonal transformation for designing feedforward neural networks. In *Third International Conference on Natural Computation*, volume 1, pages 10–14, August 2007.

[25] X. Yao. Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9):1423–1447, 1999.