ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

# A new model selection approach for the ELM network using metaheuristic optimization

Ananda L. Freire and Guilherme A. Barreto *

Federal University of Ceará (UFC), Department of Teleinformatics Engineering (DETI)
Av. Mister Hull, S/N - Center of Technology, Campus of Pici, Fortaleza, Ceará, Brazil

**Abstract**. We propose a novel approach for architecture selection and hidden neurons excitability improvement for the Extreme Learning Machine (ELM). Named *Adaptive Number of Hidden Neurons Approach* (ANHNA), the proposed approach relies on a new general encoding scheme of the solution vector that automatically estimates the number of hidden neurons and adjust their activation function parameters (slopes and biases). Due to its general nature, ANHNA's encoding scheme can be used by any metaheuristic algorithm for continuous optimization. Computer experiments were carried out using Differential Evolution (DE) and Particle Swarm Optimization (PSO) metaheuristics, with promising results being achieved by the proposed method in benchmarking regression problems.

## 1 Introduction

Single hidden layer feedforward networks work as universal approximator with any bounded non-linear piecewise continuous functions for additive nodes [1]. Among them, the Extreme Learning Machine (ELM) has become very popular due to its fast training speed by setting randomly the hidden nodes weights and biases. This configuration results in a linear model for the network output weights, which are then analytically determined by finding a least-square solution [2]. Although ELM offers good generalization performance, the random selection of input-to-hidden-layer weights (input weights, for short) may produce a set of non-optimal input weights and biases, which might suffer from overfitting [1].

To avoid that, during the past recent years, metaheuristic optimization algorithms have been used as global searching methods for finding suitable input weight values. For instance, the Evolutionary ELM [3] uses the Differential Evolution (DE) algorithm [4] to search for optimal input weights. Hidden-to-output-layer weights are estimated as the usual ELM. The Self Adaptive Evolutionary ELM [1] is similar to the E-ELM, although it allows automatic selection of different trial vector generation strategies and control parameters. In [5], the ELM network is combined with an improved Particle Swarm Optimization [6] to optimize the input weights and biases. Another method is the Group Search Optimization ELM [7] that also aims at optimizing the input weights and biases.

It should be pointed out, however, that none of the aforementioned strategies were designed to automatically determine the optimal number of hidden

*This work was supported by CAPES, an entity of the Brazilian government dedicated to the scientific and technological development.

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

neurons. In other words, in all of them the number of hidden neurons is specified beforehand. Furthermore, they do not explore the possibility of adapting the neurons' excitability parameters (i.e. the slope and bias of their activation functions). Tuning of those parameters are related to the intrinsic plasticity of a neuron cell [8] and can result in better generalization performances [9, 10].

From the exposed, we introduce a new encoding scheme for the solution vector of a given metaheuristic optimization algorithm that allows automatic estimation of the number of hidden neurons and their activation function parameter values. The proposed scheme, named *Automatic Number of Hidden Neurons Approach* (ANHNA), is very general and can be used by any metaheuristic algorithm for continuous optimization. Computer experiments using Differential Evolution (DE) and Particle Swarm Optimization (PSO) metaheuristics demonstrate the feasibility of the proposed approach.

## 2 Algorithms

**Extreme Learning Machine (ELM)** is a two layer network with random and fixed weights matrix on the hidden layer, $\mathbf{W} \in \mathbb{R}^{p \times q}$, where $p$ is the number of input units and $q$ the number of the hidden ones [2]. The output of the hidden layer is $\mathbf{h}(k+1) = \phi\left(\mathbf{W}^T(k)\mathbf{u}(k)\right)$, where $\mathbf{u}(k) \in \mathbb{R}^p$ is the current input vector and $\phi$ is a logistic activation function, and the $j$-th network output is $y_j(k+1) = \sum_{i=1}^{q} w_{ji}^{out}(k) h_i(k)$. For training, all inputs from the training sequence $((\mathbf{u}(k), \mathbf{d}(k)), k = 1...N)$ are presented to the network and the corresponding network states $(\mathbf{h}(k), \mathbf{d}(k))$ are collected (harvested) in respective matrices, where $\mathbf{d}(k)$ is the desired output. The calculation of the output weight matrix $\mathbf{W}^{out}$ is accomplished by a linear regression: $\mathbf{W}^{out} = \mathbf{H}^{\dagger}\tilde{D}$, where $\mathbf{H}^{\dagger}$ is the Moore-Penrose generalized inverse of the hidden layer output matrix $\mathbf{H}$.

