Fifth International Conference
Modelling and Development of Intelligent Systems
June 23 - 25, 2017
"Lucian Blaga" University
Sibiu - Romania

# Tuning Extreme Learning Machines with genetic algorithms

**Florin Stoica, Alina Bărbulescu, Laura Florentina Stoica**

**Abstract**

In this paper we propose a method of optimizing the predictions made with Extreme Learning Machines (ELM) by optimizing their structure. The method is based on generic algorithms to determine the optimal number of hidden nodes and also to determine the appropriate activation function. In our study the ELMs are optimized through the Breeder genetic algorithm aiming to minimize the prediction error for the sum of permanent premolars and canines dimensions from a group of 92 Romanian young people. A comparison is made with predictions obtained by multiple linear regression equations (MLRE).

## 1 Introduction

H.A.Boboc [4] developed multiple linear regression equations (MLRE) that investigates the relationship between the size of canine-premolar group and mesiodistal size of other teeth. The estimation of the mesiodistal size of permanent canine and of the two premolars before their eruption is important for early evaluation of the need for space in this area and represents an important part of diagnosis and orthodontic treatment strategy.
Our paper aims to verify if the Extreme Learning Machines with their structure optimized through genetic algorithms can improve the accuracy of the predictions provided by the original method based on MLRE.
A representative public school with a population of 321 children ages 12-15 years from Sibiu (Romania) was selected for this study. From these subjects, a random simple technique was used to select 92 students (47 females and 45 males) fulfilling the selection criteria:

- To have the parents' written consent to participate in the study;
- To present the dental arches fully erupted permanent teeth (molars 3 was not considered);
- The erupted teeth show no abnormalities of shape, size or structure;
- The teeth must not have missing of substance in the mesiodistal size, due to decay, trauma or orthodontic treatments have provided striping [8].

The measure tooth size models we used a digital calliper manufactured by Mega (Germany) with an accuracy of 0.01 mm. All models were measured 2 times by the same author and the result used was the average of two values.
For estimation the size of the unerupted canines and premolars, a recently multiple linear regression proposed equation [4] is based on known variables 21, 42 and 46. The form of this equation is: $Y = X_1 \times A_1 + X_2 \times A_2 + X_3 \times A_3 + A$, where:

- Y is the outcome expected;

- X1, X2, X3 are independent variables determined by the size of teeth 42, 46, 21;
- A1, A2 and A3 are regression coefficients for used teeth;
- A is a specific constant.

The values of constant A and regression coefficients of the equation are presented in Table 1:

| Canines premolars group | Constant A | $A_1$ ( 42) | $A_2$ (46) | $A_3$ (21) |
|---|---|---|---|---|
| Maxillary | 6,563 | 0,822 | 0,595 | 0,411 |
| Mandible | 3,350 | 0,872 | 0,710 | 0,538 |

*Table 1:* Parameters of multiple linear regression equation used [4]

In the following is presented our approach to provide a more accurate method for prediction of the mesiodistal width of unerupted permanent canines and premolars, using Extreme Learning Machines.

## 2 Extreme Learning Machines – fundamental concepts

Due to their remarkable efficiency, simplicity, and impressive generalization performance, ELMs have been applied in a variety of domains, such as control and robotics, computer vision, system identification, classification and regression.
The learning speed of feedforward neural networks is in general far slower than required and it has been a major bottleneck in their applications for past decades (the slow gradient-based learning algorithms are extensively used to train neural networks). Moreover, usually the parameters of the networks are tuned iteratively by using such learning slower algorithms.
Huang et al proposes a new learning algorithm called extreme learning machine (ELM) for single-hidden layer feedforward neural networks (SLFNs) [1].
In [1] is proved that the input weights and hidden layer biases of SLFNs can be randomly assigned if the activation functions in the hidden layer are infinitely differentiable. SLFNs can be simply considered as a linear system and the output weights (linking the hidden layer to the output layer) of SLFNs can be analytically determined through Moore–Penrose generalized inverse operation of the hidden layer output matrices.
For *N* arbitrary distinct samples $(x_i, t_i)$, where $x_i = [x_{i1}, \ x_{i2}, \ . \ . \ ., \ x_{in}]^T \in \boldsymbol{R}^n$ and $t_i = [t_{i1}, \ t_{i2}, \ . \ . \ ., \ t_{im}]^T \in \boldsymbol{R}^m$ standard SLFNs with $\widetilde{\boldsymbol{N}}$ hidden nodes and activation function *g(x)* are mathematically modeled as:

$$\sum_{i=1}^{\widetilde{N}} \beta_i \, g(w_i * x_j + b_i) = o_j$$

*j = 1, . . . ,* $\boldsymbol{N}$ where:

