Brief Papers

OP-ELM: Optimally Pruned Extreme Learning Machine

Yoan Miche, Antti Sorjamaa, Patrick Bas, Olli Simula, Christian Jutten, and Amaury Lendasse

***Abstract*—In this brief, the optimally pruned extreme learning machine (OP-ELM) methodology is presented. It is based on the original extreme learning machine (ELM) algorithm with additional steps to make it more robust and generic. The whole methodology is presented in detail and then applied to several regression and classification problems. Results for both computational time and accuracy (mean square error) are compared to the original ELM and to three other widely used methodologies: multilayer perceptron (MLP), support vector machine (SVM), and Gaussian process (GP). As the experiments for both regression and classification illustrate, the proposed OP-ELM methodology performs several orders of magnitude faster than the other algorithms used in this brief, except the original ELM. Despite the simplicity and fast performance, the OP-ELM is still able to maintain an accuracy that is comparable to the performance of the SVM. A toolbox for the OP-ELM is publicly available online.**

***Index Terms*—Classification, extreme learning machine (ELM), least angle regression (LARS), optimally pruned extreme learning machine (OP-ELM), regression, variable selection.**

I. Introduction

Since the data can be collected automatically from various and numerous sources, the global amount of information tends to grow rapidly in many fields of science. Although these data most likely improve the precision and details about the considered phenomena, they are also raising many new challenges. Storing of large data sets can get difficult, while actual processing of it can only be automated and by using very fast algorithms. "Manual" analysis is clearly impossible and the computational complexity of the used methodologies have to be kept as low as possible to be able to process even more data.

Among the most famous algorithms used for data processing through machine learning techniques lie feedforward neural networks [1]. While multilayer feedforward neural networks have been proven to be universal approximators [2], they tend not to be widely used when processing important data sets. Hence, linear models are often preferred for industrial applications, because they are much faster to build compared to the computational complexity required for a neural network, or most nonlinear models in general.

Manuscript received December 05, 2008; accepted October 29, 2009. First published December 08, 2009; current version published January 04, 2010. This work was supported in part by the Academy of Finland Centre of Excellence, by the Adaptive Informatics Research Centre, and by the Finnish Funding Agency for Technology and Innovation under the NoTeS project.

Y. Miche is with the Gipsa-Lab, INPG/CNRS, Grenoble 38402, France and also with the Department of Information and Computer Science, Helsinki University of Technology, Espoo 02015, Finland (e-mail: yoan.miche@tkk.fi).

A. Sorjamaa, O. Simula, and A. Lendasse are with the Department of Information and Computer Science, Helsinki University of Technology, Espoo 02015, Finland (e-mail: antti.sorjamaa@tkk.fi; olli.simula@tkk.fi; lendasse@tkk.fi).

P. Bas and C. Jutten are with the Gipsa-Lab, INPG/CNRS, Grenoble, France (e-mail: patrick.bas@inpg.fr; christian.jutten@inpg.fr).

Color versions of one or more of the figures in this brief are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNN.2009.2036259

The slow building of these neural networks comes from a few facts that remain inherent to the various existing training algorithms. Usually many parameters are required for a proper selection of the model structure and afterwards, the training. Moreover, these parameters are selected and tuned via slow algorithms and the whole model structure and training has to be repeated many times to make sure the model is fitting the data sufficiently well.

Recently, in [3], Huang *et al.* proposed an original algorithm called extreme learning machine (ELM). This method makes the selection of the weights of the hidden neurons very fast in the case of single-layer feedforward neural network (SLFN). Hence, the overall computational time for model structure selection and actual training of the model is often reduced even by hundreds, compared to some classical methods [2], [4]–[6]. Furthermore, the algorithm remains rather simple, which makes its implementation easy.

It is believed though that the ELM algorithm can have some issues when encountering irrelevant or correlated data. For this reason, a methodology named optimally pruned extreme learning machine (OP-ELM), based on the original ELM algorithm, is proposed in this brief. The OP-ELM extends the original ELM algorithm and wraps this extended algorithm within a methodology using a pruning of the neurons, leading to a more robust overall algorithm. Pruning of neurons in a network built using ELM has been proposed recently by Rong *et al.* in [7], for classification purposes, and using statistical tests as a measure of relevance of the neurons regarding the output. The OP-ELM presented here applies to both classification and regression problems and uses a leave-one-out (LOO) criterion for the selection of an appropriate number of neurons.

