Neurocomputing 74 (2011) 2413–2421

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

TROP-ELM: A double-regularized ELM using LARS and Tikhonov regularization

Yoan Miche [a,b,*], Mark van Heeswijk [a], Patrick Bas [b], Olli Simula [a], Amaury Lendasse [a]

[a] Information and Computer Science Department, Aalto University School of Science and Technology, FI-00076 Aalto, Finland
[b] Gipsa-Lab, INPG 961 rue de la Houille Blanche, BP46 F-38402 Grenoble Cedex, France

ARTICLE INFO

Available online 13 May 2011

Keywords:
ELM
Regularization
Ridge regression
Tikhonov regularization
LARS
OP-ELM

ABSTRACT

In this paper an improvement of the optimally pruned extreme learning machine (OP-ELM) in the form of a L_2 regularization penalty applied within the OP-ELM is proposed. The OP-ELM originally proposes a wrapper methodology around the extreme learning machine (ELM) meant to reduce the sensitivity of the ELM to irrelevant variables and obtain more parsimonious models thanks to neuron pruning. The proposed modification of the OP-ELM uses a cascade of two regularization penalties: first a L_1 penalty to rank the neurons of the hidden layer, followed by a L_2 penalty on the regression weights (regression between hidden layer and output layer) for numerical stability and efficient pruning of the neurons. The new methodology is tested against state of the art methods such as support vector machines or Gaussian processes and the original ELM and OP-ELM, on 11 different data sets; it systematically outperforms the OP-ELM (average of 27% better mean square error) and provides more reliable results – in terms of standard deviation of the results – while remaining always less than one order of magnitude slower than the OP-ELM.

1. Introduction

Data sets in machine learning and statistical modeling are becoming larger; thanks to improvements in acquisition processes it becomes possible to obtain large amounts of information about a studied phenomenon, with data to analyze more abundant, in terms of number of variables and samples. While it is usually desirable to have a large data set – as opposed to a small one from which very few information is available – it raises various problems. First of, the increase in the number of variables is likely to introduce new relevant data regarding the phenomenon at hand, but causes an accordingly high increase in the number of required samples, to avoid ill-posed problems. Irrelevant variables are also likely to appear, creating a new difficulty for the model building. The increase in the number of samples can also become problematic, for it leads to increased computational times for model building.

The extreme learning machine (ELM) as presented by Huang et al. in [19] by its very design is fast enough to accommodate such large data sets, where other traditional machine learning techniques have very large computational times. The main idea lies in the random initialization of the weights of a single hidden layer feedforward neural network (SLFN), instead of the traditional – much more time-consuming – learning of these weights through back-propagation [12], for example. In addition to its speed, which takes the computational time down by several orders of magnitude, the ELM is usually capable to compare with state of the art machine learning algorithms in terms of performance [19].

It has, however, been remarked in [22] that the ELM tends to suffer from the presence of irrelevant variables in the data set, as is likely to happen when dealing with real-world data. In order to reduce the effect of such variables on the ELM model, Miche et al. proposed in [22,24] a wrapper methodology around the original ELM, which includes a neuron ranking step (via a L_1 regularization known as Lasso [31]), along with a criterion used to prune out the most irrelevant neurons of the model (regarding this criterion): the optimally pruned extreme learning machine (OP-ELM). Section 2 gives a short introduction to the original ELM and fixes the notations for the following presentation of the OP-ELM as proposed in [22,24].

Section 2 then elaborates on one problem encountered by the original OP-ELM, in the computation of the pruning criterion. The leave-one-out criterion is originally used in the OP-ELM for the pruning, which can be a computationally costly choice. Thanks to the use of a closed form formula (Allen's PRESS statistic [1]), its computation is nevertheless very fast, but raises numerical problems which possibly "disturb" the pruning strategy.

* Corresponding author at: Information and Computer Science Department, Aalto University School of Science and Technology, FI-00076 Aalto, Finland.
E-mail address: yoan.miche@aalto.fi (Y. Miche).

0925-2312/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2010.12.042

The proposed solution to this situation in this paper is by the use of L_2 regularization in the OP-ELM. The concept of regularization – using L_1, L_2 or other norms-based penalties on the regression weights – for regression problems has been studied extensively (see for example [6,9,13,26,29–32,34,35]) and some of the most widely used methods are presented in Section 3: Lasso [31], Tikhonov regularization [32,13], but also hybrid penalties such as the elastic net [35] and the composite absolute penalties [34].

While these penalties are either of only one kind – L_1 or L_2, traditionally – or a hybrid using both simultaneously (see Owen's hybrid [26] for example), an approach that could be described as "in cascade" is used in this paper, for the OP-ELM. Indeed, a L_1 penalty is first used to rank the neurons, followed sequentially by a L_2 penalty to prune the network accordingly. Section 4 details the approach used, by a modification of Allen's PRESS statistic [1].

This newly modified OP-ELM is finally tested in Section 5 against three state of the art machine learning techniques (Gaussian processes, support vector machines and multi-layer perceptron) but also against the original ELM and OP-ELM. The experiments are carried out using eleven publicly available regression data sets and report the performances and timings for all methods.

2. The optimally pruned extreme learning machine

2.1. The extreme learning machine

The extreme learning machine (ELM) algorithm is proposed by Huang et al. in [19] as an original way of building a single hidden layer feedforward neural network (SLFN). The main concept behind the ELM is the random initialization of the SLFN internal weights and biases, therefore, bypassing a costly training usually performed by time-consuming algorithms (Levenberg–Marquardt [3], back-propagation [12], etc.).

In [19] a theorem is proposed – on which lies the efficiency of the ELM – stating that with a random initialization of the input weights and biases for the SLFN, and under the condition that the activation function is infinitely differentiable, the hidden-layer output matrix can be determined and will provide an approximation of the target values as good as wished (nonzero).

Under the conditions detailed in [17] – that is, randomly generated hidden nodes weights and bounded non-constant piecewise continuous activation function – the ELM is a universal function approximator [15,14]. It is worth noting that several possible activation functions have been investigated for the ELM nodes, for example thresholds [18], complex [21] and radial basis functions [16].

In this paper, the case of single-output regression is considered, but the ELM, OP-ELM and the proposed approach in Section 4 can be modified to solve multi-output regression and classification problems.

