Multi Mother Wavelet Neural Network based on Genetic Algorithm for 1D and 2D Functions’ Approximation

Mejda Chihaoui, Wajdi Bellil
Faculty of Sciences, University of Gafsa, 2112, Gafsa, Tunisia
mejda.chihaoui@gmail.com, wajdi.bellil@ieee.org

Chokri Ben Amar
REGIM: REsearch Group on Intelligent Machines
National School of Engineers (ENIS), University of Sfax, Sfax, Tunisia
chokri.benaamar@ieee.org

Keywords: Multi mother wavelet neural network, gradient descent, functions’ approximation, genetic algorithms.

Abstract: This paper presents a new wavelet-network-based technique for 1D and 2D functions’ approximation. Classical training algorithms start with a predetermined network structure which can be either insufficient or overcomplicated. Furthermore, the resolutions of wavelet networks training problems by gradient are characterized by their noticed inability to escape of local optima. The main feature of this technique is that it avoids both insufficiency and local minima by including genetic algorithms. Simulation results are demonstrated to validate the generalization ability and efficiency of the proposed Multi Mother Wavelet Neural Network based on genetic algorithms.

1 INTRODUCTION

Wavelet neural networks (WNN) (Daubechies, 1992), (Zhang and Benveniste, 1992) have recently attracted great interest, thanks to their advantages over radial basis function networks (RBFN) as they are universal approximators. Unfortunately, training algorithms start with a predetermined network structure for wavelet networks (predetermined number of wavelets). So, the network resulting from learning applied to predetermined architecture is either insufficient or complicated.

Besides, for wavelet network learning, some gradient-descent methods are more appropriate than the evolutionary ones in converging on an exact optimal solution in a reasonable time. However, they are inclined to fall into local optima.

The evolutionist algorithms bring in some domains a big number of solutions: practice of networks to variable architecture (Withleyet, 1990), automatic generation of Boolean neural networks for the resolution of a class of optimization problems (Gruau and Whitley, 1993).

Our idea is to combine the advantages of gradient descent and evolutionist algorithms.

In the proposed approach, an evolutionary algorithm provides a good solution.

Then, we apply a gradient-descent method to obtain a more accurate optimal solution.

The wavelet networks trained by the algorithm have global convergence, avoidance of local minimum and ability to approximate band-limited functions.

Simulation results prove that the proposed initializations’ approach reduces the wavelet network training time and improves the robustness of gradient-descent algorithms.

This paper is structured in 4 sections. After a brief introduction, we present in section 2 some basic definitions as well as initialization problems of wavenet: we focalized on two algorithms, the initialization step and the update one based on gradient-descent. In section 3, we provide the proposed approach to solve these problems and we present some results and tables achieved from the application of our new approach in 1D and 2D functions’ approximation in the last section (section 4).
2 MULTI MOTHER WAVELET NEURAL NETWORK FOR APPROXIMATION

2.1 Theoretical Background

Wavelets occur in family of functions, each one is defined by dilation $a_i$ which controls the scaling parameter and translation $t_i$ which controls the position of a single function, named mother wavelet $\psi(x)$. Wavelets are mainly used for functions’ decomposition.

Decomposing a function in wavelets consists of writing the function as a pondered sum of functions obtained from simple operations (translation and dilation) and performed on a mother-wavelet.

Let's suppose that we only have a finished number $N_w$ of wavelets $\Psi_j$ gotten from the mother wavelet.

$$f(x) = \sum_{j=1}^{N} w_j\psi_j(x)$$

We can consider the relation (1) as an approximation of the function $f$. The wavelet network has the following shape (Zhang, 1997):

$$\hat{y} = \sum_{j=1}^{N} w_j\Psi_j(x) + \sum_{k=0}^{N} a_kx_k$$

with $x_0 = 1$ (2)

Where $\hat{y}$ is the network output, $N_w$ is the number of wavelets, $W_j$ is the weight of $W_n$ and $x = \{x_1, x_2, \ldots, x_N\}$ the input vector, it is often useful to consider, besides the wavelets decomposition, that the output can have a refinement component, coefficients $a_k$ ($k=0,1,\ldots,N_i$) in relation to the variables.

