Conditioning of extreme learning machine for noisy data using heuristic optimization

Conditioning of extreme learning machine for noisy data using heuristic optimization

E Salazar[1], M Mora[1], A Vásquez[1], and E Gelvez[2]
[1] Facultad de Ciencias de la Ingeniería, Universidad Católica del Maule, Talca, Chile
[2] Facultad de Ciencias Básicas y Biomédicas, Universidad Simón Bolívar, San José de Cúcuta, Colombia

E-mail: marcomoracofre@gmail.com

Abstract. This article provides a tool that can be used in the exact sciences to obtain good approximations to reality when noisy data is inevitable. Two heuristic optimization algorithms are implemented: Simulated Annealing and Particle Swarming for the determination of the extreme learning machine output weights. The first operates in a large search space and at each iteration it probabilistically decides between staying at its current state or moving to another. The swarm of particles, it optimizes a problem from a population of candidate solutions, moving them throughout the search space according to position and speed. The methodology consists of building data sets around a polynomial function, implementing the heuristic algorithms and comparing the errors with the traditional computation method using the Moore–Penrose inverse. The results show that the heuristic optimization algorithms implemented improve the estimation of the output weights when the input have highly noisy data.

1. Introduction

A basic problem in science is to create prediction models from observations that are subject to errors. These prediction models do not necessarily have to approximate exactly the original data, since noise can take unwanted characteristics and avoid showing the fundamental. One of the techniques for obtaining prediction models are artificial neural networks, which can find nonlinear relationships between data sets through learning algorithms, obtaining models of complex problems. The extreme learning machine has become the most used algorithm for its calculation speed, however it needs many neurons in the hidden layer to find good models.

The extreme learning machine consists of randomly initializing the weights of the hidden network layer and calculating the output weights using the Moore-Penrose inverse. In this sense, the objectives of this work are: first to show that the estimation error with the Moore-Penrose inverse increases as the noise in the data increases and second that the hueristic optimization processes: simulated annealing and particle swarm improve the robustness of the extreme learning machine for highly noisy data. Heuristic algorithms are adopted because they make little or no assumption about the problem being optimized and can search in large spaces, candidates for optimal solutions at a low computational cost.

The noisy data used to validate the hypotheses are generated around a polynomial curve, since they allow studying different scenarios by increasing the number of network inputs as a function of on the degree of the polynomial. Select a 70% of the data is randomly for the network training and 30% for the test; the latter is used to validate the effectiveness of each method.

2. Background

2.1. Artificial neural networks

The artificial neural networks (ANNs) is based on a collection of connected units or nodes called artificial neurons, which are connected to each other to transmit signals. In the links the output value of the previous neuron is multiplied by a weight value and can increase or inhibit the activation state of neighboring neurons. Moreover, in the exit of the neuron there may be a limiting function or threshold which modifies the result or imposes a limit that must be exceeded before spreading to another neuron, this is known as the activation function [1].

The ANNs have been used successfully in various fields of knowledge, since through training the network, complex relationships between inputs and outputs are achieved by adjusting the weights of the connections between neurons is accomplished. Therefore, the training of a neural network can be seen as an optimization process whose objective is to find a set of weights that minimizes the error produced by the network on the training data set [1, 2].

2.2. Extreme learning machine

Many papers have been developed to obtain fast and precise algorithms in the supervised training of single layer feedforward network (SLFN), being the extreme learning machine (ELM) one of the most used for its computation speed [3, 4]. The ELM algorithm for an SLFN network with m neurons can be expressed mathematically as the Equation (1).