- $w_i = [w_{i1}, \ w_{i2}, \ . \ . \ ., \ w_{in}]^T$ is the weight vector connecting the *i*th hidden node and the input nodes;
- $\beta_i = [\beta_{i1}, \ \beta_{i2}, \ . \ . \ ., \ \beta_{im}]^T$ is the weight vector connecting the *i*th hidden node and the output nodes;
- $b_i$ is the threshold of the *i*th hidden node;

That standard SLFNs with $\widetilde{\boldsymbol{N}}$ hidden nodes with activation function *g(x)* can approximate these *N* samples with zero error if $\sum_{j=1}^{N} \|o_j - t_j\| = 0$

In other words there exist $\beta_i$ , $w_i$ and $b_i$ such that:

$$\sum_{i=1}^{\tilde{N}} \beta_i \, g\left(w_i * x_j + b_i\right) = t_j, \; j = 1, \ldots, \boldsymbol{N}$$

The above N equations can be written as

$$\boldsymbol{H} \, \beta = \boldsymbol{T}$$

where ***H*** is called the ***hidden layer output matrix*** of the neural network [2] and can be described as follows:

$$\boldsymbol{H}\,(w_1, \ldots w_{\tilde{N}}, b_1, \ldots b_{\tilde{N}}, x_1, \ldots x_{\boldsymbol{N}}) =$$

$$= \begin{bmatrix} g(w_1 * x_1 + b_1) & \cdots & g(w_{\tilde{N}} * x_1 + b_{\tilde{N}}) \\ \vdots & & \vdots \\ g(w_1 * x_{\boldsymbol{N}} + b_1) & \cdots & g(w_{\tilde{N}} * x_{\boldsymbol{N}} + b_{\tilde{N}}) \end{bmatrix}_{\boldsymbol{N} \times \tilde{\boldsymbol{N}}}$$

and $\beta$ respectively *T* are matrices of form:

$$\beta = \begin{bmatrix} \beta_1^{\boldsymbol{T}} \\ \vdots \\ \beta_{\tilde{\boldsymbol{N}}}^{\boldsymbol{T}} \end{bmatrix}_{\tilde{\boldsymbol{N}} \times m} \quad \text{and} \quad \boldsymbol{T} = \begin{bmatrix} t_1^{\boldsymbol{T}} \\ \vdots \\ t_{\boldsymbol{N}}^{\boldsymbol{T}} \end{bmatrix}_{\boldsymbol{N} \times m}$$

The ith column of H is the *i*th ***hidden node output*** with respect to inputs $x_1, \; x_2, \; . \; . \; ., \; x_{\boldsymbol{N}}$.
The input weights $w_i$ and the hidden layer biases $b_i$ are in fact not necessarily tuned and the hidden layer output matrix ***H*** can actually remain unchanged once random values have been assigned to these parameters in the beginning of learning.

For fixed input weights $w_i$ and the hidden layer biases $b_i$, ***to train*** an SLFN is simply equivalent to ***finding a least squares solution*** $\dot{\beta}$ of the linear system $\boldsymbol{H}\,\beta = \boldsymbol{T}$.
In most cases the number of hidden nodes is much less than the number of distinct training samples $\tilde{\boldsymbol{N}} \ll N$ , ***H*** is a nonsquare matrix and there may not exist $w_i, b_i, \beta_i$ such that $\boldsymbol{H}\,\beta = \boldsymbol{T}$
The smallest norm least squares solution of the above linear system is:

$$\dot{\beta} = \boldsymbol{H}^* \boldsymbol{T}$$

where $H^*$ is the *Moore–Penrose* generalized inverse of matrix *H i.e.*

$HH^*H = H$
$H^*HH^*=H^*$
$(HH^*)^T=HH^*$
$(H^*H)^T=H^*H$

$\boldsymbol{H}^*$ can be calculate using the singular value decomposition (SVD) method [6].

## 3 Tuning ELMs with Breeder genetic algorithm

The Breeder genetic algorithm, proposed by Mühlenbein and Schlierkamp-Voosen [3] represents solutions (chromosomes) as vectors of ***real numbers***, much closer to the reality than normal GAs. The ***selection*** is achieved randomly from the *T*% best elements of current population, where *T* is a constant of the algorithm (usually, *T* = 40 provide best results).

Within each generation, from the *T*% best chromosomes are selected two elements, and the ***crossover*** operator is applied over them.
On the new child obtained from the mate of the parents is applied the mutation operator. The process is repeated until are obtained *N-1* new individuals, where *N* represents the size of the initial population. The best chromosome (evaluated through fitness function) is inserted in the new population (1-elitism). Thus, the new population will have also *N* elements.

