A New Algorithm for Structure Optimization of Wavelet Neural Network

Youssef Harkouss[1], Walid Fahs[2] and Mohammad AYACHE[2]

[1]Rafic Hariri University Campus-Faculty of Engineering-Branch III
P.O.Box 14, 6573, Al Hadath, Beirut, lebanon

[2]Islamic University of Lebanon, Faculty of Engineering
Khaldeh, Lebanon

Abstract

This paper presents a new algorithm for constructing and training wavelet neural network. This algorithm is based on the variation of the number of hidden neurons dynamically during the training process. The suggested method determines the optimal number of the hidden neurons and solves the optimization problem of wavelet neural network structure. The problem of finding a good neural model is then discussed through solutions offered by wavelet neural networks trained by conjugate gradient algorithm. Finally, experimental results, which confirm the efficiency of this approach, are reported.

Keywords: *Neural Network, Wavelet, Conjugate Gradient, Regressor, Local Minimum, Hidden Neuron.*

1. Introduction

Artificial neural networks are computational models with particular characteristics such as the ability of learning, classifying, and modeling data. They seem to be a highly attractive solution, since their internal configuration and the natural parallelism inherent to those network architectures allow a higher performance than the conventional models. Many studies have been reported on the ability of feedforward neural networks for approximating nonlinear functions [1, 2]. The implementation of such networks suffers from the lack of efficient constructive method, needed to determine the neuron's parameters, as well as choosing network's architecture. Multilayer perceptrons (MLP) present a large class of feedforward neural networks [5]. An MLP is a static network with a forward direction of signal flow and without feedback loops, and is usually constructed with sigmoid neurons and trained with the back-propagation (BP) algorithm. However, due to its multilayered structure and the greedy nature of the BP algorithm, the training process often settles in undesirable local minimum of the error surface and converges too slowly. The constructing process of the MLP is very time consuming since the optimal number of hidden neurons is not known in advance, and it must be determined by trial and error. For those reasons, we propose in this paper a new algorithm for constructing and training wavelet neural network (WNN). The idea of using wavelet in neural network was proposed by Zhang and Benveniste [3]. Based on wavelet theory, WNN possesses the best function approximation ability. Since the constructing model algorithm is different from the common neural network BP algorithm, it can effectively overcome intrinsic defect in the common neural network. WNN has been proposed as an efficient universal tool for function approximation, which shows surprising efficiency in solving the poor convergence or even divergence encountered in other kinds of neural networks. It can dramatically increase the convergence speed. The structure of the WNN is almost similar to the RBF (Radial-basis Function) neural network [5] except that in the WNN, the radial basis functions are replaced by radial wavelets. The WNN performs well in comparison with the MLP and RBF networks [4]. In addition to the salient feature of approximating any non-linear function, WNN outperforms MLP and RBF networks due to its capability in dealing with the so-called curse of dimensionality and non-stationary signals. The WNN is more suitable in learning functions with local variations and discontinuities. However, the optimal number of hidden neurons of a conventional WNN is not known in advance. To solve this problem, we proceeded by changing the number of hidden neurons dynamically, that is when the network is trapped in a local minimum, a new hidden neuron is then added. Computer simulations demonstrate the advantages of the proposed algorithm. Many studies have been reported on the ability of wavelet neural networks to approximate nonlinear functions [6, 7].

Wavelet neural networks are widely applied to a variety of practical problems such as signal processing [4], modeling and design of microstrip circuits, embedded passive components, semiconductor devices, filters, and amplifiers [8, 9].

This paper is structured as follows: in section 2, we present a review of the WNN and its learning scheme. Our construction method of WNN model is described in section 3. In Section 4, we demonstrate the validation of the proposed algorithm by modeling two nonlinear functions. Section 5 presents a discussion of the main features of the novel approach. Finally in section 6, the conclusions of this study are reported.

2. Wavelet Neural Network and Learning Scheme: a Brief Review

2.1 Wavelet neural network

Interest in wavelet analysis has been grown very rapidly in recent years. The wavelet theory is a rapidly developed branch of mathematics, which has offered very efficient algorithms for approximating, estimating, analyzing and compressing nonlinear functions [10]. Due to the similarity between discrete inverse wavelet transform and one-hidden-layer feedforward neural networks (i.e. three-layer network), the idea of combining both wavelets and neural networks has been proposed by Q. Zhang and A. Benveniste [3]. A WNN is an adaptive discretization of the continuous inverse wavelet transform. It can also be considered as an one-hidden-layer feedforward neural network with radial wavelets as activation functions of its hidden neurons. A block diagram of a multiple-input-one-output feedforward wavelet network is shown in Figure 1.

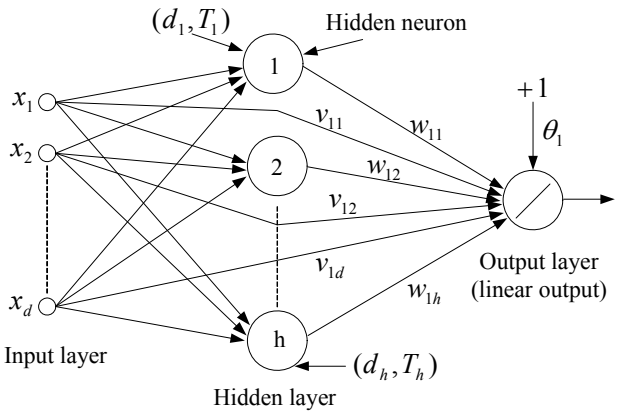

Fig. 1 Structure of wavelet neural network (q = 1).