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

where $(\mathbf{x}_i,\ y_i)$ is the set of n differents samples ($1 \le i \le n$, with $\mathbf{x}_i \in \mathbb{R}^p$ and $y_i \in \mathbb{R}$), β_i output weights, f activation function, $\mathbf{w}_i$ the weights of the hidden layer y b_i the biases. The Equation (1) can be written in matrix form $\mathbf{H}\boldsymbol{\beta} = \mathbf{y}$ (Equation (2)).

$$\mathbf{H} = \begin{bmatrix} f(\mathbf{w}_1\mathbf{x}_1 + b_1) & \cdots & f(\mathbf{w}_m\mathbf{x}_1 + b_m) \\ \vdots & \cdots & \vdots \\ f(\mathbf{w}_1\mathbf{x}_n + b_1) & \cdots & f(\mathbf{w}_m\mathbf{x}_n + b_m) \end{bmatrix}, \quad \boldsymbol{\beta} = (\beta_1, ...\beta_m)^T, \quad \mathbf{y} = (y_1, ..., y_n)^T. \tag{2}$$

The matrix H (Equation (2)) is called the hidden layer output matrix of the neural network; the ith column of H is the ith hidden neuron's output vector with respect to inputs $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n$. The ELM calculate the output weights of the SLFN network performing the following steps:

(i) Initialize randomly $\mathbf{w}_i$ y b_i, $i = 1, 2, 3, ..., m$.
(ii) The ANN is expressed as $\sum_{i=1}^{m} \beta_i f(\mathbf{w}_i \mathbf{x}_j + b_i) = y_j, \quad 1 \le j \le n$.
(iii) Calculated the output weights by means of $\boldsymbol{\beta} = \mathbf{H}^\dagger \mathbf{y}$,

where $H^\dagger$ is the Moore–Penrose inverse of H and corresponds to the solution of ordinary least squares for a regression problem. The ELM starts the weights of the hidden layer randomly, this produces an extremely fast training algorithm compared to learning machines such as the Multilayer Perceptron (MLP) and the support vector machine (SVM) [5]. However, the ELM networks require many neurons in the hidden layer to find good models.

In recent years, improvements to the ELM algorithm have been proposed under the imperfection situation. They can be grouped into two categories: incremental and pruning strategies. For incremental strategy, the model begins with a small initial network and then gradually adds new hidden nodes until the desired accuracy is achieved. Some notable incremental algorithms are: I-ELM [4], EI-ELM [6], B-ELM [7], EB-ELM [8]. For pruning strategy, the model begins with a larger network than necessary and then cuts the redundant or less effective hidden nodes. One notable destructive algorithms is the OP-ELM [9].

Traditional ELM training algorithms compute the output weights through the Moore-Penrose inverse (pseudoinverse). In this work, it is shown that the estimation error with the pseudoinverse is increased as the noisy data increases. This is why using the heuristic optimization algorithms: simulated annealing and swarm of particles in the determination of the output weights to minimize the error. The algorithms are described in section 2.3 and section 2.4.

2.3. Simulated annealing

The algorithm simulated annealing (SA) is inspired by metal annealing, where means of heating and controlled cooling of a material it is possible to increase the size of its crystals and reduce its defects. Annealing simulation can be used to find an approximation of a global minimum for a function with many variables [10, 11].

The notion of slow cooling implemented in the simulated annealing algorithm is interpreted as a slow decrease in the probability of accepting worse solutions as the solution space is explored. In general, the simulated annealing algorithm at each time step randomly selects a solution close to the current one, the algorithm measures its quality and then decides to pass or keep the solution. During the search, the temperature decreases progressively from an initial positive value to zero and affects both probabilities: at each step, the probability of moving to a better solution remains at 1 or changes to a positive value; On the other hand, the probability of moving to a non optimal solution progressively changes towards zero [11–14]. The steps for the simulated annealing algorithm are shown in the Algorithm 1.