In the next section, the actual OP-ELM and the whole wrapping methodology are presented, along with the original ELM. Section III presents the data sets used for the experiments as well as results concerning computational speed and accuracy for the OP-ELM, ELM, multilayer perceptron network (MLP), Gaussian process (GP), and support vector machines (SVMs).

II. The Methodology

The OP-ELM methodology is based on the original ELM algorithm from which it borrows the original SLFN construction. In the following, the main concepts and theory of the ELM algorithm are shortly reviewed, with an example on the possible problems encountered by the ELM on data sets with irrelevant variables.

The OP-ELM algorithm is introduced as a more robust methodology regarding irrelevant variables situation. The steps of the algorithm are detailed and the network pruning algorithm, multiresponse sparse regression (MRSR), is described, along with the validation method LOO.

There is a Matlab toolbox available online for performing the OP-ELM methodology [8], along with a detailed user's manual.[1] A version of the toolbox translated to C language is coming soon.

A. ELM and OP-ELM

1) Extreme Learning Machine: The ELM algorithm was originally proposed by Huang *et al.* in [3] and it makes use of the SLFN. The main concept behind the ELM lies in the random initialization of the SLFN weights and biases. Then, using Theorem 1 and under the conditions of the theorem, the input weights and biases do not need to be adjusted

[1]Available at: http://www.cis.hut.fi/projects/tsp/index.php?page=OPELM

and it is possible to calculate implicitly the hidden-layer output matrix and hence the output weights. The network is obtained with very few steps and very low computational cost.

Consider a set of M distinct samples $(\mathbf{x}_i, \mathbf{y}_i)$ with $\mathbf{x}_i \in \mathbb{R}^{d_1}$ and $\mathbf{y}_i \in \mathbb{R}^{d_2}$; then, an SLFN with N hidden neurons is modeled as the following sum:

$$\sum_{i=1}^{N} \boldsymbol{\beta}_i f(\mathbf{w}_i \mathbf{x}_j + b_i), \qquad 1 \le j \le M \tag{1}$$

with f being the activation function, $\mathbf{w}_i$ the input weights, b_i the biases, and $\boldsymbol{\beta}_i$ the output weights.

In the case where the SLFN perfectly approximates the data, the errors between the estimated outputs $\hat{\mathbf{y}}_i$ and the actual outputs $\mathbf{y}_i$ are zero and the relation is

$$\sum_{i=1}^{N} \boldsymbol{\beta}_i f(\mathbf{w}_i \mathbf{x}_j + b_i) = \mathbf{y}_j, \qquad 1 \le j \le M \tag{2}$$

which writes compactly as $\mathbf{H}\boldsymbol{\beta} = \mathbf{Y}$, with

$$\mathbf{H} = \begin{pmatrix} f(\mathbf{w}_1 \mathbf{x}_1 + b_1) & \cdots & f(\mathbf{w}_N \mathbf{x}_1 + b_N) \\ \vdots & \ddots & \vdots \\ f(\mathbf{w}_1 \mathbf{x}_M + b_1) & \cdots & f(\mathbf{w}_N \mathbf{x}_M + b_N) \end{pmatrix} \tag{3}$$

and $\boldsymbol{\beta} = (\beta_1^T \cdots \beta_N^T)^T$ and $\mathbf{Y} = (\mathbf{y}_1^T \cdots \mathbf{y}_M^T)^T$.

With these notations, Theorem 1 is proposed in [3], which is the pillar of the ELM idea. The theorem states that with randomly initialized input weights and biases for the SLFN, and under the condition that the activation function is infinitely differentiable, then the hidden-layer output matrix can be determined and will provide an approximation of the target values as good as wished (nonzero).

Theorem 1: Given any $\varepsilon > 0$ and an activation function $f : \mathbb{R} \mapsto \mathbb{R}$ infinitely differentiable in any interval, there exists $n < M$ such that for M distinct samples $(\mathbf{x}_i, \mathbf{y}_i)$, $\mathbf{x}_i \in \mathbb{R}^{d_1}$, $\mathbf{y}_i \in \mathbb{R}^{d_2}$, for any $\mathbf{w}_i \in \mathbb{R}^{d_1}$ and $b_i \in \mathbb{R}$, $\|\mathbf{H}_{[M\times n]}\beta_{[n\times d_2]} - \mathbf{Y}_{[M\times d_2]}\| < \varepsilon$.