**Batch Intrinsic Plasticity (BIP)** is an unsupervised learning rule that adapts bias $(b_i)$ and slope $(a_i)$ of the neurons' activation function, tuning them into more suitable regimes, maximizing information transmission and acting as a feature regularizer [9]. It is done in a way that the desired exponential distribution $f_{des}$ for the neurons activation $h_i(k) = (1 + \exp(-a_i x_i(k) - b_i))^{-1}$ is realized. For each hidden neuron, all the arriving synaptic sum $\mathbf{x}_i = \mathbf{w}_i^T \mathbf{U}$ is collected, where $\mathbf{U} = (\mathbf{u}(1), ..., \mathbf{u}(N))$. Then random targets $\mathbf{t}_{f_{des}} = (t_1, ..., t_N)^T$, from the desired output distribution, and the collected stimuli are drawn in ascending order. The model $\Phi(\mathbf{x}_i) = (\mathbf{x}_i^T, (1, ...1)^T)$ is built so we can calculate $(a_i, b_i)^T = (\Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_i) + \lambda I)^{-1} \Phi(\mathbf{x}_i)^T f^{-1}(\mathbf{t}_{f_{des}})$, where $f^{-1}$ is the inverse logistic function, $\lambda > 0$ is the regularization parameter and $I \in \mathbb{R}^{q \times q}$ is an identity matrix [9].

**Differential Evolution (DE)** aims at evolving a population of $NP$ chromosomes ($\mathbf{C}_i \in \mathbb{R}^d$, $i = 1, ..., NP$) towards the global optimum and the initial population is chosen randomly ($\mathbf{C}_i(0) \sim U(\mathbf{C}_{min}, \mathbf{C}_{max})$). The mutation is applied first, producing a trial vector, $\mathbf{V}_i$ for each chromosome of the $g$-th population:

$$\mathbf{V}_i(g) = \mathbf{C}_{i_1}(g) + \beta(\mathbf{C}_{i_2}(g) - \mathbf{C}_{i_3}(g)), \tag{1}$$

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

where $\beta \in (0, \infty)$ is the scale factor that controls the amplification of the differential variation [4]. From the several approaches that can be used for mutation, we adopted the following one: the chromosomes $\mathbf{C}_{i_2}(g)$ and $\mathbf{C}_{i_3}(g)$ are randomly chosen with $i_1 \neq i_2 \neq i_3$, and $\mathbf{C}_{i_1}(g)$ is the individual with the best fitness.

Then, the DE crossover operator implements a discrete recombination of $\mathbf{V}_i(g)$ and $\mathbf{C}_i(g)$ to produce the offspring $\mathbf{C}'_i(g)$. For each $j$-th component:

$$C'_{i,j}(g) = \begin{cases} V_{i,j}(g), & \text{if } U(0,1) < CR \\ C_{i,j}(g), & \text{otherwise.} \end{cases} \tag{2}$$

As suggested in [11], the crossover rate $CR$ adopted in this work will decrease linearly with the generations from $CR_{max} = 1$ to $CR_{min} = 0.5$ following this rule: $CR = CR_{min} + (CR_{max} - CR_{min}) \left(\frac{MAXGEN - g}{MAXGEN}\right)$, where $MAXGEN$ is the maximum number of generations possible. Such tuning of $CR$ helps to explore the search space exhaustively at the beginning, but adjust the movements of trial solutions finely during the later stages of search, so that they can explore the interior of a relatively small space in which the suspected global optimum lies [11]. Finally, the selection of chromosomes for the next generation:

$$\mathbf{C}_i(g+1) = \begin{cases} \mathbf{C}'_i(g), & \text{if } f(\mathbf{C}'_i(g)) < f(\mathbf{C}_i(g)) \\ \mathbf{C}_i(g), & \text{otherwise} \end{cases} \tag{3}$$

where $f(.)$ is the objective function to be minimized.

