A methodology for Building Regression Models using Extreme Learning Machine: OP-ELM

Yoan Miche[1,2], Patrick Bas[1,2], Christian Jutten[2], Olli Simula[1] and Amaury Lendasse[1]

1- Helsinki University of Technology - Computer and Information Science Department
P.O. Box 5400 FI-02015 HUT - Finland

2- Institut National Polytechnique de Grenoble - Gipsa-lab
961 rue de la Houille Blanche F - 38402 Saint Martin d'Hères cedex - France

Abstract. This paper proposes a methodology named OP-ELM, based on a recent development –the Extreme Learning Machine– decreasing drastically the training speed of networks. Variable selection is beforehand performed on the original dataset for proper results by OP-ELM: the network is first created using Extreme Learning Process, selection of the most relevant nodes is performed using Least Angle Regression (LARS) ranking of the nodes and a Leave-One-Out estimation of the performances. Results are globally equivalent to LSSVM ones with reduced computational time.

1 Introduction

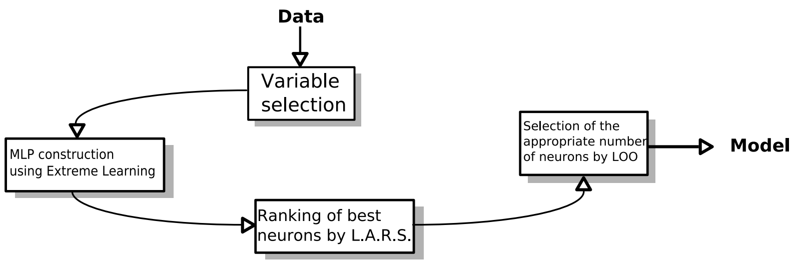
One reason why feed-forward neural networks tend not to be used widely in the industry for data mining problems lies most likely in the fact that they are very slow to train. This is due to the many parameters to be properly tuned by slow (often gradient-based) algorithms, in order to obtain a good enough model. Furthermore, the training phase has to be repeated in order to perform model structure selection, for example the selection of the number of hidden neurons or the selection of some regularization parameter. In [1], Guang-Bin Huang *et al.* propose an original algorithm for hidden nodes determination and weights selection called Extreme Learning Machine (ELM). The main advantage of this algorithm is in dividing the computational time by hundreds and making the learning process of the neural network rather simplistic. In this paper, a methodology based on ELM, called OP-ELM (for Optimal-Pruned ELM) with two main goals is proposed:

- being able to construct/select a nonlinear model in computational times close to these of linear models,
- this while keeping roughly the same performances as with the possibly best current algorithms.

For this purpose, we go through four main techniques, integrated in the OP-ELM methodology as four necessary steps, namely: variable selection [2, 3, 4], the mentioned Extreme Learning Machine [1], Least Angle Regression model selection [5] and finally a fast and exact estimation of the Leave-One-Out validation error in the training process, using PRESS statistics [6, 7]. The next section details these four steps, while section 3 presents some experiments highlighting the performances both in computational time and validation/test, as well as why variable selection is mandatory if one wants to use the Extreme Learning Machine.

2 Global methodology

Fig. 1 summarizes the four main steps of OP-ELM.



Data
Variable selection
MLP construction using Extreme Learning
Ranking of best neurons by L.A.R.S.
Selection of the appropriate number of neurons by LOO
Model

Fig. 1: The four steps of the proposed OP-ELM

2.1 Variable selection

An *a priori* variable selection has to be performed on the data set in order to remove the possibly irrelevant variables (not necessarily the redundant ones), for the problem. Experiments section 3 gives an example of the importance of this step for OP-ELM (and ELM) to perform best.

At this stage, variable selection can be achieved by any well-known technique. Since computational speed is the main advantage of OP-ELM, fast methods such as Forward selection (as in [2]) are among the most recommended ones; more elaborated techniques for selection using Markov blanket as in [3], typical mutual information as in [2] or a combination of mutual information with Forward selection and other sampling methods as proposed in [4] can also be used, at the possible penalty of a longer computational time for this step.

2.2 Extreme Learning Machine (ELM)

Once the dataset has been pruned of its irrelevant variables, the actual feedforward neural network is built, with only one hidden layer as proposed in the ELM algorithm. This algorithm has been presented by Guang-Bin Huang *et al.* in [1], although a common idea existed already in [8]. In EML, traditional multilayer perceptrons with one hidden layer is used. The weight between the input data and the hidden-layer are denoted $\mathbf{w}_i$. The weights between the hidden-layer and the output are denoted $\mathbf{b}$. The activation functions used are sigmoids in the hidden-layer and a linear function for the output layer. The novelty is in the determination of the input weights $\mathbf{w}_i$, which are randomly determined from a uniformly distributed distribution (for example between -10 and 10).

