A variable selection approach based on the Delta Test for Extreme Learning Machine models

Fernando Mateo[1] and Amaury Lendasse[2]

1- Universidad Politécnica de Valencia - Dept. Ingeniería Electrónica
Camino de Vera s/n, 46022 Valencia - Spain

2- Helsinki University of Technology - Adaptive Informatics Research Centre
Konemiehentie 2, 02150 Espoo - Finland

Abstract. Extreme Learning Machine, ELM, is a newly available learning algorithm for single layer feedforward neural networks (SLFNs), and it has proved to show the best compromise between learning speed and accuracy of the estimations. In this paper, a methodology based on Optimal-Pruned ELM (OP-ELM) for function approximation enhanced with variable selection using the Delta Test is introduced. The least angle regression (LARS) algorithm is used after variable selection to rank the input variables, and scaling is also introduced as a way to estimate the influence of each input in the output value. The performance is assessed on a dataset related to anthropometric measurements for children weight prediction. The accurate results show that this combination of techniques is very promising to solve real world problems and represents a good alternative to classic backpropagation methods.

1 Introduction

In many real-life problems it is convenient to reduce the number of involved features (variables) in order to reduce the complexity, especially when the number of features is large compared to the number of observations. There are several criteria to tackle this variable reduction problem. Three of the most common are: maximization of the mutual information (MI) between the inputs and the outputs, minimization of the k-nearest neighbors (k-NN) leave-one-out generalization error estimate and minimization of a nonparametric noise estimator (NNE).

Extreme Learning Machine (ELM)[1] is a new learning technique to train single layer feedforward neural networks (SLFN) which chooses the input weights randomly and determines the output weights analytically. This algorithm is designed to build models that provide the best possible generalization in the shortest time. Given its success, it has already been applied to several fields of machine learning such as text classification [2] or time series prediction [3].

This work intends to make use of the methodology described in [4] which proposes a combination of Extreme Learning Machine with optimal pruning (OP-ELM) and variable selection. In this case, we focus on the use of a NNE as a selection criterion, concretely by using the Delta Test (DT) as estimator. The applicability of the method is assessed on a dataset of children anthropometric measurements.

This paper is structured as follows: Section 2 explains the variable selection methodology using the Delta Test as a criterion, and how it is integrated in the forward-backward search (FBS) algorithm. In Section 3 there is a description of the LARS methodology for input ranking and Section 4 gives a mathematical perspective on the Extreme Learning Machine method. In Section 5, the experiments are described and some relevant results are presented, while Section 6 summarizes the conclusions of this work.

2 Variable selection

2.1 The Delta Test

The Delta Test, firstly introduced by Pi and Peterson for time series [5], is a technique to estimate the variance of the noise, or the mean squared error (MSE), that can be achieved without overfitting. Given N input-output pairs $(x_i, y_i) \in \mathbb{R}^M \times \mathbb{R}$, the relationship between x_i and y_i can be expressed as

$$y_i = f(x_i) + r_i, \qquad i = 1, ..., N$$

where f is the unknown function and r is the noise. The DT estimates the variance of the noise r.

The DT is useful for evaluating the nonlinear correlation between two random variables, namely, input and output pairs. The DT can be also applied to input selection: the set of inputs that minimizes the DT is the one that is selected. Indeed, according to the DT, the selected set of inputs is the one that represents the relationship between inputs and output in the most deterministic way. DT is based on hypotheses coming from the continuity of the regression function. If two points x and x' are close in the input space, the continuity of regression function implies the outputs $f(x)$ and $f(x')$ will be close enough in the output space. Alternatively, if the corresponding output values are not close in the output space, this is due to the influence of the noise.

The DT can be interpreted as a particularization of the Gamma Test [6] considering only the first nearest neighbor. Let us denote the first nearest neighbor of a point x_i in the $\mathbb{R}^M$ space as $x_{NN(i)}$. The nearest neighbor formulation of the DT estimates $\mathrm{Var}[r]$ by

$$\mathrm{Var}[r] \approx \delta = \frac{1}{2N} \sum_{i=1}^{N} (y_i - y_{NN(i)})^2, \text{ with } \mathrm{Var}[\delta] \to 0 \text{ for } N \to \infty$$

where $y_{NN(i)}$ is the output of $x_{NN(i)}$.

2.2 Forward-backward search methodology

To overcome the difficulties and the high computational time that an exhaustive search would entail (i.e. $2^N - 1$ input combinations, being N the number of variables), there are several other search strategies. These strategies are suboptimal because they do not test every input combination, and they are clearly affected

by local minima, but they are preferred to an exhaustive search if the number of variables is large.

