Approximating I/O Data Using Wavelet Neural Networks: Control the Position of Mother Wavelet

Mohammed Awad
Faculty of Engineering and Information Technology, Arab American University, Palestine

Abstract: *In this paper, we deal with the problem of function approximation from a given set of input/output data. This problem consists of analyzing training examples, so that we can predict the output of the model given new inputs. We present a new method for function approximation of the I/O data using Wavelet Neural Networks (WNN). This method is based on a new efficient method of optimizing the position of a single function called mother wavelet of the WNN; it uses the objective output of WNN to move the position of wavelet single function. This method calculates the error committed in every mother wavelet area using the real output of the WNN trying to concentrate more mother wavelets in those input regions where the error is bigger, thus attempting to homogenize the contribution to the error of every mother wavelet, this method improves the performance of the approximation system obtained, compared with other models derived from traditional algorithms.*

Keywords: *Wavelet neural networks, function approximation, controls the position of mother wavelet.*

Received April 4, 2009; accepted January 3, 2010

1. Introduction

Function approximation is the name has given to a computational task that is of interest to many sciences and engineering communities [13]. Function Approximation consists of synthesizing a complete model from samples of the function and its independent variables [7]. In supervised learning, the task is that of learning a mapping from one vector space to another with the learning based on a set of instances of such mappings. We assume that a function *F* does exist and we endeavour to synthesize a computational model of that function. As a general mathematical problem, function approximation has been studied for centuries. However, some knowledge of the function to be approximated is usually assumed, depending on the specific problem. For example, in pattern recognition, a function mapping is made whose objective is to assign each pattern in a feature space to a specific label in a class space [1]. Neural network comes formed from multiple layers of nodes interconnected with activation function in each node and weight between these nodes and the output of neural network. The output of each node is not linear function for all its inputs and the network represents an extension of unknown nonlinear relation between input and output I/O. In represented space by the activation functions of network nodes, learning is seen as an approach of a multi-dimensional function. Normally the types of activation functions are: global and local. The global activation function is activated in a big variety of input values and provides a global approximation of data. The local activation functions are active only in the immediate nearness of the value of given input [15].

It is known that the functions can be represented as a sum of orthogonal basis functions. Such extensions can be easily represented as neural networks having the basis function selected as activation functions in every node, and the coefficients of extension as the weight of the output. Several classic orthogonal functions suffer, therefore, from the disadvantages of the approximation that uses global functions. What is necessary is a set of local basis functions and orthogonal [15]. Special class of functions, known as wavelet, possesses properties of good localization and they are orthonormal basis. Although, wavelet can be use as activation functions of a neural network called Wavelet Neural Network (WNN) [16]. The idea of using wavelets in neuronal networks has been proposed recently by Zhang and Benveniste [18] and Pati and Krishnaprasad [9].

Wavelet Neural Networks (WNNs) has recently attracted great interest, because they are universal approximations, it's achieved faster convergence than Radial Basis Function Neural Networks (RBFNN) and are capable to deal with the problems of "curse of dimensionality". In addition, WNN are generalized RBFNN [1, 5]. The structure of the WNNs is similar to the RBFNNs, except the radial basis function is replaced with ortonormal basis functions. The efficiency of this type of networks is in learning of the function and its evaluation [17]. Wavelet Neural Networks use a three-layer structure and wavelet activation functions in hidden layer. The structure of a WNN with output F(x), inputs p {x1, x2, …, xp} and j mother wavelet (Neuron) as given in Figure 1.

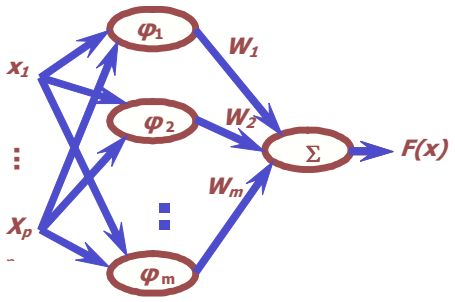


Figure 1. Wavelet neural network.