**Particle Swarm Optimization (PSO)** is a population-based search algorithm based on the simulation of the social behavior of birds within a flock. A population of chromosomes, $\mathbf{C}_i \in \mathbb{R}^d$, is initialized with random positions ($\mathbf{C}_i(0) \sim U(\mathbf{C}_{min}, \mathbf{C}_{max})$) and random velocities $\mathbf{v}_i$. Every chromosome is defined within the context of a neighborhood $k$ that comprises itself and some other particles in the population. It may be the entire swarm (*global* PSO) or adopt a social network topology which define smaller neighborhoods (*local* PSO). In this work, for the local approach, we adopted the *ring* structure, where each individual communicates with its immediate adjacent neighbors.

At each generation, the vectors of best historical individual position, $\mathbf{p}_i \in \mathbb{R}^d$, and the best historical position of the neighborhood $k$, $\mathbf{pk}_i \in \mathbb{R}^d$, are stored. The difference between $\mathbf{pk}_i$ and the $i$-th individual current position is stochastically added to its current velocity, causing the trajectory to oscillate around that best position. The velocity equation update is shown below:

$$v_{ij}(g+1) = \chi[v_{ij}(g) + \phi_1\epsilon_1(p_{ij}(g) - C_{ij}(g)) + \phi_2\epsilon_2(pk_{ij}(g) - C_{ij}(g))] \tag{4}$$

where $\chi$ is the constriction factor, $\phi_1$ and $\phi_2$ are positive constants called acceleration coefficients and $\epsilon_1$ and $\epsilon_2$ are independent random numbers uniquely generated at every update for each individual element [6]. The $i$-th chromosome is updated as follows:

$$\mathbf{C}_i(g+1) = \mathbf{C}_i(g) + \mathbf{v}_i(g+1). \tag{5}$$

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

**Adaptive Number of Hidden Neurons Approach (ANHNA)** - has as its core an encoding for chromosomes which optimize, simultaneously, the number of hidden neurons and their activation function parameters. Inspired by [11], the chromosome is a vector of real numbers of dimension $(Q_{max} + Q_{max} + Q_{max})$, where $Q_{max}$ is the maximum number of hidden neurons. The first $Q_{max}$ elements range from 0 to 1, each of which controls whether the corresponding set of $(a_{i,j}, b_{i,j})$ is activated or not. The following $Q_{max}$ elements are the activation functions' slope values $(a_{i,j})$ and the remaining are their respective bias $(b_{i,j})$. For example, the $i$-th chromosome is presented below:

$$\overrightarrow{C}_i(g) = \underbrace{\boxed{T_{i,1}}\boxed{T_{i,2}}\boxed{...}\boxed{T_{i,Q_{max}}}}_{\text{Activation Threshold}}\underbrace{\boxed{a_{i,1}}\boxed{...}\boxed{a_{i,Q_{max}}}}_{\text{Slope}}\underbrace{\boxed{b_{i,1}}\boxed{...}\boxed{b_{i,Q_{max}}}}_{\text{Bias}}$$

The activation thresholds $T_{i,j}$ behave like control genes, selecting the actual number of hidden neurons, $q$, accordingly with this rule: if $T_{i,j} \geq 0.5$ then the $j$-th hidden neuron $(a_{i,j},b_{i,j})$ is active, otherwise, this neuron does not exist to the network. Besides, some precautions must be taken at the beginning of each population (parents and offspring as well, in DE case). First, check if at least the minimum number of hidden neurons, $Q_{min}$, are active in every chromosome. For those that do not comply with this rule, in each of them, choose randomly $Q_{min}$ activation thresholds where $\mathbf{T}_i(g) \sim U(0.5, 1)$. Second and last rule, after every chromosome update, the slope values $\mathbf{a}_i$ will be replaced for their absolute values. The Algorithm 1 presents a pseudo-code for ANHNA method.