A WN can be regarded as a function approximator which estimates an unknown functional mapping: $y = f(x) + \varepsilon$, where $f$ is the regression function and the error term $\varepsilon$ is a zero-mean random variable of disturbance. There are several approaches for WN construction (Bellil, Othmani and Ben Amar, 2007): Every wavelet has different dilations following different inputs. This choice has the advantages of enriching the library, and offering a better performance for a given wavelets number. The drawback introduced by this choice concerns the library size. A library with several wavelets families is more voluminous than the one that possesses the same wavelet mother. It needs a more elevated calculation cost during the selection stage.

On the other hand, the resolutions of wavelet networks training problems by gradient are characterized by their noticed inability to escape of local optima (Michalewicz, 1993) and in a least measure by their slowness (Zhang, 1997).

We propose a genetic algorithm which provides a good solution. Then, we apply a gradient-descent method to obtain a more accurate optimal solution.

In this paper, genetic algorithm provides a good solution to these problems.

First, this algorithm will reduce the library dimension. Second, this algorithm will initialize the descent gradient in order to avoid local minima.

2.2 Training Algorithms

The wavenet learning algorithms consist of two processes: the self-construction of networks and the minimization error.

In the first process, the network structures applied to representation are determined by using wavelet analysis (Lee, 1999).

In the second process, the parameters of the initialized network are updated using the steepest gradient-descent method of minimization.

Therefore, the learning cost can be reduced. Classical training algorithms have two problems: in the definition of wavelet network structure and in update stage.

2.2.1 Initialization Problems

First, we must note that initialization step is so necessary: that if we have a good initialization, the local minimum problem can be avoided, it is sufficient to select the best regressions (the best based on the training data) from a finished set of regressors.

If the number of regressors is insufficient, not only some local minima appear, but also, the global minimum of the cost function doesn't necessarily correspond to the values of the parameters we wish to find.

For that reason, in this case, it is useless to put an expensive algorithm to look for the global minimum. With a good initialization of the network parameters the efficiency of training increases. As we previously noted, classical approaches begin often with predetermined wavelet networks. Consequently, the network is often insufficient.

After that, new works are used to construct a several mother wavelets families library for the network construction (Bellil, Othmani and Ben Amar, 2007): Every wavelet has different dilations following different inputs. This choice has the advantages of enriching the library, and offering a better performance for a given wavelets number.

The drawback introduced by this choice concerns the library size. A library with several wavelets families is more voluminous than the one that possesses the same wavelet mother. It needs a more elevated calculation cost during the selection stage.

On the other hand, the resolutions of wavelet networks training problems by gradient are characterized by their noticed inability to escape of local optima (Michalewicz, 1993) and in a least measure by their slowness (Zhang, 1997).

We propose a genetic algorithm which provides a good solution. Then, we apply a gradient-descent method to obtain a more accurate optimal solution.

In this paper, genetic algorithm provides a good solution to these problems.

First, this algorithm will reduce the library dimension. Second, this algorithm will initialize the descent gradient in order to avoid local minima.
2.2.2 Novel wavelet networks architecture

The proposed network structure is similar to the classic network, but it possesses some differences; the classic network uses dilation and translation versions of only one mother wavelet, but new version constructs the network by the implementation of several mother wavelets in the hidden layer.

The objective is to maximize the potentiality of selection of the wavelet (Yan and Gao, 2009) that approximates better the signal. The new wavelet network structure with one output $y$ can be expressed by equation (3). We consider wavelet network (Bellil et al., 2007):

$$\hat{y} = \sum_{i=1}^{N_l} \sum_{j=1}^{M_i} \omega_{ji} \psi_{ji}(x) + \sum_{k=0}^{N_i} a_k x_k$$

(3)

where $\hat{y}$ is the network output and $x = \{x_1, x_2, ..., x_{N_i}\}$ the input vector; it is often useful to consider, in addition to the wavelets decomposition, that the output can have a linear component in relation to the variables: the coefficients $a_k$ ($k = 0, 1, ... , N_i$).

$N_i$ is the number of selected wavelets for the mother wavelet family $\psi_j$.

The index $l$ depends on the wavelet family and the choice of the mother wavelet.