Consider a set of n distinct samples $(\mathbf{x}_i, y_i), 1 \le i \le n$, with $\mathbf{x}_i \in \mathbb{R}^p$ and $y_i \in \mathbb{R}$. A SLFN with m hidden neurons in the hidden layer can be expressed by the following sum

$$\sum_{i=1}^{m} \beta_i f(\mathbf{w}_i \mathbf{x}_j + b_i), \quad 1 \le j \le n, \tag{1}$$

with β_i the output weights, f an activation function, $\mathbf{w}_i$ the input weights and b_i the biases. Denoting by $\hat{y}_i$ the outputs estimated by the SLFN, in the hypothetical case where the SLFN perfectly approximates the actual outputs y_i, the relation is

$$\sum_{i=1}^{m} \beta_i f(\mathbf{w}_i \mathbf{x}_j + b_i) = y_j, \quad 1 \le j \le n, \tag{2}$$

which is written in matrix form as $\mathbf{H}\boldsymbol{\beta} = \mathbf{y}$, with

$$\mathbf{H} = \begin{pmatrix} f(\mathbf{w}_1 \mathbf{x}_1 + b_1) & \cdots & f(\mathbf{w}_m \mathbf{x}_1 + b_m) \\ \vdots & \ddots & \vdots \\ f(\mathbf{w}_1 \mathbf{x}_n + b_1) & \cdots & f(\mathbf{w}_m \mathbf{x}_n + b_m) \end{pmatrix}, \tag{3}$$

$\boldsymbol{\beta} = (\beta_1, \ldots, \beta_m)^T$ and $\mathbf{y} = (y_1, \ldots, y_n)^T$. The ELM approach is thus to initialize randomly the $\mathbf{w}_i$ and b_i and compute the output weights $\boldsymbol{\beta} = \mathbf{H}^{\dagger}\mathbf{y}$ by a Moore–Penrose pseudo-inverse [27] (which is identical to the ordinary least squares solution for a regression problem, see Section 3) of $\mathbf{H}$, $\mathbf{H}^{\dagger}$.

There has been recent advances based on the ELM algorithm, to improve its robustness (OP-ELM [24], CS-ELM [20]), or make it a batch algorithm, improving at each iteration (EM-ELM [7], EEM-ELM [33]). Here the case of the OP-ELM is studied, and specifically an approach aimed at regularizing the output layer determination and pruning.

2.2. The OP-ELM

The optimally pruned extreme learning machine (OP-ELM) is proposed in [24,10] in an attempt to solve the problem that ELM faces with irrelevant (or highly correlated) variables present in the data set that can "corrupt" some of the neurons. As described at more length in [24,22,23], it can be illustrated on a toy example as in Fig. 1: the plots give the ELM fit in light blue dots over the training points in black crosses. On the leftmost part of the figure, the fit by the ELM is good, but when a pure random noise variable is added, on the rightmost figure (the added noise variable is not pictured on the figure), the fit becomes loose and spread.

Indeed, the ELM is not designed to cope with such variables irrelevant to the problem at hand. In this spirit, the OP-ELM proposes a three-steps methodology, shortly described here, to address this issue.

The idea is to build a wrapper around the original ELM, with a neuron pruning strategy. For this matter, as can be seen in Fig. 2, the construction of the SLFN by the ELM is retained, and two steps are added afterwards. First comes a ranking of the neurons by a least angle regression (LARS [6]; in practice the MRSR [29] implementation of LARS is used for it also applies to multi-output

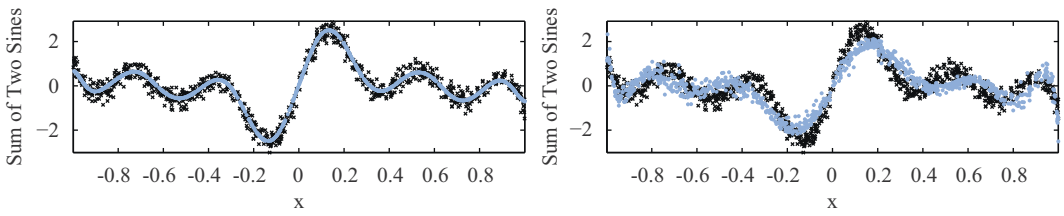


Fig. 1. Illustration of the ELM model fit (light blue dots) on a toy example (sum of sines, black crosses), for the normal data (leftmost part) and for the same data augmented with a random noise variable (not displayed), on the rightmost part. Due to the irrelevant additional variable, the fit of the ELM model is less accurate. From [24]. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Fig. 2. Illustration of the three OP-ELM steps: the SLFN is first built using the ELM approach (random initialization of internal weights and biases); then a LARS algorithm is used to rank the neurons of the hidden layer; finally the selection of the optimal number of neurons for the OP-ELM model is performed using a Leave-One-Out criterion.

cases), which sorts them by their usefulness regarding the output. And then a leave-one-out criterion is used to determine how many of the – sorted – neurons should be kept for the final OP-ELM model structure.

The LARS algorithm is not detailed here since it is described and discussed at length (or more precisely the idea it implements, Lasso) in Section 3, but it has the property of providing an exact ranking of the hidden-layer neurons in the case of the OP-ELM, since the relation between the neurons and the output is linear (by design of the OP-ELM).

The leave-one-out (LOO) method is usually a costly approach to optimize a parameter since it requires to train the model on the whole data set but one sample, and evaluate on this sample, repeatedly for all the samples of the data set. In the OP-ELM structure though, the situation is linear (between the hidden layer and the output one), and the LOO error has a closed matrix form, given by Allen's prediction sum of squares (PRESS) [1] (details of the computation of the PRESS LOO error are given in Section 4). This closed form allows for fast computation of the mean square error and hence of the output weights $\boldsymbol{\beta}$, making the OP-ELM still computationally efficient and more robust than the original ELM to irrelevant/ correlated variables.

Hence, the OP-ELM can be seen as a "regularized" ELM, by the use of a LARS approach, which is a L_1 penalty on a regression problem, here.

Meanwhile, the decision over the final number of neurons to retain (by a LOO criterion) has shown potential instabilities (numerically), due to the nature of the matrix operations performed in the PRESS formula (see Section 4 for these calculations). The proposed solution in this paper is to use regularization in the calculations of the PRESS formula. In the following are reviewed the most well-known algorithms used to perform regularization, using a L_1 and L_2 (and jointly L_1 and L_2) penalty on the regression problem. The proposed approach in Section 4 combines both L_1 and L_2 penalties in the OP-ELM, to regularize the network.

3. The problem of regularization

Here are presented some of the most widely used methods to regularize a regression problem (which is the situation between the hidden layer and the output layer of the OP-ELM).

In the following, matrices are denoted by boldface; $\mathbf{A}$ is a $n \times p$ matrix with $\mathbf{A} = (\mathbf{a}_1, \ldots, \mathbf{a}_n)^T$, $\mathbf{a}_i \in \mathbb{R}^p$. Also $\mathbf{A}$ can be referred by $\mathbf{A} = (a_{ij})_{1 \le j \le p}^{1 \le i \le n}$. Capital boldface $\mathbf{A}$ are used for matrices and low-case boldface $\mathbf{b}$ for vectors.