Since large dimensional wavelet transforms lead to a large number of coefficients, it is useful to introduce some simple ways for building multi-dimensional wavelets from one-dimensional ones. The most natural ones seem to be the radial form. Given a radial wavelet function $\psi : R^d \to R$, the expression of the kth ($1 \le k \le q$) output of the WNN can be written as:

$$(WNN(X))_k = \sum_{i=1}^{h} w_{ki}\psi_i(X) = \sum_{i=1}^{h} w_{ki}\psi(d_i(X - T_i)) \qquad (1)$$

Where q is the number of linear output neurons. In practice, this equation is modified as follows:

$$(WNN(X))_k = \sum_{i=1}^{h} w_{ki}\psi(d_i(X - T_i)) + V_k^T X + \theta_k \qquad (2)$$

where $\psi(.)$ is the mother radial wavelet function, $w_{ki} \in R$ is the linear weight between ith hidden neuron and kth output neuron, $\theta_k \in R$ is the bias parameter of the kth linear output neuron, $V_k \in R^d$ is the vector that represents the direct linear connection between input layer and kth linear output neuron, $X \in R^d$ is the input vector of network, $d_i \in R$ is the dilation parameter and $T_i \in R^d$ is the translation vector of the ith hidden neuron. *h* is the number of hidden neurons in the network. All the network parameters (w_{ki}, V_k, θ_k, d_i, T_i) must be adapted on the training data. The linear part of $(WNN(X))_k$, $V_k^T X + \theta_k$, is used in order to more easily capture linear properties in regressions. In this work, the mother radial wavelet function ψ has been chosen as:

$$\psi(X) = \left(\|X\|^2 - d\right) e^{-\frac{\|X\|^2}{2}} \qquad (3)$$

the so-called "Inverse of the Mexican hat". $\|X\|$ denotes the Euclidean norm of X and $d = \dim(X)$.

2.2 Learning algorithm

After several experiments using the WNN coupled with different training algorithms (i.e. classical BP (steepest gradient descent) technique, conjugate gradient (CG) algorithm, Newton method, etc), we have chosen in this paper, the efficient WNN coupled with the CG algorithm. The WNN is trained with one of the CG algorithms [13], in order to minimize the mean squared error (MSE) between the desired output Y^i and the WNN output:

$$MSE = \frac{1}{2N}\sum_{i=1}^{N}\sum_{k=1}^{q}(y_k^i - (WNN(X^i))_k)^2 \quad (4)$$

where X^i is the ith input vector of network, N is the number of samples in the training data set Θ_N ($\Theta_N = \{(X^i, Y^i) \in R^d \times R^q, i = 1,...,N; N < \infty\}$), and (X^i, Y^i) is the ith sample of the inputs and desired outputs of network.

The good initialization of the WNN (see section III.A) yields a fast training procedure. In this application, CG algorithm is clearly superior to the classical BP approach and is thus preferred for the WNN training. Detailed comparisons show that the CG algorithm is faster than classical BP technique and it is superior to the quasi-Newton optimization algorithm when the number of parameters in the network is large [14, 15]. During the training process the WNN adjusts automatically its parameters (i.e. w_{ki}, V_k, θ_k, d_i, T_i) so that the error MSE is minimized. The nth correction of the parameters is described as:

$$W_{n+1} = W_n + \eta D_n \quad (5)$$

where

$$D_n = -\nabla E(W_n) + \alpha_n D_{n-1} \quad (D_0 = -\nabla E(W_0)) \quad (6)$$

and

$$\alpha_n = \frac{\|\nabla E(W_n)\|^2}{\|\nabla E(W_{n-1})\|^2} \quad \text{(Fletcher-Reeves formula)} \quad (7)$$

E is the total error between the desired output (training data) and the actual output, it is expressed as:

$$E = N \times MSE \quad (8)$$

η is the learning parameter and W_n is the vector which contains all the parameters of the WNN at the nth iteration.

3. Construction of Wavelet Neural Network

3.1 Regressors-selection based approach

The association of an efficient training algorithm, such as the CG algorithm, with the initialization procedure based on the regressors selection technique, represents an efficient tool to construct WNN models. In this section we describe the technique of regressors selection "*stepwise selection by orthogonalization*" proposed by Q. Zhang [4] and used to initialize the conventional WNN. This approach consists of:

1. construct a library W of discrete dilated and translated versions of a given mother radial wavelet $\psi : R^d \to R$. This library is constructed according to the available training data set Θ_N. The dilations and translations of the mother radial wavelet function induce a multiresolution analysis [11, 12]. The construction process of this library is described below.

