Extreme Learning Machine with initialized hidden weight

L. D. Tavares
R. R. Saldanha
Graduate Program in Electrical Engineering
Federal University of Minas Gerais
Av. Antônio Carlos 6627, 31270-901,
Belo Horizonte, MG, Brazil
Email: tavares@dcc.ufmg.br, rodney@cpdee.ufmg.br

D. A. G. Vieira
ENACOM - Handcrafted Technologies
Rua Professor José Vieira de Mendonça, 770, offices 406 and 407
Belo Horizonte Techonology Park (BH-TEC)
Belo Horizonte, MG, Brazil
Email: enacom@enacom.com.br

***Abstract*—The Extreme Learning Machine (ELM) is a recent training method for feedforward neural networks. Its main advantage is a faster and simpler training procedure when it is compared with traditional global search optimization method. It is achieved by using a least square solution for the output layer and random initialization for hidden layer. In this way only one solution is attained. In this sense, a question arises: is the random initialization method really an efficient for ELM? The present work studies the influence of more sophisticated methods of initialization, in terms of performance and complexity.**

I. Introduction

Since the pioneer Huang et al.' work [1], [2], [3], the Extreme Learning Machines (ELM) has been widely discussed and used in several fields. The ELM is a Single Layer Feedforward Neural Network (SLFNs) learning method that simplifies weight initalization and trainning processes. The ELM differs from the traditional methods of learning by randomly selecting the input weights and, subsequently, analytically determining the output weights by means of linear least square solutions. Therefore, there is no need of gradient descent based algorithms, such as back propagation or other known global search methods. The universal approximation capability of SFLN has been proved in 1989 by Cybenco [4] and Funahashi [5]. More recently, Huang and Babri [6] demonstrated its learning upper bounds.

Among the main advantages of ELM, stand out: the training is extremely fast (when compared to other conventional learning algorithms); requires less parameter settings; and the result presents a good generalization performance [7]. ELM has been used in several application such as: time series prediction [8], text classification [9], pattern recognition [10], feature and variable selection [11]. A list of major applications using ELM and its state of art can be found in [12].

In this scenario, two questions rise: (i) is there any advantage in initialize the hidden layer weights? and (ii) what is the influence of random and other initialization methods over the ELM performance?

The objective of this work is to present some experiments that demonstrate that initializing the hidden layer weights can improve the ELM result (in terms of empirical risk minimization), without loss of its simplicity and without any expressive increase in the total learning time. There are several initialization methods that are computationally efficient, since those that are simpler and purely random, passing by those using information from the network structure and/or training data, to the most sophisticated using prototypes and evolutionary computation.

The paper is organized as follows: a brief explanation about ELM traning method is present in the section II, then, in the section III, the neural network initialization method used in the experiments are described. The experiments and main result are shown in the section IV. Finally the conclusion and future works are discussed in the section V.

II. Extreme Learning Machine trainning

Consider a set of N distinct samples in the form $(\mathbf{x}_i, \mathbf{t}_i)$ where $\mathbf{x}_i = [x_{i1}, x_{i2}, \ldots, x_{in}]' \in \mathbb{R}^n$ are the inputs of real system, with $i = 1, 2, ...N$, apostrophe symbol $(')$ means the transpose of the vector or matrix, and $\mathbf{t}_i = [t_{i1}, t_{i2}, \ldots, t_{im}]' \in \mathbb{R}^m$ are the real (or desired) system response. Consider also h as the number of hidden nodes and $f(\cdot)$ as activation function, the SLFN is modeled as [2]:

$$\mathbf{o}_i = \sum_{j=1}^{h} \beta_j f(\mathbf{x}_i' \mathbf{w}_j + b_j),\ i = 1, 2, \ldots, N \tag{1}$$

where $\mathbf{o}_i = [o_{i1}, o_{i2}, \ldots, o_{im}]' \in \mathbb{R}^m$ are the responses found by SLFNs, $\mathbf{w}_i = [w_{i1}, w_{i2}, \ldots, w_{ih}]' \in \mathbb{R}^h$ is the weight vector connecting the input and hidden neurons, $\beta_i = [\beta_{i1}, \beta_{i2}, \ldots, \beta_{im}]' \in \mathbb{R}^m$ connecting hidden and output layer, b_i is the threshold of the ith hidden neuron.