3.1. General case

For the general setup, assume a single-output regression problem of the form

$$\mathbf{y} = \mathbf{X}\mathbf{w} + \boldsymbol{\varepsilon}, \tag{4}$$

with $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)^T$ the inputs of the problem (data set), $\mathbf{y} = (y_1, \ldots, y_n)^T$ the actual output, $\mathbf{w} = (w_1, \ldots, w_p)^T$ the regression weights and $\boldsymbol{\varepsilon} = (\varepsilon_1, \ldots, \varepsilon_n)^T$ the residuals.

Traditionally, the ordinary least squares (OLS) solution (a.k.a. Gauss–Markov solution) is a possible approach to solve this problem. The problem can be formulated as a minimization of the mean square error as

$$\min_{\hat{\mathbf{w}}} (\mathbf{y} - \mathbf{X}\hat{\mathbf{w}})^T (\mathbf{y} - \mathbf{X}\hat{\mathbf{w}}), \tag{5}$$

or in a non-matrix form

$$\min_{\hat{\mathbf{w}}} \sum_{i=1}^{n} (y_i - \mathbf{x}_i \hat{\mathbf{w}})^2, \tag{6}$$

with $\hat{\mathbf{w}} = (\hat{w}_1, \ldots, \hat{w}_n)^T$ the estimated regression weights.

The solution of Eq. (5) is then obtained by a classical pseudo-inverse (Moore–Penrose [27]) as

$$\hat{\mathbf{w}}^{\mathrm{OLS}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}, \tag{7}$$

assuming that $\mathbf{X}$ is full rank.

This way of computing the solution involves matrix inversion (for the computation of the inverse covariance matrix $(\mathbf{X}^T\mathbf{X})^{-1}$) which tends to pose numerical problems in practice, since $\mathbf{X}$ is sometimes not full rank (there might very well be irrelevant or linear combinations of samples and/ or variables in the data set). A numerically more stable solution is to use the singular value decomposition (SVD) of $\mathbf{X}$ to compute the pseudo-inverse. The proposed approach presented in Section 4 makes use of the SVD for faster computations and numerical stability.

Two classical critiques of the OLS solution relate to the two main aspects that one expects from a model. First, the OLS is likely to perform poorly on real data (for example for the numerical reasons invoked before), while it is expected that the model should perform reasonably well on the training data. Second, it is usually desirable to have a sparse model which makes interpretation possible, regarding the relationships between variables and the output. Again, the OLS is not designed in this sense and does not provide sparse models at all.

Also, it has been shown (e.g. in [2,30]) that there exists solutions which achieve lower mean square error (MSE) than the OLS one – for numerical instability reasons, in practice – for example by the use of regularization factors, which can be seen as penalties added to the minimization problem in Eq. (5). In addition, regarding the generalization error, the OLS solution found in training is possibly not the best one (in terms of generalization MSE).

Here are detailed two different approaches to regularization, using either a L_1 or L_2 penalty term.

3.2. The L_1 penalty: LASSO

Let us first consider the case of the L_1 penalty. In this setup, the minimization problem of Eq. (6) becomes

$$\min_{\lambda, \hat{\mathbf{w}}} \left[\sum_{i=1}^{n} (y_i - \mathbf{x}_i \hat{\mathbf{w}})^2 + \lambda \sum_{j=1}^{p} |\hat{w}_j| \right], \tag{8}$$

again with $\hat{\mathbf{w}} = (\hat{w}_1, \ldots, \hat{w}_n)^T$. An instance of this very problem is studied by Tibshirani in [31] and is commonly known as the

LASSO (least absolute shrinkage and selection operator). Due to the nature of the minimization problem (L_1 penalty on the regression coefficients), the Lasso produces solutions that exhibit sparsity, making interpretability possible. Control over this sparsity of the final model is obtained by modifying the λ value: the smaller is λ, the more $\hat{w}_j$ coefficients are non-zero and hence the more variables are retained in the final solution.

Generally, the computation of the solution to Eq. (8) is a quadratic programming problem with linearity constraint which can be intensive. In [6], a computationally more efficient algorithm is presented as the LARS algorithm (least angle regression), of which the Lasso is a specific instance. LARS actually generalizes both the Lasso and the forward stagewise regression strategy (see [11] for example): the algorithm starts similarly to forward selection, with all coefficients equal to zero and finds the variable most correlated with the output. The direction of this first selected variable is followed until another variable has as much correlation with the output. LARS then follows the direction of the equiangular between first and second selected variables, until a third variable as much correlated with the output is found. The set of selected variables grows until none remain to be chosen (please refer to the original paper [6] for the computationally efficient implementation proposed by the authors).

By enforcing a restriction on the sign of the weights (which has to be the same as that of the current direction of the correlation), the LARS algorithm thus implements Lasso effectively. The authors claim an order of magnitude greater speed than the classical quadratic programming problem, using their algorithm.

Meanwhile, as noted by Zou and Hastie in [35] for example, the Lasso presents some drawbacks:

- If $p > n$, i.e. there are more variables than samples, the Lasso selects at most n variables [6].
- For classical situations where $n > p$, and if the variables are correlated, it seems (from experiments in [31]) that the Tikhonov regularization (in the following subsection 3.3) outperforms the Lasso.

A common drawback of the L_1 penalty and, therefore, of the Lasso approach is that it tends to be too sparse in some cases, i.e. there are many j such that $\hat{w}_j = 0$. In addition, the control over the sparsity by the parameter λ can be challenging to tune.

3.3. The L_2 penalty: Tikhonov regularization

Another possible approach to find a solution which deems a lower MSE than the OLS one is to use regularization in the form of Tikhonov regularization proposed in [32] (a.k.a. ridge regression [13]).

This time, the minimization problem involves a penalty using the square of the regression coefficients

$$\min_{\lambda,\hat{\mathbf{w}}} \left[\sum_{i=1}^{n} (y_i - \mathbf{x}_i \hat{\mathbf{w}})^2 + \lambda \sum_{j=1}^{p} \hat{w}_j^2 \right]. \tag{9}$$

Thanks to a bias–variance tradeoff, the Tikhonov regularization achieves better prediction performance than the traditional OLS solution. And as mentioned in the previous Subsection 3.2, it outperforms the Lasso solution in cases were the variables are correlated. One famous advantage of the Tikhonov regularization is that it tends to identify/ isolate groups of variables, enabling further interpretability (this grouping can be very desirable for some data sets, as mentioned in [35]).

The major drawback of this regularization method is similar to one mentioned for the OLS: it does not give any parsimonious solution, since all variables are retained, due to the L_2 penalty. Therefore, contrary to the Lasso which actually performs variable selection "internally" – given that λ is large enough to set some coefficients to zero – the Tikhonov regularization does not select variables directly.