## 3.1 The Breeder genetic operators

### 3.1.1 Crossover

Let be $\mathrm{x} = \{\mathrm{x}_1, \mathrm{x}_2, ..., \mathrm{x}_\mathrm{n}\}$ and $\mathrm{y} = \{\mathrm{y}_1, \mathrm{y}_2, ..., \mathrm{y}_\mathrm{n}\}$ two chromosomes, where $x_i \in R$ and $y_i \in R, i = \overline{1,n}$. The crossover operator has a result a new chromosome, whose genes are represented by values $z_i = x_i + \alpha_i(y_i - x_i),\ i = \overline{1,n}$, where $\alpha_i$ is a random variable uniformly distributed between $[-\delta, 1+\delta]$, and $\delta$ depends on the problem to be solved, typically in the interval $[0,0.5]$.

### 3.1.2 Mutation

The probability of mutation is typically selected as $1/n$. The mutation scheme is given by $x_i = x_i + s_i \cdot r_i \cdot a_i,\ i = \overline{1,n}$ where: $s_i \in \{-1, +1\}$ uniform at random, $r_i$ is the range of variation for $x_i$, defined as $r_i = r \cdot domain_{x_i}$, where $r$ is a value in the range between 0.1 and 0.5 (typically 0.1) and $domain_{x_i}$ is the domain of the variable $x_i$ and $a_i = 2^{-k\cdot\alpha}$ where $\alpha \in [0,1]$ uniform at random and $k$ is the number of bytes used to represent a number in the machine within is executed the Breeder algorithm (mutation precision).

## 3.2 The Breeder genetic algorithm

With the operators described in the previous section, the Breeder algorithm can be described as follows [7]:

```
Procedure Breeder
begin
t = 0
Randomly generate an initial population P(t) of N individuals
while (termination criterion not fulfilled) do
     Evaluate P(t) using the fitness function
     for i = 1 to N-1 do
         Randomly choose two elements from the T% best elements of P(t)
         Apply the crossover operator
         Apply the mutation operator on the child
         Insert the result in the new population P′(t)
     end for
     Choose the best element from P(t) and insert it into P′(t)
     P(t+1) = P′(t)
     t = t + 1
end while
end
```

## 3.3 Tuning parameters of ELM

The aim of the Breeder genetic algorithm is to find optimal values for the parameters of the ELM (the number of hidden nodes $\tilde{N}$ and the activation function g). Each chromosome contains two genes, representing values associated with modeled variables. The fitness function for chromosomes evaluation is represented by the train error of the represented ELM on train data sets.

In our tests, parameters of Breeder algorithm are assigned with following values: $\delta = 0$, $r = 0.1$, $a = 0.1$ and $k = 8$. The initial population has 100 chromosomes and algorithm is stopped after 100 generations.

Data provided by our study models was randomly divided in two sets: the training set, containing 50 cases and the validation set, composed by 42 study models (the data comes from 92 children ages 12-15 years fulfilling the selection criteria).

For the ELM, we chose between two activation functions, „*sig*” and „*sin*”.

The original implementation of the elm-java tool can be found at http://www3.ntu.edu.sg/home/egbhuang/elm_codes.html [5].

## 3.4 Results

Using the data from training set, the Breeder algorithm has determined as optimal values for ELM parameters the “sig” as activation function and 15 as optimal value for the number of hidden nodes $\tilde{\boldsymbol{N}}$.