**Algorithm 1** ANHNA Pseudocode $\{Q_{min}, Q_{max}\}$

```
Initialize the parameters from DE or PSO
Initialize population C(0), where T(0) and a(0) ~ U(0,1) and b(0) ~ U(−1,1)
Check activation thresholds
while g < MAXGEN do
    for each chromosome, C_i(g) do
        Train and Test an ELM   ▷ with q_i active hidden neurons and their (a_i,b_i)
        Evaluate f(C_i)         ▷ fitness objective is the RMSE from ELM's test
        if DE is chosen then
            Mutation (Eq. 1)
            Check activation thresholds and |a_i|
            Crossover (Eq. 2) and Selection (Eq. 3) → C_i(g + 1)
        end if
    end for
    if PSO is chosen then
        for each chromosome, C_i(g) do
            Set p_i and pk_i        ▷ personal and neighborhood best positions
            Update velocity (Eq. 4) and position (Eq. 5)
            Check activation thresholds and |a_i|
        end for
    end if
end while
return ELM with smaller RMSE in the last generation.
```

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

## 3 Results

The performance of ANHNA/DE, ANHNA/global-PSO, ANHNA/local-PSO, traditional ELM and ELM with BIP are compared on the regression of 4 real world datasets acquired from the UCI[1] and StatLib[2] databases. The *autompg*, that concerns city-cycle fuel consumption in miles per gallon; *Boston housing*, which involves the task of predicting housing values in areas of Boston; *machine CPU*, that concerns the instance-based prediction of relative CPU performance; and *bodyfat*, that estimates the percentage of body fat determined by underwater weighing and various body circumference measurements.

The input data was normalized between $[-0.9, 0.9]$ and the output data $[0.1, 0.9]$. Each set was divided into training and test independent sets, with an amount of (training/test) samples: for *autompg* (273/118), *Boston housing* (354/152), *machine CPU* (143/63) and *bodyfat* (176/76).

The ANHNA method's parameters were configured as $Q_{max} = 100$ and $Q_{min} = 10$. For the DE algorithm, $\beta = 0.5$, $NP = 100$ and $MAXIGEN = 300$. For PSO, as suggested in [6], $\phi_1 = \phi_2 = 2.05$ and $\chi = 0.72984$, besides a $NP = 50$ and $MAXIGEN = 150$. Considering the ELM networks, all activation functions are hyperbolic tangent and, for those whose excitability are not changed, $bias = 1$. All implementations were executed with MATLAB, where, for each dataset, 50 independent runs were executed with all methods described above. Also, the number of hidden neurons for traditional ELM and ELM/BIP is chosen from the biggest value found after those 50 runs of every ANHNAs.

Tab. 1: Comparison of the mean RMSE and the median $q$ hidden neurons.

| | **Autompg** | | | **Machine CPU** | | |
|---|---|---|---|---|---|---|
| | Train | Test | $q$ | Train | Test | $q$ |
| **ANHNA/DE** | 0.0479±0.0014 | **0.0489±0.0005** | 56±3.78 | 0.0171±0.0044 | **0.0329±0.0045** | 42±3.56 |
| **ANHNA/g-PSO** | 0.0483±0.0013 | 0.0498±0.0005 | 53±3.96 | 0.0149±0.0007 | 0.0405±0.0084 | 42±4.33 |
| **ANHNA/l-PSO** | 0.0482±0.0014 | 0.0497±0.0006 | 53±4.12 | 0.0149±0.0006 | 0.0419±0.0059 | 41±3.72 |
| **ELM** | **0.0461±0.001** | 0.0563±0.0020 | 64 | 0.0133±0.0003 | 0.4909±0.2959 | 53 |
| **ELM/BIP** | 0.0477±0.0012 | 0.0553±0.0027 | 64 | **0.0132±0.0005** | 0.1791±0.0840 | 53 |
| | **Bodyfat** | | | **Boston Housing** | | |
| | Train | Test | $q$ | Train | Test | $q$ |
| **ANHNA/DE** | **0.0123±0.0009** | **0.0148±0.0009** | 51±6.01 | **0.0341±0.0024** | **0.0385±0.0014** | 59±4.41 |
| **ANHNA/g-PSO** | 0.0130±0.0013 | 0.0155±0.0004 | 48±5.72 | 0.0356±0.0022 | 0.0405±0.0014 | 57±4.79 |
| **ANHNA/l-PSO** | 0.0129±0.0014 | 0.0156±0.0005 | 51±5.13 | 0.0345±0.0025 | 0.0398±0.0017 | 59±4.80 |
| **ELM** | 0.0124±0.0010 | 0.0417±0.0175 | 64 | 0.0383±0.0027 | 0.0542±0.0053 | 71 |
| **ELM/BIP** | 0.0230±0.0027 | 0.0608±0.0178 | 64 | 0.0373±0.0030 | 0.0523±0.0041 | 71 |

