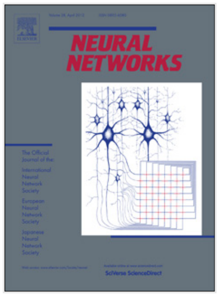


Neural architecture design based on extreme learning machine

Andrés Bueno-Crespo, Pedro J. García-Laencina, José-Luis Sancho-Gómez

PII: S0893-6080(13)00181-0
DOI: http://dx.doi.org/10.1016/j.neunet.2013.06.010
Reference: NN 3213

To appear in: *Neural Networks*

Received date: 2 October 2012
Revised date: 20 May 2013
Accepted date: 27 June 2013

Neural architecture design based on Extreme Learning Machine

Andrés Bueno-Crespo[a], Pedro J. García-Laencina[b], and José-Luis Sancho-Gómez[c]

[a] *Dpto. Informática de Sistemas, Universidad Católica San Antonio, Murcia, SPAIN.*
[b] *University Centre of Defence at the Spanish Air Force Academy, MDE-UPCT, Santiago de la Ribera (Murcia), SPAIN.*
[c] *Dpto. Tecnologías de la Información y las Comunicaciones, Universidad Politécnica de Cartagena, Cartagena(Murcia), SPAIN.*

Abstract

Selection of the optimal neural architecture to solve a pattern classification problem entails to choose the relevant input units, the number of hidden neurons and its corresponding interconnection weights. This problem has been widely studied in many research works but their solutions usually involve excessive computational cost in most of the problems and they do not provide a unique solution. This paper proposes a new technique to efficiently design the MultiLayer Perceptron (MLP) architecture for classification using the Extreme Learning Machine (ELM) algorithm. The proposed method provides a high generalization capability and a unique solution for the architecture design. Moreover, the selected final network only retains those input connections that are relevant for the classification task. Experimental results show these advantages.

Keywords: Neural Networks, Architecture Design, Extreme Learning Machine, MultiLayer Perceptron

1. Introduction

In real life, when we learn a task, we have to process a huge amount of information from multiple variables. However, human learning is able to discriminate in a quite efficient way what it is really important to learn for such task and discard irrelevant information. Similarly, when a learning machine is trained to solve a new task, it needs to select relevant inputs or remove unnecessary connections [1, 2, 3, 4, 5, 6, 7]. There are feature selection methods used before any prior learning, for example, filter methods based in information theory [4]. Other methods define the relevance of the inputs during learning, for example, by measuring the sensitivity of the output with respect to inputs until there are no more irrelevant input features to identify [5]. These types of methods are interesting in complex problems with many inputs and hidden neurons, since

Email address: abueno@ucam.edu (Andrés Bueno-Crespo)

they may provide a simple architecture. Besides the effort to reduce the data dimensionality, a high computational cost in the selection of the architecture is also required.

This paper presents a new method to efficiently design the architecture of a MultiLayer Perceptron (MLP) in order to solve a classification recognition problem. Proposed method exploits the advantages of the Extreme Learning Machine (ELM) algorithm and, in particular, the Optimal Pruned ELM (OP-ELM). Our method achieves a unique solution for the MLP architecture and automatically selects the relevant features and connections for each hidden neuron. Hidden layer size is also automatically obtained. Results obtained in different classification problems show the advantages of the proposed method. In particular, it gets a faster convergence without being affected by irrelevant or noisy input features, it reduces the computational cost by removing neurons or unnecessary connections, and it also provides a high classification performance.

The rest of the paper is organized as follows. In Section 2, an introduction of the ELM algorithm is presented. Section 3 presents the proposed method. Section 4 shows the experimental results on several data sets, and finally, conclusion and future work are given in Section 5.