3.4. Hybrid penalties

In an attempt to overcome the drawbacks of each of the two approaches, hybrid solutions have been developed, which use both the L_1 and the L_2 penalties in the same minimization problem. Below are proposed three approaches that tackle this problem: the elastic net [35], the composite absolute penalties [34], and finally an original approach by Owen [26] using an "inverted" Huber loss function.

3.4.1. The elastic net

Zhou and Hastie in [35] propose to alleviate the problems encountered by the Tikhonov regularization (lack of sparsity) while keeping its good performance thanks to the L_2 penalty. This is done using a composite of the Lasso and Tikhonov regularization, by combining the two penalties L_1 and L_2 in the form of a weighted penalty

$$\lambda_1 \sum |\hat{w}_j| + \lambda_2 \sum \hat{w}_j^2, \tag{10}$$

with λ_1 and λ_2 positive (controlling the sparsity of the model). This version of the penalty term is denoted as the "naïve" elastic net by the authors, which admits an easily computed solution, provided that λ_1 and λ_2 are defined and already optimal. As the authors mention in [35], this naïve version of the algorithm is fast to obtain and rather efficient, but creates a greater shrinkage effect (on the regression coefficients) than the original Lasso, which adds bias to the solution, while not reducing significantly the variance of it. In the end, the naïve version only seems to work well when it is close enough to the Tikhonov or Lasso case (i.e. λ_1 very small or λ_2 very small).

The "normal" version of the elastic net is then a scaled naïve one: defining $\mathbf{Y}^* = \begin{pmatrix} \mathbf{Y} \\ \mathbf{0}_{p\times 1} \end{pmatrix}$, $\mathbf{X}^* = (1/\sqrt{1+\lambda_2}) \begin{pmatrix} \mathbf{X} \\ \sqrt{\lambda_2}\, \mathbf{I}_{p\times p} \end{pmatrix}$ and $\hat{\mathbf{w}}^* = \sqrt{1+\lambda_2}\hat{\mathbf{w}}$, the minimization problem for the elastic net is then

$$\min_{\hat{\mathbf{w}}^*} \left[\sum_{i=1}^{n+p} (y_i^* - \mathbf{x}_i^* \hat{\mathbf{w}}^*)^2 + \frac{\lambda_1}{\sqrt{1+\lambda_2}} \sum_{j=1}^{p} |\hat{w}_j| \right]. \tag{11}$$

The scaling performed allows to reduce the problem of shrinkage present in the naïve version of the elastic net, while retaining the advantages of the original naïve approach (e.g. the automatic variable selection).

In practice, the algorithm is implemented as a modification of the LARS algorithm (the LARS-EN) since once λ_2 is fixed, the computations are similar to that of a Lasso.

While the LARS-EN is a very efficient way of implementing the elastic net approach, it remains that two parameters need optimizing: λ_1 and λ_2. Usually, this is done by the use of classical cross-validation (CV) which is unfortunately costly for it requires a two-dimensional search, which is hardly feasible if one wants to keep the ELM speed property.

3.4.2. Composite absolute penalty (CAP)

In [34], Zhao et al. propose to use a more generalized version of the penalty term, by using a vector of penalties on which is computed a norm. Denoting by

$$\|\mathbf{a}\|_\gamma = \left(\frac{1}{n} \sum_{i=1}^{n} |a_i|^\gamma \right)^{1/\gamma}, \tag{12}$$

the γ-norm of a vector $\mathbf{a} = (a_1, \ldots, a_n)^T$ for $\gamma \in \mathbb{N}^*$, the method of composite absolute penalties (CAP) generalizes the concept used

in the elastic net penalty to

$$\|(\|\hat{\mathbf{w}}_{G_1}\|_{\gamma_1}, \|\hat{\mathbf{w}}_{G_2}\|_{\gamma_2}, \ldots, \|\hat{\mathbf{w}}_{G_k}\|_{\gamma_k})\|_{\gamma_0}, \tag{13}$$

where G_j is a subset of $\{1,\ldots,p\}$ and $\hat{\mathbf{w}}_{G_j}$ is obtained by extracting the components denoted in G_j from $\hat{\mathbf{w}}$.

It can be seen that the penalty term is, therefore, a γ_0-norm on a vector of penalties $\|\hat{\mathbf{w}}_{G_i}\|_{\gamma_i}$. This general formulation of the penalty for example comes down to the Lasso when $\gamma_j = 1, \forall j$.

While the generalization capability of the CAP approach is clear, the determination of the groups G_j and of the γ_j is time-consuming and prone to heuristics/ *a priori* information on the variables. Again, cross-validation is typically used for the determination of the γ_j, leading to important computational times, again not "compatible" with the ELM speed.

3.4.3. Owen's hybrid

A slightly different approach is proposed by Owen in [26], by the use of an original loss function for the penalty. The problem is formulated as

$$\min\left[\sum_{i=1}^{n} L(y_i - \mathbf{x}_i\hat{\mathbf{w}}) + \sum_{j=1}^{p} P(\hat{w}_j)\right], \tag{14}$$

where the $L(\cdot)$ function can be assumed to be a 2-norm $\|\cdot\|_2^2$ in this case. The emphasis is here put on the P function, which is chosen (or more "designed") to behave like an absolute value function for small $\hat{w}_j$ for sparse solutions to arise, and like a quadratic function on large $\hat{w}_j$ to retain the properties of the Tikhonov regularization.

The author proposes an "inverted" Huber loss function for that purpose. While the Huber function is such that

$$\mathcal{H}(z) = \begin{cases} z^2 & \text{for } |z| \le 1 \\ 2|z| - 1 & \text{for } |z| \ge 1 \end{cases}, \tag{15}$$

the "inverted" version (also scaled to accommodate thresholding) is given by

$$\mathcal{B}_M(z) = \begin{cases} |z| & \text{for } |z| \le M \\ \dfrac{z^2 + M^2}{2} & \text{for } |z| \ge M \end{cases}. \tag{16}$$

The M value permits to choose where the transition between the absolute value function and the quadratic one takes place.

The minimization problem ends up as a convex one, with a large number of constraints (see [26] for more details), which is unfortunately computationally very expensive for significant data sets.

In the end, it can be noted that all the variants of the minimization problem presented here are convex problems and have hence an optimal solution that is reachable by standard convex optimization techniques. Unfortunately, the large number of parameters or constraints on the minimization problem makes it more difficult to solve, and cross-validation is often used for the determination of the parameters. This in turn implies large computational times.

While the properties of both the Lasso and the Tikhonov regularization are desirable, the penalties combining both lead to complex minimization problems which take too long for the application to OP-ELM.

The following Section 4 proposes to use the two approaches in turn, instead of together, along with fast matrix computations.