In a matrix compact form we get:

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

where:

$$\mathbf{H} = \begin{bmatrix} f(\mathbf{x}_1' \mathbf{w}_1 + b_1) & \ldots & f(\mathbf{x}_N' \mathbf{w}_h + b_h) \\ \vdots & \ldots & \vdots \\ f(\mathbf{x}_1' \mathbf{w}_1 + b_1) & \ldots & f(\mathbf{x}_N' \mathbf{w}_h + b_h) \end{bmatrix} \in \mathbb{R}^{N \times h} \tag{3}$$

978-1-4799-4905-2/14/$31.00 ©2014 IEEE

$$\beta = [\beta_1, \ldots, \beta_h]' \text{ and } \mathbf{T} = [\mathbf{t}_1, \ldots, \mathbf{t}_N]' \tag{4}$$

As aforementioned, the ELM adjust randomly the values of w and b. Now the objective is to evaluate the value of β such that:

$$\min_{\beta} \|\mathbf{T} - \mathbf{H}\beta\|_p \tag{5}$$

where p indicates the kind of norm used (1, 2, ∞, ...). For $p = 2$ the value of this β can be found by using least squares solution, as:

$$\begin{aligned} \xi &= (\mathbf{T} - \mathbf{H}\beta)'(\mathbf{T} - \mathbf{H}\beta) \\ &= \mathbf{T}'\mathbf{T} - \mathbf{T}'\mathbf{H}\beta - \mathbf{H}'\beta'\mathbf{T} + \mathbf{H}'\beta'\mathbf{H}\beta \end{aligned} \tag{6}$$

Solving $\frac{\partial \xi}{\partial \beta} = 0$ we get:

$$\frac{\partial \xi}{\partial \beta} = -(\mathbf{T}'\mathbf{H})' - (\mathbf{H}'\mathbf{T}) + (\mathbf{H}'\mathbf{H} + \mathbf{H}'\mathbf{H})\beta \tag{7a}$$

$$\mathbf{H}'\mathbf{H}\beta = \mathbf{H}'\mathbf{T} \tag{7b}$$

$$\beta = \mathbf{H}^{+}\mathbf{T} \tag{7c}$$

where equation (7b) is called normal equation, $\mathbf{H}^{+}$ is the pseudoinverse of $\mathbf{H}$, which most widely known type of matrix pseudoinverse is the Moore-Penrose pseudoinverse.

III. Neural networks initialization methods

This section will briefly describe the initialization methods used in the experiments and its main characteristics. The first weight initialization method is the easiest one, that consists in randomly initiate in an uniform $[-r, r]$ range. In this method the input-output values and neural network structure, such as number of hidden neurons, are ignored. In the same way, Haffner et. al. proposed that the weights might be initialized in a normal distribution with 0 means and unit variance [13]. Approaches of this type are recommended when nothing is known about the training data and is used generally in the early stages of the neural network design.

On the other hand, some methods use the neural network structure in order to generate the weight values. Wessels and Barnard [14] proposed using $r = 3/\sqrt{d_{in}}$ where d_{in} is the number of neuron input. In the same way, Le Cun [15] proposed $r = \sqrt{d_{in}}$ and, according to the author, this range favors that the activation function works in its active region.

Other methods use the neural network structure with the knowledge of the input data. Yam and Chow [16] proposed $r = \frac{-8.72}{b}\sqrt{\frac{3}{d_{in}}}$, where $b = \sqrt{\sum_{i=1}^{m}(max(x_i) - min(x_i))^2}$.

The two last initialization methods used in this work are: (i) Nguyen & Widrow method [17] and (ii) Statistically Controlled Activation Weight Initialization [18].