The way to calculate the output weights β from the knowledge of the hidden-layer output matrix $\mathbf{H}$ and target values is proposed with the use of a Moore–Penrose generalized inverse of the matrix $\mathbf{H}$, denoted as $\mathbf{H}^\dagger$ [9]. Overall, the ELM algorithm is summarized as follows.

Algorithm 1: ELM

Given a training set $(\mathbf{x}_i, \mathbf{y}_i)$, $\mathbf{x}_i \in \mathbb{R}^{d_1}$, $\mathbf{y}_i \in \mathbb{R}^{d_2}$, an activation function $f : \mathbb{R} \mapsto \mathbb{R}$, and the number of hidden nodes N:

1: Randomly assign input weights $\mathbf{w}_i$ and biases b_i, $1 \le i \le N$;

2: Calculate the hidden-layer output matrix $\mathbf{H}$;

3: Calculate output weights matrix $\beta = \mathbf{H}^\dagger \mathbf{Y}$.

The proposed solution to the equation $\mathbf{H}\beta = \mathbf{Y}$ in the ELM algorithm, as $\beta = \mathbf{H}^\dagger \mathbf{Y}$ has three main properties making it an appealing solution.

1) It is one of the least squares solutions of the mentioned equation, hence the minimum training error can be reached with this solution.
2) It is the solution with the smallest norm among the least squares solutions.
3) The smallest norm solution among the least squares solutions is unique and it is $\beta = \mathbf{H}^\dagger \mathbf{Y}$.

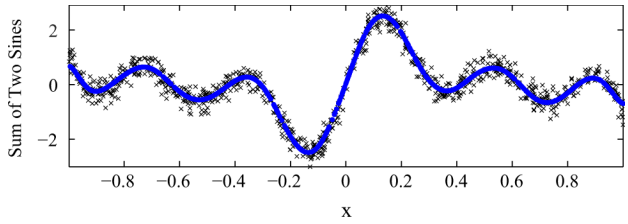

Fig. 1. Example of a training result using ELM, on a sum of two sines. Dots represent the ELM model fitting the data points (crosses).

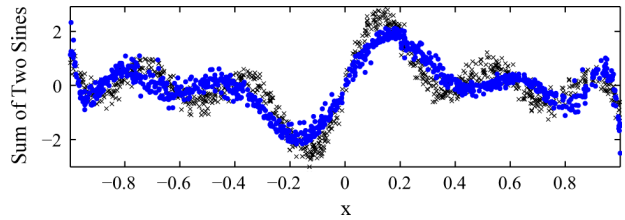

Fig. 2. Example using the same sum of sine as in Fig. 1 and an additional noisy variable (not represented here) for training. The obtained ELM model is much more spread and approximate, due to the irrelevant variable included.

Theoretical proofs and a more thorough presentation of the ELM algorithm are detailed in the original paper [3]. In Huang *et al.*'s later work, it has been proved that the ELM is able to perform universal function approximation [10].

2) The Problem of ELM With Irrelevant Variables: As already mentioned, the ELM models tend to have problems when irrelevant or correlated variables are present in the training data set. As an illustration of this, a toy example with two cases, without and with an irrelevant variable, are tested and compared.

Fig. 1 shows the ELM model obtained by training on the sum of sines example. In this case, the ELM model fits very well to the training data, with no apparent perturbation or distortion.

In Fig. 2, an additional variable containing a pure Gaussian noise, totally unrelated to the actual data, is also used as an input. The additional noise variable is not shown in the figure. The ELM model on top of the data is much more spread and approximate than in the previous case. Overall, the global fitting of the ELM model to the actual data is not as good as before.

For this reason, it is proposed in the OP-ELM methodology, to perform a pruning of the irrelevant variables, via pruning of the related neurons of the SLFN built by the ELM.

3) Optimally Pruned ELM: The OP-ELM is made of three main steps summarized in Fig. 3.

The very first step of the OP-ELM methodology is the actual construction of the SLFN using the original ELM algorithm with a lot of neurons.

Second and third steps are presented in more details in Sections II-A4 and II-A5 and are meant for an effective pruning of the possibly unuseful neurons of the SLFN: MRSR algorithm enables to obtain a ranking of the neurons according to their usefulness, while the actual pruning is performed using the results of the LOO validation.