The WNNs possesses an unique attribute, besides the formation of an orthogonal basis they are also capable of explicit and represent the behaviour of a function in several resolutions of input variables [15]. The fundamental concept in the formulation and the design of WNN as basis function is the representation of multi-resolution of functions that use wavelet. This provides the essential frame for completely learning for WNN [15]. The wavelets are a family of functions where each one comes defined by one parameter of dilation A_j that controls the scaling parameter and translation B_j which controls the position of a single function, called mother wavelet $\varphi(x)$ [14]. The position of functions in a space of input data make that a WNN can reflect the properties of the function more exactly than the RBFNNs. Considering a set of learning of n-elements, The output of the WNN is given by the following expression:

$$F(\vec{x}) = \sum_{j=1}^{m} w_j \cdot \varphi_j \left(\frac{\vec{x} - B_j}{A_j} \right) \quad (1)$$

Where m the number of the mother wavelet nodes in the hidden layer, w_j the weight connecting the *jth* unit of the hidden layer to the output layer unit. $\varphi(x)$ is the wavelet activation function (mother wavelet) of *jth* unit of the hidden layer. In terms of wavelet transformation theory, mother wavelets given by the following form:

$$\varphi = \left| A_j \right|^{-1/2} \varphi \left(\frac{\vec{x} - B_j}{A_j} \right) \quad (2)$$

Where $x = \{x_1, x_2, \ldots, x_p\}$, $A_j = \{A_{j1}, A_{j2}, \ldots, A_{jp}\}$ and $B_j = \{B_{j1}, B_{j2}, \ldots, B_{jp}\}$ are a family of functions generated from one single function $\varphi(x)$. Note that for the p-dimensional input space, the mother wavelet can be calculated by the product of p single mother wavelets as follows [4]:

$$\varphi(\vec{x}) = \prod_{i=1}^{p} \varphi(\vec{x}_i) \quad (3)$$

A WNN can be considered as a function approximation which estimates an unknown functional using the following equation:

$$y = F(x) + \varepsilon \quad (4)$$

Where F is the regression function and ε is error term is a zero-mean random variable of disturbance [14]. The localization of the *jth* units of the hidden layer is determined by the scale parameter A_j and the translation parameter B_j: these parameters are modified as well as the weight parameters during the process of training. This flexibility simplifies more reliability towards optimum learning solution. According to other models derived from traditional algorithms, the two parameters A_j, B_j can either be predetermined based upon the wavelet transformation theory or be determined by a learning algorithms [4]. Note that the above WNN is a kind of basis function neural network in the sense of that the wavelet consists of the basis functions [4]. A learning method for a WNN is optimizing the translation B_j which controls the position of a single basis function. The WNN translations B_j can be optimize by many algorithms. The author in [10] used Adaptive Immune Genetic Algorithm (AIGA) to optimize the WNN structure and parameters (weights, dilation and translation), other algorithm applies Ant Colony Algorithm (ACA) with embedded deterministic searching strategy to optimize dilation factor, translation factor and output weight of WNN [3]. Other algorithm that we compare the result with is clustering for function approximation method which uses the cluster as mother wavelet of WNN [6]. This algorithm describes a technique specially designed for function approximation problems, analyzes the output variability of the target function during the optimization process and augments the number of prototypes in those input zones where the target function is more variable, increasing the variance explained by the approximation [6].

In this paper, we present a proposed method of optimization the translation B_j of WNNs for the function approximation problem, this method uses the target output of the WNN to migrate and fine-tune the translation instead of just the input values of the I/O data. This method calculates the error committed in every mother wavelet zone using the real output of the WNN trying to concentrate more mother wavelets in those input regions where the error is bigger, thus attempting to homogenize the contribution to the error of every mother wavelet.

The organization of the rest of this paper is as follows. Section 2 presents an overview of the proposed algorithm. In section 3, we present in details the proposed algorithm for the determination of the pseudo-optimal WNN parameters. Then, in section 4 we show some results that confirm the performance of the proposed methodology. Some final conclusions are drawn in section 5.

2. The Proposed Method

As mentioned before, the problem of function approximation consists of synthesizing a complete model from samples of the function and its independent variables. Consider a function $y = F(\vec{x})$ where $\vec{x}$ is a vector $(x_1,\ldots,x_p)$ in n-dimensional space from which there is available set of input/output data pairs. The idea is to approximate these data with another function $F(\vec{x})$. The accuracy of the approximation is generally measured by a cost function which takes into account the error between the output of the WNN and the target output. In this paper, the cost function we are going to use is the so-called Normalized Root Mean Squared Error (NRMSE). This performance index is defined as:

$$NRMSE = \sqrt{\sum_{i=1}^{P} (y_i - F(\vec{x}_i))^2 / \sum_{i=1}^{P} (y_i - \bar{y})^2} \quad (5)$$

Where $\bar{y}$ is the mean of the target output and p is the data number.