Among the typical search strategies, there are three that share similarities:

- Forward search
- Backward search (or pruning)
- Forward-backward search (or forward stagewise regression)

The difference between the first two is that the forward search starts from an empty set of selected variables and adds variables to it according to the optimization of a search criterion, while the backward search starts from a set containing all the variables and removes those for which the elimination optimizes the search criterion.

Both forward and backward search suffer from incomplete search. The forward-backward search (FBS) is a combination of them. It is more flexible in the sense that a variable is able to return to the selected set once it has left it, and vice versa, a previously selected variable can be discarded later. This method can start from any initial input set: empty set, full set, custom set or randomly initialized set. If we consider a set of N input-output pairs $(x_i, y_i) \in \mathbb{R}^M \times \mathbb{R}$, the forward-backward search algorithm can be described as follows

1. Initialization:

   Let S be the selected input set, which can contain any input variables, and F the unselected input set, which contains the variables not present in S. Compute Var[r] using Delta Test (see section 2.1) on the set S.

2. Forward-Backward selection step:

   Find

   $$x^S = \arg\, min_{x_i,x_j}\{(\text{Var}[r]|S \cup x^j) \cup (\text{Var}[r]|S \setminus x^i)\}, \;\; x^i \in S, x^j \in F$$

3. If the old value of Var[r] on the set S is lower than the new result, stop; otherwise, update set S and save the new Var[r]. Repeat step 2 until S is equal to any former selected S.

4. The selected input set is S

3 The LARS algorithm

Least angle regression (LARS)[7] is a stylized version of the stagewise procedure that uses a simple mathematical formula to accelerate the computations. Only m steps are required for the full set of solutions, where m is the number of covariates.

The LARS procedure works roughly as follows. Supposing that the initial estimate is μ_0, the algorithm takes a first step in the direction of the most correlated predictor, say x_1. When another predictor, say x_2, becomes sufficiently correlated to become one of the chosen variables, the next step is taken in the bisecting angle between x_1 and x_2. This happens again when a third predictor, x_3, gains the sufficient importance to contribute to the model. The process continues until all the covariates have entered the model. This method is illustrated in Fig. 1

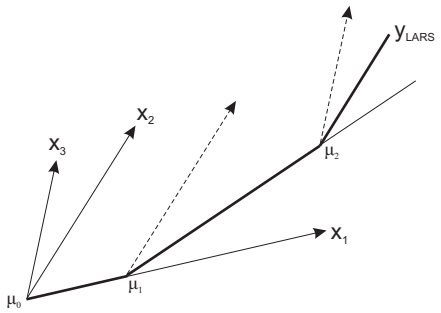


Fig. 1: LARS algorithm evolution for m = 3 covariates.

The entire sequence of steps in the LARS algorithm with $m < n$ variables, where n is the number of observations, requires $\mathrm{O}(m^3 + nm^2)$ computations (the cost of a least squares fit on m variables). The LARS algorithm works gracefully for the case where there are many more variables than observations ($m >> n$).

It is important to take into account that the procedure requires that the variables and outputs are previously scaled to have zero mean and variance equal to 1.

4 Extreme Learning Machine

Let us consider a set of N points $(\boldsymbol{x}_i, \boldsymbol{t}_i) \in \mathbb{R}^n \times \mathbb{R}^m$, where $i = 1, ..., N$. A standard single layer feedforward neural network (SLFN) with L hidden neurons and activation function $g(x)$ can be mathematically modeled as

$$\sum_{i=1}^{L} \beta_i g(\boldsymbol{w}_i \boldsymbol{x}_j + b_i) = \boldsymbol{d}_j \qquad j = 1, ..., N$$

where $\boldsymbol{w}_i$ is the weight vector connecting inputs and the ith hidden neuron, β_i is the weight vector connecting the ith hidden neuron and output neurons, b_i is the threshold of the ith hidden neuron, and $\boldsymbol{d}_j$ is the output given by the ELM for data point j. The standard SLFN with n hidden neurons and activation function $g(x)$ can approximate these N samples with zero error in the ideal case, meaning that $\sum_{j=1}^{L} \|\boldsymbol{d}_j - \boldsymbol{t}_j\| = 0$, and thus, there exist β_i, $\boldsymbol{w}_i$ and b_i such that

$$\sum_{i=1}^{L} \beta_i g(\boldsymbol{w}_i \boldsymbol{x}_j + b_i) = \boldsymbol{t}_j \qquad j = 1, ..., N$$

The above equations can be written compactly as:

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

where

$$\mathbf{H} = \begin{pmatrix} g(\boldsymbol{w}_1 \boldsymbol{x}_1 + b_1) & \dots & g(\boldsymbol{w}_L \boldsymbol{x}_1 + b_L) \\ \vdots & \dots & \vdots \\ g(\boldsymbol{w}_1 \boldsymbol{x}_N + b_1) & \dots & g(\boldsymbol{w}_L \boldsymbol{x}_N + b_L) \end{pmatrix}_{N \times L}$$

$$\beta = \begin{pmatrix} \beta_1^T \\ \vdots \\ \beta_L^T \end{pmatrix}_{L \times m} \quad \text{and} \quad \mathbf{T} = \begin{pmatrix} \boldsymbol{t}_1^T \\ \vdots \\ \boldsymbol{t}_N^T \end{pmatrix}_{N \times m}$$

The matrix $\mathbf{H}$ is called the hidden layer output matrix of the neural network. When the number of neurons in the hidden layer is equal to the number of samples, $\mathbf{H}$ is square and invertible. Otherwise, the system of equations needs to be solved by numerical methods, concretely by solving

$$\min_\beta \|\mathbf{H}\beta - \mathbf{T}\|$$

The solution that minimizes the norm of this least squares equation is

$$\hat{\beta} = \mathbf{H}^\dagger \mathbf{T}$$

where $\mathbf{H}^\dagger$ is called Moore-Penrose generalized inverse [1]. The most important properties of this solution are:

- Minimum training error.
- Smallest norm of weights and best generalization performance.
- The minimum norm least-square solution of $\mathbf{H}\beta = \mathbf{T}$ is unique, and is $\hat{\beta} = \mathbf{H}^\dagger \mathbf{T}$.

Hence, the ELM algorithm for SLFNs can be summarized in these steps:

Given a training set $(\boldsymbol{x}_i, \boldsymbol{t}_i) \in \mathbb{R}^n \times \mathbb{R}^m$, $i = 1, ..., N$ activation function $g(x)$ and L hidden neurons:

1. Assign arbitrary input weights w_i and bias b_i, $i = 1, ..., L$.
2. Calculate the hidden layer output matrix $\mathbf{H}$.
3. Calculate the output weights β:

$$\beta = \mathbf{H}^\dagger \mathbf{T}$$

where $\mathbf{H}$, β and $\mathbf{T}$ are as defined before.

5 Experiments and results

5.1 The "anthrokids" dataset

The dataset used for testing the described methodology was a collection of anthropometric data that represents the results of a three-year study on 3900 infants and children representative of the U.S. population of year 1977, ranging in age from newborn to 12 years of age. The dataset comprises 121 variables and the target variable to predict is children's weight. The data repository can be found in http://ovrt.nist.gov/projects/anthrokids/.

This dataset is characterized by the presence of many missing values. Therefore, a first sample and variable discrimination had to be done to build a robust and reliable dataset. The approach to do this was to keep a minimum amount of 1000 samples out of the possible 3900. The number of variables was chosen by means of an iterative routine which attacked the data set reduction both in terms of number of samples and number of variables. The variables were removed one by one (every time the one with the highest amount of missing values) while the number of samples removed per iteration could be tuned. The best compromise, with 43 samples removed per iteration, was a set of 54 variables which led to a set of 1019 samples, free of missing values. Figures 2(a) and 2(b) describe this process. One extra variable was removed because it had a constant value for all samples, yielding a final set of 53 variables.

The resulting dataset was subdivided into training and test sets, with 70% and 30% of the samples respectively. The variables were normalized to have zero mean and standard deviation 1 before being processed.

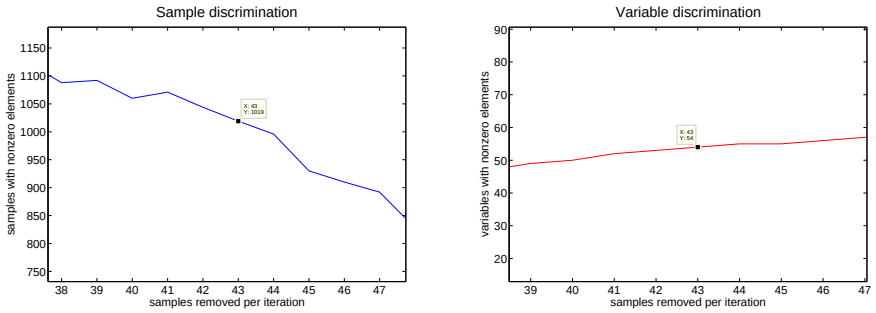
5.2 Forward-backward search

Forward-backward search with Delta Test as the performance criterion and empty initial search set was applied to the training set. The FBS algorithm was applied to several training set combinations to find the best (lowest) DT value. The method selected the 12 variables listed in Table 1 and the value of Var[r] versus the number of variables is shown in Fig. 3. The lowest DT value achieved was 0.0070 and the algorithm converged in 281.63 seconds.