The wavelet library W is considered as a set of regressor candidates. It should contain a finite number of wavelets. Given a wavelet function ψ, the construction of W consists in selecting some subset of the continuously parameterized family:

$$\{\psi(d(X - T)):\ d \in R_+, T \in R^d\} \quad (9)$$

The standard discretization of this family is a regular lattice:

$$\{\psi(d_0^l X - mt_0):\ l \in Z, m \in Z^d\} \quad (10)$$

where d_0, $t_0 > 0$ are two scalar constants defining the discretization step sizes for dilation and translation, respectively. Typically we take a dyadic grid for the sake of simplicity and we choose $d_0 = 2$ and $t_0 = 1$. Usually we only want to estimate a multidimensional nonlinear function on a compact domain $D \subset R^d$ and the wavelet function ψ is chosen to be compactly supported or rapidly vanishing. Hence we can replace in (10) $m \in Z^d$ by $m \in S_t$ with a finite set $S_t \subset Z^d$; on the other hand, $l \in Z$ should be replaced by $l \in S_d$ with a finite set $S_d \subset Z$ corresponding to the desired resolution levels of the estimation. l is then called the number of scale levels scanned during the construction phase of the wavelet library. In practice, 4 or 5 consecutive levels are usually sufficient. In this case, family (10) becomes:

$$\{\psi(d_0^l X - mt_0):\ l \in S_d, m \in S_t\} \quad (11)$$

In the next step, we should scan the training data set Θ_N, for each sample point (X^i, Y^i) ($i = 1...N$) we should determine the wavelets in (11) whose supports contain the sample point. We denote by W the set of wavelets

resulting from the refining of family (11) and call it the wavelet library. For computational convenience the wavelets are normalized:

$$W = \{\psi_i : \psi_i(X) = \alpha_i \psi(d_i(X - T_i)), \quad \alpha_i = \left(\sum_{k=1}^{N} [\psi(d_i(X^k - T_i))]^2 \right)^{-1/2}, \ i = 1,...,L \} \quad (12)$$

where X^k is the kth sample of the inputs in Θ_N, L is the number of wavelets in the library W, and d_i, T_i correspond respectively to the dilation and translation parameters d_0^l and $mt_0 d_0^{-l}$ of the wavelet ψ_i.

2. select the best regressors (i.e. wavelets) from this library in an iterative way which they best fit the training data set Θ_N. The aim of the algorithm 'stepwise selection by orthogonalization' is then to select a number $h \leq L$ wavelets from W, the best ones based on the training data set Θ_N for building the regression $WNN(X)$ (i.e. equation of WNN). The problem is then to find h that minimizes the error function MSE.

For convenience of presentation, let us define some vectorial notations:

$$u_i = \left(\psi_i(X^1) \ \ldots \ \psi_i(X^N)\right)^T \quad (13)$$

where $\psi_i \in W$ and $X^1,...,X^N$ are the sampled inputs in Θ_N.

Now collect all the $u_i, i = 1,...,L$, in a set U:

$$U = \{u_1,...,u_L\} \quad (14)$$

The problem of regressor selection is equivalent to selecting h vectors u_i from U that minimize the MSE. The problem is then to select, for the first stage, the wavelet in W that best fits the training data set Θ_N, and then repeatedly select the wavelet that best fits the set Θ_N while working together with the previously selected wavelets. When the initialization procedure of the conventional WNN is achieved, this network is then trained by the CG algorithm. The choice of hidden neurons number h (i.e. number of model parameters) depends on the complexity of the multidimensional nonlinear function, as well as the requested accuracy for the approximation. In this construction procedure, h must be determined by trial and error. h is determined according to the evolution of the error function MSE during the initialization procedure. In the next section, we present a new algorithm, which can be used to determine the optimal number of hidden neurons during the training procedure.

3.2 Construction procedure using the proposed optimization approach

The flowchart of the proposed algorithm is shown in Figure 2. At first, once we have the wavelet library, we select the best regressor (i.e. wavelet) from this library. Thus, the WNN contains initially one hidden neuron, which makes the learning process trapped sometimes in a local minimum, and the network cannot produce then the required response.

The error function MSE defined as a function of the free parameters of the network WNN. This function may be visualized as a multidimensional error-performance surface or simply error surface, with the network parameters as coordinates. Any given operation of the network WNN during the training process is represented as a point on the error surface. For the WNN, to improve performance over time, the operating point has to move down successively toward a minimum point of the error surface; the minimum point may be a local minimum or a global minimum. On the other hand, the peculiarity of the error surface that impacts the network's performance is the presence of local minima where the values of the error function MSE are not acceptable. In the parameter space there is another set of network parameters for which the error function MSE is smaller than the local minimum there where the network is stuck.

Therefore, in our algorithm, a new hidden neuron is to be added each time the network becomes trapped in a local minimum. The addition of this new hidden neuron to the WNN changes the number of network parameters and creates a new WNN model (i.e. creates a new set of network parameters). This causes variations in the shape of the parameter space so that the network can easily avoid the local minimum. In this case, the training process then proceeds because the shape of the parameter space is changed. We must, of course, consider when and how a new hidden neuron should be added. It is usually not clear whether the network is trapped in a local minimum that is why the MSE is used instead. Therefore, MSE is checked after every n parameter corrections. We choose $n = 10$, n must not have a high value because the construction process of the WNN could become less faster. If the error MSE does not decrease by more than one percent of its previous value, new hidden neuron is then added from the wavelet library (one percent is sufficient in order to know if the error continues to decrease). Otherwise, if MSE decreases by more than one percent, the parameters will be corrected another n times. We considered that the network have converged when the MSE is less than a

predetermined value ε. We tested the proposed algorithm for different values of n (10, 20, 30, 40, 50, 100,…). We noticed that if n increases, the construction phase gives the same results but this phase becomes slow.