The objective of our algorithm is to increase the density of mother wavelets (Neurones) in the input domain areas where the error committed in every mother wavelets using the real output of the WNN is bigger. The WNN is completely specified by choosing the following parameters: the number m of mother wavelets (Neurones), the translation B_j of every mother wavelet, scale parameter A_j, and the weights w.

The number of mother wavelets (Neurones), is a critical choice. In our proposed algorithm we have used a simple incremental method to determine the number of mother wavelets. We will stop adding new mother wavelets when the error falls below a certain target error. In section 3, we present in detail the proposed algorithm for the determination of the pseudo-optimal WNN parameters.

3. Parameters Optimization of WNN

The locality property inherent to the mother wavelet of the WNN allows us to use algorithm to optimize the mother wavelet position (translation B_j). Many algorithms may get stuck in a local minimum ignoring a better placement of some of the translation B_j, i.e., the algorithm is trapped in a local minimum which is not the global one. Therefore we need an optimization algorithm capable to solve this local minimum problem. To avoid this problem we endow our supervised algorithm with a migration technique. This modification allows the algorithm to escape from local minimum and to obtain a Neurones allocation independent of the initial configuration. To optimize the other parameters of the WNN (scale parameter A_j, and the weights w) we used well-known heuristics; the K-Nearest Neighbour technique (KNN) [12] for the initialization of the scale parameter A_j, Singular Value Decomposition (SVD) [8] to optimize directly the weights. Finally, the Levenberg-Marquardt algorithm fine-tunes the obtained WNN [11].

Therefore, in this section we will concentrate on the proposed algorithm. In Figure 2, we show a flowchart representing the general description of our proposed algorithm. As seen from this figure, the initial values of the translation B_j are initialized randomly followed by a local displacement process which locally minimizes the Distortion (D) within each mother wavelet.

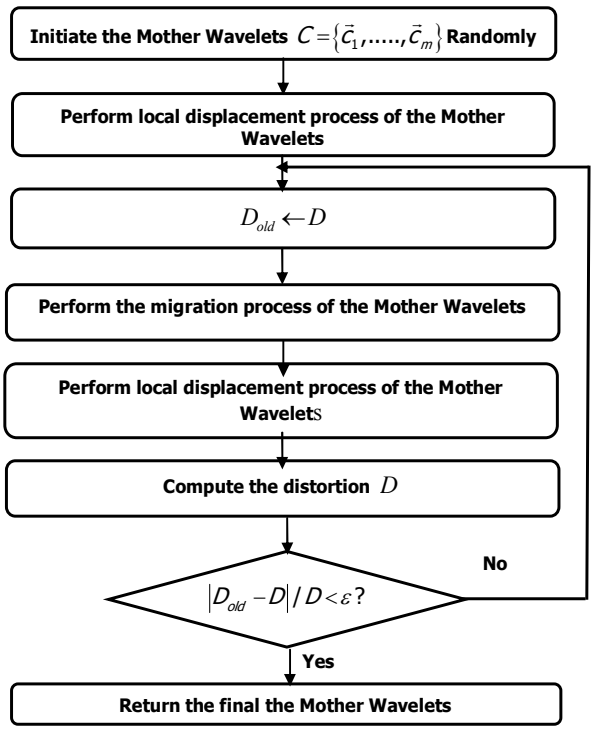


Figure 2. General description of the proposed algorithm.

The distortion is defined as:

$$D = \frac{\sum_{j=1}^{m} \sum_{\vec{x}_i \in C_j} \left\| \vec{x}_i - \vec{B}_j \right\|^2 E_{ij}}{\sum_{j=1}^{m} \sum_{\vec{x}_i \in C_j} E_{ij}} \quad (6)$$

Where m is the number of mother wavelets C_j in WNN, B_j is the translation of mother wavelet and E_{ij} is the error committed by the WNN when the input vector $\vec{x}_p$ belongs to mother wavelet as in the following equation:

$$E = \left| y - F(\vec{x}) \right| \quad (7)$$

In the local displacement of the position of mother wavelets C_j, we start by making a hard partition of the training set.