Algorithm 1. Heuristic optimization process: simulated annealing [14].

```
Input: ProblemSize, $iterations_max$, $temp_max$
Output: $P_{gbest}$
$S_{current} \leftarrow$ CreateInitialSolution (ProblemSize);
$S_{best} \leftarrow S_{current}$;
for $i = 1$ to $iterations_{max}$ do
    $S_i \leftarrow$ CreateNeighborSolution ($S_{current}$);
    $temp_{curr} \leftarrow$ CalculateTemperature ($i, temp_{max}$);
    if $Cost\ (S_i) \leq Cost\ (S_{current})$ then
        $S_{current} \leftarrow S_i$;
        if $Cost\ (S_i) \leq Cost\ (S_{best})$ then
            $S_{best} \leftarrow S_i$;
        end
    else if $Exp\left(\frac{Cost(S_{current}) - Cost(S_i)}{temp_{curr}}\right) > Rand()$ then
        $S_{current} \leftarrow S_i$;
    end
end
end
Return $S_{best}$
```

2.4. Particle swarm optimization

The heuristic of particle swarm optimization (PSO) was developed by Kennedy and Eberhart when studying the social transport observed in swarms of insects and fish banks; Such social behavior is based on the transmission of the experience of each individual to the others in the group, which results in a synergistic process that allows individuals to move in a search space in an optimal way according to their needs. PSO heuristics has proven very efficient in solving optimization problems with rapid convergence rates [14–16].

Basically, the PSO heuristic consists of an iterative algorithm based on a population of individuals called swarm, in which each individual, called a particle, moves in the decision space in search of optimal solutions. The movements of the particles are guided by their own experience in the search space as well as by the best position known throughout the swarm. When better positions are discovered, these will guide the swarm movements. The process is repeated and in doing so an optimal solution is eventually discovered.

The pseudocode for PSO optimization process is shown in the Algorithm 2, for which the following assumptions are taken into account: $Population_{size}$ is the number of particles in the swarm, each with a position $P_{position} \in \mathbb{R}^n$ and a speed $P_{velocity} \in \mathbb{R}^n$ in the search space; P_{pbest} the experience where the best position of each particle was obtained and P_{gbest} the best position of the all swarm.

```
Algorithm 2. Heuristic optimization process: particle swarm [14].
Input: ProblemSize, $Population_{size}$
Output: $P_{gbest}$
Population $\leftarrow \emptyset$;
$P_{gbest} \leftarrow \emptyset$;
for $i = 1$ to $Population_{size}$ do
    $P_{velocity} \leftarrow$ RandomVelocity();
    $P_{position} \leftarrow$ RandomPosition($Population_{size}$);
    $P_{cost} \leftarrow$ Cost($P_{position}$);
    $P_{pbest} \leftarrow$ ($P_{position}$);
    if $P_{cost} \leq P_{gbest}$ then
        $P_{gbest} \leftarrow P_{pbest}$;
    end
end
while $\neg StopCondition()$ do
    foreach $P \in Population$ do
        $P_{velocity} \leftarrow$ UpdateVelocity ($P_{velocity}, P_{gbest}, P_{pbest}$);
        $P_{position} \leftarrow$ UpdatePosition ($P_{position}, P_{velocity}$);
        $P_{cost} \leftarrow$ Cost ($P_{position}$);
        if $P_{cost} \leq P_{pbest}$ then
            $P_{pbest} \leftarrow P_{position}$;
            if $P_{cost} \leq P_{gbest}$ then
                $P_{gbest} \leftarrow P_{pbest}$;
            end
        end
    end
end
Return $P_{gbest}$
```

3. Materials and methods
Matlab software is used for the implementation of the monolayer neural network. Training and test data for the network is generated with the Matlab rand command around a polynomial function. The robustness validation against the noise of the SA and PSO heuristic algorithms is performed by studying the error as a function of on the noisy data. In addition, different scenarios for the data are considered increasing the degree of the polynomial.

The following steps describe the process to the construction of noisy data:

(i) A polynomial function $y = p(x)$ is defined, where x is the input vector and y is the required output.

