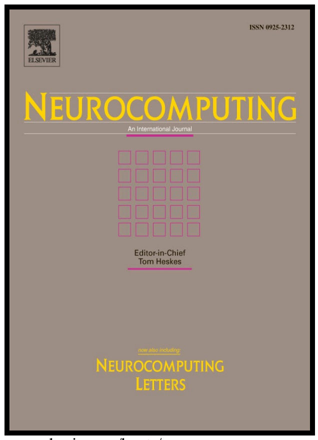
Extreme learning machine with a deterministic assignment of hidden weights in two parallel layers

Pablo A. Henríquez, Gonzalo A. Ruz



www.elsevier.com/locate/neucom

PII: S0925-2312(16)31441-2
DOI: http://dx.doi.org/10.1016/j.neucom.2016.11.040
Reference: NEUCOM17796

To appear in: *Neurocomputing*

Received date: 15 June 2016
Revised date: 15 November 2016
Accepted date: 19 November 2016

Cite this article as: Pablo A. Henríquez and Gonzalo A. Ruz, Extreme learning machine with a deterministic assignment of hidden weights in two parallel layers *Neurocomputing,* http://dx.doi.org/10.1016/j.neucom.2016.11.040

Extreme learning machine with a deterministic assignment of hidden weights in two parallel layers

Pablo A. Henríquez[a], Gonzalo A. Ruz[a,b,*]

[a]Facultad de Ingeniería y Ciencias, Universidad Adolfo Ibáñez, Av. Diagonal Las Torres 2640, Peñalolén, Santiago, Chile
[b]Center of Applied Ecology and Sustainability (CAPES), Santiago, Chile

Abstract

Extreme learning machine (ELM) is a machine learning technique based on competitive single-hidden layer feedforward neural network (SLFN). However, traditional ELM and its variants are only based on random assignment of hidden weights using a uniform distribution, and then the calculation of the weights output using the least-squares method. This paper proposes a new architecture based on a non-linear layer in parallel by another non-linear layer and with entries of independent weights. We explore the use of a deterministic assignment of the hidden weight values using low-discrepancy sequences (LDSs). The simulations are performed with Halton and Sobol sequences. The results for regression and classification problems confirm the advantages of using the proposed method called PL-ELM algorithm with the deterministic assignment of hidden weights. Moreover, the PL-ELM algorithm with the deterministic generation using LDSs can be extended to other modified ELM algorithms.

Keywords: Extreme learning machine, Low-discrepancy points, Parallel layers, Regression, Classification.

*Corresponding author
Email addresses: pabhenriquez@alumnos.uai.cl (Pablo A. Henríquez), gonzalo.ruz@uai.cl (Gonzalo A. Ruz)

1. Introduction

The extreme learning machine (ELM) is an effective and efficient learning algorithm originally proposed by Guang-Bin Huang et al. [1, 2]. It is based on the single-hidden layer feedforward neural network (SLFN), one of the most popular models of neural networks. The main difference of ELM with the common SLFN trained for example, using backpropagation [3], is that the iterative form in the training process is avoided, moreover ELM uses a single layer network which can approximate any continuous non-linear function. This property was verified by Cybenco [4].

The basic idea of ELM is to generate a random weight matrix among the layer of entry and the hidden layer and later calculate the weights of exit using the least-squares method. Research in ELM has become active in the machine learning community where different variations have been proposed. B.Y. Qu et al. [5] developed an algorithm with two hidden layers denoted by TELM, where they introduce a new method to calculate the parameters of the second hidden layer. Junpeng Li et al. [6] generates a quick ELM based on an algorithm for matrix decomposition which reduces the computational cost. Liang et al. [7] proposes an online sequential extreme learning machine (OS-ELM) which can learn from data one-by-one or chunk-by-chunk, either with a fixed or variable size of the data. Qin-Yu Zhu et al. [8] uses the evolutionary differential algorithm to select the weights of entry, achieving a better performance with more compact networks. Rong et al. [9] describes the algorithm P-ELM which uses statistical methods to measure the relevance of the hidden nodes, pruning those that are irrelevant. The determination of the ideal number of neurons in the hidden layer has been analyzed in many studies, such as [10, 11], but it still continues to be an opened problem.

On the other hand, the optimization (computational time) of ELM has been studied at a hardware level, Heeswijk et al. [12] optimized the computational time using the graphical processing unit (GPU), instead of the processor (CPU). Fei Han et al. [13], proposed an optimization of the weights of entry and biases, using particle swarm optimization (PSO). In general, based on previous works, one can conclude that most of the research efforts have been concentrated on how to update or assign the input weight matrix and to modify the structure of the hidden layer, in order to improve the performance of the original ELM.

Many applications have used ELM; for example: big data classification [14], evaluation of credit risk [15], prediction of the blooming of algae [16], the quick recognition of objects and the classification of images [17, 18, 19, 20], among many others.

L.D. Travers et al. [21] proposed a new learning algorithm that considers parallel layers, named Parallel Layer Perceptron (PLP), in which they consider a non-linear layer in parallel with a linear layer, providing more freedom for a suitable adjustment, in this form it combines the characteristics of the ELM with the architecture PLP.

On the other hand, recently, Cervellera et al. [22] assigned the weights among the layer of entry and the hidden layer of the ELM using a family of sampling methods commonly used for numerical integration, called low-discrepancy sequences (LDSs).

The deterministic assignment of the weights presents a great potential for ELM, with only a few works reported in this topic. Therefore, in this work we propose a new architecture in parallel based on a non-linear layer in parallel by another non-linear layer avoiding the problem of the number of neurons in the hidden layer of the ELM-PLP and with entries of independent weights.

In addition, the random assignment of the hidden weights for each parallel layer is generated using a uniform distribution and LDS, later the linear coefficients of the hidden layer of exit are computed analytically. The proposed algorithm is called PL-ELM, and we will study this architecture for different datasets. The experimental results presented in this paper for several problems of regression and classification demonstrates the superiority of the PL-ELM when compared to the original ELM and other variations of the ELM. Besides we demonstrate that the low-discrepancy points for the deterministic generation of the weights matrix, is simple to implement and can be efficiently integrated in other variations of the ELM.