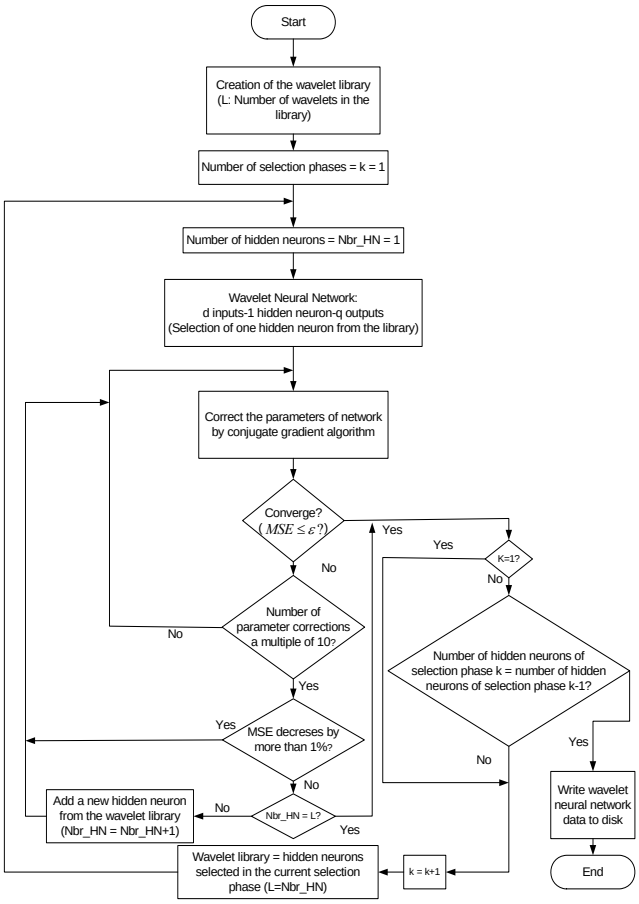

Fig. 2 Flowchart of the proposed algorithm.

If the network is trapped in a local minimum after 10 parameter corrections, then this network cannot avoid this local minimum after 20, 30, 40, 50, or 100 parameter corrections. And if the error surface is a very flat plateau, we could not know the necessary time in order to reach during the learning process a point on the error surface where the MSE begins to decrease, and many iterations may be required to produce a significant reduction in the error performance of the network. We found that $n = 10$ is sufficient in order to know if we could change the shape of the error surface (i.e. in order to know if we could add a new hidden neuron) or not and to obtain a faster construction procedure of the WNN model.

To determine the value of ε we used the method of cross-validation [5]. It is a commonly used method for evaluating the performance of WNN models where the data set is partitioned into two subsets: a subset used for the estimation of the WNN model (called the training set) and another subset used for the evaluation of the generalization performance of the WNN model (called the validation set or testing set). The CG algorithm is considered to have converged when the MSE reaches a sufficiently small error threshold ε. The learning process is then stopped when the training performance has peaked. At the end of the learning process, the network WNN is tested for its generalization performance measured on the test data set. If the generalization performance is satisfactory then the building process of the WNN is finished. The initial values of the added hidden neuron parameters are determined as follows: The dilation parameter d_j and the translation vector T_j of the jth added hidden neuron are selected from the wavelet library. In this case, the expression of the kth output of the WNN can be written as:

$$\begin{aligned}(WNN(X))_k &= \sum_{i=1}^{j} w_{ki}\psi_i(X) + V_k^T X + \theta_k \\ &= \sum_{i=1}^{j} w_{ki}\psi(d_i(X - T_i)) + V_k^T X + \theta_k\end{aligned} \tag{15}$$

The linear coefficients w_{ki}, $V_k = (v_{k1}, v_{k2}, ..., v_{kd})^T$ and θ_k are determined by the least squares solution of the system of linear equations:

$$O_k = A.C_k \text{ and } C_k = A^+.O_k = \left(\left(A^T A\right)^{-1} A^T\right) O_k \tag{16}$$

where $C_k = \left(w_{k1}, w_{k2}, ...w_{kj}, v_{k1}, v_{k2}, ..., v_{kd}, \theta_k\right)^T \in R^{j+d+1}$ is the vector that contains the linear coefficients,

$$A = \begin{pmatrix} \psi_1(X^1) ..... \psi_j(X^1) & x_1^1 ..... x_d^1 & 1 \\ . & . & . \\ . & . & . \\ . & . & . \\ \psi_1(X^N) ..... \psi_j(X^N) & x_1^N ..... x_d^N & 1 \end{pmatrix} \in R^{N \times (j+d+1)},$$

$X^i = \left(x_1^i, x_2^i, ..., x_d^i\right)^T \in R^d$ is the ith input vector of network in Θ_N, $O_k = \left(y_k^1, y_k^2, ..., y_k^N\right)^T \in R^N$ is the desired output of the kth output neuron, and A^+ is the pseudoinverse of the matrix A.

At the end of the first selection phase, the WNN contains $h_1 \le L$ wavelets selected from the library W and that best fit the training data set Θ_N. In order to know if the number of selected wavelets is optimal, we

introduced a new selection phase. The wavelet library W_1 in this phase contains the wavelets selected in the previous selection phase and that best fit the data set Θ_N. This library W_1 contains of course the wavelets of optimal WNN. The new selection phase allows to select $h_2 \le h_1$ vectors u_i from $U_1 = \{u_{l_1},...,u_{l_{h_1}}\}$ that minimize the MSE where $l_1,...,l_{h_1}$ are the indices of wavelets previously selected from W ($l_1,...,l_{h_1} \in \{1,...,L\}$). During this phase, if the threshold ε has been reached with h_1 ($h_2 = h_1$) wavelets then the optimal network WNN will contain h_1 hidden neurons, else the optimal WNN will contain $h_3 \le h_2$ ($h_2 < h_1$) hidden neurons and that will best fit the data set Θ_N. These neurons will be selected during a new selection phase. This procedure is repeated as long as the number of hidden neurons in a selection phase is less than the number of hidden neurons selected in the previous selection phase. Note that the optimal number of hidden neurons is the smaller number of hidden neurons that gives $MSE = \varepsilon$.