The OP-ELM algorithm uses a combination of three different types of kernels, for robustness and more generality, where the original ELM proposed to use only sigmoid kernels. The used types are linear, sigmoid, and Gaussian kernels. Having the linear kernels included in the network helps when the problem is linear or nearly linear.

The Gaussian kernels have their centers taken randomly from the data points, similarly as in [11], and widths randomly drawn between

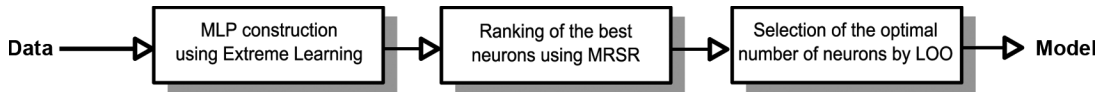

Fig. 3. Three steps of the OP-ELM algorithm.

percentile 20% and percentile 80% of the distance distribution of the input space, as suggested in [12].

The sigmoid weights are drawn randomly from a uniform distribution in the interval $[-5, 5]$ in order to cover the whole zero mean and unit variance data range.

The OP-ELM methodology can also handle multiple-output—multiple-class problems in both regression and classification using multiple inputs.

4) Multiresponse Sparse Regression: In order to get rid of the unuseful neurons of the hidden layer, the MRSR, proposed by Similä and Tikka [13], is used.

The main idea of the algorithm is as follows. Denote by $\mathbf{X} = [\mathbf{x}_1 \dots \mathbf{x}_m]$ the $n \times m$ regressor matrix. MRSR adds each column of the regressor matrix one by one to the model $\hat{\mathbf{Y}}^k = \mathbf{X}\mathbf{W}^k$, where $\hat{\mathbf{Y}}^k = [\hat{\mathbf{y}}_1^k \dots \hat{\mathbf{y}}_p^k]$ is the target approximation of the model. The $\mathbf{W}^k$ weight matrix has k nonzero rows at kth step of the MRSR. With each new step, a new nonzero row and a new column of the regressor matrix are added to the model.

More specific details of the MRSR algorithm can be found from the original paper [13].

It can be noted that the MRSR is mainly an extension of the least angle regression (LARS) algorithm [14] and hence, it is actually a variable ranking technique, rather than a selection one. An important detail shared by the MRSR and the LARS is that the ranking obtained is exact, if the problem is linear. In fact, this is the case with the OP-ELM, since the neural network built in the previous step is linear between the hidden layer and the output. Therefore, the MRSR provides an exact ranking of the neurons for our problem. Because of the exact ranking provided by the MRSR, it is used to rank the kernels of the model. The target is the actual output $\mathbf{y}_i$, while the "variables" considered by the MRSR are the outputs of the kernels $\mathbf{h}_i = \mathrm{Ker}(\mathbf{x}_i^T)$, the columns of $\mathbf{H}$.

5) Leave-One-Out: Since the MRSR only provides a ranking of the kernels, the decision over the actual best number of neurons for the model is taken using an LOO validation method.

One problem with the LOO error is that it can be very time consuming, if the data set has a high number of samples. Fortunately, the PREdiction Sum of Squares (PRESS) statistics provide a direct and exact formula for the calculation of the LOO error for linear models (see [15] and [16] for details of this formula and its implementations)

$$\varepsilon^{\mathrm{PRESS}} = \frac{y_i - \mathbf{h}_i \mathbf{b}_i}{1 - \mathbf{h}_i \mathbf{P} \mathbf{h}_i^T} \tag{4}$$

where $\mathbf{P}$ is defined as $\mathbf{P} = (\mathbf{H}^T\mathbf{H})^{-1}$ and $\mathbf{H}$ is the hidden-layer output matrix.

The final decision over the appropriate number of neurons for the model can then be taken by evaluating the LOO error versus the number of neurons used. Here, the neurons are already ranked by the MRSR.

In order to give an overview of the usefulness of the ranking step performed by the MRSR algorithm, the final model structure selection for the OP-ELM model using the Ailerons data set (see Section III) is shown in Fig. 4.