The rest of the work is organized in the following way: in section 2 a brief description of the low-discrepancy points is provided. The detail of the algorithm PL-ELM is presented in section 3. The evaluation of the performance and comparison with other algorithms is presented in section 4. Finally, section 5 presents the conclusions of the works.

2. Low-discrepancy sequences

Low-discrepancy sequences (LDSs) are sequences of numbers that cover a space without clustering and without gaps, in such a way that adding another number to the sequence also avoids clustering and gaps. They give the appearance of randomness although they are deterministic. They are used for estimating integrals numerically, often in high dimensions. The definition and the analysis of the discrepancy are given on the n-dimensional unitary hypercube $[0, 1]^n$.

Definition 1. For N points $\xi = [\omega_1, \dots, \omega_N] \in [0, 1]^n$, then $\jmath$ denotes the family of all subintervals B of the form $\Pi_{i=1}^{s}[a_i, b_i] \in [0, 1]$, and let $A(B, \xi)$ be the counting function for the number of points of ξ that belong to B, we define the discrepancy $\Psi(\xi)$ as [22, 23]

$$\Psi_N(\xi) = \sup_{B \in \jmath} \left| \frac{A(B, \xi)}{N} - \lambda(B) \right| \tag{1}$$

where $\lambda(B)$ is the Lebesgue measure of B. For the interval $[0, 1]^n$, $\lambda(B)$ generalizes the notion of length (length in R, area in R^2 and volume in R^n for $n \geq 3$).

At present, efforts are focused on generating sequences of points deterministic and efficiently. This has led to the development of a family of sequences called low-discrepancy sequences. An LDS aims at keeping the discrepancy of the resulting points in $[0, 1]^n$ as small as possible, and provides a favorable asymptotical rate of convergence of the discrepancy itself. Examples of such sequences are the Halton and Sobol. Fig. 1 shows the sampling of the 2-D unit cube by means of 1000 samples obtained from a Halton, Sobol and the uniform distribution.

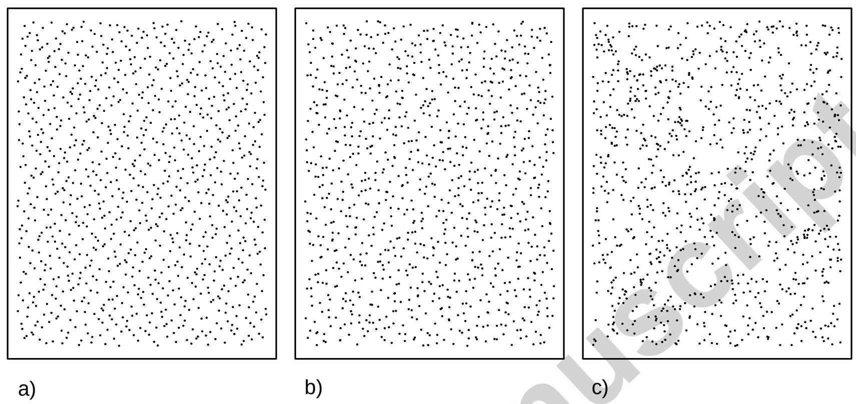


Fig. 1: Sampling of the 2-D unit cube by: a) Halton b) Sobol c) Uniform distribution.

It can be clearly seen how the low-discrepancy sampling scheme covers the space in a more uniform and regular way.

3. Two parallel layers with LDS

3.1. Universal approximation theorem

Let us consider the universal approximation theorem [4, 24]

$$F(x_n) = \sum_{j=1}^{M} \alpha_j \phi \left(\sum_{i=1}^{n} (\omega_{ij} x_j + b_i) \right) \tag{2}$$

where $j = \{1, \ldots, \mathrm{M}\}$, $i = \{1, \ldots, n\}$, α_j, ω_{ij}, b_i are real values, M is an integer, and ϕ is a nonlinear activation function. Let I_n denote the n-dimensional unit hypercube. The space of continuous functions on I_n is denoted by $C(I_n)$. That is, if $f \in C(I_n)$ and $\varepsilon > 0$, such that

$$|F(x_n) - f(x_n)| < \varepsilon \quad \forall x \in I_n. \tag{3}$$

But Eq. (2) can be generalized as a product of two nonlinear functions, similar to the Parallel Percetron Layer (PLP)

$$F(x_n) = \sum_{j=1}^{M} \alpha_j \gamma \left(\sum_{i=1}^{n} (m_{ij} x_j + b_i) \right) \phi \left(\sum_{i=1}^{n} (\omega_{ij} x_j + b_i) \right) \tag{4}$$

This configuration has some computational advantages as discussed in [25].

3.2. Proposed PL-ELM method

We propose the PL-ELM algorithm (PL-ELM network structure is illustrated in Fig. 2) using two parallel neurons γ and ϕ (similar to Eq. (4)) as the product of two sigmoidal functions proposed in [26, 27]. For N distint samples (x_j, y_j) with $x_j \in R^m$ and $y_j \in R^n$ and a SLFN with L hidden neurons the output of the network can be formulated as

$$y_j = \sum_{i=1}^{L} \beta_i \gamma(m_i x_j) \phi(\omega_i x_j + b_i) \quad 1 \leq j \leq N \tag{5}$$

where m_i, ω_i are the weights matrix for each input layer, x_j the jth input sample, γ and ϕ are nonlinear activation function and b_i is the bias.

Eq. (5) can be written compactly as

$$\mathbf{H}\boldsymbol{\beta} = \mathbf{Y} \tag{6}$$

where

$$\mathbf{H} = \mathbf{H_1} \circ \mathbf{H_2} \tag{7}$$

$$\mathbf{H_1} = \begin{bmatrix} \gamma(m_1 x_1) & \cdots & \gamma(m_L x_1) \\ \vdots & \vdots & \vdots \\ \gamma(m_1 x_N) & \cdots & \gamma(m_L x_N) \end{bmatrix} \in R^{N \times L} \tag{8}$$

$$\mathbf{H_2} = \begin{bmatrix} \phi(\omega_1 x_1 + b_1) & \cdots & \phi(\omega_L x_1 + b_L) \\ \vdots & \vdots & \vdots \\ \phi(\omega_1 x_N + b_1) & \cdots & \phi(\omega_L x_N + b_L) \end{bmatrix} \in R^{N \times L} \tag{9}$$

$$\mathbf{Y} = \begin{bmatrix} y_1^T \\ \vdots \\ y_N^T \end{bmatrix} \in R^{N \times m} \tag{10}$$

$\boldsymbol{\beta} = [\beta_1, \beta_2, \ldots, \beta_L]^T \in R^{L \times m}$ is the connection-weight matrix between the hidden layer and the output layer. $\mathbf{H}$ is the hidden layer output matrix of the neural network. $\mathbf{Y}$ is the target matrix of the output layer.