4. Validation of the Proposed Algorithm: Modeling Examples

In this section, we present experimental results of the proposed approach on approximating two nonlinear functions. The run of our neural network software is executed on a 2 GHz Pentium PC.

4.1 Approximation of the heart beat function

In this section, we present experimental results of the proposed approach for approximation of the real nonlinear function, which is: beat of the heart. First, no noisy data is considered and the training data set is composed of 70 samples ($N = 70$, $d = 1$, $q = 1$). The learning parameter is $\eta = 0.1$. Figure 3 shows the variation of the hidden neurons number of the WNN during the selection phases. The wavelet library contains initially 128 wavelets. Figure 7 shows that the construction phase of the WNN is done with 6 selection phases. In the first selection phase, the threshold $\varepsilon = 3.67 \times 10^{-6}$ has been reached with 13 hidden neurons. These 13 neurons represent the wavelet library for the second selection phase. In the second phase, 10 hidden neurons have been selected, and the hidden neurons number has been decreased. For the third selection phase, the wavelet library contains 10 wavelets. During this phase, the number of hidden neurons has been again decreased and this number became equal to 7. The threshold ε is always equal to 3.67×10^{-6}. The numbers of hidden neurons determined by the fourth and fifth selection phases are 5 and 4, respectively. However, in the sixth selection phase the number of hidden neurons is 4; thus all the wavelets of the wavelet library containing 4 hidden neurons have been selected. Since the last two selection phases gave the same number of hidden neurons, therefore we can conclude that the optimal number of hidden neurons is obtained, and it is equal to 4 where $\varepsilon = 3.67 \times 10^{-6}$. The WNN obtained in the sixth selection phase, containing 4 hidden neurons, is the neural model of the non-noisy heart beat data. Figure 4 shows the results obtained with the WNN and the original non-noisy data (desired output). This figure shows the results computed from the training data set and the test data set. The test data set contains 25 samples and the calculated error on this set is $MSE = 5.44 \times 10^{-6}$.
The time of computation for building the WNN of the heart beat function is 33.21 seconds.
In figure 5 we present the results obtained at the output of different networks (i.e. MLP and conventional WNN) and computed from the training data set. In table 1 we listed the error MSE calculated at the end of the learning process. All networks are trained by the conjugate gradient algorithm. The number of hidden neurons used in each network is equal to 4 and the learning parameter is $\eta = 0.1$.

Table 1: Performance evaluation of different neural networks of the heart beat function.

Neural network	MSE
Proposed WNN	3.67×10^{-6}
MLP	1.28×10^{-2}
Conventional WNN	1.92×10^{-3}

Next, we apply our algorithm to the heart beat function, but in the case of noisy data. The training data set is composed of 129 samples. The learning parameter is $\eta = 0.1$. The construction procedure of the WNN of this function demands 4 selection phases (Fig. 6). The wavelet library contains initially 128 wavelets. In the first phase, we have reached the threshold $\varepsilon = 3.8 \times 10^{-6}$ with 33 hidden neurons. The wavelet library of the second phase contains the 33 hidden neurons previously selected, and at the end of this phase, 26 hidden neurons are selected. For the third selection phase, the wavelet library contains 26 wavelets. During this phase, the number of hidden neurons

has been again decreased and the number of hidden neurons became 16. During the fourth selection phase, we have selected all the neurons of the wavelet library, which contains 16 wavelets. The threshold ε is always equal to 3.8×10^{-6}. The WNN obtained by the fourth selection phase, and which contains 16 hidden neurons, represents the neural model of heart beat noisy data. Figures 7 and 8 show the original noisy data (desired output) and the reconstruction result obtained at the output of the WNN, respectively.

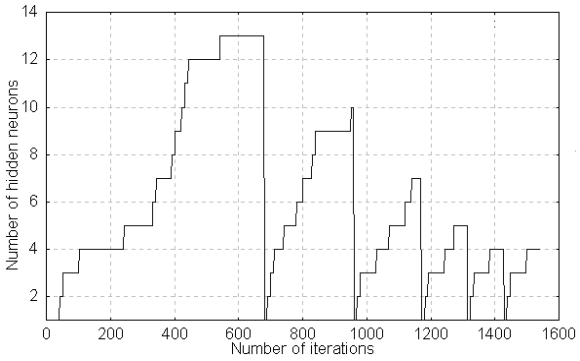

Fig. 3 Number of hidden neurons as a function of the number of iterations.

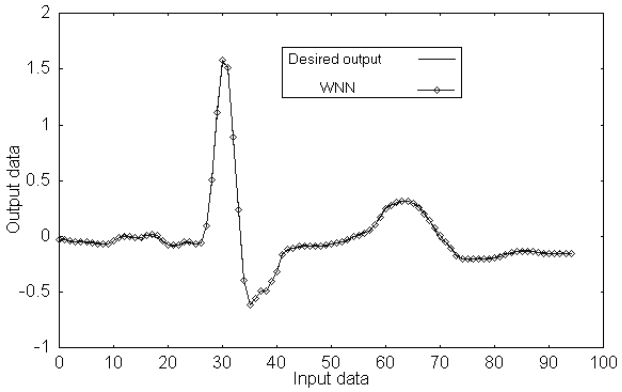

Fig. 4 The reconstruction result by the proposed WNN.