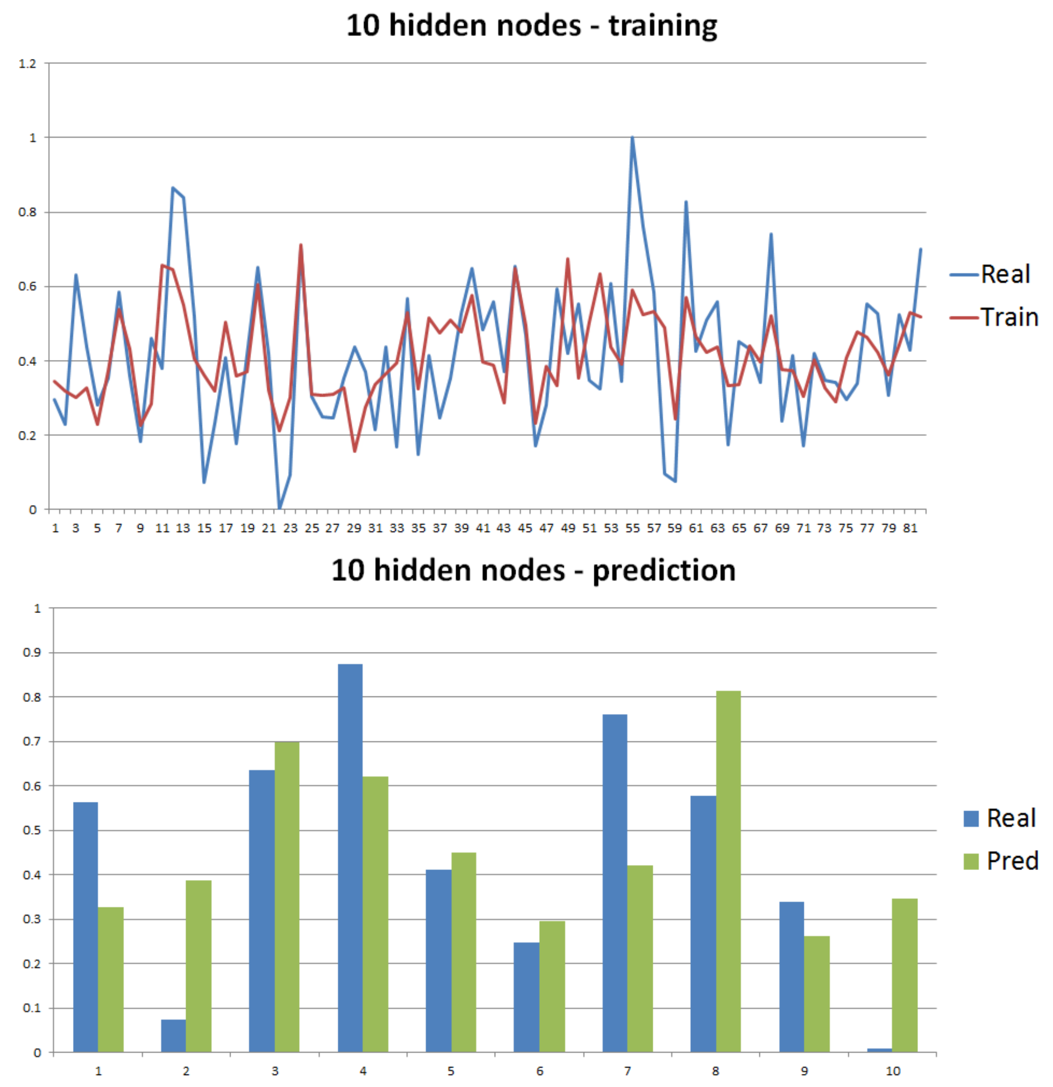
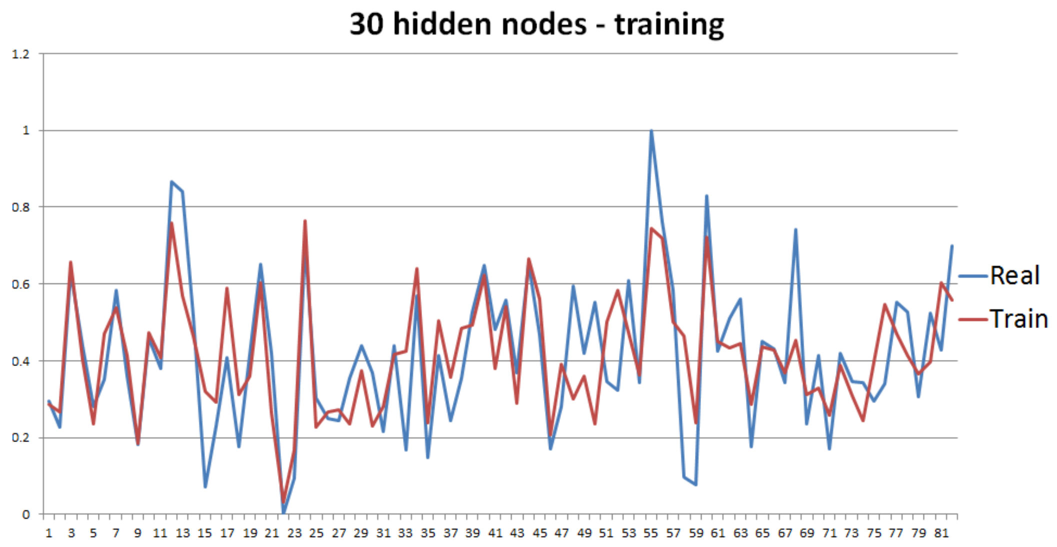