Theorem 1. *[2]: Given any small positive value $\epsilon > 0$ and activation function $\phi : R \to R$ which is infinitely differentiable in any interval, there exist $\tilde{N} \leq N$ such that for N arbitrary distinct samples $(x_j,\ y_j)$, where $x_j \in R^m$ and $y_j \in R^n$, for any w_j and b_j randomly chosen from any intervarls of R^m and R, respectively, according to any continuous probability distribution, then with probability one, $\|\mathbf{H}_{N \times \tilde{N}} \boldsymbol{\beta}_{\tilde{N} \times m} - \mathbf{Y}_{N \times m}\| < \epsilon$.*

Eq. (6) becomes a linear system and the output weight $\widehat{\boldsymbol{\beta}}$ can be obtained by a least-squares solution of Eq. (6), as follows

$$\left|\left|\mathbf{H}\widehat{\boldsymbol{\beta}} - \mathbf{Y}\right|\right| = \min_{\boldsymbol{\beta}} ||\mathbf{H}\boldsymbol{\beta} - \mathbf{Y}|| \tag{11}$$

$$\widehat{\boldsymbol{\beta}} = \mathbf{H}^{\dagger}\mathbf{Y} \tag{12}$$

where $\mathbf{H}^{\dagger}$ denotes the Moore-Penrose generalized inverse of matrix $\mathbf{H}$. $\mathbf{H}^{\dagger}$ can be calculated through several techniques such as orthogonal projection method, orthogonalization method, singular value decomposition (SVD) [28], etc. Then the output function of PL-ELM can be modeled as follows

$$f(x) = h(x)\widehat{\boldsymbol{\beta}} = h(x)\mathbf{H}^{\dagger}\mathbf{Y} \tag{13}$$

Where $h(x) = [\gamma(m_1 x)\phi(w_1 x + b_1) \ldots \gamma(m_L x)\phi(w_L x + b_L)] \in R^{N \times L}$. The workflow of the PL-ELM architecture is depicted in Fig. 3. Given a set of N training samples (x_j, y_j) and L hidden neurons for each parallel layer with two activation functions γ and ϕ, we first randomly initialize the connection weight matrix m_i between the input layer and the first activation function, then we repeat the process with the second parallel layer. Both parallel layers produce the hidden layer $\mathbf{H_1}$ and $\mathbf{H_2}$ respectively. Finally, we calculate the weight matrix $\widehat{\boldsymbol{\beta}}$ between $\mathbf{H}$ and the output layer using Eq. (12).

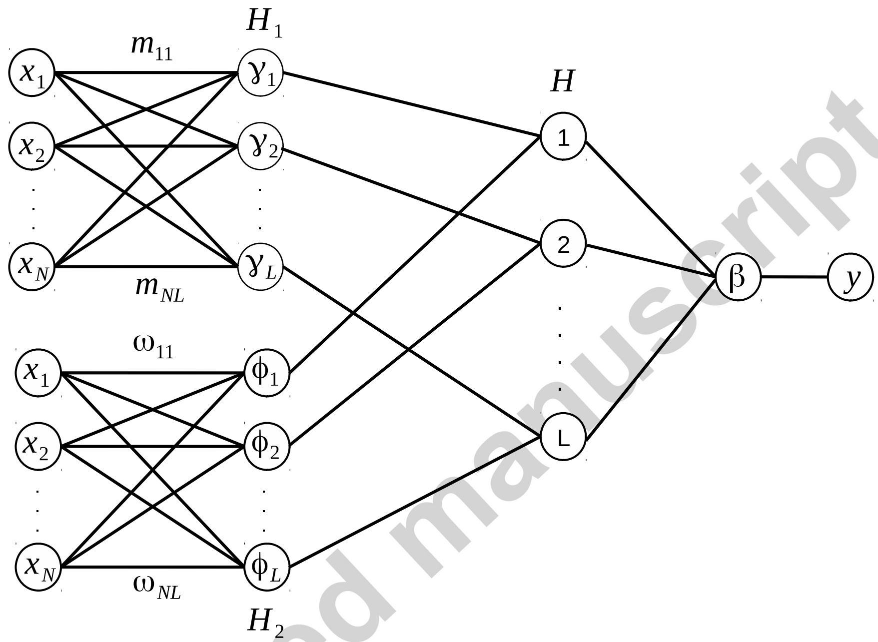


Fig. 2: Structure of the proposed PL-ELM approach.

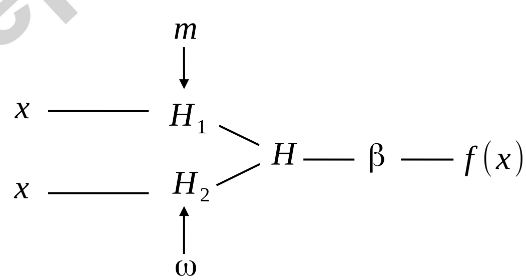


Fig. 3: Workflow of the proposed PL-ELM approach.

Therefore, the proposed algorithm called PL-ELM can be summarized as follows:

Algorithm 1 PL-ELM

Require: N training set $x_j \in R^m$, $y_j \in R^n$ and L hidden neurons in total with two activation functions $(\gamma(\cdot),\phi(\cdot))$

1: Randomly generate input weights using low-discrepancy points $[0,1]^n$ or the uniform distribution $[-1,1]$ (m_i, ω_i and biases b_i)

2: Calculate the first parallel layer matrix $\mathbf{H_1}$ using Eq. (8).