2. Extreme Learning Machine

The design and training of MLP architectures based on traditional techniques has several drawbacks, such as the high computational time required and the convergence to suboptimal solutions [8, 9]. Recently, the Extreme Learning Machine (ELM) algorithm has mitigated these drawbacks [10, 11]. Essentially, ELM provides a fast and accurate training algorithm which applies random computational nodes in the hidden layer of the feedforward neural network and only the output weights adjusted during training (i.e., the hidden layer in ELM need not be tuned). The basic idea, in which ELM is based on, was previously analyzed in other works [12, 13]. Nevertheless, ELM is an unified framework for generalized single hidden layer feedforward neural networks, it has the universal approximation capability for a wide range of random computational nodes and, in this algorithm, all the hidden node parameters can randomly be generated according to any continuous probability distribution without any prior knowledge [14]. In [15], a deep study about the differences between the ELM and other previous works based on randomly fixed hidden neurons can be found. The standard ELM algorithm is based on an MLP consisting of M hidden neurons, whose input weights are randomly initialized being able to learn N different n-dimensional input vectors producing zero error [10]. As the MLP input weights are fixed to random values, the MLP can be considered as a linear system and the output weights can be easily obtained using the pseudo-inverse hidden neurons outputs matrix $\mathbf{H}$ for a given training set. Thus, given a set of N input vectors, an MLP can approximate N cases with zero error, $\sum_{i=1}^{N} \|\mathbf{y}_i - \mathbf{t}_i\| = 0$, being $\mathbf{y}_i$ the output network for the input vector $\mathbf{x}_i$ with target vector $\mathbf{t}_i$. Thus, there exist β_j, $\mathbf{w}_j$ and b_j such that,

$$\mathbf{y}_i = \sum_{j=1}^{M} \beta_j f(\mathbf{w}_j \cdot \mathbf{x}_i + b_j) = \mathbf{t}_i, \quad i = 1, ..., N. \tag{1}$$

where $\beta_j = [\beta_{j1}, \beta_{j2}, ..., \beta_{jm}]^T$ is the weight vector connecting the jth hidden node and the output nodes, $\mathbf{w}_j = [w_{j1}, w_{j2}, ..., w_{jn}]^T$ is the weight vector connecting the jth hidden node and the input nodes, and b_j is the bias of the jth hidden node.
The previous N equations can be expressed by:

$$\mathbf{HB} = \mathbf{T}, \tag{2}$$

where

$$\mathbf{H}(\mathbf{w}_1, \dots, \mathbf{w}_M, b_1, \dots, b_M, \mathbf{x}_1, \dots, \mathbf{x}_N) =$$

$$= \begin{bmatrix} f(\mathbf{w}_1 \cdot \mathbf{x}_1 + b_1) & \dots & f(\mathbf{w}_M \cdot \mathbf{x}_1 + b_M) \\ \vdots & \dots & \vdots \\ f(\mathbf{w}_1 \cdot \mathbf{x}_N + b_1) & \dots & f(\mathbf{w}_M \cdot \mathbf{x}_N + b_M) \end{bmatrix}_{N \times M} \tag{3}$$

$$\mathbf{B} = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_M^T \end{bmatrix}_{M \times m} \quad \text{and} \quad \mathbf{T} = \begin{bmatrix} \mathbf{t}_1^T \\ \vdots \\ \mathbf{t}_N^T \end{bmatrix}_{N \times m} \tag{4}$$

where $\mathbf{H} \in \Re^{N \times M}$ is the matrix of hidden neurons output layer of the MLP, $\mathbf{B} \in \Re^{M \times m}$ is the output weight matrix, and $\mathbf{T} \in \Re^{N \times m}$ is the target matrix of the N training cases. Thus, as $\mathbf{w}_j$ and b_j with $j = 1, ..., N$, are randomly selected, the MLP training is given by the solution of the least square problem of (2), i.e., the optimal output weight layer is $\hat{\mathbf{B}} = \mathbf{H}^\dagger \mathbf{T}$, where $\mathbf{H}^\dagger$ is the Moore-Penrose pseudo-inverse [16].
Finally, ELM for training MLPs can be summarized as follows:

Algorithm 1 Extreme Learning Machine (ELM).

Require: Given a training set $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{t}_i) | \mathbf{x}_i \in \mathbb{R}^n, \mathbf{t}_i \in \mathbb{R}^m, i = 1, \dots, N\}$, an activation function f and an hidden neuron number M,
1: Assign arbitrary input weights $\mathbf{w}_j$ and biases b_j, $j = 1, \dots, M$.
2: Compute the hidden layer output matrix $\mathbf{H}$ using (3).
3: Calculate the output weight matrix $\mathbf{B} = \mathbf{H}^\dagger \mathbf{T}$, where $\mathbf{B}$ and $\mathbf{T}$ are both defined in (4).