*Figure 1:* Training and prediction errors for ELM with $\tilde{\boldsymbol{N}} = 10$



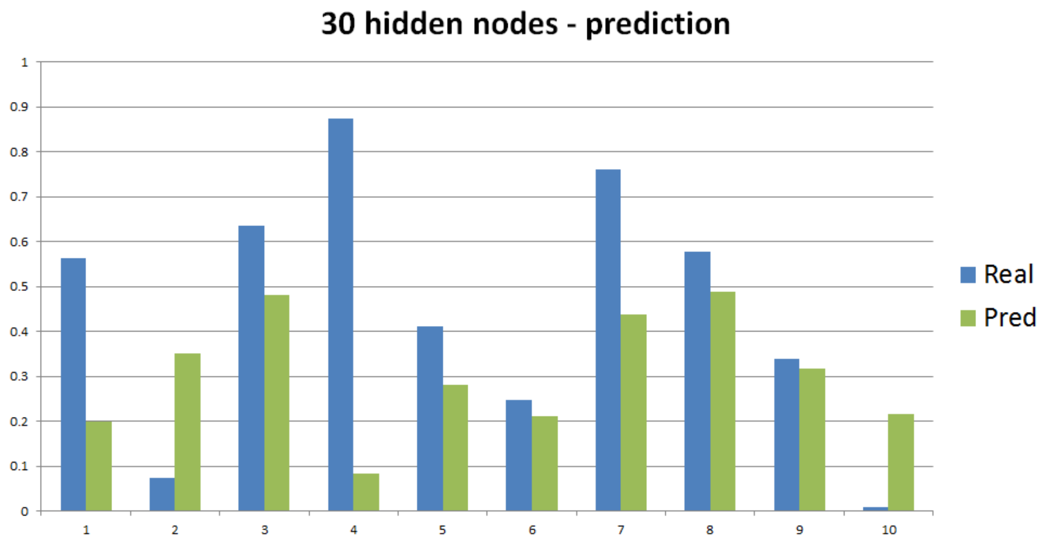
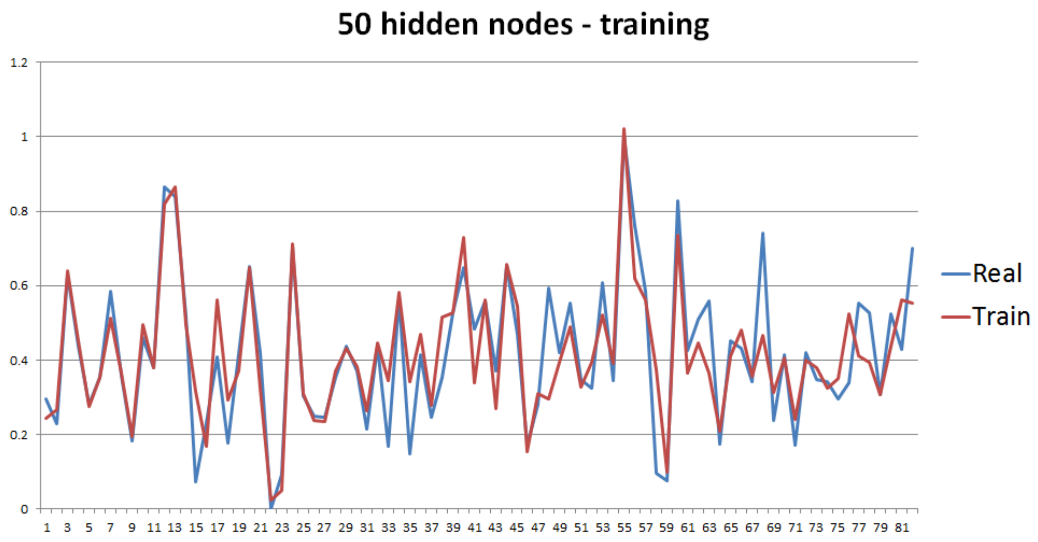


*Figure 2:* Training and prediction errors for ELM with $\widetilde{\boldsymbol{N}} = 30$