3: Calculate the second parallel layer matrix $\mathbf{H_2}$ using Eq. (9).

4: Matrix $\mathbf{H}$ output $\mathbf{H} = \mathbf{H_1} \circ \mathbf{H_2} \in R^{N \times L}$

5: Calculate output weights matrix $\widehat{\boldsymbol{\beta}} = \mathbf{H}^{\dagger}\mathbf{Y}$

---

4. Experiments and results

The performace of PL-ELM algorithm is compared with the original ELM algorithm on three real regression problems and nine classification problems which all come from the UCI database [29]. All the simulations are carried out using the free R software environment for statistical computing environment running in 2.6 GHz Intel Core i5 and 8 GB-RAM computer. We developed the ELM and PL-ELM using the R packages fOptions, RSNNS, MASS and car. We divided all the dataset in 70% for training and 30% for testing, the division was carried out randomly. The activation functions used in PL-ELM algorithm are

$$\gamma(x) = \phi(x) = \frac{1}{1+e^{-x}}. \tag{14}$$

In all the simulations, the input and output attributes of the three regression problems are normalized into the range $[0, 1]$, and the input attributes of the nine classification problems are normalized into the range $[-1, 1]$. For classification performance, we use the measure accuracy. For regression problems, the performance is measured by the root mean squared error (RMSE) defined as

$$\text{RMSE} = \sqrt{\frac{1}{K}\sum_{i=1}^{K}(y_i - \widehat{y_i})^2}. \tag{15}$$

where y_i, $\widehat{y_i}$ are the desired value and actual prediction value, respectively. K is the number of examples. All the experiments runs were performed 50 times and averages and standard deviations are reported. When PL-ELM considers a low-discrepancy points approach to deterministically assign the hidden weights using the Halton sequences, then we call the model PL-ELM(Halton). When the model uses the Sobol sequences, then we call the model PL-ELM(Sobol). Finally, if we use the traditional uniform distribution for the random assignments of the hidden weights we call the model PL-ELM.

4.1. Regression

For this simulation, the four neural networks (ELM, PL-ELM, PL-ELM(Halton) and PL-ELM(Sobol)) are used to approximate the $sinc(\cdot)$ function definded as follows

$$y = \begin{cases} \frac{sin(x)}{x}, & x \neq 0, \\ 1, & x = 0. \end{cases} \tag{16}$$

A training set and a testing set with 2000 points, was used. Where $x \in [-10, 10]$, also uniformly distributed noise in $[-0.2, 0.2]$ has been added to all the training samples while testing data remained noise-free. Fig. 4 shows the true function and approximation for 500 neurons. Fig. 5 shows the relationship between the generalization performance for the four algorithms and its network size for the $sinc(\cdot)$ function approximation using 2000 training samples. From Fig. 5, the generalization performance of PL-ELM is stable on a wide range of number of hidden neurons outperforming the other algorithms. The generalization performance is better for the low discrepancy points, when compared to the original ELM.

Table 1 shows the comparison of the results presented in [21] with the proposed algorithm. In this experiment, we used 500 samples. We can see that the RMSE for PL-ELM(Sobol) is the lowest.

In the second simulation we use three real-world problems, collected from the UCI Machine Learning Repository [29]. The number of observations (separated

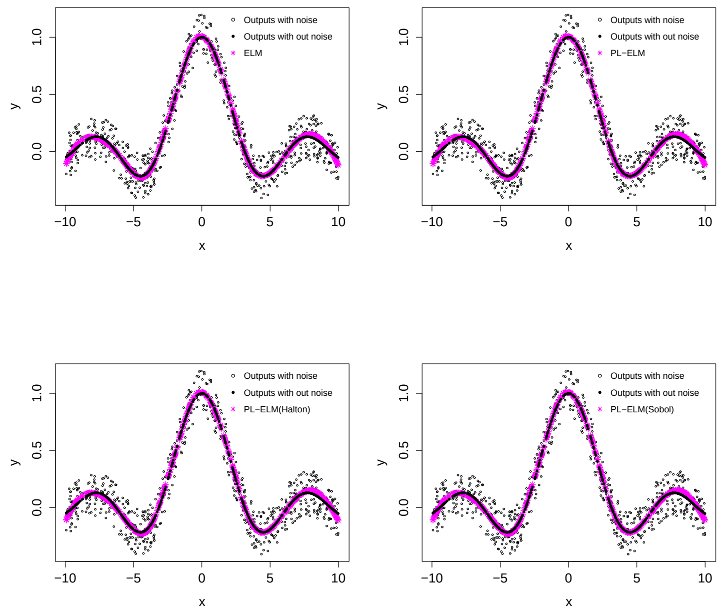


Fig. 4: Comparison of the actual $sinc(\cdot)$ graph and the output predicted by diferents model using 500 neurons.

Table 1: Comparison between ELM, PLP-ELM [21] and PL-ELM for 500 samples.

Algorithms	# of neurons	RMSE (testing)
ELM	8	0.0773
PLP-ELM	4	0.0645
PL-ELM	8	0.0625
PL-ELM(Sobol)	8	**0.0612**
PL-ELM(Halton)	8	0.0887

into training and testing) and the number of attributes of each dataset are shown in Table 2. For each case, the training dataset and testing dataset are randomly generated from the complete dataset before each trial of simulation. For the

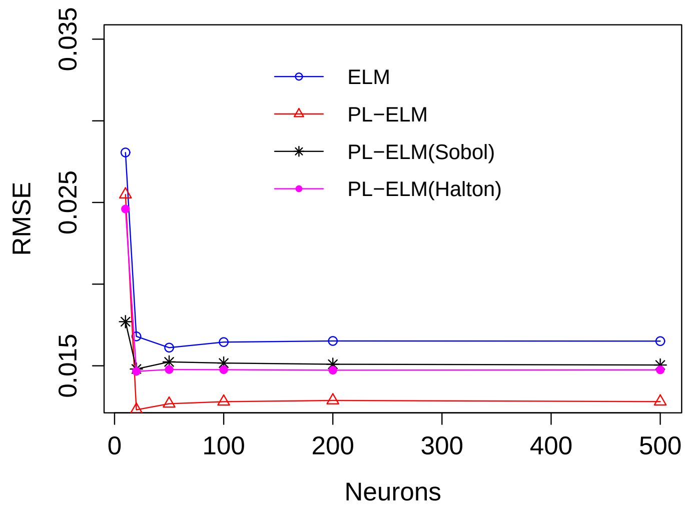


Fig. 5: The variations of RMSE with respect to number of neurons in the hidden layer for the four types of neural networks.

four neural networks (ELM, PL-ELM, PL-ELM(Sobol) and PL-ELM(Halton)), the number of hidden nodes is gradually increased.