It can be seen from Fig. 4 that the OP-ELM benefits greatly from the MRSR ranking step. The convergence is faster, because the LOO error gets to the minimum faster when the MRSR is used than when it is not. Also, the number of neurons is far fewer in the LOO error

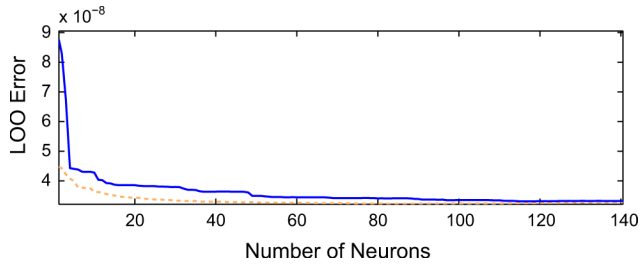

Fig. 4. Comparison of LOO error with and without the MRSR ranking. The solid line represents the LOO error without and the dashed line with the MRSR ranking.

TABLE I
INFORMATION ABOUT THE SELECTED DATA SETS. NUMBER OF VARIABLES AND NUMBER OF SAMPLES FOR BOTH TRAINING AND TESTING, TWO THIRDS OF THE WHOLE SET FOR TRAINING AND ONE THIRD FOR TEST. FOR CLASSIFICATION PROBLEMS, THE VARIABLES COLUMN ALSO INCLUDES THE NUMBER OF CLASSES IN THE DATA SET

		Samples	
Regression	# of Variables	Train	Test
Abalone	8	2784	1393
Ailerons	5	4752	2377
Elevators	6	6344	3173
Computer	12	5461	2731
Auto price	15	106	53
CPU	6	139	70
Servo	4	111	56
Breast Cancer	32	129	65
Bank	8	2999	1500
Stocks	9	633	317
Boston	13	337	169
Classification			
Iris	4/3	100	50
Wisconsin Breast Cancer	30/2	379	190
Pima Indians Diabetes	8/2	512	256
Wine	13/3	118	60

minimum point when using the MRSR ranking, thus leading to more sparse network with the same performance.

In the end, an SLFN possibly using a mix of linear, sigmoid, and Gaussian kernels is obtained, with a highly reduced number of neurons, all within a small computational time.

III. EXPERIMENTS

In the following, five methodologies are compared using several regression and classification tasks. The compared methods are GP, SVM, MLP, the original ELM, and the proposed OP-ELM.

A. Data Sets

Fifteen different data sets have been chosen for the experiments, 11 for regression and four for classification problems. The data sets are collected from the University of California at Irvine (UCI) Machine Learning Repository [17] and they have been chosen by the overall heterogeneity in terms of number of samples, variables, and classes for classification problems.

Table I summarizes the different attributes for the 15 data sets. All data sets have been preprocessed in the same way. Ten different random permutations of the whole data set are taken without replacement, and two thirds are used to create the training set and the remaining third is

TABLE II
COMPUTATIONAL TIMES (IN SECONDS) FOR ALL FIVE METHODOLOGIES ON THE REGRESSION DATA SETS. ALGORITHMS HAVE BEEN SORTED BY COMPUTATIONAL TIME. "AUTO P." STANDS FOR AUTO PRICE DATA SET AND "BREAST C." FOR BREAST CANCER DATA SET

	Abalone	Ailerons	Elevators	Computer	Auto P.	CPU	Servo	Breast C.	Bank	Stocks	Boston
SVM	6.6e+4	4.2e+2	5.8e+2	3.2e+5	2.6e+2	3.2e+2	1.3e+2	3.2e+2	1.6e+3	2.3e+3	8.5e+2
MLP	2.1e+3	3.5e+3	3.5e+3	8.2e+3	7.3e+2	5.8e+2	5.2e+2	8.0e+2	2.7e+3	1.2e+3	8.2e+2
GP	9.5e+2	2.9e+3	6.5e+3	6.3e+3	2.9	3.2	2.2	8.8	1.7e+3	4.1e+1	8.5
OPELM	5.7	16.8	29.8	26.2	2.7e-1	2.0e-1	2.1e-1	4.2e-1	8.03	1.54	7.0e-1
ELM	4.0e-1	9.0e-1	1.6	1.2	3.8e-2	4.2e-2	3.9e-2	4.8e-2	4.7e-1	1.1e-1	7.4e-2

used for the test set. Then, the training set is normalized, zero mean, and unit variance, and the test set is also normalized using the same mean and variance used for the training set. Because the test set is normalized using the same normalization parameters as for the training, it is most likely not exactly zero mean and unit variance.