3 GENETIC ALGORITHMS

3.1 Network Initialization Parameters

Once $t_i$ and $d_i$ are obtained from the initialization by a dyadic grid (Zhang, 1997), they are used in computing a least square solution for $\omega$, $a$, $b$.

$$\begin{bmatrix} \psi_1(x_1) & \psi_{N_i}(x_1) & x_1 \\ \psi_2(x_1) & \psi_{N_i}(x_1) & x_2 \\ \vdots & \vdots & \vdots \\ \psi_{N_i}(x_1) & \psi_{N_i}(x_{N_i}) & x_{N_i} \end{bmatrix} \begin{bmatrix} \omega_i \\ a_k \\ b \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_{N_i} \end{bmatrix}$$

(4)

The variable $N_i$ represents the number of displacement pair’s data.

Using families of wavelets, we have a library that contains $N_i$ wavelets. To every wavelet $\psi_{ji}$ we associate a vector whose components are the values of this wavelet according to the examples of the training sequence. We constitute a matrix that is constituted of $V_{Mw}$ of blocks of the vectors representing the wavelets of every mother wavelet where the expression is:

$$V_{Mw} = \{V_{ji}\}_{i=1}^{N_i} \times \{j=1, ..., M_i\}$$

The $V_{Mw}$ matrix is defined as follows:

$$V_{Mw} = \begin{bmatrix} v_{j1}(x_1) & v_{j1}(x_2) & \cdots & v_{j1}(x_{N_i}) \\ v_{j2}(x_1) & v_{j2}(x_2) & \cdots & v_{j2}(x_{N_i}) \\ \vdots & \vdots & \ddots & \vdots \\ v_{jM_i}(x_1) & v_{jM_i}(x_2) & \cdots & v_{jM_i}(x_{N_i}) \end{bmatrix}$$

(6)

3.2 Selection of the best Wavelets

The library being constructed, a selection method is applied in order to determine the most meaningful wavelet for modelling the considered signal. Generally, the wavelets in $W$ are not all meaningful to estimate the signal. Let’s suppose that we want to construct a wavelets network $g(x)$ with $m$ wavelets, the problem is to select $m$ wavelets from $W$.

The proposed selection is based on Orthogonal Least Square (OLS) (Titsias and Likas, 2001), (Colla, Reyneri and Sgarbi, 1999).

3.3 Change of the library dimension

3.3.1 Crossover Operators

This algorithm used two crossover operators.

One of them changes the number of columns of chromosome so it changes the number of mother wavelets and introduce in the library a new version of wavelets issued of the new mother wavelet.

The second operator does not change the number of columns of each chromosome.

3.3.1.1 The Crossover1 Operator

After the selection of the two chromosomes to which we will apply this operator, we choose an arbitrary position $a$ in the first chromosome and a position $b$ in the second according to $a$. After that, we exchange the second parts of the two chromosomes.

Figure 1: Crossover1 operator.
3.3.2 The Crossover2 Operator
For the second operator, we choose an arbitrary position c in the first chromosome and a position d in the second chromosome according to c.

Let Min_point= Min (c,d). First, we change the values of c and d to Min_point. Then, we exchange the second parts of the two chromosomes.

In this case, we have necessarily first children having the same length as the second chromosome and the second children having the same length as the first chromosome.

![Crossover2 operator](image)

3.3.3 Mutation Operator
Generally, the initial population does not have all the information that is essential to the solution. The goal of applying the mutation operator is to inject some new information (wavelets) into the population.

Mutation consists in changing one or more gene(s) in chromosome chosen randomly according to a mutation probability pm. Nevertheless, the muted gene may be the optimal one therefore, the new gene will not replace the old but it will be added to this chromosome.

3.4 Change of settings wavelets
In this step, we have a uniform crossover operator and a mutation operator applied to structural parameters of WN (translations and dilatations).

3.4.1 Uniform Crossover Operator
Let T = (t₁, t₂, ..., tₙ) the vector representing the translations: A coefficient is chosen and a vector T = (t₁', t₂', ..., tₙ') is constructed as follows:

\[ t'_1 = \alpha t_1 + (1 - \alpha) t_2 \]
\[ t'_2 = \alpha t_2 + (1 - \alpha) t_1 \]

(7) (8)

Where \( \alpha \) is a real random value chosen in [-1 1]. The same operator is applied to the vector dilation D.