Table 2: Information of the regression benchmark problems.

Datasets	Training	Testing	Attributes	Predictors
Concrete	693	337	9	1
PPPT	31935	13795	9	1
Combined cycle power plant	6596	2972	4	1

Table 3 shows the average results of 50 trials of simulations for all these four methods. Small RMSE values indicate a better accuracy of regression. For the Concrete data we can see that for 50 to 100 neurons the proposed methods are more efficient, improving even when using low-discrepancy points. In addition, it can be concluded that both the average training RMSE and the average testing

RMSE of the PL-ELM algorithm is superior to those of the ELM algorithm when the number of hidden neurons ranges from 50 to 100 for Physicochemical properties of protein Tertiary(PPPT) data and the range from 50 to 70 for Combined cycle power plant data. Based on these results one might infer that the proposed PL-ELM algorithm reaches a superior performance under conditions where there is a relatively high number of hidden neurons. From the time-cost aspect, the average training speed for each of the four algorithms considered is extremely fast, and of a similar order of magnitude. We statistically analyzed the experimental results with a paired t-test with a confidence level 0.05. From the p-values in Table 3 , we can confirm statistically that the proposed algorithm PL-ELM outperforms ELM.

Table 3: Results for regressions of real-world datasets.

Datasets	Algorithms	Nodes	Training time (s)	Training		Testing		p-value
				RMSE	Dev	RMSE	Dev	
	ELM*	20	0.00338	**0.11283**	0.00392	**0.12291**	0.00558	$p > 0.05$
	PL-ELM**		0.00526	0.11405	0.00530	0.12408	0.00627	
	PL-ELM(Halton)		0.00636	0.11448	0.00468	0.12331	0.00525	
	PL-ELM(Sobol)		0.00548	0.11356	0.00469	0.12315	0.00654	
Concrete	ELM*	50	0.01176	0.09131	0.00256	0.10569	0.00392	$p < 0.05$
	PL-ELM		0.01496	0.08990	0.00180	0.10455	0.00346	
	PL-ELM(Halton)**		0.01570	**0.08914**	0.00164	**0.10347**	0.00333	
	PL-ELM(Sobol)		0.01490	0.08946	0.00161	0.10602	0.00318	
	ELM*	100	0.04126	0.07231	0.00244	0.10145	0.00483	$p > 0.05$
	PL-ELM		0.04490	0.07227	0.00219	0.10121	0.00527	
	PL-ELM(Halton)**		0.04340	0.07020	0.00194	**0.10043**	0.00520	
	PL-ELM(Sobol)		0.04570	**0.06931**	0.00211	0.10203	0.00627	
	ELM*	20	0.11744	0.05612	0.08382	0.05792	0.00119	$p > 0.05$
	PL-ELM		0.15740	0.05609	0.00040	0.05769	0.00097	
	PL-ELM(Halton)		0.18366	0.05727	0.00065	0.06083	0.00206	
	PL-ELM(Sobol)**		0.18758	**0.05607**	0.00036	**0.05752**	0.00124	
PPPT	ELM*	50	0.49356	0.05397	0.00028	0.05359	0.00674	$p < 0.05$
	PL-ELM		0.61506	0.05381	0.00020	0.05341	0.00959	
	PL-ELM(Halton)**		0.60452	**0.05372**	0.00022	**0.05159**	0.00113	
	PL-ELM(Sobol)		0.59844	0.05380	0.00029	0.05328	0.00891	
	ELM*	100	1.51344	0.05214	0.00024	0.05210	0.00243	$p < 0.05$
	PL-ELM		1.68790	0.05216	0.00021	0.05172	0.00479	
	PL-ELM(Halton)		1.70588	**0.05119**	0.00013	0.04963	0.00417	
	PL-ELM(Sobol)**		1.67750	0.05201	0.00013	**0.04950**	0.00621	
	ELM *	20	0.0.0220	0.05668	0.00021	0.08046	0.00047	$p < 0.05$
	PL-ELM**		0.02790	0.05667	0.00021	**0.08002**	0.00037	
	PL-ELM(Halton)		0.02872	**0.05666**	0.00017	0.08038	0.00038	
	PL-ELM(Sobol)		0.02804	0.05667	0.00017	0.08035	0.00017	
Power plant	ELM*	50	0.08732	0.05519	0.00007	0.07948	0.00021	$p < 0.05$
	PL-ELM		0.11514	**0.05518**	0.00008	0.07945	0.00025	
	PL-ELM(Halton)		0.13702	0.05534	0.00012	0.07938	0.00023	
	PL-ELM(Sobol)**		0.12756	0.05531	0.00011	**0.07936**	0.00020	
	ELM*	70	0.32692	0.05412	0.00013	0.07914	0.00029	$p < 0.05$
	PL-ELM		0.37070	**0.05405**	0.00014	0.07925	0.00032	
	PL-ELM(Halton)		0.37266	0.05443	0.00008	0.07894	0.00021	
	PL-ELM(Sobol)**		0.36326	0.05443	0.00007	**0.07802**	0.00018	

* and ** are the two models considered for the statistical test.

4.2. Evaluation on classification using benchmark datasets

A third simulation related to the classification performance of PL-ELM was carried out using the datasets and partitions (training and testing) shown in Table 4.