(ii) With the instruction $R = r\left(2\,\mathrm{rand}\left(\left[\begin{array}{cc}\mathrm{length}(x) & 1\end{array}\right]\right) - 1\right)$ a R vector of random numbers with uniform distribution between $-r$ and r is generated, it of the same size of the vector x; r defines the radius of the noise in the data.

(iii) With previous steps the output with noise $T = y + R$ is obtained.

3.1. Extreme learning machine algorithm and heuristic optimization processes

For the training of the neural network 400 data are used, of which 70% are randomly chosen for the training and the remaining 30% for the test. Then the output weights of the network are calculated with the pseudoinverse and with the optimization algorithms SA and PSO with cost function the mean square error. Finally the errors in the test of each of the methods are compared.

4. Results

The following parameters are used to obtain the results: 15 neurons and 400 data generated around a polynomial of degree 50 (50 descriptors) with a noise of radius 0.1 and 0.3. Table 1 shows the root mean square error (RMSE) in the test and the execution time in determining the weights of the hidden layer of the neural network as indicative of the performance of the inverse of Moore-Penrose and the optimization algorithms SA and PSO. The table shows that using noisy data of radius 0.1 the pseudoinverse delivers good models in a low time, however when the noise radius is increased to 0.3 the pseudoinverse loses accuracy. Nevertheless, the proposed algorithms make better approximations to the real curves using noisy data of radius 0.3, having a greater cost in time.

Table 1. Performance results of the Moore-Penrose inverse and the proposed optimization algorithms AS and PSO for the determination of the output weights using noisy data of radius 0.1 and 0.3.

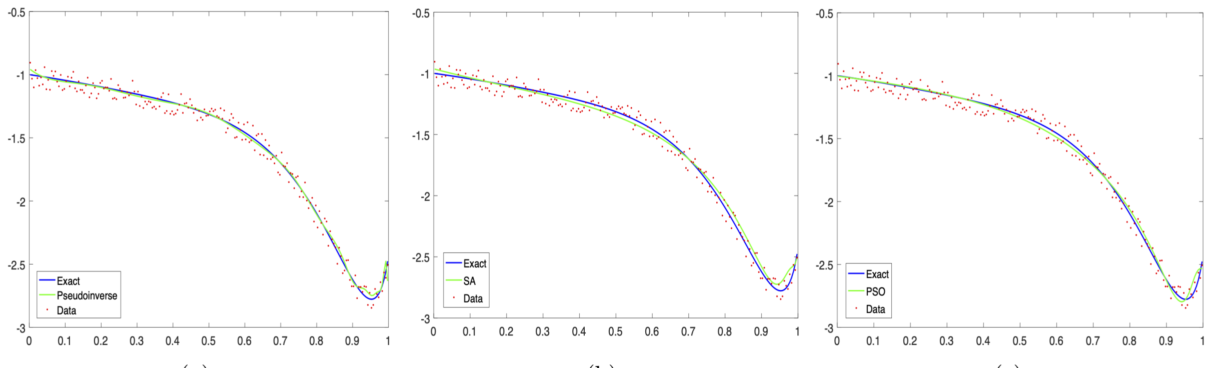
Noise radius	Pseudoinverse		Simulated annealing		Particle swarm	
	Time (s)	RMSE	Time (s)	RMSE	Time (s)	RMSE
0.1	0.017	0.020	3.654	0.035	0.402	0.025
0.3	0.019	0.066	6.706	0.041	4.348	0.036

The test behavior for each of the optimization processes is shown in the Figure 1 and the Figure 2. For a noise of radius 0.1 the three methods have good approximations as shown in Figure 1(a), Figure 1(b) and Figure 1(c), however the pseudoinverse generates a better fit according to Figure 1(a). For a noise of radius 0.3, the SA and the PSO have good approximations to the real model as shown in Figure 2(b) and Figure 2(c), but the model generated by computing the output weights of the ELM algorithm with the pseudoinverse begins to oscillate as shown in Figure 2(a), which leads to obtaining unreliable models.