ELM provides a fast and efficient MLP training [10], but it needs to fix the number of hidden neurons. In order to avoid the exhaustive search for the optimal value of M, the Optimal Pruned ELM method (OP-ELM) [17] sets a very high initial number of hidden neurons ($M \gg N$) and, by using Least Angle Regression algorithm (LARS) [18], sorts the neurons according to their importance to solve the problem (2). The pruning of neurons is done using Leave-One-Out Cross-Validation (LOO-CV) by choosing that combination of neurons (which have been previously sorted by the LARS algorithm) that provides lower LOO error. The LOO-CV error is efficiently computed using the Allen's formula [17]. An importance result from the Huang's work is related to the bias weight values [19]. It can be proved that the biases b_j are not required in the ELM's optimization for classification since the separate hyperplane in the ELM feature space passes through the origin. This result will be used in the method presented here.

3. Proposed method

The objective of this work is to automatically obtain a unique architecture design for an MLP network in order to solve a particular pattern classification problem. The proposed method employs the OP-ELM algorithm to design the network in a very fast and complete way. It is fast because of ELM is fast and complete because not only the number of hidden nodes is calculated but also the input connections that are useful to solve the classification problem. We will refer to it as ASELM (Architecture Selection based on ELM).
The ASELM takes advantage of the capabilities of the OP-ELM algorithm and introduces some variants on its implementation for performing architecture selection. Thus, instead of assigning random values for input weights, ASELM assigns binary values to them. In particular, they are given by all possible binary combinations of the number of input features. Thus, for example, if the problem is defined by four input features, the initial architecture will be composed by $2^4 - 1 = 15$) hidden neurons, being its corresponding inputs weights equal to $\mathbf{w}_1 = [0, 0, 0, 1]$, $\mathbf{w}_2 = [0, 0, 1, 0]$, $\mathbf{w}_3 = [0, 0, 1, 1]$, ..., $\mathbf{w}_{15} = [1, 1, 1, 1]$. Note that the case $[0, 0, 0, 0]$ is not considered and the biases are set to zero according the previously commented Huang's work result [19]. In general, if the classification problem presents n input features, the initial number of hidden neurons M is fixed by $2^n - 1$, being each hidden neuron connected to the input layer by means of the corresponding binary combination of the input weights.

Once the initial MLP architecture is defined, it is trained using the OP-ELM algorithm. Through the combined use of the LARS algorithm and the LOO cross-validation technique, the OP-ELM optimally discards those hidden neurons whose combination of input variables is not relevant to the target task. Note that, because of the binary value of the input weights, the selection of hidden nodes implies also the selection of the relevant connections between the input and hidden layers. Thus, only input connections corresponding to selected hidden neurons and with weights values equal to 1 will be part of the final architecture. Figure 1 shows an example of how the ASELM method works on a set of four-dimensional data ($n = 4$).
Once the final MLP structure has been obtained, it needs to be trained by a learning algorithm which learns all the parameters of the network. In this work, the classical Back-Propagation (BP) gradient based algorithm has been used to train the final architecture. A summary of the ASELM algorithm is shown next.

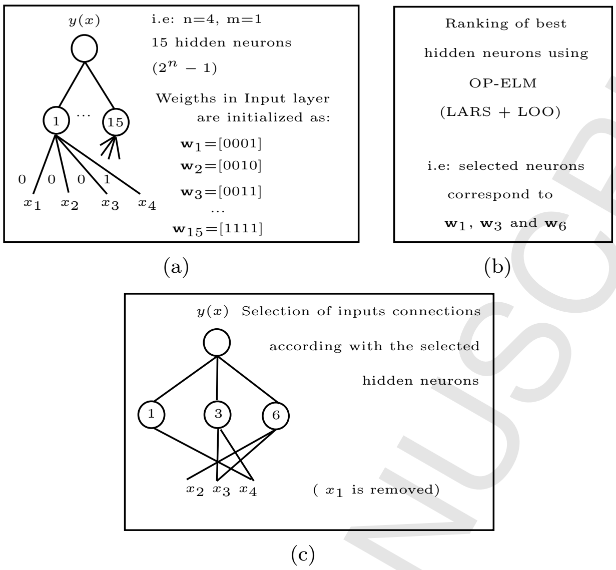


Figure 1: Example of the 3-steps of ASELM method to solve 4 dimensional classification problem. a) Initial network and weights; b) Training with OP-ELM algorithm; and c) Final architecture.