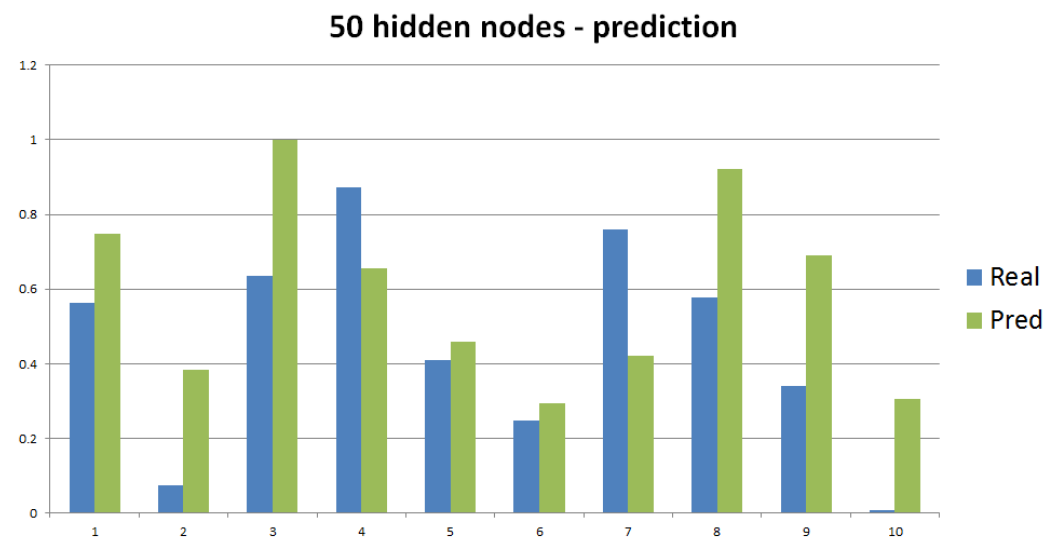
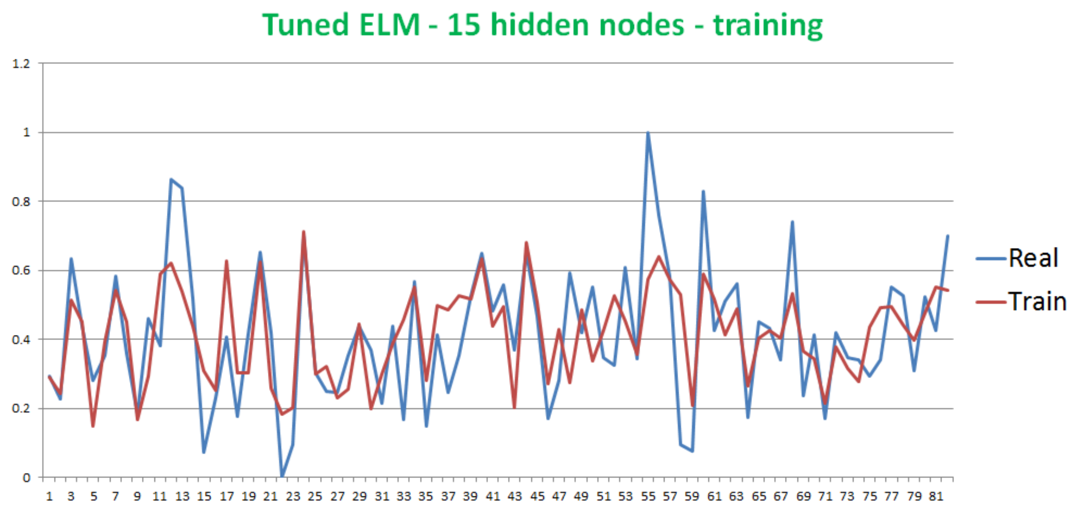


*Figure 3:* Training and prediction errors for ELM with $\tilde{\boldsymbol{N}} = 50$



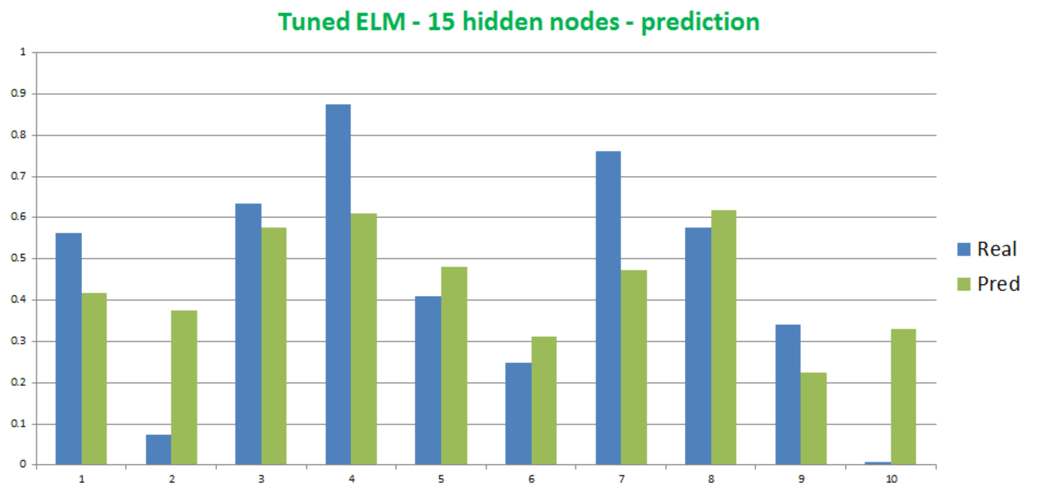


*Figure 4:* Training and prediction errors for ELM with $\tilde{\boldsymbol{N}} = 15$