Indeed, with this done and following the mandatory hypothesis that the activation functions f of the hidden layer are indefinitely differentiable in any interval of their domain, the output weights $\mathbf{b}$ can be simply calculated from the hidden layer output matrix $\mathbf{H}$. Each column of $\mathbf{H}$ is given by the product of the weight vector and the input vectors: $\mathbf{h_i} = \mathbf{x_i}^T \mathbf{w_i}$. The output weights are calculated by $\mathbf{b} = \mathbf{H}^\dagger \mathbf{y}$, where $\mathbf{H}^\dagger$ stands for the Moore-Penrose inverse [9] and $\mathbf{y} = (y_1, \ldots, y_M)^T$ is the output. The choice of the number of neurons N to be used in the hidden layer remains the only arbitrary parameter; since the next step of the methodology is meant to prune the unuseful neurons of the hidden layer, it is wise to have sufficient number of neurons for the ELM part.

2.3 Least Angle Regression (LARS)

The LARS algorithm was proposed by Efron *et al.* in [5] for variable selection for regression. The LARS algorithm provides the ranking of a set of possible input variables. The solution that is produced is exact if the problem is linear. In the case of our neural network built in the previous stage, the hidden layer neurons are ranked by the LARS algorithm. Since the part between the hidden and the output layer of the neural network is linear, LARS is guaranteed to find the best ranking. Finally, the selection of the final model structure is achieved through Leave-One-Out validation in the last step of OP-ELM that selects the number of neurons. Hence, only the most important neurons are used and their number is optimized.

2.4 Leave-One-Out (LOO)

For the estimation of the validation error and the actual selection of the best neurons for the problem, a Leave-One-Out is used. Calculating the LOO error ϵ can be very time consuming when data sets tend to have an important number of samples. Fortunately, the PRESS (or PREdiction Sum of Squares) statistics provide a direct and exact formula for the calculation of this error for linear models (see [6, 7] for details on this formula and implementations):

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

where $\mathbf{P}$ is defined as $\mathbf{P} = (\mathbf{H}^T \mathbf{H})^{-1}$ and $\mathbf{H}$ the hidden layer output matrix defined previously. Finally, evaluating the LOO error versus the number of neurons used (which have been previously properly ranked by the LARS algorithm) enables to select the best number of ranked neurons.

3 Experiments

This section presents three different cases where OP-ELM is applied: a typical one dimension sine example, the same sine with an additional irrelevant variable in the data set and the UCI machine learning repository Abalone dataset.

Table 1 sums up the results in Normalized Mean Square Error form for all experiments below, as well as the calculation times.

3.1 Sine in one dimension (sine1)

A set of 1000 training points are generated, following a sum of two sines. This gives a one-dimensional example where no feature selection has to be performed. Fig. 3 plots the obtained model on top of the training data.

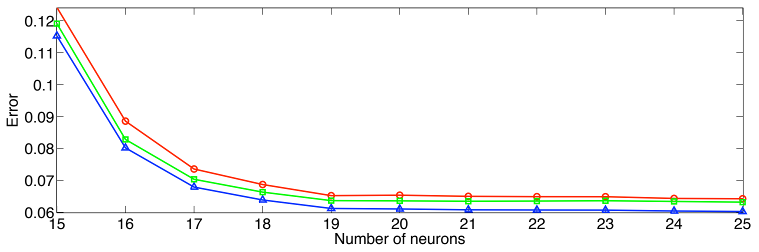
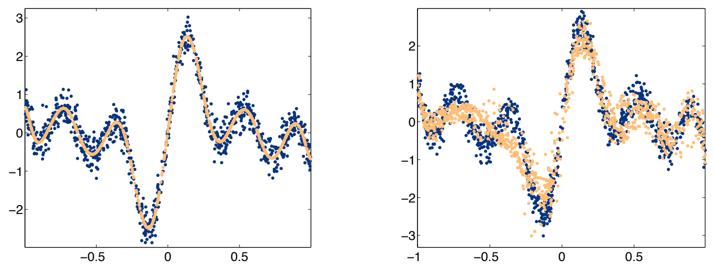


Fig. 2: Mean Square error (△), Leave-One-Out error (□) and Test error (○).

The model seems to approximate the data very nicely, and using a number of neurons around 20, one reaches an error already equal to the noise introduced in the dataset (0.0625) as can be seen on Fig. 2.



(a) Training, sine1 (b) Training, sine2

Fig. 3: (a) Plot of a one dimensional sum of sines (1000 blue dots) and the model obtained by OP-ELM (red dots). (b) Plot of the model fit to the data for the sine with irrelevant variable case.

3.2 Sine with irrelevant additional variable (sine2)

The sine dataset previously generated is added an irrelevant variable (gaussian noise), and feature selection step is skipped; this to show the importance of the first step of OP-ELM, if it wants to succeed. Fig. 3 shows the obtained model on the sine-only data. It is clear, by simple comparison with the previous experiment, that the artificially added irrelevant variable has highly perturbated the model. This enhances the fact that OP-ELM (as well as ELM) is very sensitive to variable selection and performs well only if it is applied beforehand.