The Nguyen & Widrow method is a well known method and it is currently implemented at Matlab neural network toolbox. This method performs the initialization into two stages, being the first completely random and the second stage of adjustment, as follows:

1) $\mathbf{w} \approx U(-1, 1)$
2) $\mathbf{w} = \frac{\beta \mathbf{w}}{\gamma}$

where U is a random uniform distribution, $\beta = 0.7^{\frac{1}{d_{in}}}$ and $\gamma = \sqrt{\sum(w)^2}$. The justification given by the authors is that all input values must excite at most 70% of the active region of the activation function.

Also a well known method, but with a more refined statistical justification, is called Statistically Controlled Activation Weight Initialization (SCAWI) described by Drago and Ridella in [18]. The main purpose of the method is to allow a small percentage of values that are not in the active region of neurons, called PNP - percentage of neurons paralyzed. This value is obtained by testing how many times the neuron enters a saturated area and the magnitude of the largest error rate. Furthermore, the initialization of weights also give the statistical properties of the input data, such as variance, for example. The initialization is made as:

$$\mathbf{w} = \frac{1.3}{\sqrt{1 + d_{in}v^2}} r_{i,j} \tag{8}$$

where $v = \sum(w)^2/m$ and $r \approx U(-1, 1)$.

IV. Experiments and results

This section evaluates the performance of initialized ELM learning methods. All simulations were carried out in MATLAB environment version 7.12 (R2011a) 32 bits running in a Intel(R) Core(TM) i3, 2.40GHz CPU under Ubuntu 13.10 operating system. All initialized ELMs used the simple sigmoidal activation function $f(u) = 1/(1 + exp(-u))$ in the hidden layer. Twelve well known benchmarks problems were chosen for experiments. The benchmarks data set collected from UCI Machine Learning Repository [19], all of them for regression purpose. In all benchmark experiments the inputs have been normalized into range $[0, 1]$ (even for nominal features), and the targets into range $[-1, 1]$.

Three measures were used to compare the initialization methods. The first is the empirical error, the second is the norm of the weights of the output layer and the third, and last, is the measure of time spent by the initialization method. The Root-Mean-Square-Error (RMSE) was chosen as the empirical error measure. The norm of parameters β is the euclidean norm as $\|\beta\|_2$.

The characteristic of each data set, number of data used for training and testing are presented in table I.

For each data set were arbitrarily setted with a distinct number of hidden node. The number of hidden nodes are presented in the table II.

Each experiment was executed 50 times the average and standard deviation are presented. The first comparison is related to the empirical risk. The results for the training and testing phase, are presented in tables III and IV, respectively. The best results in both, i.e. the lowest results for RMSE, are marked with the symbol *.

Method	Random	D&R [18]	N&W [17]	Y&C [16]	W&B[14]	L[15]	H et. al. [13]
Data set							
Abalone	0.1207(0.0018)	0.0733(0.0013)	0.0733(0.0013)	0.0740(0.0013)	0.0734(0.0011)	*0.0730(0.0013)	0.0737(0.0012)
Auto price	0.0679(0.0054)	0.0021(0.0007)	*0.0019(0.0005)	0.0020(0.0005)	0.0022(0.0006)	0.0020(0.0004)	0.0022(0.0005)
Bank	0.0479(0.0003)	*0.0114(0.0002)	0.0116(0.0002)	0.0115(0.0002)	0.0157(0.0014)	0.0117(0.0003)	0.0117(0.0004)
Breast cancer	0.2731(0.0105)	$*< 10^{-5}(-)$	$< 10^{-5}(-)$	$< 10^{-5}(-)$	$< 10^{-5}(-)$	$< 10^{-5}(-)$	$< 10^{-5}(-)$
California housing	0.1196(0.0006)	0.0661(0.0009)	*0.0660(0.0008)	0.0664(0.0009)	0.0662(0.0010)	0.0662(0.0010)	0.0662(0.0010)
Census (house 8L)	0.1079(0.0017)	0.0710(0.0020)	0.0713(0.0019)	0.0705(0.0016)	0.0731(0.0021)	*0.0704(0.0020)	0.0706(0.0015)
Computer activity	0.0472(0.0010)	0.0097(0.0005)	0.0094(0.0004)	0.0097(0.0005)	*0.0088(0.0003)	0.0095(0.0004)	0.0090(0.0006)
Delta ailerons	0.0352(0.0005)	*0.0191(0.0003)	0.0192(0.0004)	0.0193(0.0004)	0.0196(0.0003)	0.0192(0.0003)	0.0193(0.0003)
Delta elevators	0.0301(0.0004)	0.0284(0.0003)	0.0283(0.0004)	0.0284(0.0003)	0.0283(0.0003)	*0.0282(0.0003)	0.0283(0.0003)
Machine cpu	0.0391(0.0063)	0.0050(0.0008)	0.0048(0.0006)	*0.0044(0.0006)	0.0047(0.0005)	0.0046(0.0005)	0.0047(0.0006)
Servo	0.1053(0.0075)	0.0425(0.0059)	0.0407(0.0058)	0.0405(0.0073)	*0.0388(0.0073)	0.0405(0.0068)	0.0411(0.0079)
Triazines	0.1250(0.0039)	0.0080(0.0019)	0.0091(0.0021)	0.0087(0.0021)	0.0069(0.0019)	0.0075(0.0017)	*0.0060(0.0016)