The six plots show (Fig. 6 to Fig. 11) the accuracy rate in the test sets of Table 4, the same range between 5 to 500 neurons was used. For the Magic

Gamma Telescope dataset the best performance was obtained with the algorithm PL-ELM (Fig. 6). For the Skin Segmentation dataset, the best performance was achieved with the algorithm PL-ELM (Sobol) (Fig. 7). In Fig. 8, we see overfitting for PL-ELM algorithm with low-discrepancy points, but the algorithm PL-ELM showed a good performance with the weights assigned using the uniform distribution. In Fig. 9 and Fig. 11, the best performance was archived with the algorithm PL-ELM. In Fig. 10, PL-ELM and ELM obtained similar performances. In addition, all the results of the statistical analysis on the accuracy was with $p < 0.05$, therefore, we may confirm statistically that the proposed algorithm PL-ELM outperforms ELM.

Datasets shown in Table 5 were used to compare PL-ELM with a composite function wavelet neural network with extreme learning machine (CFWNN-ELM) [30]. It can be observed in Table 6 that the proposed PL-ELM was better in two out of the three datasets when evaluated in the test set.

The proposed algorithm has some similarities and differences with respect to the work presented in [30]. CFWNN-ELM considers in the hidden nodes a composition of functions, i.e., the input to the standard activation function (sigmoid function) is another function (wavelet function), which depends on two parameters: dilation and translation. Our proposed approach does not consider the composition of functions, but instead, considers the product of two standard activation functions. The similarity between these two methods, is that they both modify the original ELM, by changing the activations in the hidden layer, but in a different way.

The flexibility of the proposed algorithm allows the use of different activation functions, in Table 7 we combine four activation functions (Sig/RBF, Sin/ Sin, Hardlim/Sin and Hardlim/Sig). The results presented in Table 7 shows that the combination of activation functions impact on the performance of the proposed algorithm.

Table 4: Information of the classification benchmark problems.

Datasets	Training	Testing	Attributes	Classes
Magic Gamma Telescope	1320	5795	11	2
Skin Segmentation	171774	73283	4	2
Wine	178	60	14	3
Glass	141	73	10	7
Sonar	138	70	61	2
Page blocks	2700	2773	10	5

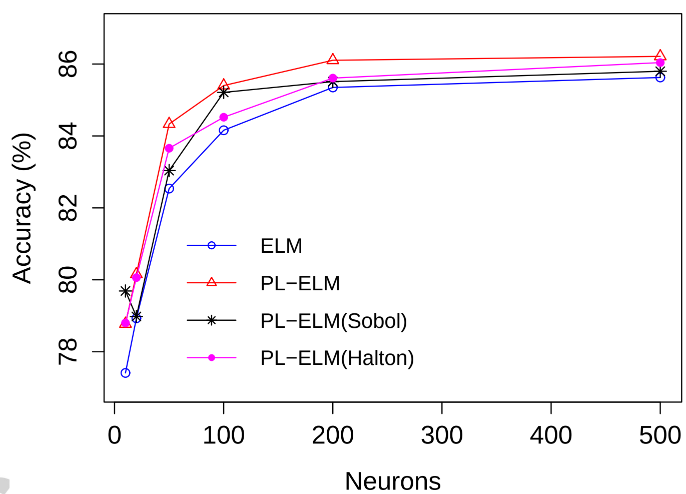


Fig. 6: Performance of the four models in the test set using the Magic Gamma Telescope dataset.

5. Conclusion

The PL-ELM algorithm is proposed for regression and classification problems. The general idea is that with the two parallel layers, the model generates two independent projections to the feature space, each projection goes through

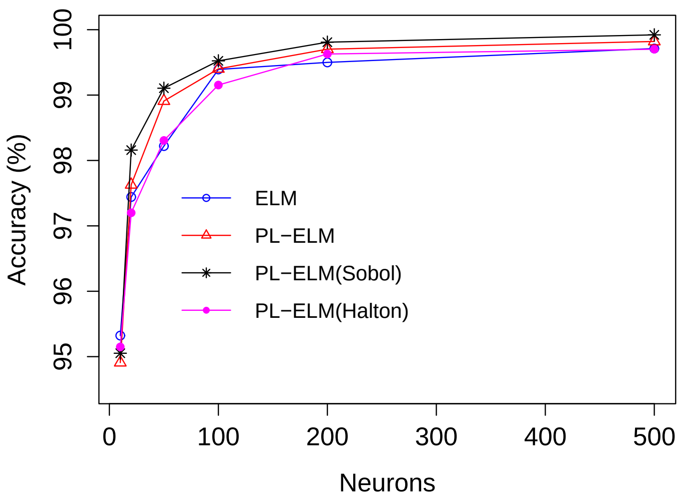


Fig. 7: Performance of the four models in the test set using the Skin Segmentation dataset.

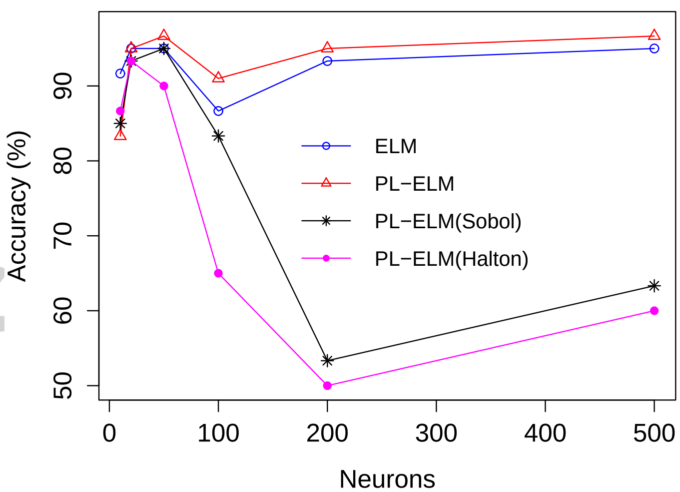


Fig. 8: Performance of the four models in the test set using the Wine dataset.