Algorithm 2 Architecture Selection ELM (ASELM)

Require: Given a training set $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{t}_i) | \mathbf{x}_i \in \mathbb{R}^n, \mathbf{t}_i \in \mathbb{R}^m, i = 1, \ldots, N\}$, activation function f, an hidden neuron number $2^n - 1$, where n is the number of input features, proceed as follows:

1: The weights of the input layer are initialized with binary values by considering all possible combinations of inputs. The case of all weights set to zero is discarded.

2: MLP network is trained by the OP-ELM and, then, useless hidden neurons are discarded according to the ranking given by LARS and LOO-CV procedure.

3: The final MLP architecture is given by the selected hidden neurons with its corresponding input(s) weight(s) equal to one.

---

4. Experiments

In order to show the plausibility of the ASELM method to design an appropriate MLP network, the accuracy classification results of four widely-used design methods for MLP networks are compared. Firstly, the MLP obtained with ASELM is trained with standard Back Propagation (BP) algorithm; secondly, an MLP network has also been trained with the BP algorithm being its hidden layer optimized by means of a Cross-Validation (CV) procedure; thirdly, the obtained MLP from CV is pruned using two well-known methods: Optimal

Table 1: Input and output features, and number of training and testing samples from the selected data sets.

Data set	Inputs	Outputs	Training	Testing
Logical	5	1	20	12
Iris	4	3	150	50
Pima	7	1	300	232
CoverType	10	4	2265	1132
Abalon	8	3	3133	1044
Cancer	9	1	480	202
Monk 1	6	1	288	144
Monk 2	6	1	288	144
Monk 3	6	1	288	144
Breast Tissue	10	4	70	36
Balance Scale Weight	4	3	416	209
Mammographic Mass	5	1	550	280
MAGIC Gamma Telescope	10	1	12680	6339
Wine (Red) recognition	11	1	1066	533

Brain Surgeon (OBS) [20, 21] and Optimal Brain Damage (OBD) [22]. The pruning of MLP by means of the OBS and OBD methods will be referred in this paper as MLP-OBS and MLP-OBD, respectively; and, finally, the MLP architecture is trained by the OP-ELM algorithm. Note that the simulations with MLP-OBS[1] , MLP-OBD[2] and OP-ELM[3] have been respectively performed using the Matlab Toolboxes developed in [23], [24] and [17].

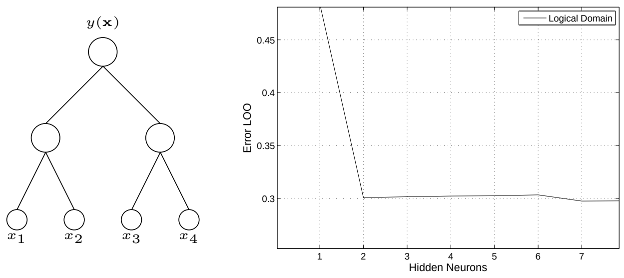
With respect to the cost function, the mean square error has been used (Mean Squared Error, MSE) as cost function to be minimized, and a 10-fold CV with 30 trials (weight initializations) have been done to obtain the classification accuracy of the four networks. Inappropriate initializations are discarded from the averaged accuracy results by considering a t-student distribution with a confidence interval of 95%. All the simulations have been carried out over the same computer (Intel Xeon 2.93GHz, 8GB of RAM).

A set of twelve different data sets have been selected with the criterion of having a large variability, i.e., different input and output dimensions and number of samples. A summary of the main features of these data sets can be seen in Table 1. These data sets are available in the UCI ML Repository [25], except ”Logical Domain” problem which comes from [26].