5.3 OP-ELM model construction

After this initial selection, an OP-ELM model was built using these 12 variables and the same percentages of training and test samples. The algorithm was initialized with a high number of kernels (100) so that the pruning did not affect the accuracy of the result.

The criterion to optimize during training was the estimation of the leave-one-out (LOO) validation error. Usually, LOO estimation is too time consuming, especially when the number of samples is large. For that reason, the estimation was done using PRESS (PREdiction Sum of Squares) statistics, which gives an exact formula for LOO calculation when the problem is linear. This exact formula defines the LOO error ϵ as



(a) (b)

Fig. 2: Composition of the dataset. The first step was the determination of the number of samples to remove per iteration to achieve the minimum number of samples (>1000) with nonzero elements (a) and the second was to find the number of variables with nonzero elements determined by this amount of samples removed between iterations (b).

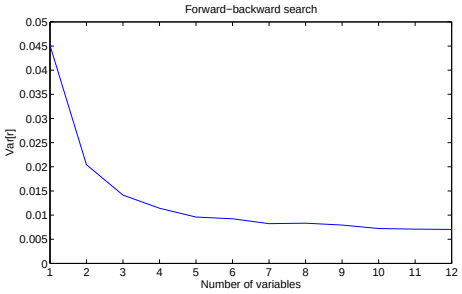


Fig. 3: Forward-backward search algorithm evolution. The algorithm reaches the minimum value of Var[r] for 12 variables.

$$\epsilon = \frac{\boldsymbol{t}_i - \mathbf{h}_i}{1 - \mathbf{h}_i \mathbf{P} \mathbf{h}_i^T}$$

where $\mathbf{P}$ is defined as $\mathbf{P} = (\mathbf{H}^T\mathbf{H})^{-1}$, $\mathbf{H}$ is the hidden layer output matrix, $\mathbf{h}_i$ are the column vectors of matrix $\mathbf{H}$, $\boldsymbol{t}_i$ are the target values and β are the output weights (see section 4 for details). Table 2 lists the results of several OP-ELM models, using different activation functions for the hidden layer. A linear component is always maintained because it generally helps fitting the data if there is any linearity between the inputs and the outputs of the model.

The results show that variable selection allows building a model with the same performance as with the full set of variables (or even improving it when linear and sigmoid activation functions are used) in a much shorter time. The reduction in the number of required neurons is also noticeable.

Variable number	Variable name	LARS ranking
1	Stature	12
8	Erect sitting height	7
13	Buttock-knee length	10
34	Shoulder breadth	4
38	Upper arm circumference	5
40	Elbow-hand length	8
62	Chest circumference	1
65	Waist circumference	6
70	Hip circumference	3
76	Upper thigh circumference	11
80	Calf circumference	2
85	Foot length	9

Table 1: Variables selected by FBS (DT = 0.0070) and ranking given by LARS algorithm. The variable numbers correspond to their position in the original dataset

Number of variables	Activation function	Comput. time (s)	LOO error	Test error	Number of neurons
53 (full set)	L + S	7.79	0.0221	0.0354	88
	L + G	8.12	0.0060	0.0167	118
	L + S + G	7.98	0.0069	0.0169	143
12 (FBS)	L + S	3.15	0.0110	0.0214	47
	L + G	2.71	0.0060	0.0170	17
	L + S + G	3.24	0.0062	0.0173	22

Table 2: Performance achieved by several OP-ELM models using different combinations of activation functions. L: linear, S: sigmoid, G: gaussian.

5.4 LARS ranking

The LARS algorithm is guaranteed to find the best ranking among all possible inputs if a problem is linear. The OP-ELM model makes use of LARS to rank the neurons of the hidden layer, since this layer can be considered an input layer to the last (linear) stage of the neural network. We also used the LARS algorithm to provide the best ranking of the variables selected. Despite the non-linearities of the problem, LARS managed to find a coherent order for the set of selected variables. The resulting ranking of variables appears in Table 1.