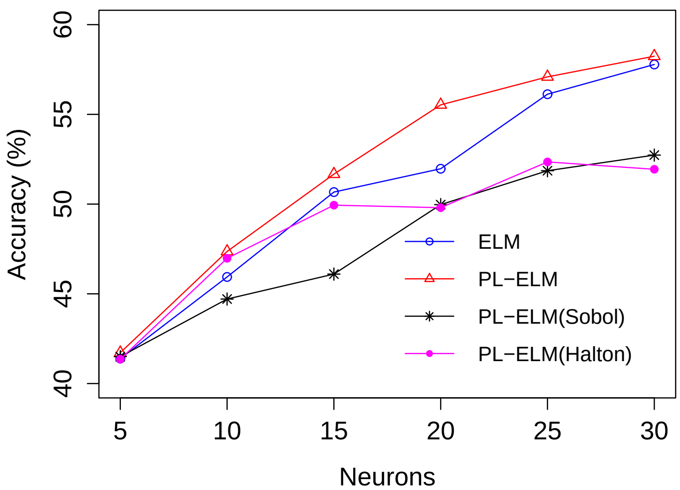


Fig. 9: Performance of the four models in the test set using the Glass dataset.

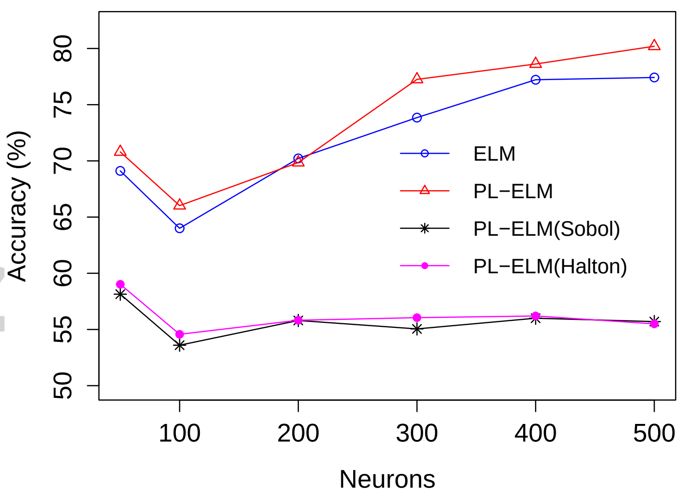


Fig. 10: Performance of the four models in the test set using the Sonar dataset.

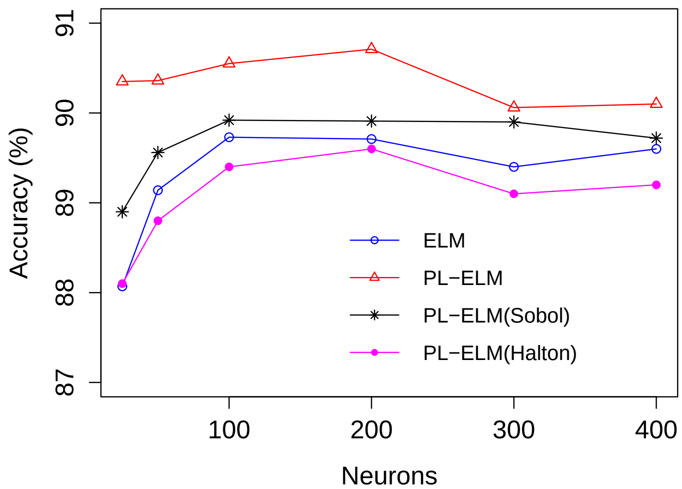


Fig. 11: Performance of the four models in the test set using the Page blocks dataset.

Table 5: Information of the classification datasets used for comparison between PL-ELM and CFWNM-ELM [30].

Datasets	Training	Testing	Attributes	Classes	Nodes
Satellite image	4400	2000	36	7	500
Diabetes	576	192	8	2	35
Image segmentation	1500	910	18	7	300

a nonlinear activation function which then are combined through the product, generating a more powerful nonlinear mapping than just using one activation function, which in turn, affects positively the predictive capacity of the proposed approach. In particular, for classification problems, we observed that not only PL-ELM obtained the best accuracy in the test sets, but also with the lowest dispersion (standard deviation) making the prediction more reliable. The use of low-discrepancy points represents a possibility of improvement in other algorithms that assign the weights of entry randomly. The LDS are easy to

Table 6: Comparison of training and testing accuracy rate (%) between the proposed PL-ELM and CFWNN-ELM [30]. The best results are marked in bold.

Datasets	Algorithms	Accuracy (%)			
		Training		Testing	
		Rate	Dev	Rate	Dev
Satellite image	CFWNM-ELM	93.29	0.37	88.82	0.48
	PL-ELM	**93.62**	0.18	**89.64**	0.32
	PL-ELM(Sobol)	68.40	3.81	54.97	4.16
	PL-ELM(Halton)	61.37	5.71	55.39	5.53
Diabetes	CFWNM-ELM	**79.59**	1.11	78.02	2.56
	PL-ELM	79.49	0.74	75.08	1.72
	PL-ELM(Sobol)	79.55	0.86	**78.61**	1.82
	PL-ELM(Halton)	78.10	0.97	75.89	1.53
Image segmentation	CFWNM-ELM	**98.46**	0.13	**95.40**	0.80
	PL-ELM	98.21	0.17	94.91	0.32
	PL-ELM(Sobol)	96.57	0.29	95.26	0.29
	PL-ELM(Halton)	96.48	0.34	95.11	0.41

implement and to integrate in many programming languages.

Acknowledgement

The authors would like to thank CONICYT-Chile under grant CONICYT Doctoral scholarship (2015- 21150790) (P.H.), Basal(CONICYT)-CMM (G.A.R), and the Research Center Millennium Nucleus Models of Crisis (NS130017) (G.A.R), for financially supporting this research.

References

[1] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: a new learning scheme of feedforward neural networks, Proceedings. 2004 IEEE

Table 7: Performance of the model for different combination of activation functions in the test set using the Sonar dataset.