4. Regularized ELM

Recently, Deng et al. in [5] proposed a regularized extreme learning machine algorithm, which is essentially a L_2 penalized ELM, with a possibility to weight the sum of squares in order to address outliers interference. Using the notations from the previous section, the minimization problem is here

$$\min_{\lambda,\mathbf{d},\hat{\mathbf{w}}}\left[\lambda\sum_{i=1}^{n}(d_i(y_i - \mathbf{x}_i\hat{\mathbf{w}}))^2 + \sum_{j=1}^{p}\hat{w}_j^2\right], \tag{17}$$

where the d_i are the weights meant to address the outliers.

This extension of the ELM clearly (from the results in [5]) brings a very good robustness to outliers to the original ELM. Unfortunately, it suffers from the problems related to L_2 penalties, that is the lack of sparsity for example.

As described before, the original OP-ELM already implements a L_1 penalty on the output weights, by performing a LARS between the hidden and output layer.

It is here proposed to modify the original PRESS LOO criterion for the selection of the optimal number of neurons by adding a Tikhonov regularization factor in the PRESS, therefore, making the modified PRESS LOO a L_2 penalty applied on the L_1 penalized result from the LARS.

In the following are used matrix operations such as $\mathbf{A}/\mathbf{B}$ to refer to the matrix $\mathbf{C}$ such that $(c_{i,j}) = a_{i,j}/b_{i,j}$. Also the diag$(\cdot)$ operator is used to extract the diagonal of a matrix, $\text{diag}(\mathbf{A}) = (a_{1,1},\ldots,a_{n,n})^T$.

4.1. L_1 and L_2 regularized OP-ELM

4.1.1. Allen's PRESS

The original PRESS formula used in the OP-ELM was proposed by Allen in [1]. The original PRESS formula can be expressed as

$$\text{MSE}^{\text{PRESS}} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \mathbf{x}_i(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T y}{1 - \mathbf{x}_i(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{x}_i^T}\right)^2, \tag{18}$$

which means that each observation is "predicted" using the other $n-1$ observations and the residuals are finally squared and summed up. Algorithm 1 proposes to implement this formula in an efficient way, by matrix computations.

Algorithm 1. Allen's PRESS algorithm, in a fast matrix form.

1: Compute the utility matrix $\mathbf{C} = (\mathbf{X}^T\mathbf{X})^{-1}$
2: And $\mathbf{P} = \mathbf{X}\mathbf{C}$;
3: Compute the pseudo-inverse $\mathbf{w} = \mathbf{C}\mathbf{X}^T\mathbf{y}$;
4: Compute the denominator of the PRESS $\mathbf{D} = \mathbf{1} - \text{diag}(\mathbf{P}\mathbf{X}^T)$;
5: And finally the PRESS error $\boldsymbol{\varepsilon} = \frac{\mathbf{y} - \mathbf{X}\mathbf{w}}{\mathbf{D}}$;
6: Reduced to a MSE, $\text{MSE}^{\text{PRESS}} = \frac{1}{n}\sum_{i=1}^{n}\varepsilon_i^2$.

The main drawback of this approach lies in the use of a pseudo-inverse in the calculation (in the Moore–Penrose sense), which can lead to numerical instabilities if the data set $\mathbf{X}$ is not full rank. This is unfortunately very often the case, with real-world data sets.

The following approach proposes two improvements on the computation of the original PRESS: regularization and fast matrix calculations.

4.1.2. Tikhonov-regularized PRESS (TR-PRESS)

In [9], Golub et al. note that the singular value decomposition (SVD) approach to compute the PRESS statistic is preferable to the traditional pseudo-inverse mentioned above, for numerical reasons. In this very same paper is proposed a generalization of Allen's PRESS, as the generalized cross-validation (GCV) method, which is technically superior to the original PRESS, for it can handle cases were the data is extremely badly defined—for example if all $\mathbf{X}$ entries are 0 except the diagonal ones.

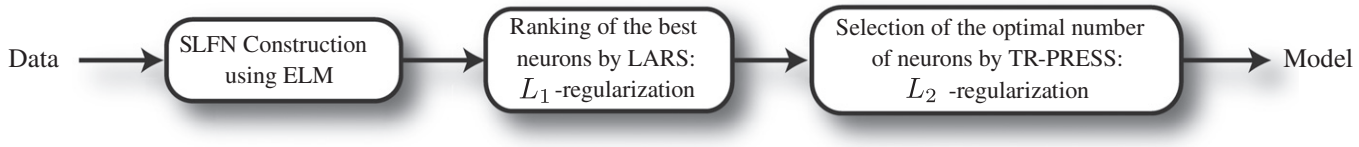


Fig. 3. The proposed regularized OP-ELM (TROP-ELM) as a modification of Fig. 2.

In practice, from our experiments, while the GCV is supposably superior, it leads to identical solutions with an increased computational time, compared to the original PRESS and the Tikhonov-regularized version of PRESS presented below.

Algorithm 2 gives the computational steps used, in matrix form, to determine the $\mathrm{MSE}^{\mathrm{TR-PRESS}}(\lambda)$ from

$$\mathrm{MSE}^{\mathrm{TR-PRESS}}(\lambda) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \mathbf{x}_i(\mathbf{X}^T\mathbf{X}+\lambda\mathbf{I})^{-1}\mathbf{x}^T y}{1-\mathbf{x}_i(\mathbf{X}^T\mathbf{X}+\lambda\mathbf{I})^{-1}\mathbf{x}_i^T}\right)^2, \tag{19}$$

which is the regularized version of Eq. (18). The notation $A \circ B$ denotes element-wise product between matrices A and B (Schur product). It is used in step 4 of Algorithm 2 for it is faster than standard matrix product.

Algorithm 2. Tikhonov-regularized PRESS. In practice, the REPEAT part of this algorithm (convergence for λ) is solved by a Nelder–Mead approach [25], a.k.a. downhill simplex.

1: Decompose $\mathbf{X}$ by SVD: $\mathbf{X} = \mathbf{USV}^T$;
2: Compute the products (used later): $\mathbf{A} = \mathbf{XV}$ and $\mathbf{B} = \mathbf{U}^T\mathbf{y}$;
3: **repeat**
4: Using the SVD of $\mathbf{X}$, compute the $\mathbf{C}$ matrix by:

$$\mathbf{C} = \mathbf{A} \circ \begin{pmatrix} \frac{S_{11}}{S_{11}^2+\lambda} & \cdots & \frac{S_{nn}}{S_{nn}^2+\lambda} \\ \vdots & \vdots & \vdots \\ \frac{S_{11}}{S_{11}^2+\lambda} & \cdots & \frac{S_{nn}}{S_{nn}^2+\lambda} \end{pmatrix};$$

5: Compute the $\mathbf{P}$ matrix by: $\mathbf{P} = \mathbf{CB}$;
6: Compute $\mathbf{D}$ by: $\mathbf{D} = \mathbf{1} - \mathrm{diag}(\mathbf{CU}^T)$;
7: Evaluate $\boldsymbol{\varepsilon} = \frac{\mathbf{y}-\mathbf{P}}{\mathbf{D}}$ and the actual MSE by $\mathrm{MSE}^{\mathrm{TR-PRESS}} = \frac{1}{n}\sum_{i=1}^{n}\varepsilon_i^2$;
8: **until** convergence on λ is achieved
9: Keep the best $\mathrm{MSE}^{\mathrm{TR-PRESS}}$ and the λ value associated.