The second step of the process of local displacement is the calculation of the error of the WNN using the K-nearest neighbours algorithm to initiate the scale parameter A_j, and the singular value decomposition to calculate the weights of the mother

wavelets C_j. This is carried out by an iterative process that updates each mother wavelets position as the weighted mean of the training data belonging to those mother wavelets and we repeat this process until the total distortion of the WNN reaches a minimum as in the following equation:

$$\vec{B}_m = \sum_{\vec{x}_i \in C_m} E_{im} \cdot \vec{x}_i \; / \sum_{\vec{x}_i \in C_m} E_{im} \qquad (8)$$

After this process we must update the mother wavelets C_j position to minimize the total distortion. The algorithm stops when the value of the distortion is less than the value of a threshold ε. Figure 3 presents a flowchart with the general description of the local displacement process.

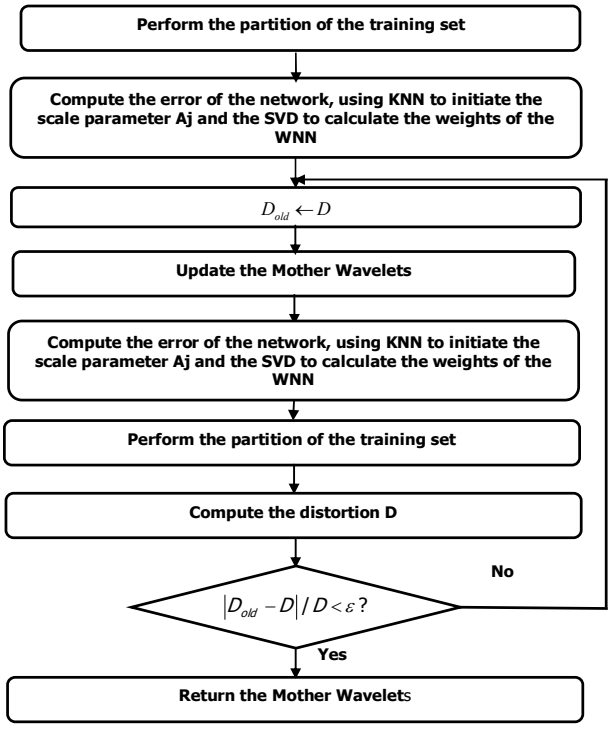


Figure 3. Local displacement of mother wavelets.

The migration process migrates mother wavelets from the better zones toward zones where the error is worse, attempting to make equal their contribution to the total distortion. Our main hypothesis is that the best initial mother wavelets configuration will be the one that equalizes the error committed by every mother wavelets. The probability of choosing a given mother wavelets inversely proportional to what we call "the utility" of those mother wavelets, which is defined as:

$$U_j = D_j / \bar{D} \quad j = 1, \ldots, m \qquad (9)$$

In this way, the proposed algorithm selects one mother wavelets that has the utility less than one and moves these mother wavelets to the zone nearby a new selected mother wavelets having utility more than one as shown as in Figure 4 This migration step is necessary because the local displacement of mother wavelets only moves mother wavelets in a local manner.

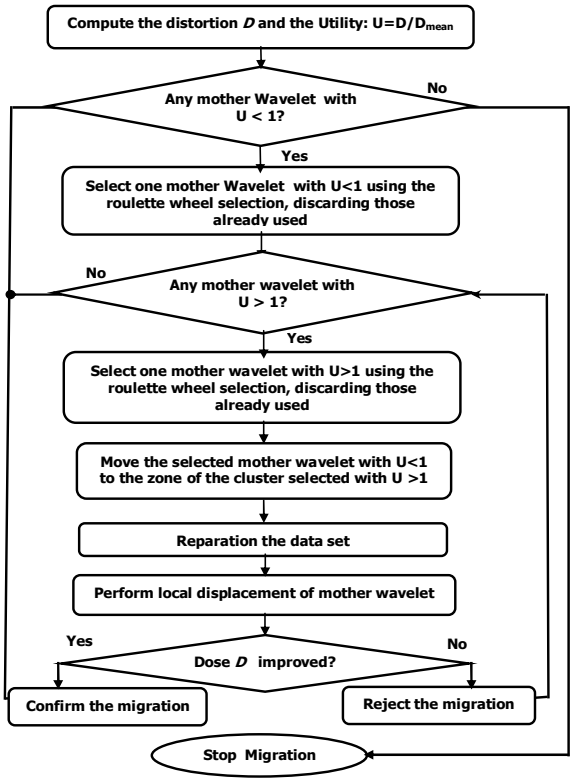


Figure 4. Migration process.