5.5 Scaling

The next step to take in order to optimize the variable selection is to scale the selected variables according to their influence on the output value. Let us consider f as the unknown function that determines the relationship between

the inputs and outputs of a regression problem, $y = f(x) + r$, with $x \in \mathbb{R}^M, y \in \mathbb{R}, r \in \mathbb{R}$. Let d be the number of selected variables that minimize the Delta Test without any scaling. Thus, the estimate of the output, $\hat{y} \in \mathbb{R}$, can be expressed as $\hat{y} = g(x') + r$, with $x' \in \mathbb{R}^d$ and g is the model that best approximates the function f. The objective is to find a scaling vector $\alpha \in \mathbb{R}^d$ such that

$$\hat{y} = g(\alpha_1 x'_1, \alpha_2 x'_2, \ldots, \alpha_d x'_d) + r$$

minimizes the variance of the noise (DT) of the problem.

Intuitively, the scaling will assign high values of α to the variables that are most correlated with the output, and low values to those less correlated. It can happen that some variable that initially was not selected, now enters the set of selected variables with a low scaling factor.

The method of FBS with scaling was applied to the anthropometrics example, providing the ranking shown in Table 3. The result includes all the previously selected variables with scaling factor 1, and adds 6 more with lower scaling factors ranging from 0.1 to 0.5. The computational time employed was 1495.95 seconds and the DT with scaling was reduced to 0.0064. Finally, Table 4 shows a comparison of several OP-ELM models built using the scaled variables.

Scaling factor	Variable number	Variable name
1.0	1	Stature
1.0	8	Erect sitting height
1.0	13	Buttock-knee length
1.0	34	Shoulder breadth
1.0	38	Upper arm circumference
1.0	40	Elbow-hand length
1.0	62	Chest circumference
1.0	65	Waist circumference
1.0	70	Hip circumference
1.0	76	Upper thigh circumference
1.0	80	Calf circumference
1.0	85	Foot length
0.5	42	Forearm circumference
0.4	36	Shoulder-elbow length
0.2	22	Lower face height
0.1	47	Hand breadth
0.1	57	Maximum fist circumference
0.1	112	Birth order

Table 3: Variables selected by FBS with scaling factors (DT = 0.0064).

6 Conclusion

This work has presented the use of variable selection using the Delta Test as a selection criterion, based on the minimization of the variance of the noise

Number of variables	Activation function	Comput. time (s)	LOO error	Test error	Number of neurons
18 (scaled)	L + S	3.61	0.0106	0.0210	68
	L + G	3.71	0.0058	0.0168	17
	L + S + G	3.72	0.0058	0.0166	17

Table 4: Study of the performance of several OP-ELM models using scaled variables. The different activation functions are: L: linear, S: sigmoid, G: gaussian.

in a regression problem. This selection of variables has been combined with optimally pruned ELM models that effectively accelerate the learning process of single layer feedforward neural network.

The OP-ELM models by themselves produced short training times (of the order of 8 seconds for a problem involving 53 variables and around 1000 samples) and relatively accurate estimations in terms of validation and test error. In the example studied, the combination with variable selection using DT reduces the computational time to less than half of the achieved with OP-ELM, maintaining, or improving in some cases, the calculated errors. It also proved to reduce substantially the necessary number of nodes in the network. The best performing models for this application were those which included gaussian kernels, either with or without sigmoid components.

The use of scaling factors to weight the variables according to their importance in the model slightly increases the accuracy but on the other hand it increases the computational time too. Therefore, the convenience of using scaling or not will depend on each specific application.

In particular, we consider that this methodology could be effectively used for time series prediction as it is done in [3] but automatizing the choice of hidden neurons and saving computational time by reducing the number of variables.

References

[1] G.-B. Huang, Q.-Y. Zhu and C.-K. Siew, Extreme learning machine: A new learning scheme of feedforward networks, *Neurocomputing*, 70:489–501, 2006.

[2] Y. Liu, H.-T. Loh and S.-B. Tor, Comparison of extreme learning machine with support vector machine for text classification, LNAI 3533:390–399, Springer-Verlag Berlin Heidelberg, 2005.

[3] R. Singh and S. Balasundaram, Application of extreme learning machine method for time series analysis, *Int. Jour. Int. Tech.*, 2(4):256–262, 2007.

[4] Y. Miche, P. Bas, Ch. Jutten, O. Simula and A. Lendasse, A methodology for building regression models using extreme learning machine: OP-ELM, In *Proceedings of ESANN 2008, European Symposium on Artificial Neural Networks*, pp. 247–252, 2008.

[5] H. Pi and C. Peterson, Finding the embedding dimension and variable dependencies in time series, *Neural Computation*, 6(3):509–520, 1994.

[6] A.J. Jones, New tools in non-linear modelling and prediction, *Comput. Manage. Sci.*, 1:109–149, 2004.

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