3.4.2 Mutation Operator
After crossing, the string is subject to mutation. We consider the optimal wavelet, we reset the network with these wavelets that will replace the old in the library and the optimization algorithm will be continued using new wavelets.

Finally, after N iterations, we construct a network of wavelets \( N_w \) wavelet layer which hides the approximation signal Y. As a result, the network parameters are:

\[ y'^{opt} = \left[ y^{opt}_{pert} \right]_{pert}^{opt} = [1...N_w] \]

\[ y'^{pert} = \left[ y^{pert}_{pert} \right]_{pert}^{opt} = [1...N_w] \]

The model \( f(x) \) can be written as:

\[ g(x) = \sum_{i=0}^{N_w} a_i^{opt} * y_{pert}^{opt} + \sum_{i=0}^{N_w} a_i x_i \]

(10)

The proposed algorithm is resumed in this figure:

![Chart of genetic algorithm](image)

4 EXPERIMENTS AND RESULTS
In this section, we present some experimental results of the proposed Multi Mother Wavelet Neural Networks based genetic algorithm (MMWNN-GA) on approximating 1D and 2D functions.

First, simulations on 1D function approximation are conducted to validate and compare the proposed algorithm with some others wavelets neural networks.

The input \( x \) is constructed by the uniform distribution, and the corresponding output \( y \) is functional of \( y = f(x) \).

Second, we approximate four 2D functions using MMWNN-GA and some others wavelets networks...
to illustrate the robustness of the proposed algorithm.

We compare the performances using the Mean Square Error (MSE) defined by:

$$\text{MSE} = \frac{1}{M} \sqrt{\sum_{m=1}^{M} (\hat{f}(x) - y)^2}$$

(11)

Where $\hat{f}$ is the network output.

### 4.1 1D approximation using the initialization by genetic Algorithm

We want to rebuild three signals $F_1(x)$, $F_2(x)$ and $F_3(x)$ defined by equations (12), (13) and (14).

$$F_1(x) = \begin{cases} 
-2.186x-12.864 & \text{for } x \in [-10,-2] \\
4.246x & \text{for } x \in [-2,0] \\
10\exp(0.05x-0.5)\sin(x(0.03x+0.7)) & \text{for } x \in [0,10] 
\end{cases}$$

(12)

$$F_2(x) = 0.5x\sin(x) + \cos(x) \text{ for } x \in [-2.5,2.5]$$

(13)

$$F_3(x) = \text{sinc}(1.5x) \text{ for } x \in [-2.5,2.5]$$

(14)

Table 1 gives the MSE after 100 training for classical and multi-mother wavelet network and only 40 iteration for MMWWN-GA algorithm.

The best approximated functions $F_1$, $F_2$ and $F_3$ are displayed in Figure 4.

For $F_1$, the MSE of Mexhat is 1.39e-2, comparing to 4.20e-3 for MMWWN-GA.

Beta2 approximates $F_2$ with an MSE equal to 9.25e-7 where the MSE using the MMWWN-GA is equal to 1.01e-10. Finally, the MSE is 4.58e-6 for MMWWN comparing to 2.72e-6 for MMWWN-GA.

Table 1: Comparison between CWNN, MMWNN and MMWWN-GA in term of MSE for 1D functions approximation.

<table>
<thead>
<tr>
<th>Function</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nb of wavelets</td>
<td>8</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>CWNN (100 iterations)</td>
<td>Mexhat</td>
<td>1.39e-2</td>
<td>2.64e-5</td>
</tr>
<tr>
<td></td>
<td>Pwog1</td>
<td>4.70e-2</td>
<td>2.63e-5</td>
</tr>
<tr>
<td></td>
<td>Slog1</td>
<td>2.08e-3</td>
<td>3.70e-6</td>
</tr>
<tr>
<td></td>
<td>Beta1</td>
<td>1.93e-2</td>
<td>9.24e-7</td>
</tr>
<tr>
<td></td>
<td>Beta2</td>
<td>1.92e-2</td>
<td>9.25e-7</td>
</tr>
<tr>
<td></td>
<td>Beta3</td>
<td>1.93e-2</td>
<td>1.39e-5</td>
</tr>
<tr>
<td>MLWN (100 iterations)</td>
<td>3.46e-4</td>
<td>8.81e-11</td>
<td>4.58e-6</td>
</tr>
<tr>
<td>MMWWN-GA (40 iterations)</td>
<td>4.20e-3</td>
<td>1.01e-10</td>
<td>2.72e-6</td>
</tr>
</tbody>
</table>