4. Experiments

Experiments have been performed to test the proposed algorithm. The system is simulated in MATLAB 7.0 under Windows XP with processor Pentium IV running at 2.4 Ghz. To test the effects caused by the proposed algorithm on the initialization and avoid local minimum on the placement of the mother wavelet, a training set one and two dimensions was generated. In this work, the mother wavelet is used for the experiments as in the following expression:

$$\varphi(\vec{x}) = \exp\left(\frac{\vec{x} - B}{A}\right) \qquad (10)$$

$x = \{x_1, x_2, \ldots, x_p\}$, $A = \{A_{j1}, A_{j2}, \ldots, A_{jp}\}$ and $B = \{B_{j1}, B_{j2}, \ldots, B_{jp}\}$ are a family of functions generated from one single function $\varphi(x)$.

The results of the proposed algorithm compared with the cluster for function approximation algorithm, which resulted to be the best algorithm for this example in [6], with $\mu_{\min}$ = 0.001, are represented in Tables 1, 2 and 3. In these tables, $NRMSE_C$ is the NRMSE of the training data. $NRMSE_T$ is the final error index (for 10000 test data) obtained after the application of the Levenberg–Marquardt method. It must be noted, that both algorithms the cluster for function approximation and the proposed algorithm were designed to provide an initial WNN configuration to be subsequently optimized using a local

optimization method to find the global minimum. Finally, $Time_C$ is the optimization process execution time *(in seconds)*. As seen from the tables, the proposed algorithm reaches better approximations using less time than the CFA algorithm always.

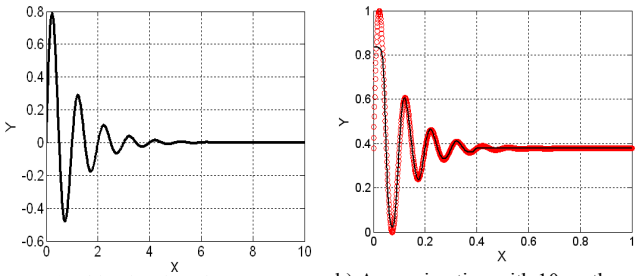
4.1. One Dimension Example

This example *f1(x)* has been chosen to demonstrate the importance of the equidistribution of the approximation errors throughout the clusters. This function as shown as in Figure 5-a, has a very variable output when the input x is near the value zero. This function is defined as:

$$f_1(x) = \frac{\sin 2\pi x}{e^x} \qquad x\ [0, 10] \qquad (11)$$

To test the effects caused by the proposed algorithm on the initialization and avoid local minimum on the placement of the mother wavelet activation function, a training set of 2000 samples of the function was generated by evaluating inputs taken uniformly from the interval [0, 10], from which we have removed 1000 points for test.

Figure 5-b presents the approximation of the WNN using 10 mother wavelets activation functions. We can see how the WNN is capable of making practically a perfect approximation.



a) Objective function $f_1(x)$. b) Approximation with 10 mother wavelet.

Figure 5. One dimension example.

Table 1. Comparison between the proposed approach and cluster for function approximation algorithm for the function $f_1(x)$.

Proposed Approach				Cluster Function Approx.		
Neurons #	NRMSE $_C$	NRMSE$_T$	Time$_C$	NRMSE $_C$	NRMSE$_T$	Time$_C$
4	0.88	0.27	2.8	0.84	0.60	14.2
5	0.85	0.10	1.3	0.79	0.5	11.1
6	0.86	0.15	6.1	0.79	0.36	18.9
7	0.84	0.11	9.4	0.77	0.27	30.2
8	0.80	0.07	20.2	0.77	0.24	14.2
9	0.83	0.03	12.7	0.76	0.14	81.9
10	0.78	0.02	43.2	0.77	0.15	167

We show in Table 1 best results with very much less times of execution. Both performances show that for this example of one dimension the proposed algorithm outperforms the cluster for function approximation algorithm.

In Figure 6, it's clear that the NRMSE$_T$ and execution time of the proposed algorithm outperforms CFA algorithm. The proposed approach minimizes the approximation error and execution time with much accuracy than CFA algorithms.