In Tab. 1, for each dataset, the mean training and testing RMSE and its standard deviation are shown, as well as the median number of hidden neurons and its standard deviation. From those results, it may be observed that ELM and ELM/BIP has the best training performance in half of the sets, however, ANHNA methods show better generalization potentiality with less neurons than

[1] `http://archive.ics.uci.edu/ml/index.html`
[2] `http://lib.stat.cmu.edu/index.php`

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

the traditional ELM and the improved ELM/BIP. The ANHNA/DE, specifically, has shown the best results both in training and testing in most of the datasets.

## 4 Conclusion

This work presented a hybrid method based on metaheuristics for optimizing the ELM network, where the number of hidden neurons and their excitability capacity are improved in one single evolutionary process. It is build up with a new chromosome encoding within a basic DE or PSO algorithms altered by few improvements and general parameters values, which are suggested in the literature. From this basic setup, we achieved better generalization capacity and network size selection at once in real world regression problems. In addition, ANHNA demonstrates that is possible to improve ELM's performance without resorting to the optimization of input weights and biases, whose randomness is the main characteristic of the ELM network.

As for future work, we aim at evaluating ANHNA's performances in the presence of outliers by embedding automatic robust statistics tools directly into the encoding of the solution vector. Further studies on the impact of the metaheuristics' choice of strategies and control parameters are also being executed.

## References

[1] J. Cao, Z. Lin, and G.-B. Huang. Self-adaptive evolutionary extreme learning machine. *Neural Processing Letters*, 36(3):285 – 305, 2012.

[2] G.-B. Huang, D. Wang, and Y. Lan. Extreme learning machines: a survey. *International Journal of Machine Learning and Cybernetics*, 2(2):107 – 122, 2011.

[3] Qin-Yu Zhu, A.K. Qin, P.N. Suganthan, and Guang-Bin Huang. Evolutionary extreme learning machine. *Pattern Recognition*, 38(10):1759 – 1763, 2005.

[4] R. Storn and K. Price. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 1997.

[5] F. Han, H.-F. Yao, and Q.-H. Ling. An improved evolutionary extreme learning machine based on particle swarm optimization. *Neurocomputing*, 116:87 – 93, 2013.

[6] D. Bratton and J. Kennedy. Defining a standard for particle swarm optimization. In *Swarm Intelligence Symposium, 2007. SIS 2007. IEEE*, pages 120 – 127, 2007.

[7] D. N G Silva, L. D S Pacifico, and T.B. Ludermir. An evolutionary extreme learning machine based on group search optimization. In *2011 IEEE Congress on Evolutionary Computation (CEC)*, pages 574 – 580, 2011.

[8] J. Triesch. A gradient rule for the plasticity of a neurons intrinsic excitability. In *Int. Conf. on Artificial Neural Networks*, page 6579, 2005.

[9] K. Neumann and J. Steil. Batch intrinsic plasticity for extreme learning machines. In *Artificial Neural Networks and Machine Learning - ICANN*, pages 339–346. 2011.

[10] K. Neumann and J. Steil. Optimizing extreme learning machines via ridge regression and batch intrinsic plasticity. *Neurocomputing*, 102:23 – 30, February 2013. Advances in Extreme Learning Machines (ELM 2011).

[11] S. Das, A. Abraham, and A. Konar. *Metaheuristic Clustering*. Studies in Computational Intelligence, Vol. 178. Springer, 2009.