It should also be noted that the proportions of the classes, for the classifications cases, have been kept balanced: each class is represented in an equal proportion, in both training and test sets. This is important in order to have relevant test results.

B. Experiments

Experiments have been conducted using the online versions of the methodologies, unaltered. All experiments have been run on the same x86_64 Linux machine with at least 4 GB of memory (no swapping for any of the experiments) and $2+$ GHz processor. It should be noted that even though some methodologies are using parallelization of the tasks, the computational times are reported considering single-threaded execution on one single core, for the sake of comparisons.

The hyperparameters for the SVM and the MLP are selected using a tenfold cross validation.

The SVM is performed using the SVM toolbox [6] with the default settings for the hyperparameters and the grid search: the grid is logarithmic between 2^{-2} and 2^{10} for each hyperparameter; nu-SVC has been used for classification and epsilon-SVR for regression, with radial basis function kernel. The original grid search has been replaced by a parallelization process, which distributes parts of the grid over different machines.

The MLP [4] is performed using a neural network toolbox, which is part of the Matlab© software from the MathWorks, Inc. (Natick, MA). The training of the MLP is performed using the Levenberg–Marquardt backpropagation.

In order to decrease the possibility of local minima with the MLP, the training is repeated ten times for each fold and the best network according to the training error is selected for validation. For example, in order to validate the MLP network using 12 hidden neurons, we have to train a total of 100 MLP networks with 12 hidden neurons to evaluate the validation error. This procedure is done for each number of hidden nodes from 1 to 20 and the appropriate number according to the validation MSE is selected.

The GP is performed using a GPML toolbox for Matlab from Rasmussen and Williams [5]. The GP is performed using the default settings taken from the examples of usage of the toolbox.

Finally, the OP-ELM was used with all possible kernels, linear, sigmoid, and Gaussian, using a maximum number of 100 neurons.

1) Computational Times: Computational times are first reviewed for all five methodologies. Tables II and III give the computational times for training and test steps (sum of both), for each methodology. It can be noted that for all five methodologies, the computational times for the test steps are negligible compared to the training times; this is especially clear for large training times, like the SVM or MLP ones.

According to Tables II and III, the ELM is the fastest algorithm by several orders of magnitude compared, for example, to the SVM. This is in line with the claims of the ELM authors. The proposed OP-ELM is between one and three orders of magnitude slower than the original ELM, but still much faster than the rest of the compared methods in all data sets.

TABLE III
COMPUTATIONAL TIMES (IN SECONDS) COMPARED FOR ALL FIVE METHODOLOGIES FOR CLASSIFICATION DATA SETS. "WISC. B.C." FOR WISCONSIN BREAST CANCER DATA SET AND "PIMA I.D." FOR PIMA INDIANS DIABETES DATA SET

	Iris	Wisc. B.C.	Pima I.D.	Wine
SVM	2.3e+2	2.9e+3	3.3e+3	3.8e+2
MLP	7.6e+2	1.7e+3	4.1e+2	1.2e+3
GP	7.6e-1	6.1	5.8	1.9
OPELM	7.4e-2	1.1	9.6e-1	4.4e-1
ELM	2.4e-2	4.3e-2	4.8e-2	2.7e-2

However, the ranking of the SVM, MLP, and GP regarding the computational times is not exactly the same in all data sets, but in every case they are clearly slower than the ELM and OP-ELM.

The main reason why the OP-ELM has been designed in the first place is to add more robustness to the very simple and fast ELM algorithm. Experimental results for this robustness are presented in Section III-B2 through test results.

2) Test Errors: Because the validation results, while providing a good measure of the model fit to the data, do not measure the actual interpolation properties of the model, only the test results for the five models are presented in Tables IV and V.

According to the test results, the SVM is very reliable on average. Meanwhile, as mentioned earlier, the ELM can have good results with respect to its computational speed, but also it can have very high mean square errors (MSEs) on some test sets, for example, in Auto price and central processing unit (CPU) data sets.

In this regard, the OP-ELM manages to keep a good MSE, when comparing to other algorithms, and even rather close to the performance of the SVM (and of the GP) on many data sets used in the experiments. This comforts the earlier claims that the OP-ELM keeps a part of the speed of the ELM and, therefore, is much faster than most common algorithms, while remaining robust and accurate and providing good interpolation models.