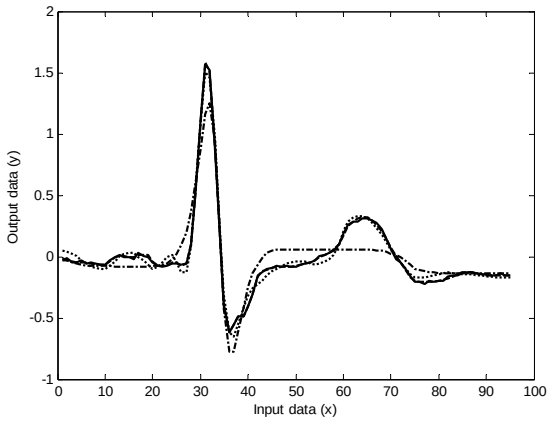

Fig. 5 Comparison between MLP, proposed WNN, and conventional WNN (desired output (solid line), proposed WNN (dashed line), conventional WNN (dotted line), MLP (dashdot line)).

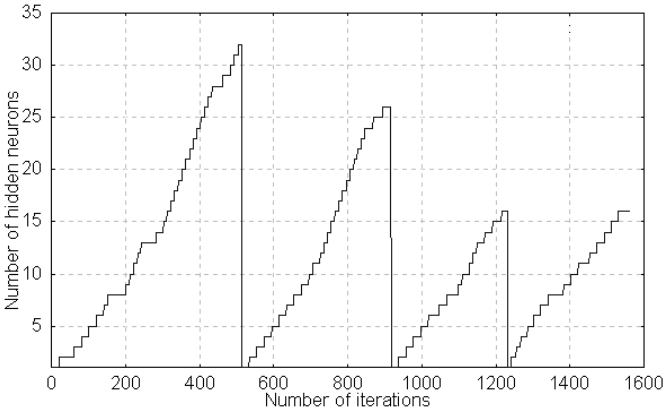

Fig. 6 Number of hidden neurons as a function of the number of iterations.

The results presented in this section have demonstrated the efficiency and the ability of using our WNN training technique in modeling noisy data and functions with local variations and discontinuities.

The time of computation for building the WNN of the noisy data is 41.56 seconds.

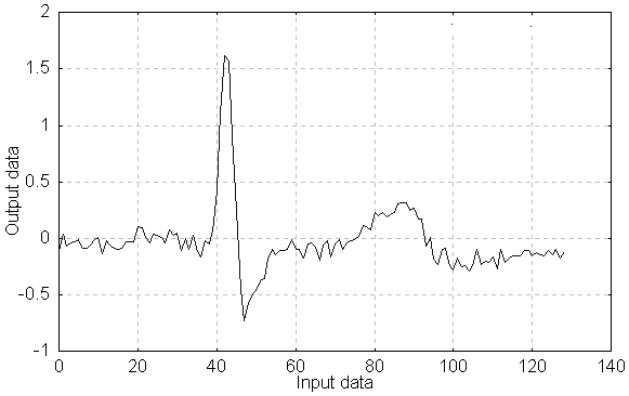

Fig. 7 The original noisy data of the heart beat.

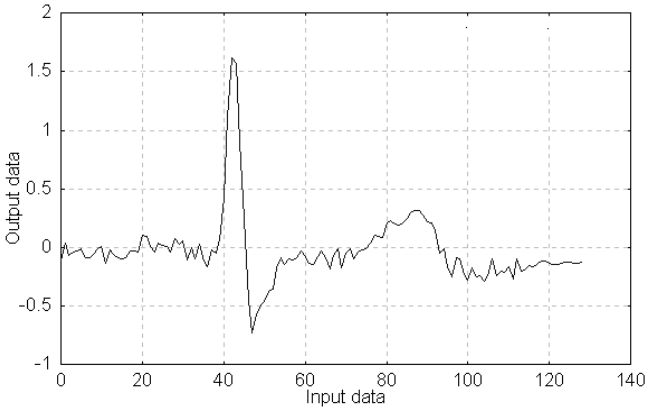

Fig. 8 The reconstruction result by the proposed WNN.

4.2 Approximation of two-dimension nonlinear function

Finally, we apply our algorithm to the two-dimension function:

$$f(x_1, x_2) = (x_1^2 - x_2^2)\sin(5x_1) \quad (17)$$

The training data set is composed of 441 samples ($N = 441$, $d = 2$, $q = 1$). The learning parameter is $\eta = 0.1$. Figure 9 shows the variation of the hidden neurons number of the WNN during the selection phases. The result shows that the number of selection phases is 4. The wavelet library contains initially 233 wavelets. We have selected for each phase of the 4 phases the following numbers of wavelets: 46 wavelets for the first phase, 33 for the second, 20 for the third, and also 20 for the fourth phase. The threshold ε is equal to 5.1×10^{-6}. The WNN obtained in the second phase, and which contains 20 hidden neurons, represents the neural model of the function f . Figures 10 and 11 show respectively the primary function f and the reconstructed result by the WNN. Figure 9 shows the results computed from the training data set and the test data set. The test data set contains 100 samples and the calculated error on this set is $MSE = 6.76\times10^{-6}$.

The time of computation for building the WNN of the nonlinear function f is 184.32 seconds. Figures 16 and 17 show the results obtained at the output of each network and computed from the training data. In table 2 we listed the error MSE calculated at the end of the learning process. All networks are also trained by the conjugate gradient algorithm.