Globally, the algorithm uses the SVD of $\mathbf{X}$ to avoid computational issues, and introduces the Tikhonov regularization parameter in the calculation of the pseudo-inverse by the SVD. This specific implementation happens to run very quickly, thanks to the pre-calculation of utility matrices ($\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$) before the optimization of λ.

In practice, the optimization of λ in this algorithm is performed by a Nelder–Mead [25] minimization approach, which happens to converge very quickly on this problem (`fminsearch` function in Matlab).

Through the use of this modified version of PRESS, the OP-ELM has an L_2 penalty on the regression weights (regression between the hidden and output layer), for which the neurons have already been ranked using an L_1 penalty. Fig. 3 is a modified version of Fig. 2 illustrating the TROP-ELM approach.

Fig. 4 illustrates the effect of the regularization factor introduced in the TR-PRESS: the mean square error is more stable regarding the increase of the number of neurons following the ranking provided by LARS (L_1 penalty). The introduction of the L_2 penalty has a very visible regularization effect here (the situation is similar for the other data sets), avoiding numerical instabilities, for example.

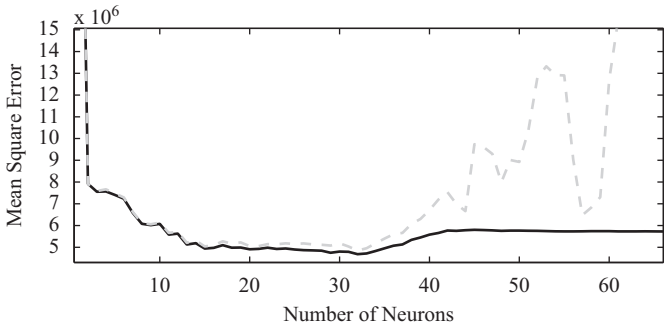


Fig. 4. Comparison of the MSE for the original OP-ELM (gray dashed line) and the proposed TROP-ELM (solid black line) for one data set (Auto Price, see Section 5) for a varying amount of neurons (in the order ranked by the LARS). The regularization enables here to have a more stable MSE along the increase of the number of neurons.

The following section proposes a comparison of the modified OP-ELM[1] (denoted TROP-ELM for Tikhonov-regularized OP-ELM) with the original OP-ELM on 11 different regression data sets from the UCI machine learning repository [8], along with other classical machine learning methods.

5. Experiments

In order to compare the proposed TROP-ELM with the original OP-ELM and other typical machine learning algorithms, eleven data sets from UCI machine learning repository [8] have been used. They are chosen for their heterogeneity in terms of problem, number of variables, and sizes.

Table 1 summarizes the details of each data set.

The data sets have all been processed in the same way: for each data set, 10 different random permutations are taken without replacement; for each permutation, two thirds are taken for the training set, and the remaining third for the test set (see Table 1). Training sets are then normalized (zero-mean and unit variance) and test sets are also normalized using the very same normalization factors than for the corresponding training set. The results presented in the following are hence the average of the 10 repetitions for each data set. This also enables to obtain an estimate of the standard deviation of the results presented (see Table 2).

It should be noted that most of the results presented in Tables 2 and 3 are from [24] and are reproduced here for comparison purposes.

As mentioned in the original paper [24], experiments are performed using the online available versions of the methodologies, unaltered. All experiments have been run on the same *x*86_64 Linux machine with at least 4 GB of memory (no swapping

[1] A toolbox implementing the used TROP-ELM will be available at http://www.cis.hut.fi/projects/eiml/research/downloads.

Table 1
Details of the data sets used and the proportions for training and testing sets for each (two thirds of the whole set for training and one third for testing), along with the number of variables.

	Abalone	Ailerons	Elevators	Computer	Auto P.	CPU	Servo	Breast C.	Bank	Stocks	Boston
# of Variables	8	5	6	12	15	6	4	32	8	9	13
Training	2784	4752	6344	5461	106	139	111	129	2999	633	337
Test	1393	2377	3173	2731	53	70	56	65	1500	317	169

Table 2
Mean Square Error results in boldface (standard deviations in regular) for all six methodologies for regression data sets. "Auto P." stands for Auto Price and "Breast C." for Breast Cancer data sets. Results for all algorithms but TROP-ELM are originally in [24] and are reproduced for comparison purposes.

	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
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
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
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	7.0e−4	9.35	2.1e+1
OP-ELM	**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
TROP-ELM	**4.8**	**2.7e−8**	**2.0e−6**	**2.4e+1**	**7.0e+6**	**4.1e+3**	**6.1e−1**	**1.1e+3**	**1.1e−3**	**8.4e−1**	**1.9e+1**
	4.2e−1	1.5e−9	5.2e−8	6.2	2.2e+6	2.9e+3	2.2e−1	1.7e+2	3.4e−5	5.8e−2	4.4

Table 3
Computational times (in seconds) for all five methodologies on the regression data sets. "Auto P." stands for Auto Price and "Breast C." for Breast Cancer data sets. Timings for all algorithms but TROP-ELM are originally in [24] and are reproduced for comparison purposes.

	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
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
OP-ELM	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
TROP-ELM	12.2	14.6	44.3	13.9	4.8e−1	1.2	8.4e−1	7.8e−1	4.4	1.1	1.5

for any of the experiments) and 2+ GHz processor. Also, even though some methodologies implementations are taking advantage of parallelization, computational times are reported considering single-threaded execution on one single core, for the sake of comparisons.

The SVM is performed using the SVM toolbox [4]; MLP [3] is using a neural network toolbox, part of the Matlab software from the MathWorks, Inc; the GPML toolbox for Matlab from Rasmussen and Williams [28] is used for the GP; finally, the OP-ELM was used with all possible kernels, linear, sigmoid, and Gaussian, using a maximum number of 100 neurons and similarly for the TROP-ELM. For more details on the parameters used for each toolbox, please refer to [24].