TABLE III. RMSE RESULT FOR REAL-WORLD BENCHMARK DATA SETS IN TRAINING FASE. THE BEST RESULT ARE MARKED BY *. BETWEEN PARENTHESES ARE THE STANDARD DEVIATION. THE VALUES SMALLER THAN 0.0001 FOR WERE IGNORED.

Method	Random	D&R [18]	N&W [17]	Y&C [16]	W&B[14]	L[15]	H et. al. [13]
Data set							
Abalone	0.1203(0.0017)	*0.0781(0.0031)	0.0785(0.0040)	0.0785(0.0045)	0.0793(0.0051)	0.0810(0.0069)	0.0780(0.0037)
Auto price	0.0824(0.0044)	0.0038(0.0077)	0.0031(0.0044)	0.0020(0.0018)	0.0016(0.0001)	*0.0002(−)	0.0027(0.0031)
Bank	0.0481(0.0003)	*0.0117(0.0002)	0.0117(0.0003)	0.0119(0.0003)	0.0160(0.0015)	0.0118(0.0003)	0.0120(0.0004)
Breast cancer	0.3186(0.0997)	0.2836(0.0589)	0.2341(0.0402)	0.2567(0.0667)	0.2980(0.0728)	*0.0703(0.0026)	0.1972(0.0302)
California housing	0.1195(0.0004)	0.0674(0.0010)	0.0672(0.0006)	0.0672(0.0010)	*0.0668(0.0008)	0.0674(0.0008)	0.0669(0.0012)
Census (house 8L)	0.1091(0.0013)	0.0722(0.0017)	*0.0716(0.0015)	0.0726(0.0020)	0.0747(0.0019)	0.0721(0.0017)	0.0719(0.0028)
Computer activity	0.0466(0.0009)	0.0113(0.0020)	0.0108(0.0013)	0.0114(0.0017)	*0.0099(0.0016)	0.0107(0.0011)	0.0102(0.0020)
Delta ailerons	0.0351(0.0004)	0.0197(0.0003)	0.0197(0.0003)	*0.0196(0.0003)	0.0199(0.0003)	0.0197(0.0002)	0.0196(0.0003)
Delta elevators	0.0301(0.0005)	*0.0284(0.0004)	0.0286(0.0005)	0.0285(0.0004)	0.0287(0.0004)	0.0286(0.0004)	0.0286(0.0005)
Machine cpu	0.0423(0.0029)	0.0339(0.0022)	0.0391(0.0034)	0.0282(0.0019)	*0.0274(0.0018)	0.0418(0.0007)	0.0388(0.0032)
Servo	0.1040(0.0068)	0.0600(0.0098)	0.0609(0.0068)	0.0622(0.0111)	0.0638(0.0140)	0.0614(0.0088)	*0.0596(0.0087)
Triazines	0.1229(0.0961)	0.0890(0.0088)	0.0554(0.0058)	0.1979(0.0259)	0.0991(0.0015)	*0.0039(0.0004)	0.0182(0.0014)

TABLE IV. RMSE RESULT FOR REAL-WORLD BENCHMARK DATA SETS IN TESTING FASE. THE BEST RESULT ARE MARKED BY *. BETWEEN PARENTHESES ARE THE STANDARD DEVIATION. THE VALUES SMALLER THAN 0.0001 FOR WERE IGNORED.