The above shows the efficiency of the heuristic algorithms for particular cases (noise radius 0.1 and 0.3), now a general study is done of the behavior of the RMSE in function of noise radius for each of the optimization processes. The Figure 3 corresponds to the evolution of the RMSE, where it is observed that for low noisy data (noise radius less than 0.2) the pseudoinverse generates good models given that it presents the least error. However, it loses precision as the noise increases. In the SA and PSO optimization algorithms the growth of the RMSE as a function of noise is significantly lower compared to the pseudoinverse.

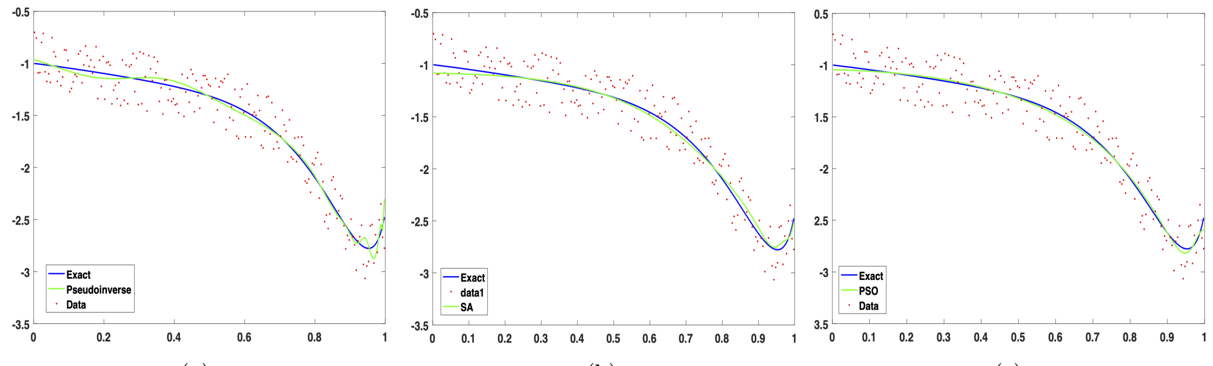
The foregoing shows that the ELM algorithm with the pseudoinverse has good results as long as the data have little noise. Though a better the ELM performance to highly noisy data can be improved using the SA and PSO heuristic algorithms.

Now the behavior of the error in the test is analyzed when the number of descriptors changes (varying the degree of the polynomial) for a noise of radius 0.2 since that is where the psudoinverse loses precision. The Figure 4 shows a bar chart corresponding to the test error as a function of the number of descriptors, where it is observed that the pseudoinverse tends to produce a greater error.



(a) (b) (c)

Figure 1. The test behavior of each optimization process in the calculation of the output weights of the ELM algorithm for a noisy data of radius 0.1. (a) pseudoinverse, (b) simulated annealing, and (c) particle swarm.



(a) (b) (c)

Figure 2. The test behavior of each optimization process in the calculation of the output weights of the ELM algorithm for a noisy data of radius 0.3. (a) pseudoinverse, (b) simulated annealing, and (c) particle swarm.

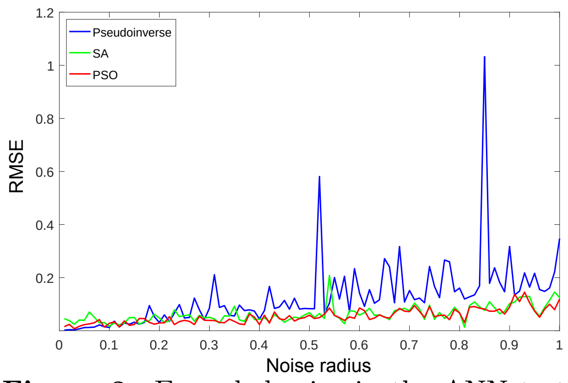


Figure 3. Error behavior in the ANN test as a function of noise.