First are reported the mean square errors (and standard deviations) for the six algorithms tested. It can be seen that the proposed TROP-ELM is always at least as good as the original OP-ELM, with an improvement on the standard deviation of the results, over the 10 repetitions for each data set (only for the Boston housing case is the standard deviation larger for the TROP-ELM than the OP-ELM): over the 11 data sets, the TROP-ELM performs on average 27% better than the original OP-ELM and gives a standard deviation of the results 52% lower than that of the OP-ELM (also on average over the 11 data sets).

Also, the TROP-ELM is clearly as good (or better) as the GP in six out of the 11 data sets – Ailerons, Elevators, Auto Price, Breast Cancer, Bank and Boston – in which cases it has a similar (or lower) standard deviation of the results. This with a computational time usually two or three orders of magnitude lower than the GP.

Table 3 gives the computational times for each algorithm and each data set (average of the 10 repetitions).

It can be seen that the TROP-ELM keeps computational times of the same order as that of the OP-ELM (although higher on average), and remains several orders of magnitudes faster than the GP, MLP or SVM. Of course, as for the OP-ELM, the computational times remain one to two orders of magnitude above the original ELM.

Finally, in Table 4 are reported the average number of neurons (average over the 10 repetitions for each data set) selected for the final model structure of the OP-ELM and TROP-ELM. It can be seen that only in the cases of computer activity and stocks data sets are all the neurons selected for the final model (which suggests that a larger number of neurons given initially to the model might lead to better

Table 4
Average (over the ten repetitions) of the number of neurons selected for the final model (out of 100 initially in the model) for both OP-ELM and TROP-ELM.

	Abalone	Ailerons	Elevators	Computer	Auto P.	CPU	Servo	Breast C.	Bank	Stocks	Boston
OP-ELM	36	75	74	100	14	33	36	12	98	100	66
TROP-ELM	42	80	53	100	15	28	42	14	93	100	59

results…). Otherwise, the selected amount varies largely over the data sets and slightly between the OP-ELM and TROP-ELM.

The effect of the L_1 penalty is here obvious, on the number of neurons retained in the final model structure (compared to the ELM or regularized ELM, for example, which are less parsimonious), while the L_2 penalty introduced enables to regularize the weights chosen and improve the performances of the final model (compared to the OP-ELM, that is).

6. Conclusions and future work

In this paper is proposed a modification of the original optimally pruned extreme learning machine (OP-ELM), by the use of a L_2 penalty in the PRESS leave-one-out estimation and a fast matrix computation strategy.

The OP-ELM was proposed in the first place as a wrapper around ELM to improve its robustness by adding a neuron pruning strategy based on LARS (L_1 penalty) and leave-one-out (LOO). Here the LOO criterion is modified to add a L_2 penalty (Tikhonov regularization) to the estimate, in order to regularize the matrix computations and hence make the MSE computation more reliable. The modified OP-ELM (TROP-ELM), therefore, uses "in cascade" L_1 and L_2 penalties, avoiding the large computational times problems commonly encountered when attempting to intertwine the two penalties (as in the elastic net or the composite absolute penalty approaches).

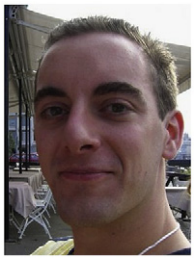
The TROP-ELM shows better performance than the original OP-ELM, with an average of 27% better MSE for the considered data sets (and improvements between 0% and 96% over the data sets used). Also notable is the decrease of the standard deviation of the results over the multiple repetitions for each data set, illustrating that the regularization introduced has a visible effect. In the end, the TROP-ELM performs rather similarly to the Gaussian processes on more than half the data sets tested, for a computational time which remains two to three orders of magnitude below—and less than an order of magnitude slower than the OP-ELM, in the worst case among the data sets used.

Future work on the TROP-ELM includes a generalization to multi-output regression and classification (binary and multi-class), by the use of the MRSR implementation of LARS which makes it possible.