Data sets	# Observations		# Attributes	
	Training	Testing	Continuous	Nominal
Abalone	2,000	2,177	7	1
Auto price	80	79	14	1
Bank	4,500	3,692	8	0
Breast cancer	100	94	32	0
California housing	8,000	12,460	8	0
Census (house 8L)	10,000	12,784	8	0
Computer activity	4,000	4,192	8	0
Delta ailerons	3,000	4,19	6	0
Delta elevators	4,000	5,517	6	0
Machine cpu	100	109	6	0
Servo	80	87	0	4
Triazines	100	86	60	0

TABLE I. SPECIFICATION OF REAL-WORLD BENCHMARK DATA SETS.

Data sets	# of hidden nodes
Auto price	75
Bank	180
Breast cancer	570
California housing	30
Census (house 8L)	480
Computer activity	375
Delta ailerons	135
Delta elevators	30
Machine cpu	30
Servo	90
Triazines	30

TABLE II. SPECIFICATION OF ELM HIDDEN NODES FOR EACH DATA SET.

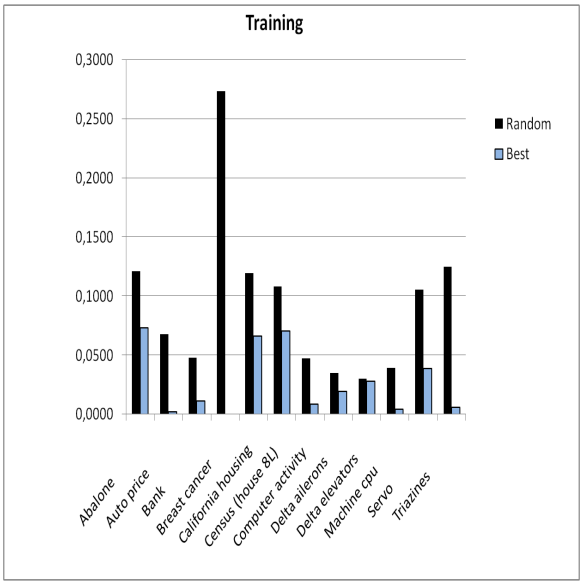

Fig. 1. Graphical comparison between the random initalization and the best result found, in training phase.

It is possible to note that in both cases (training and testing), despite initialization with random values present good results, this type of initalization has never achieved the best result. A graphical comparison between the best result and the random method, for training and testing phases, are shown in Figures 1 and 2, respectively. In the case of Breast cancer in training phase, the Drago and Ridella method had a better result, even if the result was below 10^{-5} as the others.

It is easy to observe that all other more sophisticated methods, than random initialization method, obtained best results. This can be explained by the fact that other methods take into account some characteristics of neural network topology and, in some cases, the data used in the training phase.

In this experiment it was verified that there is no a better initialization method. At the table III, for example, Drago & Ridella and Le Cun methods were the best in 3 cases each one, Nguyen & Widrow and Wessels & Barnard were the best in 2 cases each one, and, finally, Yam & Chow and Haffner et. al. where the best 1 case, each one.

The second comparison measure is the norm of the weights of the output layer. The average result is presented in table V. In this case the standard deviation is not show.

Once again the random initialization method has never achieved the best result. In some cases the difference between the best result and the random initialization method is around one order of magnitude.