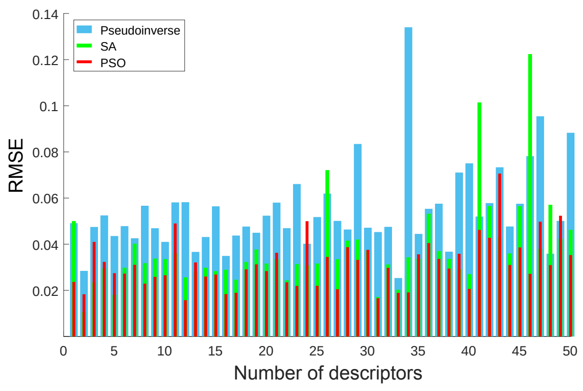


Figure 4. RMS as a function of the descriptors number.

5. Conclusion

The classic ELM algorithm using Moore Penrose inverse loses precision for highly noisy data, however, by implementing the SA and PSO heuristic optimization algorithms, the robustness of ELM is achieved for noisy data. Thus, different investigations can use this tool to obtain good models when noisy data is inevitable.

References

[1] Varela E and Campbells E 2011 Redes Neuronales Artificiales: Una revisión del estado del arte, aplicaciones y tendencias futuras *Revista Investigación y Desarrollo en TIC* **2(1)** 18

[2] Karayiannis N and Venetsanopoulos A 2013 Learning algorithms, performance evaluation, and applications *Artificial Neural Networks* vol 209 (New York: Springer Science+Business Media)

[3] Hornik K, Stinchcombe M and White H 1989 Multilayer feedforward networks are universal approximators *Neural Networks* **2(5)** 359

[4] Huang G B, Chen L and Siew C K 2006 Universal approximation using incremental constructive feedforward networks with random hidden nodes *IEEE Trans. Neural Networks* **17(4)** 879

[5] Zhang L, Zhang D and Tian F 2016 SVM and ELM: Who Wins? Object recognition with deep convolutional features from ImageNet *Proceedings of ELM-2015* **1** 249

[6] Huang G B and Chen L 2008 Enhanced random search based incremental extreme learning machine *Neurocomputing* **71(16-18)** 3460

[7] Yang Y, Wang Y and Yuan X 2012 Bidirectional extreme learning machine for regression problem and its learning effectiveness *IEEE Transactions on Neural Networks and Learning Systems* **23(9)** 1498

[8] Cao W, Ming Z, Wang X and Cai S 2019 Improved bidirectional extreme learning machine based on enhanced random search *Memetic Computing* **11(1)** 19

[9] Miche Y, Sorjamaa A, Bas P, Simula O, Jutten C and Lendasse A 2009 OP-ELM: Optimally pruned extreme learning machine *IEEE transactions on Neural Networks* **21(1)** 158

[10] Ranganathan S, Nakai K and Schonbach C 2018 *Encyclopedia of Bioinformatics and Computational Biology: ABC of Bioinformatics* (Cambridge: Elsevier)

[11] Khachaturyan A, Semenovskaya S and Vainshtein B 1979 Statistical-thermodynamic approach to determination of structure amplitude phases *Sov. Phys. Crystallography* **24(5)** 519

[12] Kirkpatrick S, Gelatt J and Vecchi 1983 Optimization by simulated annealing *Science* **220(4598)** 671

[13] Semenovskaya S V, Khachaturyan K A and Khachaturyan A G 1985 Statistical mechanics approach to the determination of a crystal *Acta Cryst.* **A41** 268

[14] Brownlee J 2011 *Clever algorithms: Nature-inspired Programming Recipes* (Autralia: Jason Brownlee)

[15] Eberhart R and Kennedy J 1995 Particle swarm optimization *Proceedings of ICNN'95 - International Conference on Neural Networks* (Perth: IEEE) 1942

[16] Eberhart R, Shi Y and Kennedy J 2001 *Swarm Intelligence* (San Diego: Academic Press)