3.3 Abalone dataset from UCI machine learning repository

The Abalone dataset from UCI [10] contains 4177 samples, 8 variables (one discrete). The dataset is divided into a training set of 2000 samples and a testing set of 2177 samples. Table 1 gives results for OP-ELM, also when variable selection (VS) is not performed beforehand [2, 3, 4], and with original ELM algorithm, for comparison. It appears that in "proper" cases (Sine and Abalone (VS)) –when

	OP-ELM			LS-SVM		
	LOO	Test	Time	LOO	Test	Time
Sine	0.0636	0.0651	**2.4s**	0.0628	0.0643	**40s**
Sine2 (no VS)	0.5645	0.5467	**5.9s**	0.0886	0.0774	**45s**
Abalone (no VS)	0.5045	0.4313	**23s**	0.4711	0.3981	**40min**
Abalone (VS)	0.4716	0.4206	**21s**	0.4953	0.4041	**13min**
Abalone (ELM)	0.3606	0.7377	**15s**	X	X	X

Table 1: Normalized Mean Square Error (NMSE) for the three different examples (also for Abalone dataset without variable selection (VS)) in the case of OP-ELM and for LS-SVM.

the whole methodology with variable selection step is performed–, results of OP-ELM are at most 8% behind LS-SVM [11] results (test case of Abalone (VS)), and most of the time 2% around. This while having computational times 15 to 100 times faster. It should also be noted, that without the variable selection step, the proposed OP-ELM can fail spectacularly: Abalone without variable selection case remains less than 10% behind LS-SVM results, but the Sine2 case gives 6.5 times worse results than LS-SVM, in terms of NMSE. Finally, it should be noted that on the example of the Abalone dataset (with variable selection for both OP-ELM and ELM only), OP-ELM performs much better even though the Leave-One-Out error is higher.

4 Conclusions and perspectives

The proposed methodology (OP-ELM) based on the Extreme Learning Machine performs better than the original version of ELM, mostly thanks to the variable selection and the selection of the number of neurons performed by LARS algorithm and PRESS statistics. This selection not only ensures that the whole OP-ELM will work properly but also brings interpretability over the variables selected out of the full set. Moreover, results most of the time comparable to the ones obtained by a LS-SVM are achieved, and this in a computational time divided by 15 to 200. With such small current calculation times, future research will investigate other extensions of OP-ELM for performance enhancement. Improving the *a priori* variable selection, with possible scalings of the variables could also be a way to increase performance. Finally, extension of OP-ELM to classification problems is currently considered for high dimensional problems.

Acknowledgments

This work was supported (in part) by the European Commission under grant IST-2002-507932 ECRYPT and French national funding under grants ANR-06-SETI-009 Nebbiano, RIAM Estivale and ARA TSAR.

References

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

[2] Rossi F., Lendasse A., François D., Wertz V., and Verleysen M. Mutual information for the selection of relevant variables in spectrometric nonlinear modelling. *Chemometrics and Intelligent Laboratory Systems*, 80:215–226, 2006.

[3] Herrera L. J., Pomares H., Rojas I., Verleysen M., and Guilén A. Effective input variable selection for function approximation. In Springer Berlin / Heidelberg, editor, *Artificial Neural Networks: ICANN 2006*, volume 4131/2006 of *Lecture Notes in Computer Science*, pages 41–50, 2006.

[4] D. François, F. Rossi, V. Wertz, and M. Verleysen. Resampling methods for parameter-free and robust feature selection with mutual information. *Neurocomput.*, 70(7-9):1276–1288, 2007.

[5] Efron B., Hastie T., Johnstone I., and Tibshirani R. Least angle regression. In *Annals of Statistics*, volume 32, pages 407–499. 2004.

[6] Myers R. H. *Classical and Modern Regression with Applications, 2nd edition.* Duxbury, Pacific Grove, CA, USA, 1990.

[7] Bontempi G., Birattari M., and Bersini H. Recursive lazy learning for modeling and control. In *European Conference on Machine Learning*, pages 292–303, 1998.

[8] Miller W. T., Glanz F. H., and Kraft L. G. Cmac: An associative neural network alternative to backpropagation. In *Proceedings of the IEEE*, volume 70, pages 1561–1567. October 1990.

[9] Rao C. R. and Mitra S. K. *Generalized Inverse of Matrices and Its Applications.* John Wiley & Sons Inc, January 1972.

[10] Asuncion A. and Newman D. J. (uci) machine learning repository, 2007.

[11] Suykens J.A.K., Van Gestel T., De Brabanter J., B. De Moor B., and Vandewalle J. *Least Squares Support Vector Machines*. World Scientific, 2002.