One of the arguments used at work [1], is that the method will always result in a minimun norm of the weights, even if

Method	Random	D&R [18]	N&W [17]	Y&C [16]	W&B[14]	L[15]	H et. al. [13]
Data set							
Abalone	1.83e03	9.85e02	*2.56e02	3.22e05	1.63e05	5.79e03	2.60e02
Auto price	1.51e04	6.40e03	3.06e03	7.38e06	2.28e08	*2.69e03	2.92e04
Bank	8.21e01	*1.48e01	4.92e01	1.72e01	1.23e04	6.79e03	1.87e03
Breast cancer	9.39e02	5.58e02	*2.28e02	1.33e06	7.12e05	8.15e02	8.61e02
California housing	3.01e03	1.74e03	*3.69e02	5.66e05	2.81e05	5.15e03	4.17e02
Census (house 8L)	1.20e03	*1.87e02	9.01e02	2.13e02	2.55e05	1.26e05	1.93e04
Computer activity	1.48e03	7.28e02	*1.57e02	4.68e05	2.53e05	1.11e03	2.98e02
Delta ailerons	7.62e02	3.89e02	7.18e01	6.19e04	4.56e04	2.00e05	*6.29e01
Delta elevators	9.50e02	4.81e02	*1.02e02	1.05e05	5.87e04	2.85e05	1.05e02
Machine cpu	3.44e03	1.91e03	*3.07e02	3.17e02	4.96e05	2.52e05	4.02e03
Servo	2.30e03	*1.47e02	1.24e03	2.61e02	1.48e05	1.45e05	1.24e04
Triazines	5.86e02	4.18e02	*2.89e02	1.02e06	1.22e06	1.14e03	1.08e03

TABLE V. OUTPUT WEIGHT NORM RESULT FOR REAL-WORLD BENCHMARK DATA SETS. THE BEST RESULT ARE MARKED BY *.

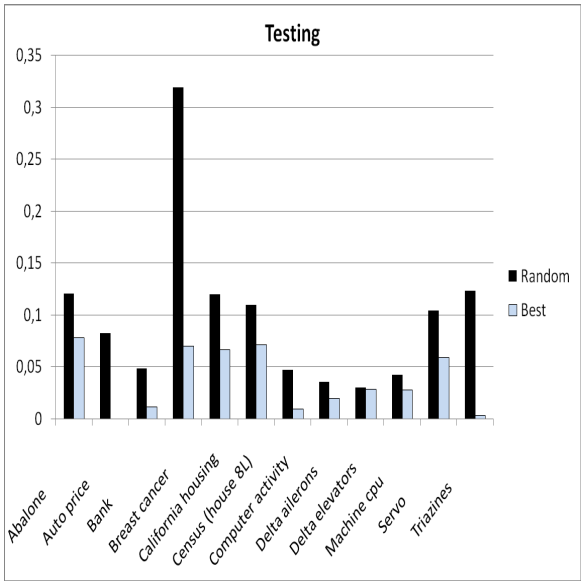

Fig. 2. Graphical comparison between the random initalization and the best result found, in testing phase.

the neural network is initialized randomly. However, as can be seen in table V, it is possible to achieve lower norms using others initalization methods. Therefore, we expected that by reducing the training error and, at the same time, the norm of the weight of the output layer, the testing error should be smaller [20]. The results demonstrated that the effect occurred as expected.

It is possible to observe that for most dataset Drago & Ridella [18] and Nguyen & Widrow [17] methods had a better result. One hypothesis for this is that both methods prioritize for the data of the hidden layer have the majority of their points of entry into the active region of the activation function.

In the case of multiple criteria comparison, a statistical comparison of the mean results must then be used, disregarding the standard deviation. First, all algorithms are ranked for each data set separately, the best performing algorithm getting rank of 1, the second best 2, and so on. In case of ties, average ranks are assigned. Average ranks by themselves provide a fair comparison of the algorithms as pointed out in [21]. The average ranks are presented in table VI

Based on table VI it is clear that the methods Drago & Ridella [18] and Nguyen & Widrow [17] have better performance in the three comparison criteria.