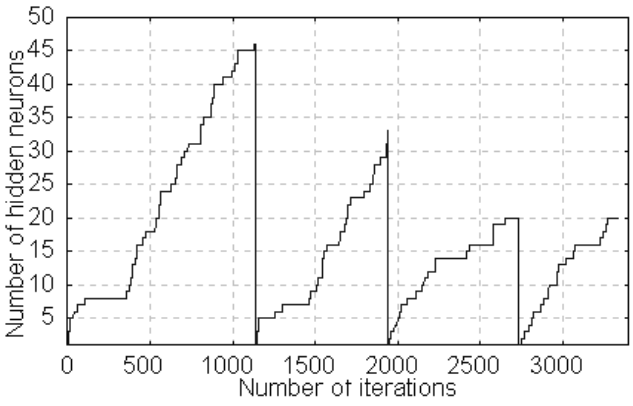

Fig. 9 Number of hidden neurons as a function of the number of iterations.

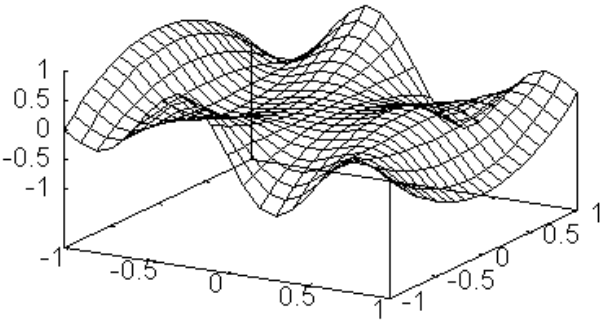

Fig. 10 The original function f .

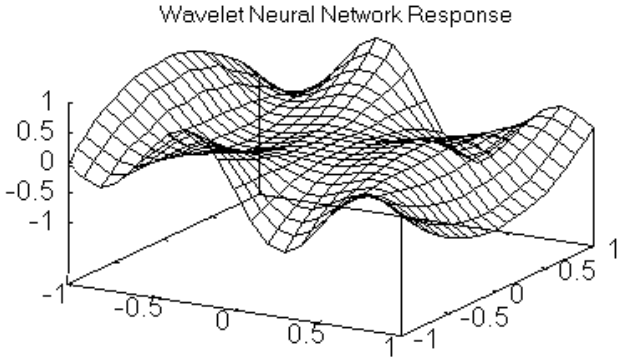

Fig. 11 The reconstruction result by the proposed WNN.

Table 2: Performance evaluation of different neural networks of f .

Neural network	MSE
Proposed WNN	5.1×10^{-6}
MLP	4.91×10^{-3}
Conventional WNN	1.37×10^{-4}

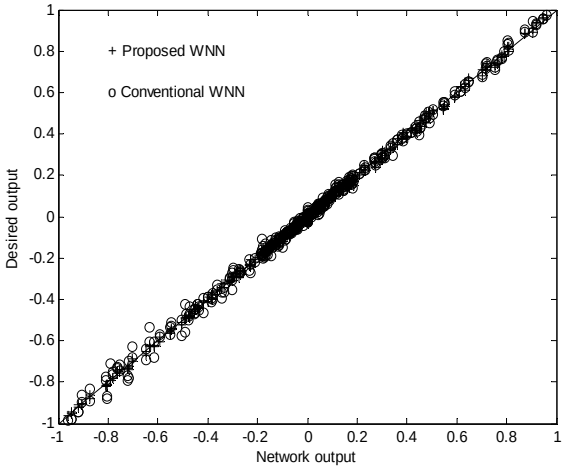

Fig. 12 Comparison between the proposed WNN and the conventional WNN.

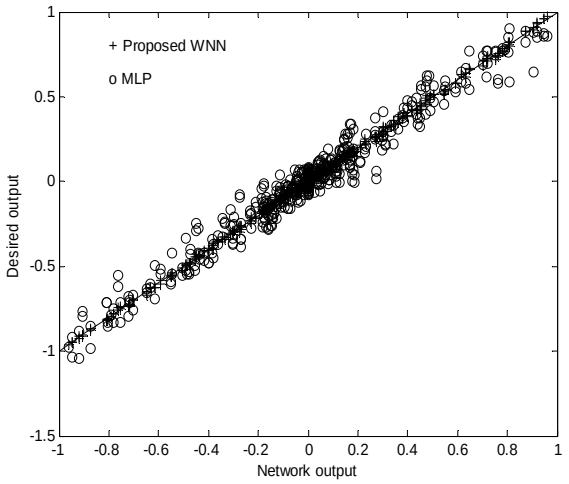

Fig. 13 Comparison between the proposed WNN and MLP.

5. Discussion

Experimental results have shown that the WNN constructed by our method has a smaller number of hidden neurons, a faster learning speed, and a smaller approximation error (MSE) in comparison with the other existing networks [3, 9]. However, in case of MLP (or conventional WNN), the number of hidden neurons is not known in advance, and the network architecture is usually determined by trial-and-error tests. Therefore, the construction method of MLP is very time consuming and in most cases, the optimal number of hidden neurons is not obtained. On the other hand, the proposed construction procedure of the WNN is very fast because the construction process of the wavelet library is based on the training data set, and the samples of this set are used to determine the different parameters of the library wavelets. There are several advantages of the proposed approach, for instance, it provides useful guidelines for the construction of an optimal WNN model and this model is more suitable in learning noisy data and functions with local variations and discontinuities, where the MLP (and RBF network) does not have the ability to model similar type of these functions.