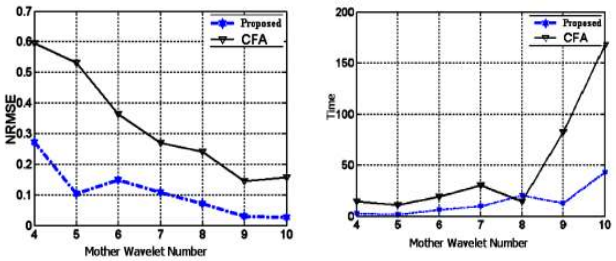
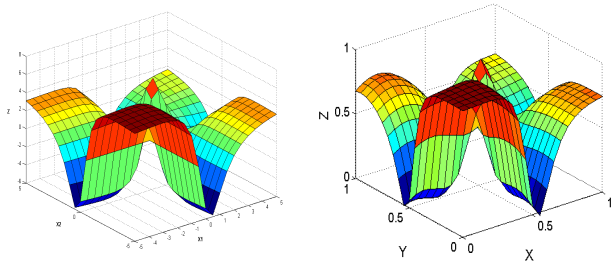


Figure 6. Comparison NRMSE$_T$ and time of optimization between the proposed algorithm and CFA algorithm of $f_1(x)$.

4.2. Two Dimensions Example

In this part, we used function $f_2(x_1, x_2)$ of two-dimensions as shown as in Figure 7-a, widely used in the bibliography related with the function approximation. This function of two-dimensions uses a set of training data formed by 441 points distributed as 21 x 21 cells in the input space.



a) Objective function $f_2(x_1, x_2)$. b) Approximation with 10 mother wavelet.

Figure 7. Two dimensions example.

This example of two dimensions $f_2(x_1, x_2)$ used to demonstrate the ability of the proposed algorithm in approximate two dimension examples, this function defined as:

$$f_2(x_1, x_2) = \left(\frac{(x_1 - 2)(2x_1 + 1)}{(1 + x_1^2)}\right)\left(\frac{(x_2 - 2)(2x_2 + 1)}{(1 + x_2^2)}\right) \quad x_1, x_2 \in [-5, 5] \qquad (12)$$

In Figure 8, the NRMSE$_T$ and execution time of the proposed algorithm outperforms CFA algorithm, comparing these two parameters (NRMSE$_T$, time). It is clear that the proposed approach minimizes the approximation error and execution time with much accuracy than CFA algorithms.

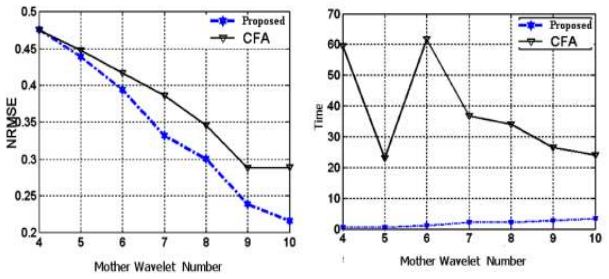


Figure 8. Comparison $NRMSE_T$ and time of optimization between the proposed algorithm and cluster for function approximation algorithm of $f_2(x_1, x_2)$.

Table 2 shows that the proposed algorithm obtains best results with less times of execution. Figure 7-b represents the approximation of the WNN using 10 mother wavelets activation functions.

Table 2. Comparison between the proposed approach and cluster for function approximation algorithm for the function $f_2(x_1, x_2)$.

Proposed Approach				Cluster Function Approx.		
Neurons #	$NRMSE_C$	$NRMSE_T$	$Time_C$	$NRMSE_C$	$NRMSE_T$	$Time_C$
4	0.85	0.48	0.42	0.86	0.48	59.51
5	0.83	0.44	0.60	0.85	0.45	23.1
6	0.75	0.39	1.2	0.85	0.42	61.7
7	0.66	0.33	2.1	0.79	0.39	36.7
8	0.63	0.30	2.1	0.68	0.35	34.0
9	0.62	0.24	2.8	0.58	0.29	26.6
10	0.60	0.22	3.3	0.59	0.28	24.1

4.3. Normalized WNN Example

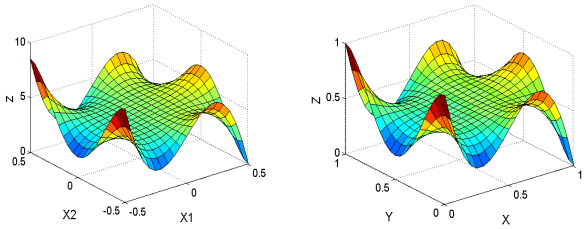
In Normalized Wavelet Networks (NWNN), hidden nodes perform a function similar to a Voronoi tessellation [2, 3] of the input space, and the output weights become the network's output over the partition defined by the hidden nodes. Consequently, a NWNN lose the localized characteristics of standard WNN and exhibit excellent generalization properties, to the extent that hidden nodes need to be recruited only for training data at the boundaries of class domains. In NWNN, the output activity is normalized by the total input activity in the hidden layer as in the following expression:

$$F(\vec{x}) = \sum_{j=1}^{m} \frac{\varphi_j(\vec{x})}{\sum_{j=1}^{m} \varphi_j(\vec{x})} \cdot w_j \qquad (13)$$