Finally, in order to give an overview of the pruning result for the OP-ELM, Table VI lists the selected neurons for two data sets, one for regression and one for classification, namely, Ailerons and Iris.

One can see that the total number of kept neurons is fairly stable, and so is the number of linear neurons. It is interesting to note that the amount of neurons for each type is more stable for classification data sets than for regression one. On average, the situation depicted here is globally similar for other data sets.

Whether the stability of the number of neurons is a consequence of the size of the data set or the type of the problem, warrants further investigation.

IV. CONCLUSION

In this brief, the OP-ELM methodology has been detailed through the presentation of the three steps: the plain original ELM as the first step to build the SLFN, followed by a ranking of the neurons by the MRSR algorithm, and finally, the selection of the neurons that will remain in the final model through LOO validation.

TABLE IV
MEAN SQUARE ERROR RESULTS IN BOLDFACE (AND STANDARD DEVIATIONS IN REGULAR) FOR ALL FIVE METHODOLOGIES FOR THE REGRESSION DATA SETS. "AUTO P." STANDS FOR AUTO PRICE DATA SET AND "BREAST C." FOR BREAST CANCER DATA SET

	Abalone	Ailerons	Elevators	Computer	Auto P.	CPU	Servo	Breast C.	Bank	Stocks	Boston
SVM	**4.5**	**1.3e-7**	**6.2e-6**	**1.2e+2**	**2.8e+7**	**6.5e+3**	**6.9e-1**	**1.2e+3**	**2.7e-2**	**5.1e-1**	**3.4e+1**
	2.7e-1	2.6e-8	6.8e-7	8.1e+1	8.4e+7	5.1e+3	3.3e-1	7.2e-1	8.0e-4	9.0e-2	3.1e+1
OPELM	**4.9**	**2.8e-7**	**2.0e-6**	**3.1e+1**	**9.5e+7**	**5.3e+3**	**8.0e-1**	**1.4e+3**	**1.1e-3**	**9.8e-1**	**1.9e+1**
	6.6e-1	1.5e-9	5.4e-8	7.4	4.0e+6	5.2e+3	3.3e-1	3.6e+2	1.0e-6	1.1e-1	2.9
ELM	**8.3**	**3.3e-8**	**2.2e-6**	**4.9e+2**	**7.9e+9**	**4.7e+4**	**7.1**	**7.7e+3**	**6.7e-3**	**3.4e+1**	**1.2e+2**
	7.5e-1	2.5e-9	7.0e-8	6.2e+1	7.2e+9	2.5e+4	5.5	2.0e+3	7e-4	9.35	2.1e+1
GP	**4.5**	**2.7e-8**	**2.0e-6**	**7.7**	**2.0e+7**	**6.7e+3**	**4.8e-1**	**1.3e+3**	**8.7e-4**	**4.4e-1**	**1.1e+1**
	2.4e-1	1.9e-9	5.0e-8	2.9e-1	1.0e+7	6.6e+3	3.5e-1	1.9e+2	5.1e-5	5.0e-2	3.5
MLP	**4.6**	**2.7e-7**	**2.6e-6**	**9.8**	**2.2e+7**	**1.4e+4**	**2.2e-1**	**1.5e+3**	**9.1e-4**	**8.8e-1**	**2.2e+1**
	5.8e-1	4.4e-9	9.0e-8	1.1	9.8e+6	1.8e+4	8.1e-2	4.4e+2	4.2e-5	2.1e-1	8.8

TABLE V
CORRECT CLASSIFICATION RATES IN BOLDFACE (AND STANDARD DEVIATIONS IN REGULAR) FOR ALL FIVE METHODOLOGIES FOR CLASSIFICATION DATA SETS. "WISC. B.C." FOR WISCONSIN BREAST CANCER DATA SET AND "PIMA I.D." FOR PIMA INDIANS DIABETES DATA SET

	Iris	Wisconsin B.C.	Pima I.D.	Wine
SVM	**95.4**	**91.6**	**72.7**	**95.83**
	1.9	1.7	1.5	2.9
OPELM	**95.0**	**95.6**	**74.9**	**90.7**
	2.1	1.3	2.4	4.9
ELM	**72.2**	**95.6**	**72.2**	**81.8**
	1.01	1.2	1.9	6.2
GP	**95.6**	**97.3**	**76.3**	**96.1**
	2.3	0.9	1.8	2.1
MLP	**94.8**	**95.6**	**75.2**	**96.0**
	3.8	1.9	1.9	2.4