From these simulations we can deduce the superiority of the MMWWN-GA algorithm over classical WNN and MLWN in term of 1D functions’ approximation as they much reduce the number of gradient iterations since initialization step has already achieved acceptable results.

### 4.2 2D Approximation

The Table 2 gives the final mean square error after 100 training for classical networks and multi-mother wavelet network and only 40 iteration for MMWWN-GA 4 levels decomposition to approximate some 2D functions ($S_1$, $S_2$, $S_3$ and $S_4$ given on figure 5).

$$S_1(x, y) = e^{-\frac{1}{30}(x+y)^2 + (1-y)^2}$$

(15)

$$S_2(x, y) = \left(\frac{2+x}{2}\right)\left(\frac{2-x}{2}\right)\left(\frac{y+2}{2}\right)\left(\frac{2-y}{2}\right)$$

(16)

$$S_3(x, y) = \frac{3.2(0.25+\cos(5.4y))}{(6+6(3x-1))^2}$$

(17)

$$S_4(x, y) = (x^2 - y^2)\sin(5x)$$

(18)
Table 2: Comparison between the MSE of CWN, MMWNN and MMWNN-GA in term of 2D functions approximation.

<table>
<thead>
<tr>
<th>Function</th>
<th>Nb of wavelets</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
</tr>
</thead>
<tbody>
<tr>
<td>CWN (100 iterations)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mexh</td>
<td>17</td>
<td>2.58e-3</td>
<td>1.00e-2</td>
<td>4.93e-2</td>
<td>1.05e-2</td>
</tr>
</tbody>
</table>
Pwog      | 19            | 4.06e-3 | 1.73e-2 | 4.94e-2 | 1.08e-3 |
|Slog1     | 14            | 2.31e-3 | 2.60e-3 | 4.88e-2 | 6.37e-3 |
|Beta1     | 9             | 4.22e-3 | 1.44e-2 | 4.63e-2 | 2.25e-4 |
|Beta2     |               | 2.80e-3 | 6.19e-3 | 4.56e-2 | 3.94e-4 |
|Beta3     |               | 4.23e-3 | 6.19e-3 | 4.65e-2 | 4.85e-4 |

MLWNN (100 iterations) | 3.49e-7 | 1.50e-5 | 2.54e-4 | 8.4e-3 |

MMWNN-GA (40 iterations) | 8.48e-7 | 5.78e-7 | 7.89e-4 | 4.86e-3 |

From table 2, we can see that MLWNN-GA are more suitable for 2D function approximation then the others wavelets neural networks.

For example we have an MSE equal to 8.4877e-7 to approximate the surface S1 using MLWNN-GA after 40 iterations over 2.5803e-3 if we use the Mexican hat wavelet after 100 iterations.

The MMWNN approximates S2 with an MSE equal to 1.50e-5 where the MSE using the MLWNN-GA is 5.78e-7.

For S3, the MSE is equal to 4.65e-2 for Beta3 WNN comparing to 7.89e-4 for MLWNN-GA.

Finally, Slog1 approximates S4 with MSE equal to 6.37e-3 comparing to 4.86e-3 with MMWNN-GA.

5 CONCLUSIONS

In this paper, we presented a genetic algorithm for the design of wavelet network.

The problem was to find the optimal network structure and parameters. In order to determine the optimal network, the proposed algorithms modify the number of wavelets in the library.

The performance of the algorithm is achieved by evolving the initial population and by using operators that alter the structure of the wavelets library.

Comparing to classical algorithms, results show significant improvement in the resulting performance and topology.

As future work, we propose to combine this algorithm with GCV (Othmani, Bellil, Ben Amar and Alimi, 2010) to optimize the number of wavelets in hidden layer.

REFERENCES