The third and last comparison is about the time spent for random initialization method and the best RMSE result in

Training RMSE	Testing RSME	Output weight norm
L[15] = 2.71	H et. al. [13] = 3.13	N & W [17] = 1.67
N & W [17] = 3.25	N & W [17] = 3.25	D & R [18] = 2.33
D & R [18] = 3.54	D & R [18] = 3.29	H et. al. [13] = 3.50
Y & C [16] = 3.79	L[15] = 3.42	Random = 4.33
W & B[14] = 3.79	Y & C [16] = 3.67	L[15] = 4.92
H et. al. [13] = 3.92	W & B[14] = 3.13	Y & C [16] = 5.00
Random = 7.00	Random = 7.00	W & B[14] = 6.42

TABLE VI. METHODS AVERAGE RANKS

training phase. The result are present in table VII.

Method	Random	Best RMSE result
Data set		
Abalone	0.0002(0.0001)	L[15]: 0.0005(−)
Auto price	0.0001(−)	N&W [17]: 0.0002(−)
Bank	0.0001(−)	D&R [18]: 0.0005(−)
Breast cancer	0.0004(−)	D&R [18]: 0.0004(−)
California housing	0.0001(−)	N&W [17]: 0.0008(0.0001)
Census (house 8L)	0.0004(−)	L[15]: 0.0006(−)
Computer activity	0.0003(−)	W&B[14]: 0.0015(0.0001)
Delta ailerons	0.0002(−)	D&R [18]: 0.0003(−)
Delta elevators	0.0001(−)	L[15]: 0.0003(−)
Machine cpu	0.0002(−)	Y&C [16]: 0.0003(−)
Servo	0.0002(−)	W&B[14]: 0.0003(−)
Triazines	0.0002(−)	H et. al. [13]: 0.0002(0.0001)

TABLE VII. OUTPUT WEIGHT NORM RESULT FOR REAL-WORLD BENCHMARK DATA SETS. BETWEEN PARENTHESES ARE THE STANDARD DEVIATION. THE VALUES SMALLER THAN 0.0001 FOR WERE IGNORED.

In the last comparison, it is evident that the use of more sophisticated methods for initialization of weights of the hidden layer does not increase the time spent significantly. In 11 of the 12 tested data sets, the time spent for the initialization process was less than 10 milliseconds, which can be considered an inconsiderable time.

V. CONCLUSION AND FUTURE WORKS

In the original Huang et al.' work [1] it was proposed a SLFN which initialization and training phases have been simplified. This was called ELM. But, in this context, some questions may be done. In this sense, this paper proposes a series of experiments related to neural networks initialization methods and what their influence in the ELM performance.

The seven methods chosen and used for the experiments are: (i) random initalization, (ii) Drago and Ridella [18], (iii) Nguyen & Widrow [17], (iv) Yam and Chow [16], (v) Wessels and Barnard [14], (vi) Le Cun [15] and (vii) Haffner et. al. [13]. Twelve well known UCI benchmark dataset were chosen for the experiments. For the experiments, three comparison measures were used: (i) empirical error RMSE, (ii) the norm of the output layer weights $\|\beta\|_2$ and (iii) time spent by the initialization method.

For comparison related to the RMSE was possible to verify that for all tested data base, the random initialization method had the worst performance. However, it has not been possible

to define, among the methods chosen for comparison, which was what was better for both, training and testing phase.

For the second measure of comparison, the norm of the weights of the output layer, the method of random initialization obtained, again, the worst results. In this sense, the methods Drago & Ridella [18] and Nguyen & Widrow [17] obtained the best results and, sometimes, with one order of magnitude smaller than the random method.

The last comparison, the time spent, showed that more sophisticated methods had initalization time viable and competitive, when compared to the random method. In most cases, the elapsed time is less than 10 milliseconds.

In conclusion, this study presented experiments which it was possible to verify that the initialization methods of neural networks are interesting when used with the ELM. Accordingly, further study should be conducted in order to further improve its performance.

As future work, we suggest: (i) a study of statistical analysis in more sophisticated initialization methods, (ii) use of other initialization methods, eg, those based on prototypes and evolutionary methods and (iii) study of a hybrid approach of initalization that is simple and fast as random method, which is as efficient as more sophisticated methods.