TABLE VI
DETAILS OF NUMBERS OF SELECTED NEURONS IN OP-ELM FOR THE DELTA AILERONS AND IRIS DATA SETS. "L" STANDS FOR LINEAR NEURONS, "S" FOR SIGMOID ONES, AND "G" FOR GAUSSIAN ONES

	Ailerons				Iris			
Run #	L	S	G	Total	L	S	G	Total
1	5	50	25	80	2	16	6	24
2	5	50	30	85	3	17	4	24
3	5	49	21	75	2	16	6	24
4	5	50	45	100	2	8	4	14
5	5	50	40	95	2	13	4	19
6	4	43	13	60	2	4	3	9
7	5	48	17	70	2	7	5	14
8	4	36	10	50	2	5	2	9
9	5	50	45	100	2	10	2	14
10	3	27	5	35	2	13	4	19

By the use of these steps, the speed and accuracy of the OP-ELM methodology has been demonstrated, through experiments using 12 different data sets for both regression and classification problems, all very different in terms of number of samples, variables, and outputs. The OP-ELM achieves roughly the same level of accuracy than the other well-known methods such as SVM, MLP, or GP. Even though the original ELM is much faster than the OP-ELM based on it, the accuracy of the ELM can be problematic in many cases, while the OP-ELM remains robust to all tested data sets.

The main goal in this brief was **not** to show that the OP-ELM is either the best in terms of MSE or the computational time. The main goal is to prove that it is a very good compromise between the speed of the ELM and the accuracy and robustness of much slower and complicated methods. Indeed, very accurate results, close to the SVM accuracy, can be obtained in a very small computational time. This makes the OP-ELM a valuable tool for the applications in need for a small response time with a good accuracy.

REFERENCES

[1] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 1998.

[2] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Netw.*, vol. 2, no. 5, pp. 359–366, 1989.

[3] G. B. Huang, Q. Y. Zhu, and C. K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, no. 1–3, pp. 489–501, Dec. 2006.

[4] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford, U.K.: Oxford Univ. Press, 1995.

[5] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA: MIT Press, 2006.

[6] C. C. Chang and C. J. Lin, LIBSVM: A Library for Support Vector Machines, 2001 [Online]. Available: http://www.csie.ntu.edu.tw/ cjlin/libsvm

[7] H. jun Rong, Y.-S. Ong, A.-W. Tan, and Z. Zhu, "A fast pruned-extreme learning machine for classification problem," *Neurocomputing*, vol. 72, no. 1–3, pp. 359–366, 2008.

[8] A. Lendasse, A. Sorjamaa, and Y. Miche, OP-ELM Toolbox, 2008 [Online]. Available: http://www.cis.hut.fi/projects/tsp/index.php?page=OPELM

[9] C. R. Rao and S. K. Mitra, *Generalized Inverse of Matrices and Its Applications*. New York: Wiley, 1972.

[10] G.-B. Huang, L. Chen, and C.-K. Siew, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *IEEE Trans. Neural Netw.*, vol. 17, no. 4, pp. 879–892, Jul. 2006.

[11] T. Poggio and F. Girosi, *A Theory of Networks for Approximation and Learning*. Cambridge, MA: MIT Press, 1989, vol. 1140.

[12] B. Schölkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond (Adaptive Computation and Machine Learning)*. Cambridge, MA: MIT Press, Dec. 2001, 0262194759.

[13] T. Similä and J. Tikka, "Multiresponse sparse regression with application to multidimensional scaling," in *Proc. Int. Conf. Artif. Neural Netw.*, 2005, vol. 3697/2005, pp. 97–102.

[14] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, "Least angle regression," *Ann. Statist.*, vol. 32, no. 2, pp. 407–499, 2004.

[15] R. Myers, *Classical and Modern Regression With Applications*, 2nd ed. Pacific Grove, CA: Duxbury, 1990.

[16] G. Bontempi, M. Birattari, and H. Bersini, "Recursive lazy learning for modeling and control," in *Proc. Eur. Conf. Mach. Learn.*, 1998, pp. 292–303.

[17] A. Asuncion and D. Newman, UCI Machine Learning Repository, Univ. California Irvine, Irvine, CA, 2007 [Online]. Available: http://archive.ics.uci.edu/ml/