In order to evaluate the performance of the proposed approach, we carried out comparisons of results obtained by the tuned ELM with results provided by other ELMs with empirically established architecture. Also we made a comparison of the prediction accuracy between the two methods (tuned ELM vs. Boboc's multiple linear regression equations) using the following performance metrics and their formula:

- The metric ***r*** (Pearson's correlation coefficient) with formula: $\dfrac{\sum_{i=1}^{n}(p_i - m_p)(a_i - m_a)}{\sqrt{\sum_{i=1}^{n}(p_i - m_p)^2}\sqrt{\sum_{i=1}^{n}(a_i - m_a)^2}}$

- The metric RMSE (Root Mean Square Error) with formula: $\sqrt{\frac{1}{n}\sum_{i=1}^{n}(p_i - a_i)^2}$

- The metric MSE (Mean Squared Error) with formula: $\frac{1}{n}\sum_{i=1}^{n}(p_i - a_i)^2$

- The metric MAE (Mean Absolute Error) with formula: $\frac{1}{n}\sum_{i=1}^{n}|a_i - p_i|$

where notations are: $p_i$ - predicted data; $a_i$ - observed (actual) data; $n$ - number of data; $m_a$ - mean of observed data; $m_p$ - mean of predicted data.

The comparison of performances of tuned vs. non-tuned ELMs are presented in Table 2.

| | $\tilde{\boldsymbol{N}}$ | 10 | 15 | 30 | 50 |
|---|---|---|---|---|---|
| Training | ***r*** | 0.6007745 | 0.6864645 | 0.7676291 | 0.8751079 |
| | RMSE | 0.1580064 | 0.143725 | 0.1266738 | 0.09564912 |
| | MSE | 0.02496602 | 0.02065688 | 0.01604626 | 0.009148755 |
| | MAE | 0.127673 | 0.1103865 | 0.09795375 | 0.06541024 |
| Prediction | ***r*** | 0.5519677 | **0.7211824** | 0.09139431 | 0.5684539 |
| | RMSE | 0.2272806 | **0.199109** | 0.3211489 | 0.2759233 |
| | MSE | 0.05165646 | **0.03964439** | 0.1031366 | 0.07613369 |
| | MAE | 0.1943521 | **0.1671411** | 0.2393244 | 0.2504432 |

*Table 2*: Performance comparison with tuned ELM ($\tilde{\boldsymbol{N}} = 15$)

Comparing predictions provided by the proposed (ELM) and respectively MLRE method, we can conclude that the ELM tuned by a Breeder genetic algorithm is capable to provide greater accuracy in prediction of the mesiodistal width of unerupted teeth as can be seen in the figure 5 and respectively in table 3.