Activation functions	Algorithms	Nodes	Accuracy (%)	
			Rate	Dev
sig/RBF	PL-ELM	400	73.32	1.09
		500	75.02	1.96
	PL-ELM(Sobol)	400	62.37	1.37
		500	61.82	1.49
	PL-ELM(Halton)	400	60.85	1.10
		500	62.84	1.03
sin/sin	PL-ELM	400	80.17	1.07
		500	81.20	1.26
	PL-ELM(Sobol)	400	76.71	1.88
		500	77.91	1.51
	PL-ELM(Halton)	400	73.68	1.34
		500	74.17	1.21
hardlim/sin	PL-ELM	400	78.31	1.99
		500	80.54	1.20
	PL-ELM(Sobol)	400	72.80	1.10
		500	72.11	1.04
	PL-ELM(Halton)	400	68.60	1.88
		500	69.71	1.51
hardlim/sig	PL-ELM	400	73.97	1.78
		500	73.25	1.56
	PL-ELM(Sobol)	400	70.68	1.71
		500	72.14	1.44
	PL-ELM(Halton)	400	68.31	1.82
		500	69.94	1.69

International Joint Conference on, Vol. 2, 2004, pp. 985–990 vol.2.

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

[3] G.-B. Huang, D. H. Wang, Y. Lan, Extreme learning machines: a survey, International Journal of Machine Learning and Cybernetics 2 (2) (2011) 107–122.

[4] G. Cybenko, Approximation by superpositions of a sigmoidal function, Mathematics of Control, Signals and Systems 2 (4) (1989) 303–314.

[5] B. Qu, B. Lang, J. Liang, A. Qin, O. Crisalle, Two-hidden-layer extreme learning machine for regression and classification, Neurocomputing 175, Part A (2016) 826–834.

[6] J. Li, C. Hua, Y. Tang, X. Guan, A fast training algorithm for extreme learning machine based on matrix decomposition, Neurocomputing 173, Part 3 (2016) 1951–1958.

[7] N. y. Liang, G. b. Huang, P. Saratchandran, N. Sundararajan, A fast and accurate online sequential learning algorithm for feedforward networks, IEEE Transactions on Neural Networks 17 (6) (2006) 1411–1423.

[8] Q.-Y. Zhu, A. Qin, P. Suganthan, G.-B. Huang, Evolutionary extreme learning machine, Pattern Recognition 38 (10) (2005) 1759 – 1763.

[9] H.-J. Rong, Y.-S. Ong, A.-H. Tan, Z. Zhu, A fast pruned-extreme learning machine for classification problem, Neurocomputing 72 (1-3) (2008) 359 – 366, machine Learning for Signal Processing (MLSP 2006) / Life System Modelling, Simulation, and Bio-inspired Computing (LSMS 2007).

[10] A. R. Lima, A. J. Cannon, W. W. Hsieh, Nonlinear regression in environmental sciences using extreme learning machines: A comparative evaluation, Environmental Modelling and Software 73 (2015) 175 – 188.

[11] Y. Miche, A. Sorjamaa, P. Bas, O. Simula, C. Jutten, A. Lendasse, OP-ELM: Optimally pruned extreme learning machine, IEEE Transactions on Neural Networks 21 (1) (2010) 158–162.

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

[13] F. Han, H.-F. Yao, Q.-H. Ling, An improved evolutionary extreme learning machine based on particle swarm optimization, Neurocomputing 116 (2013) 87–93.

[14] J. Chen, H. Chen, X. Wan, G. Zheng, MR-ELM: a mapreduce-based framework for large-scale ELM training in big data era, Neural Computing and Applications 27 (1) (2016) 101–110.

[15] H. Zhou, Y. Lan, Y. C. Soh, G. B. Huang, R. Zhang, Credit risk evaluation with extreme learning machine, in: 2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2012, pp. 1064–1069.

[16] I. Lou, Z. Xie, W. K. Ung, K. M. Mok, Freshwater algal bloom prediction by extreme learning machine in macau storage reservoirs, Neural Computing and Applications 27 (1) (2016) 19–26.

[17] J. Xu, H. Zhou, G. B. Huang, Extreme learning machine based fast object recognition, in: Information Fusion (FUSION), 2012 15th International Conference on, 2012, pp. 1490–1496.

[18] K. Yang, E. Y. Du, E. J. Delp, P. Jiang, F. Jiang, Y. Chen, R. Sherony, H. Takahashi, An extreme learning machine-based pedestrian detection method, in: Intelligent Vehicles Symposium (IV), 2013 IEEE, 2013, pp. 1404–1409.

[19] A. Samat, P. Du, S. Liu, J. Li, L. Cheng, E^2LMs : Ensemble extreme learning machines for hyperspectral image classification, IEEE Journal of

Selected Topics in Applied Earth Observations and Remote Sensing 7 (4) (2014) 1060–1069.

[20] J. Cao, K. Zhang, M. Luo, C. Yin, X. Lai, Extreme learning machine and adaptive sparse representation for image classification, Neural Networks 81 (2016) 91–102.

[21] L. Tavares, R. Saldanha, D. Vieira, Extreme learning machine with parallel layer perceptrons, Neurocomputing 166 (2015) 164–171.

[22] C. Cervellera, D. Macci, Low-discrepancy points for deterministic assignment of hidden weights in extreme learning machines, IEEE Transactions on Neural Networks and Learning Systems 27 (4) (2016) 891–896.

[23] P. Bratley, B. L. Fox, Algorithm 659: Implementing sobol's quasirandom sequence generator, ACM Trans. Math. Softw. 14 (1) (1988) 88–100.

[24] K.-I. Funahashi, On the approximate realization of continuous mappings by neural networks, Neural Networks 2 (3) (1989) 183–192.

[25] W. M. Caminhas, D. A. Vieira, J. A. Vasconcelos, Parallel layer perceptron, Neurocomputing 55 (3-4) (2003) 771–778, evolving Solution with Neural Networks.

[26] A. R. Barron, Universal approximation bounds for superpositions of a sigmoidal function, IEEE Transactions on Information Theory 39 (3) (1993) 930–945.

[27] D. A. G. Vieira, R. H. C. Takahashi, V. Palade, J. A. Vasconcelos, 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 19 (8) (2008) 1415–1430.

[28] K. S. Banerjee, Generalized inverse of matrices and its applications, Technometrics 15 (1) (1973) 197–197.

[29] M. Lichman, UCI machine learning repository (2013). URL http://archive.ics.uci.edu/ml

[30] J. Cao, Z. Lin, G. bin Huang, Composite function wavelet neural networks with extreme learning machine, Neurocomputing 73 (7-9) (2010) 1405–1416.