As a result of the normalization, the output activity becomes an activity-weighted average of the input weights in which the weights from the most active inputs contribute more on the value of the output activity. This results in novel computational properties which have attracted little attention in the neural network community. In standard WNN, the weights determine how much each hidden node contributes on the output. In NWNN, the activities of the hidden nodes determine which weights contribute more on the output. For instance, in the extreme case where only one hidden node is active, then the output of the network becomes equal to the weight corresponding with that hidden node, whatever its level of activity [2].

In this part, we used function $f_3(x_1, x_2)$ of two-dimensions as shown as in Figure 9-a. This function uses a set of training formed by 441 points distributed as 21 x 21 cells in the input space. This example $f_3(x_1, x_2)$ used to demonstrate the ability of the proposed algorithm in approximate functions using NWNN, this function defined as:

$$f_3(x_1, x_2) = 42.659(0.1 + x_1(0.05 + x_1^4 - 10x_1^2x_2^2 + 5x_2^4)) \qquad (14)$$
$$x_1 x_2 \in [-0.5, 0.5]$$



a) Objective function $f_3(x_1, x_2)$. b) Approximation with 10 mother wavelet.

Figure 9. Normalized WNN example.

As shown in Table 3, the proposed algorithm obtains best results with less times of execution. Figure 9-b represents the approximation of the WNN using 10 mother wavelets activation functions.

Table 3. Comparison between the proposed approach and cluster for function approximation algorithm for the function $f_3(x_1, x_2)$.

Proposed Approach				Cluster Function Approx.		
Neurons #	$NRMSE_C$	$NRMSE_T$	$Time_C$	$NRMSE_C$	$NRMSE_T$	$Time_C$
4	0.86	0.51	0.67	0.88	0.71	12.9
5	0.87	0.41	0.50	0.89	0.58	12.2
6	0.84	0.19	0.69	0.83	0.29	12.2
7	0.91	0.10	1.4	0.78	0.28	26.2
8	0.84	0.05	2.9	0.76	0.25	32.1
9	0.82	0.05	2.9	0.79	0.15	38.0
10	0.73	0.02	3.4	0.77	0.14	28.8

As shown in Figure 10, the proposed approach minimizes the approximation error and execution time with much accuracy than CFA algorithms.

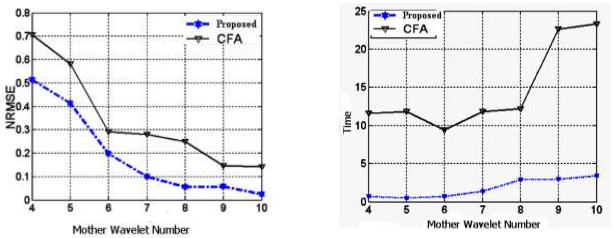


Figure 10. Comparison $NRMSE_T$ and time of optimization between the proposed algorithm and cluster for function approximation algorithm of $f_3(x_1, x_2)$.

5. Conclusions

In this paper, we have proposed an algorithm for optimizing the position of the mother wavelet function especially suited for function approximation problems. This method divides the input data space depend on the position of each mother wavelet function and calculates the error committed in every wavelet function using the real output of the WNN or NWNN (depends on the type used), trying to concentrate more mother wavelet functions in those input regions where the approximation error is large, this process has done by using a migration process which depends on the value of the performance of each mother wavelet, thus attempting to homogenize the contribution to the error of every mother wavelet area. This proposed technique is easy to implement and is superior in both performance and computation time to other algorithms such as the cluster for function approximation method, with which we have compared. We have also shown how it is possible to use this algorithm to find the minimal number of mother wavelets that satisfy a certain error target for a given function approximation problem.

References

[1] Awad M., Pomares H., Herrera. L., González J., Guillén A., and Rojas F., "Approximating I/O Data Using Radial Basis Functions: A New Clustering-Based Approach," *in Proceedings of 8th International Workshop on Artificial Neural Networks,* Spain, pp. 289-296, 2005.