A reduced version of the original “Logical Domain” problem will be used to explain in more detail how ASELM works. This problem consists of a synthetic task, where the target is a function given by the logical combination of four binary input features: $t = (x_1 \vee x_2) \wedge (x_3 \vee x_4)$. In addition, there is another input feature, x_5, which is not relevant to learn the logic function, but it will be used to demonstrate how the method discards irrelevant input features. The

[1] http://eu.wiley.com/WileyCDA/Section/id-105036.html
[2] http://www.iau.dtu.dk/research/control/nnsysid.html
[3] http://research.ics.aalto.fi/eiml/software/OPELM.zip

OP-ELM is initialized to 31 hidden neurons ($i.e., 31 = 2^5 - 1$) with hidden weight vectors from $[0, 0, 0, 0, 1]$ to $[1, 1, 1, 1, 1]$. After training the initial network with the OP-ELM procedure, the ASELM selects only 2 hidden nodes: one learns the function $(x_1 \vee x_2)$ and the other learns $(x_3 \vee x_4)$. This is due to ASELM has selected the two neurons whose input weights are $[1, 1, 0, 0, 0]$ and $[0, 0, 1, 1, 0]$. Then, we can see that the first neuron receives x_1 and x_2 as inputs and the second one x_3 and x_4. It can also be observed how the irrelevant feature x_5 is not included (see Figure 2). This new architecture provides a more compact MLP scheme where unnecessary connections and irrelevant inputs are removed. Moreover, it can be observed from the simulation results that the MLP provided by the ASELM requires a lower number of epochs to reach the convergence than the MLP designed by CV. Figure 3 shows an example of this case.



(a) (b)

Figure 2: Solution of the ASELM method over the Logical Domain problem. a) Selected architecture; b) selection of hidden neurons by LOO cross-validation provide by the OP-ELM method. It can be observed how the LOO error saturates after adding the second hidden neuron.

Table 2 shows the classification results obtained by the five procedures described before (ASELM, MLP-BP, MLP-OBS, MLP-OBD and OP-ELM) over the fourteen classification data sets. The column NC represents the number of connections between the input layer and the hidden layer. This allows us to see how ASELM, MLP-OBS and MLP-OBD reduces the number of effective connections versus full-connected scheme (MLP-BP and OP-ELM). The column RT represents the number of seconds required for the design and training of the network (RunTime). Thus, RT includes the total 10-fold CV with 30 simulations by fold for each MLP architecture. Note that, in cases of MLP-OBS and MLP-OBD method, RT is given by the RT of MLP-BP plus the corresponding time for pruning using MLP-OBS or MLP-OBD. In the case of MLP-BP, this process is repeated with different hidden layer sizes to get the optimal one. As it can be observed, ASELM provides a very competitive classification accuracy compared with MLP-BP, MLP-OBS and MLP-OBD methods, but with a much lower training time and smaller architecture. In addition, it clearly outperforms the results given by the OP-ELM procedure both in classification accuracy and

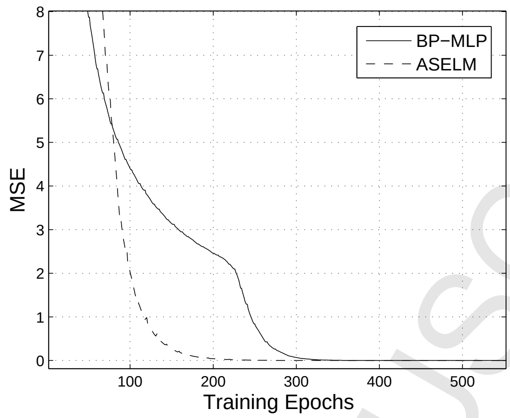


Figure 3: Reduction of training epochs of the ASELM scheme versus the standard BP-MLP scheme for the "Logical Domain" data set.

size of the final network. It should also be noted that ASELM provides an unique solution for designing the MLP architecture, performs feature selection stage and pruning the irrelevant hidden nodes, by giving a final simplified MLP architecture which converges faster than a fully interconnected scheme.