ACKNOWLEDGMENT

The authors would like to thank CNPq, FAPEMIG and CAPES for the financial support.

REFERENCES

[1] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: a new learning scheme of feedforward neural networks," in *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, vol. 2, 2004, pp. 985–990.

[2] ——, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, no. 1-3, pp. 489–501, 2006.

[3] G.-B. Huang, D. Wang, and Y. Lan, "Extreme learning machines: a survey," *International Journal of Machine Learning and Cybernetics*, vol. 2, no. 2, pp. 107–122, 2011.

[4] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals, and Systems (MCSS)*, vol. 2, no. 4, pp. 303–314, 1989.

[5] K. Funahashi, "On the Approximate Realization of Continuous Mappings by Neural Networks," *Neural Network*, vol. 2, no. 3, pp. 183–192, 1989.

[6] G.-B. Huang and H. A. Babri, "Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions." *IEEE Transactions on Neural Networks*, vol. 9, no. 1, pp. 224–229, 1998.

[7] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme Learning Machine for Regression and Multiclass Classification." *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 42, no. 2, pp. 513–529, 2012.

[8] R. Singh and S. Balasundaram, "Application of Extreme Learning Machine Method for Time Series Analysis," *International Journal of Intelligente Technology*, vol. 2, no. 4, pp. 256–262, 2007.

[9] D. Liu, H. Zhang, and S. Hu, "Neural networks: Algorithms and applications," *Neurocomputing*, vol. 71, pp. 471–473, 2008.

[10] N. Liang, P. Saratchandran, G. Huang, and N. Sundararajan, "Classification of mental tasks from EEG signals using extreme learning machine," *INTERNATIONAL JOURNAL OF NEURAL SYSTEMS*, vol. 16, no. 1, pp. 29–38, 2006.

[11] F. Mateo and A. Lendasse, "A variable selection approach based on the Delta Test for Extreme Learning Machine models," in *European Symposium on Time Series Prediction*, 2008.

[12] R. Rajesh and J. S. Parkash, "Extreme learning machine - A review and State-of-art," *Internationa Journal of Wisdom Based Computing*, vol. 1, no. 1, pp. 35–49, 2011.

[13] P. Haffner, K. Shikano, and A. Waibel, "Fast Back-Propagation Learning Methods for Neural Networks in Speech." Proceedings of the Fall Meeting of the Acoustical Society of Japan, 1988, pp. 619–624.

[14] L. F. Wessels and E. Barnard, "Avoiding false local minima by proper initialization of connections," *Trans. Neur. Netw.*, vol. 3, no. 6, pp. 899–905, 1992.

[15] Y. L. Cun, "Efficient learning and second-order methods," 1993, aT&T Bell Laboratories.

[16] J. Y. F. Yam and T. W. S. Chow, "Feedforward networks training speed enhancement by optimal initialization of the synaptic coefficients." *IEEE Transactions on Neural Networks*, vol. 12, no. 2, pp. 430–434, 2001.

[17] D. Nguyen and B. Widrow, "Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights," in *International Joint Conference on Neural Networks, 1990., 1990 IJCNN* , vol. 3, 1990, pp. 21–26.

[18] G. P. Drago and S. Ridella, "Statistically controlled activation weight initialization (SCAWI)," *Trans. Neur. Netw.*, vol. 3, no. 4, pp. 627–631, Jul. 1992.

[19] K. Bache and M. Lichman, "UCI Machine Learning Repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[20] D. A. G. Vieira, R. H. C. Takahashi, V. Palade, J. A. Vasconcelos, and W. M. Caminhas, "The Q-Norm Complexity Measure and the Minimum Gradient Method: A Novel Approach to the Machine Learning Structural Risk Minimization Problem," *IEEE Transactions on Neural Networks*, vol. 19, no. 8, pp. 1415–1430, 2008.

[21] J. Demšar, "Statistical Comparisons of Classifiers over Multiple Data Sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, 2006.