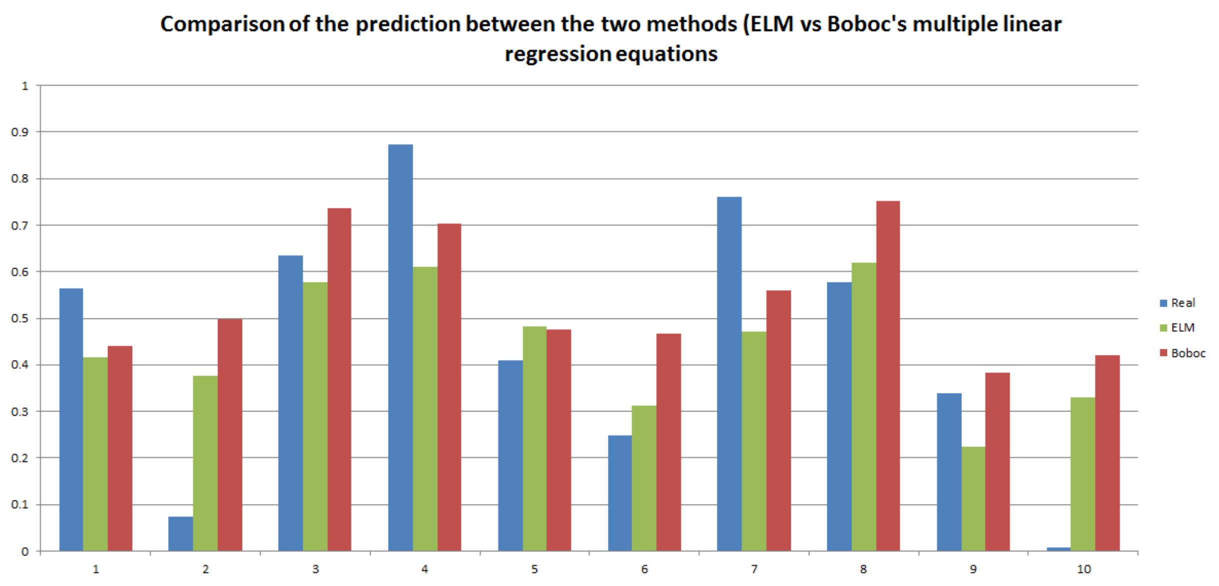


*Figure 5:* Comparing prediction errors for the two methods: ELM and MLRE

| | | ELM (15 nodes) | Boboc (MLRE) |
|---|---|---|---|
| Prediction | ***r*** | **0.7211824** | 0.6519263 |
| | RMSE | **0.199109** | 0.230186 |
| | MSE | **0.03964439** | 0.05298558 |
| | MAE | **0.1671411** | 0.1938037 |

*Table 3:* Comparing prediction errors (ELM and MLRE) using performance metrics

## 4 Conclusions

Tuning the ELM using a genetic algorithm provide better results than empirical approach. The fast learning speed of an Extreme Learning Machine is crucial in obtaining a reasonable time for the tuning process. After evaluation, we found that our proposed method is providing a better prediction than original MLRE method. Thus, the prediction error rates of tuned ELM using the Breeder genetic algorithm are smaller than those provided by the multiple linear regression equations proposed in [4]. We intend to do more tests to be done on large data sets. Also, will be interesting to compare an ELM with a backpropagation trained neural network in terms of speed and prediction accuracy.

## References

[1] G.-B. Huang, Q.-Y. Zhu, C.-K. *Siew Extreme learning machine: Theory and applications*, Neurocomputing *70* (2006) 489–501
[2] G.-B. Huang, *Learning capability and storage capacity of two hidden-layer feedforward networks*, IEEE Trans. Neural Networks, 14 (2) (2003) 274–281.
[3] Mühlenbein H, Schlierkamp-Voosen D, *The science of breeding and its application to the breeder genetic algorithm*, Evolutionary Computation, vol. 1, 1994, 335-360
[4] Boboc A, Dibbets J, *Prediction of the mesiodistal width of unerupted canines and premolars: a statistical approach, American Journal of Orthodontics and Dentofacial Ortopedics*, 2010, vol 137(4);503-507
[5] Dong Li, *Java version of ELM*, http://www3.ntu.edu.sg/home/egbhuang/elm_codes.html.
[6] J.M. Ortega, *Matrix Theory*, Plenum Press, New York, London, 1987.

[7] Stoica F., Simian D., *Optimizing a New Nonlinear Reinforcement Scheme with Breeder genetic algorithm*, Proceedings of the 11th International Conference on EVOLUTIONARY COMPUTING (EC'10),13-15 June 2010, Iaşi, Romania, ISSN: 1790-2769, ISBN: 978-960-474-194-6, pp. 273-278

[8] Florin Stoica, Cornel Gheorghe Boitor, *Using the Breeder genetic algorithm to optimize a multiple regression analysis model used in prediction of the mesiodistal width of unerupted teeth*, International Journal of Computers, Communications & Control, Vol 9, No 1, pp. 62-70, ISSN 1841-9836, february 2014, Impact Factor 2014 (JCR2013/Science Edition) 0.694

Florin STOICA
“Lucian Blaga” University of Sibiu
Faculty of Sciences
Mathematics and Informatics Department
Dr. Ion Rațiu St, 5-7 No, Sibiu, 550012
ROMANIA
E-mail: florin.stoica@ulbsibiu.ro

Alina BĂRBULESCU
Higher College of Technology
Sharjah
EMIRATELE ARABE UNITE
E-mail: alinadumitriu@yahoo.com

Florentina Laura STOICA
“Lucian Blaga” University of Sibiu
Faculty of Sciences
Mathematics and Informatics Department
Dr. Ion Rațiu St, 5-7 No, Sibiu, 550012
ROMANIA
E-mail: laura.cacovean@ulbsibiu.ro