World Academy of Science, Engineering and Technology
International Journal of Electrical and Computer Engineering
Vol:10, No:6, 2016

Continuous Functions Modeling with Artificial Neural Network: An Improvement Technique to Feed the Input-Output Mapping

A. Belayadi, A. Mougari, L. Ait-Gougam, F. Mekideche-Chafa

***Abstract*—**The artificial neural network is one of the interesting techniques that have been advantageously used to deal with modeling problems. In this study, the computing with artificial neural network (CANN) is proposed. The model is applied to modulate the information processing of one-dimensional task. We aim to integrate a new method which is based on a new coding approach of generating the input-output mapping. The latter is based on increasing the neuron unit in the last layer. Accordingly, to show the efficiency of the approach under study, a comparison is made between the proposed method of generating the input-output set and the conventional method. The results illustrated that the increasing of the neuron units, in the last layer, allows to find the optimal network's parameters that fit with the mapping data. Moreover, it permits to decrease the training time, during the computation process, which avoids the use of computers with high memory usage.

***Keywords*—** Neural network computing, information processing, input-output mapping, training time, computers with high memory.

I. Introduction

THE artificial neural networks have been widely used in different applications [1]-[3], these new techniques of networks have been proven very suitable for solving such problems with fewer adjustable parameters [4], [5], because they are known as universal approximators [6] that can learn from its environment by using a mathematical process which leads in adjusting the synaptic weight and bias. The requested processes are typically obtained through a learning operation which consists to adjust the synaptic weights. For this reason, a variety of learning algorithms have been proposed to adjust these connection links [7]-[10]. The algorithms of first order may converge very slowly to the optimal solution of weights if the choice of learning rate is taken very small; however, if the learning rate is large, the learning process may get over fitting or be stuck in local optima [11], [12]. The algorithms of second order, which take into account the second derivative, are ranked as one of the most efficient learning algorithms. Among these algorithms, we will use in this study the Levenberg Marquardt algorithm [13]. Besides, it has been proven that the choice of the activation functions is as important as the choice of the network architecture and the learning algorithms. The sigmoid is considered as one of the most important activation functions for the performance of neural network models [14].

In this study, we develop an artificial neural network which is able to modulate the information processing of one-dimensional task. The model involves: supervised learning technique, second order algorithms to update the weights, and sigmoid as activation function. This paper is organized as follows: in Section II, we give a mathematical description of CANN. In Section III, we provide a theoretical formalism of synaptic weights updating. Moreover, we explain the way of feeding the input-output data by the proposed model and this by using first and second order algorithms. In Section IV, we show the results obtained through the neural network. The conclusion is provided in Section V.

A. Belayadi is with the Laboratory of Physics, Laboratory of coating, Materials and Environment (LRME), University M'hamed Bougara, Boumerdes, Algeria (phone: +213-551-221-302; e-mail: adelphys@gmail.com).

A. Mougari is with Electronic Department, National College of Technology, Biomedical Centre Dergana-Bordj El Kiffan, Algeria, and also with the Laboratory of coating, Materials and Environment (LRME), University M'hamed Bougara, Boumerdes, Algeria (e-mail: alnetdz@gmail.com).

L. Ait-Gougam, is with University of Bab Ezzouar, BP 32, El Alia, 16111, Algiers, Algeria (e-mail: ag-liela@yahoo.fr).

F. Mekideche-Chafa is Vice Rector in charge of External Relations of Cooperation, Animation, Communication and Scientific Events, University of Science and Technology Houari Boumedienne, Algeria (e-mail: fazia-mekideche@yahoo.com).

II. Neural Network Architecture

Artificial neural networks are known to be universal approximators of nonlinear functions [6]. In this approach, every multivariate continuous signal can be represented by the superposition of a small number of invariant continuous functions. In terms of neural network, every continuous function of two variables can be computed by a network with one hidden layer whose hidden units compute continuous functions.

Fig. 1 shows the architecture of a typical feedforward neural network model. The given network is a model of information processing whose design is very schematically inspired from real neuron functioning.