5. Conclusion and Future Work

In most problems, learning from data using traditional training methods for MLP schemes (such as BP) involves excessive computational cost and it also requires trial and error or cross-validation procedures for the design of the architecture. Furthermore, the existence of irrelevant or noisy input features can seriously hinder the training. Some approaches (e.g. the well-known OBS and OBD methods) have been developed for removing useless input connections and hidden nodes. However, given the previously trained full-connected scheme, they entail additional training time for pruning irrelevant connections and neurons and, also, it does not ensure an enhancement of the full-connected MLP trained by BP in terms of the generalization capabilities. In this paper, a new algorithm based on OP-ELM to automatically construct a simplified MLP architecture is presented. This architecture selection method (known as ASELM) discards irrelevant input features and, also, eliminates unnecessary hidden neurons and input connections. In particular, the proposed method uses the OP-ELM algorithm, which is based on the Least Angle Regression (LARS) for ranking hidden neurons and the Allen's formula of the LOO-CV error for pruning useless neurons. ASELM automatically provides a fast and unique solution for the MLP scheme with a very competitive generalization capability and without user intervention: there is not any tunable parameter.
In the experimental results section, a toy problem to explain the operation of the

proposed method and thirteen real problems from the UCI ML Repository have been used to test the proposed method and compare it with four well-known approaches for designing MLP networks: MLP-BP, MLP-OBS, MLP-OBD and OP-ELM. In general, the simplified architectures obtained by ASELM for these data sets have improved the performance with respect to the fully interconnected MLP schemes and the pruned MLP architectures with OBS and OBD in terms of the required computational training time and the generalization capability. The extension of ASELM to other machine learning models, such as Radial Basis Functions (RBF) Networks, and regression problems, constitutes the immediate future work to be done. Another important issue is related to the limitation of the proposed method dealing with data set with very large number of input features. It is due to the nature of the ELM method, which is based on pseudoinverse computation. This constitutes the third ongoing research line. In this sense, a sequential computation of the Moore-Penrose pseudoinverse is being under investigation [27, 28].

References

[1] Bartlett, E. B. (1994). Self determination of input variable importance using neural networks. *Neural, Parallel & Scientific Computations, 2*, 103-114.

[2] Kwok, T. Y. & Yeung, D. Y. (1997). Constructive algorithms for structure learning in Feedforward Neural Networks. *IEEE Transactions on Neural Networks, 8(3)*, 630-645.

[3] Hagiwara, M. (1994). A simple and effective method for removal of hidden units and weights. *Neurocomputing, 6*, 207-218.

[4] Setiono, R. & Liu, H. (1996). Improving Backpropagation learning with feature selection. *Applied Intelligence: The International Journal of Artificial Intelligence, Neural Networks, and Complex Problem-Solving Technologies, 6(2)*, 129-139.

[5] Belue, L. M. & Baver, K. W. (1995). Determining input features for multilayer perceptrons. *Neurocomputing, 7(2)*, 111-121.

[6] Durbin, B., Dudoit, S. & Van der Laan, M. L. (2008). A Deletion/Substitution/Addition algorithm for classification neural networks, with applications to biomedical data. *Journal of Statistical Planning and Inference, 138*, 464-488.

[7] Narasimhaa, P. L., Delashmitb, W. H., Manrya, M. T., Li, J. & Maldonado, F. (2008). An integrated growing-pruning method for feedforward network training. *Neurocomputing, 71*, 2831-2847.

[8] Duda, R. O., Hart, P. E. & Stork, D. G. (2000). *Pattern Recognition and Neural Networks.* Wiley-interscience.

[9] Bishop, C. (2006). *Pattern Recognition and Machine Learning.* Springer.

[10] Huang, G. B., Zhu, Q. Y. & Siew, C. K. (2006). Extreme Learning Machine: Theory and Applications, *Neurocomputing, 70(1-3)*, 489-501.

[11] Huang, G. B., Wang, D. H. & Lan, Y. (2011). Extreme Learning Machines: A survey. *International Journal of Machine Leaning and Cybernetics, 2 (2)*, 107122.

[12] Pao Y. H., Park, G. H. & Sobajic, D. J. (1994). Learning and generalization characteristics of the Random Vector Functional-Link Net. *Neurocomputing, 6(2)*, 163-180.