[2] Bugmann G., "Normalized Gaussian Radial Basis Function Networks," *Technical Report*, Centre for Neural and Adaptive Systems, School of Computing, University of Plymouth, United Kingdom, 1998.

[3] Cao Z., Guo F., and Wang J., "Research on Flux Observer Based on Wavelet Neural Network Adjusted by Antcolony Optimization," *in Procceedings of International Conference on Machine Learning and Cybernetics*, Beijing, pp. 862-866, 2007.

[4] Chen Y., Yang B., and Dong J., "Time-Series Prediction Using A Local Linear Wavelet Neural Network," *Computer Journal of Neurocomputing*, vol. 69, no. 4-6, pp. 449-465, 2006.

[5] Delyon B., Juditsky A., and Benveniste A., "Accuracy Analysis for Wavelet Approximations," *IEEE Transactions Neural Networs*, vol. 6, no. 2, pp. 332-348, 1995.

[6] Gonzalez J., Rojas I., and Pomares H., "A New Clustering Technique for Function Approximation," *IEEE Transactions Neural Networks*, vol. 13, no. 1, pp. 132-142, 2002.

[7] Higgins C., "Classifications and Approximation with Rule-Based Networks," *Dissertation PhD*, California Institute of Technology, pp. 33-36, 1993.

[8] Kanjilal P. and Banerjee N., "On the Application of Orthogonal Transformation for the Design and Analysis of Feedforward Networks," *IEEE Transactions Neural Network*s, vol. 6, no. 5, pp. 1061-1070, 1995.

[9] Krishnaprasad. S. and Pati C., "Analysis and Synthesis of Feed Forward Neural Networks Using Discrete Affine Wavelet Transformations," *IEEE Transactions on Neural Networks*, vol. 4, no. 1, pp. 73-85, 1993.

[10] Li X. and Liu D., "Modeling and Optimal for Vacuum Annealing Furnace Based on Wavelet Neural Networks with Adaptive Immune Genetic Algorithm," *in Procceedings of International Conference on Advances in Natural Computation*, China, pp. 922-930, 2005.

[11] Marquardt W., "An Algorithm for Least-Squares Estimation of Nonlinear Inequalities," *Computer of Journal on Applied Mathematics*, vol. 11, no. 2, pp. 431-441, 1963.

[12] Moody E. and Darken C., "Fast Learning in Networks of Locally Tuned Processing Units," *Neural Computation*, vol. 2, no. 1, pp. 281-294, 1989.

[13] Pomares H., Rojas I., Ortega J., González J., and Prieto A., "A Systematic Approach to Self-Generating Fuzzy Rule-Table for Function Approximation," *in Procceedings of IEEE Transactions System*, vol. 30, no. 3, pp. 431-447, 2000.

[14] Sheng L. and Shu-Ching C., "Function Approximation Using Robust Wavelet Neural Networks," *in Procceedings of 14th IEEE International Conference on Tools with Artificial Intelligence*, Taiwan, pp. 483-488, 2002.

[15] Vachtsevanos G. and Wang P., "A Wavelet Network Framework for Diagnostics of Complex Engineered Systems," *in Proceedings of MARCON Maintenance and Reliability Conference*, Atlanta, pp. 623-637, 1998.

[16] Wallich P., "Wavelet Theory: An Analysis Technique that's Creating Ripples," *Technical Document*, Scientific American, pp. 34-35, 1991.

[17] Zhang J., Walter G., Miao Y., and Lee W., "Wavelet Neural Networks for Function Learning," *IEEE Transactions Signal Processing*, vol. 43, no. 6, pp. 1485-1497, 1995.
[18] Zhang Q. and Benveniste A., "Wavelet Networks," *IEEE Transactions on Neural Networks*, vol. 3, no. 6, pp. 889-898, 1992.

Mohammed Awad received his BSc degree in industrial automation engineering in 2000, from the Palestine Polytechnic University, and his PhD degree in 2005 from University of Granada, Spain. From February, 2006 till now he is an assistant professor in the Faculty of Engineering and Information Technology at the Arab American University, Palestine. From February, 2006 till now he is the chair of the Department of Computer Information Technology (CIT) at the Arab American University, Palestine. His current areas of research interest include artificial neural networks and evolutionary computation, function approximation using radial basis function neural networks, input variable selection and fuzzy and neuro-fuzzy system.