The architecture contains three layers with q input parameters, a layer of n nodes, and p output parameters. The connections are from each input node to each intermediate layer node and from each intermediate layer node to each output node in a feed forward manner.

Mathematically, the output $t_i^{(k)}$ to the *i*th node is given as [15]:

$$t_i^{(k)} = f^{(k)}\left(\sum_{j=1}^{N} w_{ij}^{(k)}\, t_j^{(k-1)}\right), \quad (1)$$

where w_{ij} are weight coefficients, which are connection links between the nodes in different layers, and N is the number of neurons in the layer $(k-1)$.

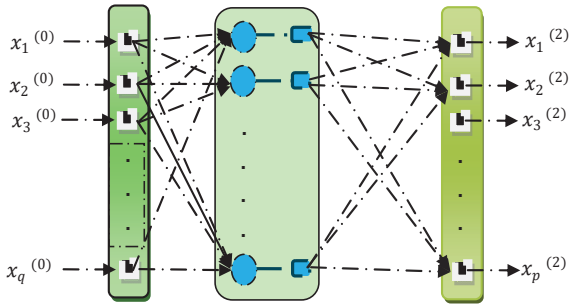

Fig. 1 Feedforward neural network, with architecture of q inputs, one hidden layer, and p outputs

III. Main Basis Functions in Neural Approach

Before we proceed, it is important to bring an outline of the approach and the architecture of the neural network. Consider an input x which must be processed into an output defined as a continuous signal given as a function $F(x)$. Next, let us deal with neural elementary units which obtain the same input x. Each unit of the network delivers an output $f(x)$ which relates on two parameters: the translation parameter b and the scale parameter λ. Output synaptic weights $w(b,\lambda)$ linearly regroup these elementary outputs into a unique output $F_{ap}(x)$. We have then the expansion [16]:

$$F_{ap}(x) = \int w(b,\lambda).f\left(\frac{x-b}{\lambda}\right).db.d\lambda \tag{2}$$

The unknown function $F(x)$ is approximated by a function $F_{ap}(x)$ of a set of N units of basic functions$f\left(\frac{x(p)-b_i}{\lambda_i}\right)$. For this reason, we can discretize, in the context of approximation theory, the approximation scheme to become as [13]:

$$F_{ap}(x(p)) = \sum_{j=1}^{N} w_j(b_j,\lambda_j).f_j\left(\frac{x(p)-b_j}{\lambda_j}\right). \tag{3}$$

where b_j and λ_j are the network parameters (synapses). N is neurons number in the last layer.

If we set the neural architecture to be a one hidden layer, and the last-layer activation functions to be linear then the (3) will be considered exactly as the (1). In this case, we will use (3) as expression to deal with our approaches.

In order to find the best approximation of the given task, we minimize the mean square error ε, first in terms of synaptic weight $w_i(b_i,\lambda_i)$, and second in terms of the parameters λ_i. For this, we minimize the square norm of the error ε given by:

$$\varepsilon = \langle F(x)|F_{ap}(x)\rangle \tag{4}$$

Compared to the synaptic weights, the minimization of the mean squared error is to solve the following equations:

$$\frac{\partial}{\partial w_i}\left(\langle F(x)-F_{ap}(x)|F(x)-F_{ap}(x)\rangle\right) = 0, \text{ and}$$

$$\frac{\partial}{\partial \lambda_i}\left(\langle F(x)-F_{ap}(x)|F(x)-F_{ap}(x)\rangle\right) = 0, \tag{5}$$

where the coefficients w_i and λ_i can be determined during the learning stage, using the gradient descent method or the Levenberg-Marquardt algorithm.

A. Training Algorithms

In the network learning processes, error back propagation (EBP) is considered as one of the most used training algorithm for feedforward artificial neural networks [17]. However, this algorithm is very slow if the size of the network is too large and different regions of the error surface may have a dynamic change of the learning rate coefficient. Second order algorithms help to converge much faster than the first order algorithms. Furthermore, by combining the training speed of second order algorithms and the stability of EBP algorithm, we obtain a very effective model of updating the network parameters. The training and optimization of neural networks to perform the approximation tasks is well documented in the literature [18], [19].