[13] Igelnik, B. & Pao Y. H. (1997). Stochastic choice of basis functions in adaptive function approximation and the Functional-Link Net. *IEEE Transactions on Neural Networks,8(2)*, 452-454.

[14] Huang, G. B. & Chen, L. (2007). Convex incremental Extreme Learning Machine. *Neurocomputing, 70*, 3056-3062.

[15] Wang, L. & Chunru, W. (2008). Comments on "the Extreme Learning Machine". *IEEE Transactions on Neural Networks, 19(8)*, 1494-1495.

[16] Björk, A. (1996). Numerical methods for Least Squares problems. *SIAM, Philadelphia.*

[17] Miche, Y., Sorjamaa, A., Bas, P., Simula, O., Jutten, C. & Lendasse, A. (2010). OP-ELM: optimally pruned extreme learning machine. *IEEE Trans Neural Networks, 21(1)*, 158-162.

[18] Efron, B., Hastie, T., Johnstone, I. & Tibshirani, R. (2004). Least Angle Regression. *Annal of Statistics, 32*, 407-499.

[19] Huang, G. B., Ding, X. & Zhou, H. (2010), Optimization method based Extreme Learning Machine for classification. *Neurocomputing, 74*, 155-163.

[20] Hassibi, B., Stork, D. G. & Wolff, G. J. (1993), Optimal Brain Surgeon and general network pruning. *IEEE International Conference on Neural Networks, 1*, 293 - 299.

[21] Weishui Wan, W., Mabu, S., Shimada, K., Hirasawa, K. & Hu J. (2009), Enhancing the generalization ability of Neural Networks through controlling the Hidden Layers. *Applied Soft Computing. 9*, 404-414.

[22] Cun, L., Denker, J. S. & Solla, S. A. (1990), Optimal Brain Damage. *In Advances in Neural Information Processing Systems, 2*, 598-605.

[23] Stork, D. G. & Yom-Tov, E. (2004). *Computer Manual in MATLAB to Accompany Pattern Classification, Second Edition.* Wiley-Interscience.

[24] Norgaard, M., Ravn, O., Hansen, L. K. & Poulsen, N. K. (1996), The NNSYSID Toolbox - A MATLAB Toolbox for System Identification with Neural Networks. *Proceedings of the 1996 IEEE International Symposium on Computer-Aided Control System Design.*

[25] Asuncion, A. & Newman D. J. (2007). UCI Machine Learning Repository. *http://www.ics.uci.edu/~mlearn/MLRepository.html.*

[26] Silver, D. L. & Mercer, R. E. (2001). Selective functional transfer: Inductive bias from related tasks. *Proceedings of the IASTED International Conference on Artificial Intelligence and Soft Computing*, 182-191.

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

[28] Tapson, J. & van Schaik, A. (2013). Learning the pseudoinverse solution to network weights. *Neural Networks*, Available online 13 March 2013.

Table 2: Test classification accuracy (CA)(mean ± std) with different methods and data sets, HN represents the number of neurons in the hidden layer, NC represents the number of connections between the input layer and the hidden layer, RT the runtime in seconds (entire process: architecture selection and training), and RF is the input features removed by ASELM.