The main difference of the proposed method from other methods is that originates from a wavelet frame with a multiscale structure. The proposed approach is presented for quick construction of WNN with a smaller MSE, and is more constructive in the sense that it automatically determines the network size and estimates the network parameters in a reasonable number of iterations.

It should be noted that the size of the neural network depends on the size of the training data set. These reflect the complexity of the problem. A small network is not able to solve the problem for any size of the training data set, but increasing the network size makes it difficult to optimally estimate a large number of parameters from the moderate-size training set. This type of network is considered as an overtrained network [5] that tends to memorize rather than to generalize from data. A neural network that is designed to generalize well will produce a correct input-output mapping, even when the input is slightly different from the examples used in the training set.

The proposed algorithm can be applied to model any nonlinear multidimensional function. The WNN modeling procedure offers an accurate mathematical equation representing the network and which can be lead to calculate the derivatives (first order derivative, second order derivative, …) of the network output with respect to its inputs.

6. Conclusion

We presented in this paper a new approach for structure optimization of WNN, based on the variation of the hidden neurons number during the training process. In this approach, there is no need to determine the number of hidden neurons by trial and error because the system adjusts them automatically. The proposed algorithm determines the optimal number of hidden neurons and solves the problem of the wavelet network structure optimization in a new way. Experimental results demonstrate its superiority over the construction procedure of the MLP and the conventional WNN. We used the proposed algorithm to model two nonlinear functions. The results presented in this study have demonstrated the efficiency and the ability of using a WNN model, implemented by the proposed approach, for efficient modeling of nonlinear multidimensional functions.

References

[1] A K. Hornik, M. Stinchcombe, and H. White, “Multilayer feedforward network are universal approximators”, Neural Networks, 2, 5, pp. 359-366, 1989.

[2] W A.R. Barron, “Neural net approximation”, in Procceding of the Seventh Yale Workshop on Adaptative and Learning Systems, pp. 69-72, 1992.

[3] Q. Zhang, and A. Benveniste, “Wavelets networks”, IEEE Trans. on Neural Network, 3, 6, pp.889-898, 1992.

[4] Q. Zhang, “Using wavelet network in non parametric estimation”, IEEE Trans. on Neural Networks, 8, 2, pp. 227-236, 1997.

[5] S. Haykin, “Neural networks: a comprehensive foundation”, Macmillan College Publishing Company, 2nd ed., New York, 1998.

[6] D.W.C. Ho, P.A. Zhang, and J. Xu, "Fuzzy wavelet networks for function learning", IEEE Trans. on Fuzzy Systems, 9, 1, pp. 200-211, 2001.
[7] H.L. Shashidhara, S. Lohani, and V.M. Gadre, "Function learning using wavelet neural networks", Proceedings of IEEE International Conference on Industrial Technology, pp. 335-340, 2000.
[8] Y. Harkouss, J. Rousset, H. Chéhadé, E. Ngoya, D. Barataud, and J.P. Teyssier, "The use of artificial neural networks in nonlinear microwave devices and circuits modeling: an application to telecommunication system Design", International Journal of RF and Microwave Computer-Aided Engineering-Special Issue on Applications of Artificial Neural Networks to RF and Microwave Design (Invited Paper), pp. 198-215, 1999.
[9] **Y.** Harkouss, E. Ngoya, J. Rousset, and D. Argollo, "Accurate radial wavelet neural-network model for efficient CAD modelling of microstrip discontinuities", IEE Proceedings – Microwaves, Antennas and Propagation, 147, 4, pp. 277-283, 2000.
[10] C. Chui, "WAVELETS: A tutorial in theory and applications", Academic Press, San diego, 1992.
[11] I. Daubechies, "Ten lectures on wavelets", Philadelphia: SIAM Press, 1992.
[12] S. Mallat, "Multiresolution approximation and wavelets orthonormal bases of $L^2(R)$", Trans Amer Math, 315, 1, pp. 69-88, 1989.
[13] C. Charalambous, "Conjugate gradient algorithm for efficient training of artificial neural networks", IEE PROCEEDINGS-G, 139, 3, pp. 301-310, 1992.
[14] A. Cichocki, and R. Unbehauen, "Neural networks for optimization and signal processing", John Wiley & Sons, Stuttgart, 1994.
[15] S. MCLOONE, and G.W. IRWIN, "Fast parallel off-line training of multilayer perceptrons", IEEE Truns. Neural Network, **8,** 3, pp. 646-653, 1997.

Youssef Harkouss received the Ph.D. degree in electronics from the University of Limoges, France, in 1998. From 1999 to 2000, he was Research Engineer at the CNRS, France. He joined the Lebanese University in 2001 as an Associate Professor. His research interests include advanced fuzzy neural networks software development, fuzzy neural network modeling and optimization for microwave devices and circuits, CAD of passive devices, and fuzzy neural networks for controlling dynamic systems.

Walid Fahs received the Ph.D degree in computer communication from the University of Blaise Pascal, France, in 2008. He is the coordinator of the department of Computer and Communication at the faculty of engineering at the Islamic University of Lebanon. His research interests include advanced neural networks software development and wireless networks.

Mohammad Ayache received the Ph.D degree in medical Image Processing from the University of Tours, France, in 2007. He is the coordinator of the department of biomedical at the faculty of engineering at the Islamic University of Lebanon. His research interests include advanced neural networks software development and advanced signal and image processing techniques.