References

[1] M.A. David, The relationship between variable selection and data augmentation and a method for prediction, Technometrics 16 (1) (1974) 125–127.
[2] J. Berger, Minimax estimation of a multivariate normal mean under arbitrary quadratic loss, Journal of Multivariate Analysis 6 (2) (1976) 256–264.
[3] C.M. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, USA, 1996.
[4] C.-C. Chang, C.-J. Lin, LIBSVM: a library for support vector machines, Software available at ⟨http://www.csie.ntu.edu.tw/∼cjlin/libsvm⟩, 2001.
[5] W. Deng, Q. Zheng, L. Chen, Regularized extreme learning machine, in: IEEE Symposium on Computational Intelligence and Data Mining, CIDM '09, March 30th–April 2nd 2009, pp. 389–395.
[6] B. Efron, T. Hastie, I. Johnstone, R. Tibshirani, Least angle regression, Annals of Statistics 32 (2004) 407–499.
[7] G. Feng, G.-B. Huang, Q. Lin, R. Gay, Error minimized extreme learning machine with growth of hidden nodes and incremental learning, IEEE Transactions on Neural Networks 20 (8) (2009) 1352–1357.
[8] A. Frank, A. Asuncion, UCI machine learning repository, ⟨http://archive.ics.uci.edu/ml⟩, 2010.
[9] G.H. Golub, M. Heath, G. Wahba, Generalized cross-validation as a method for choosing a good ridge parameter, Technometrics 21 (2) (1979) 215–223.
[10] EIML Group. The op-elm toolbox, Available online at ⟨http://www.cis.hut.fi/projects/eiml/research/downloads/op-elm-toolbox⟩, 2009.
[11] T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction, second ed., Springer, 2009.
[12] S. Haykin, Neural Networks: A Comprehensive Foundation, second ed., Prentice Hall, 1998 ISBN 0132733501.
[13] A.E. Hoerl, Application of ridge analysis to regression problems, Chemical Engineering Progress 58 (1962) 54–59.
[14] G.-B. Huang, L. Chen, Convex incremental extreme learning machine, Neurocomputing 70 (16–18) (2007) 3056–3062 ISSN 0925-2312.
[15] G.-B. Huang, L. Chen, Enhanced random search based incremental extreme learning machine, Neurocomputing 71 (16–18) (2008) 3460–3468 ISSN 0925-2312.
[16] G.-B. Huang, C.-K. Siew, Extreme learning machine with randomly assigned rbf kernels, International Journal of Information Technology 11 (1) (2005) 16–24.
[17] G.-B. Huang, L. Chen, C.-K. Siew, Universal approximation using incremental constructive feedforward networks with, IEEE Transactions on Neural Networks 17 (2005) 879–892.
[18] G.-B. Huang, Q.-Y. Zhu, K.Z. Mao, C.-K. Siew, P. Saratchandran, N. Sundararajan, Can threshold networks be trained directly? IEEE Transactions on Circuits and Systems II: Express Briefs 53 (3) (2006) 187–191 ISSN 1549-7747.
[19] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: theory and applications, Neurocomputing 70 (1–3) (2006) 489–501.
[20] Y. Lan, Y.C. Soh, G.-B. Huang, Constructive hidden nodes selection of extreme learning machine for regression, Neurocomputing 73 (16-18) (2010) 3191–3199.
[21] M.-B. Li, G.-B. Huang, P. Saratchandran, N. Sundararajan, Fully complex extreme learning machine, Neurocomputing 68 (2005) 306–314 ISSN 0925-2312.
[22] Y. Miche, P. Bas, C. Jutten, O. Simula, A. Lendasse, A methodology for building regression models using extreme learning machine: OP-ELM, in: M. Verleysen (Ed.), ESANN 2008, European Symposium on Artificial Neural Networks, Bruges, Belgium, d-side publ, (Evere, Belgium), 23–25 April 2008, pp. 247–252.
[23] Y. Miche, A. Sorjamaa, A. Lendasse, OP-ELM: theory, experiments and a toolbox, in: R. Neruda, V. Kurková, J. Koutník (Eds.), LNCS-Artificial Neural Networks—ICANN 2008—Part I, Lecture Notes in Computer Science, vol. 5163, Springer Berlin / Heidelberg, September 2008, pp.145–154.
[24] 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.
[25] J.A. Nelder, R. Mead, A simplex method for function minimization, The Computer Journal 7 (4) (1965) 308–313.
[26] A.B. Owen, A robust hybrid of lasso and ridge regression, Technical Report, Stanford University, 2006.
[27] C. Radhakrishna Rao, S.K. Mitra, Generalized Inverse of Matrices and Its Applications, John Wiley & Sons Inc, 1971.
[28] C.E. Rasmussen, C.K.I. Williams, Gaussian Processes for Machine Learning, The MIT Press, 2006.
[29] T. Similä, J. Tikka, Multiresponse sparse regression with application to multidimensional scaling, in: International Conference on Artificial Neural Networks (ICANN), Lecture Notes in Computer Science, vol. 3697, Warsaw, Poland, 11–15 September 2005, pp. 97–102.
[30] R.A. Thisted, Ridge regression, minimax estimation, and empirical bayes methods, Technical Report 28, Division of Biostatistics, Stanford University, 1976.
[31] R. Tibshirani, Regression shrinkage and selection via the lasso, Journal of the Royal Statistical Society, Series B 58 (1994) 267–288.
[32] A.N. Tychonoff, Solution of incorrectly formulated problems and the regularization method, Soviet Mathematics 4 (1963) 1035–1038.
[33] L. Yuan, S. Yeng Chai, G.-B. Huang, Random search enhancement of error minimized extreme learning machine, in: M. Verleysen (Ed.), European Symposium on Artificial Neural Networks (ESANN) 2010, d-side Publications, Bruges, Belgium, April 28–30th 2010, pp. 327–332.
[34] P. Zhao, G.V. Rocha, B. Yu, Grouped and hierarchical model selection through composite absolute penalties, Annals of Statistics 37 (6A) (2009) 3468–3497.
[35] H. Zou, T. Hastie, Regularization and variable selection via the elastic net, Journal of the Royal Statistical Society Series B 67 (2) (2005) 301–320.

Yoan Miche was born in 1983 in France. He received an Engineer's Degree from Institut National Polytechnique de Grenoble (INPG, France), and more specifically from TELECOM, INPG, on September 2006. He also graduated with a Master's Degree in Signal, Image and Telecom from ENSERG, INPG, at the same time. He recently received his Ph.D. degree in Computer Science and Signal and Image Processing from both the Aalto University School of Science and Technology (Finland) and the INPG (France). His main research interests are steganography/steganalysis and machine learning for classification/regression.



Mark van Heeswijk has been working as an exchange student in both the EIML (Environmental and Industrial Machine Learning, previously TSPCi) Group and Computational Cognitive Systems Group on his Master's Thesis on "Adaptive Ensemble Models of Extreme Learning Machines for Time Series Prediction", which he completed in August 2009. Since September 2009, he started as a Ph.D. student in the EIML Group, ICS Department, Aalto University School of Science and Technology. His main research interest is in the field of high-performance computing and machine learning. In particular, how techniques and hardware from high-performance computing can be applied to meet the challenges one has to deal with in machine learning. He is also interested in biologically inspired computing, i.e. what can be learned from biology for use in machine learning algorithms and in turn what can be learned from simulations about biology. Some of his other related interests include: self-organization, complexity, emergence, evolution, bioinformatic processes, and multi-agent systems.

Patrick Bas received the Electrical Engineering degree from the Institut National Polytechnique de Grenoble, France, in 1997 and the Ph.D. degree in Signal and Image processing from Institut National Polytechnique de Grenoble, France, in 2000. From 1997 to 2000, he was a member of the Laboratoire des Images et des Signaux de Grenoble (LIS), France where he worked on still image watermarking. During his post-doctoral activities, he was a Member of the Communications and Remote Sensing Laboratory of the Faculty of Engineering at the Université Catholique de Louvain, Belgium. His research interests include synchronisation and security evaluation in watermarking, and steganalysis.

Olli Simula received the Doctor of Science (Tech.) degree in computer science and engineering from Helsinki University of Technology (TKK), Finland, in 1979. Dr. Simula is Professor of Computer Science and Engineering at the Department of Information and Computer Science at Aalto School of Science and Technology (formerly TKK). He is also Dean of the Faculty of Information and Natural Sciences at Aalto. During the academic year 1977–78 Dr. Simula was a research fellow at Delft University of Technology, Delft, The Netherlands.

Amaury Lendasse was born in 1972 in Belgium. He received the M.S. degree in Mechanical Engineering from the Universite Catholique de Louvain (Belgium) in 1996, M.S. in control in 1997 and Ph.D. in 2003 from the same university. In 2003, he has been a post-doctoral researcher in the Computational Neurodynamics Lab at the University of Memphis. Since 2004, he is a chief research scientist and a docent in the Adaptive Informatics Research Centre in the Aalto University School of Science and Technology (previously Helsinki University of Technology) in Finland. He has created and is leading the Environmental and Industrial Machine Learning (previously Time Series Prediction and Chemoinformatics) Group. He is chairman of the annual ESTSP conference (European Symposium on Time Series Prediction) and member of the editorial board and program committee of several journals and conferences on machine learning. He is the author or the coauthor of around 140 scientific papers in international journals, books or communications to conferences with reviewing committee. His research includes time series prediction, chemometrics, variable selection, noise variance estimation, determination of missing values in temporal databases, nonlinear approximation in financial problems, functional neural networks and classification.