Data set	Method	CA(mean ± std)	HN	NC	RT	RF
Logical	$ASELM$	1.000±0.000	2	4	13	x_5
	$MLP-BP$	0.941±0.059	7	35	143	
	$MLP-OBS$	0.854±0.064	7	32±1	143+14	
	$MLP-OBD$	0.911±0.072	7	32±2	143+8	
	$OP-ELM$	0.838±0.045	6±1	30±5	<1	
Iris	$ASELM$	0.986±0.001	5	8	144	x_1
	$MLP-BP$	0.982±0.006	6	24	658	
	$MLP-OBS$	0.967±0.021	6	10±5	658+30	
	$MLP-OBD$	0.979±0.011	6	16±4	658+20	
	$OP-ELM$	0.947±0.026	8±2	32±8	3	
Pima	$ASELM$	0.791±0.003	8	24	43	none
	$MLP-BP$	0.791±0.006	21	147	14879	
	$MLP-OBS$	0.775±0.007	21	137±5	14879+337	
	$MLP-OBD$	0.764±0.015	21	128±8	14879+496	
	$OP-ELM$	0.768±0.015	7±2	49±14	4	
CoverType	$ASELM$	0.924±0.004	12	42	5044	x_5
	$MLP-BP$	0.926±0.003	15	150	25532	
	$MLP-OBS$	0.832±0.008	15	138±3	25532+4707	
	$MLP-OBD$	0.867±0.004	15	112±5	25532+9216	
	$OP-ELM$	0.871±0.006	45±2	450±20	204	
Abalon	$ASELM$	0.762±0.010	9	14	4248	x_5 and x_7
	$MLP-BP$	0.775±0.027	19	152	75592	
	$MLP-OBS$	0.791±0.024	19	140±7	75592+5974	
	$MLP-OBD$	0.775±0.015	19	137±6	75592+9587	
	$OP-ELM$	0.760±0.017	33±3	264±24	270	
Cancer	$ASELM$	0.966±0.009	2	10	128	x_5
	$MLP-BP$	0.973±0.010	10	90	2080	
	$MLP-OBS$	0.975±0.015	10	63±8	2080+795	
	$MLP-OBD$	0.952±0.016	10	61±14	2080+238	
	$OP-ELM$	0.955±0.023	9±3	90±30	6	
Monk 1	$ASELM$	0.998±0.005	8	21	899	none
	$MLP-BP$	0.986±0.017	8	48	42465	
	$MLP-OBS$	0.993±0.022	8	40±3	42465+75	
	$MLP-OBD$	0.984±0.028	8	16±5	42465+18	
	$OP-ELM$	0.705±0.035	22±5	132±30	7	
Monk 2	$ASELM$	0.754±0.032	21	60	1020	none
	$MLP-BP$	0.820±0.043	26	156	49794	
	$MLP-OBS$	0.689±0.032	26	126±6	49794+929	
	$MLP-OBD$	0.816±0.045	26	146±6	49794+711	
	$OP-ELM$	0.682±0.038	19±7	114±42	7	
Monk 3	$ASELM$	0.970±0.008	6	13	477	x_1
	$MLP-BP$	0.990±0.005	5	30	42730	
	$MLP-OBS$	0.835±0.006	5	22±3	42730+57	
	$MLP-OBD$	0.945±0.046	5	10±3	42730+12	
	$OP-ELM$	0.834±0.043	25±6	150±36	7	
Balance	$ASELM$	0.933±0.007	10	20	529	none
	$MLP-BP$	0.917±0.003	14	56	27528	
	$MLP-OBS$	0.901±0.007	14	45±5	27528+168	
	$MLP-OBD$	0.898±0.018	14	38±8	27528+478	
	$OP-ELM$	0.909±0.010	33±7	132±28	11	
Breast T.	$ASELM$	0.918±0.018	27	86	82	none
	$MLP-BP$	0.919±0.021	19	190	10032	
	$MLP-OBS$	0.928±0.028	19	129±12	10032+328	
	$MLP-OBD$	0.904±0.031	19	135±6	10032+720	
	$OP-ELM$	0.920±0.024	24±12	216±108	3	
Mammogr.	$ASELM$	0.811±0.010	2	5	281	x_5
	$MLP-BP$	0.782±0.052	9	45	9612	
	$MLP-OBS$	0.799±0.018	9	35±6	9612+167	
	$MLP-OBD$	0.786±0.012	9	28±9	9612+84	
	$OP-ELM$	0.808±0.011	24±8	120±40	14	
MAGIC	$ASELM$	0.858±0.011	44	310	1820	none
	$MLP-BP$	0.855±0.014	9	90	193018	
	$MLP-OBS$	0.795±0.018	9	48±8	193018+10231	
	$MLP-OBD$	0.861±0.004	9	57±9	193018+9261	
	$OP-ELM$	0.815±0.006	50±6	500±60	659	
Wine	$ASELM$	0.744±0.007	32	127	8112	none
	$MLP-BP$	0.738±0.004	21	231	74061	
	$MLP-OBS$	0.719±0.015	21	199±8	74061+4468	
	$MLP-OBD$	0.717±0.010	21	226±4	74061+1636	
	$OP-ELM$	0.722±0.011	34±7	374±77	68	