To start the training stage, we would like to provide a theoretical formalism of updating the synaptic weights of our model. In this approach, the Levenberg Marquardt algorithm is used to train the neural network connection links [20], [21]. The weights are adjusted as follows:

$$\mathbf{w}(k+1) = \mathbf{w}(k) + [\mathbf{J}^{\mathrm{T}}(k)\mathbf{J}(k) + \mu\,\mathbf{I}]\,\mathbf{J}^{\mathrm{T}}(k)\,\boldsymbol{\varepsilon}(k), \text{ and}$$

$$\boldsymbol{\lambda}(k+1) = \boldsymbol{\lambda}(k) + [\mathbf{J}^{\mathrm{T}}(k)\mathbf{J}(k) + \mu\,\mathbf{I}]\,\mathbf{J}^{\mathrm{T}}(k)\,\boldsymbol{\varepsilon}(k), \tag{6}$$

The Hessian matrix $\mathbf{H}$ is approximated as $\mathbf{H} = \mathbf{J}^{\mathrm{T}}\mathbf{J}$, and the gradient is computed as $\mathbf{g} = \mathbf{J}^{\mathrm{T}}\boldsymbol{\varepsilon}$, where $\mathbf{J}$ is the Jacodian matrix, which contains first derivatives of the network errors, $\boldsymbol{\varepsilon}$ is a vector of mean square network errors, and μ is a constant.

IV. Result of Simulation

In this section, the experimental results of artificial neural networks are discussed by dealing with two different methodologies in generating the input-output mapping.

The task under study includes continuous functions given as:

$$F(x) = \begin{cases} sin(x) + (1/x^2+1).exp(-x^2), & -6 < x \le 0 \\ sin(x) + (1/x^2+1).cos(-x^2), & 0 < x < 6 \end{cases} \tag{7}$$

The network that we have applied is a feedforward artificial neural network with one neuron in the input layer. The hidden layer is powered with 3 neurons, while the output layer contains one single neuron. In this study, the sigmoid activation function has been employed at hidden layers, i.e. $f(u) = 1/(1+e^{-u})$. However, for the output layer a linear function has been used as activation, i.e. $f(u) = u$.

For evaluating the optimal weight values, we have chosen the mean square error (ε) as a criterion. The ε indicates the average deviation from the target task and the neural response, and it is given as:

$$\varepsilon = \frac{1}{Q}\sum_{i=1}^{K}\sum_{p=1}^{Q}\left(y_i(p) - t_i^{(2)}(p)\right)^2, \quad (8)$$

where $t_i^{(2)}(p)$ are the set of the neural network response, $y_i(p)$ represent the desired values given by the task under study, Q is the total number of training data, and K is the neural number in the last layer.

A. Results through the First Methodology

This first methodology is commonly used and based on a direct mapping of the training input-output data. The data generation is given as:

- The neuron number in the first layer is equal to one since we have a continuous function with one variable.
- The neuron number in the last layer is equal to one since the desired values are given as one-output function $F(x)$.

We sample the function $F(x)$ to yield 150 training data generated uniformly over the interval $I = [-6, 6]$. In this case, the input-output mapping is organized as follow

- The training patterns $x(p)$ are uniformly and randomly distributed on the interval I, where i goes from 1 to Q.
- The output is generated directly as a single valued function defined as $y(p) = F(x(p))$.

Finally, the training set $\{\{x(p)\};\{y(p)\}\}$ is created, where the set $\{x(p)\}$ and $\{y(p)\}$ are the p-th input patterns and its output values over the interval I.

The main purpose, in this work, is to compare the exact function $F(x)$ to the neural network's approximated function $F_{ap}(x)$, by using the first methodology. To do so, we use the created training set $\{\{x(p)\};\{y(p)\}\}$. Furthermore, we initialize the network weights and bias by using Nguyen and Widrow's initialization algorithm [22]. The results are depicted in Figs. 2- 9.

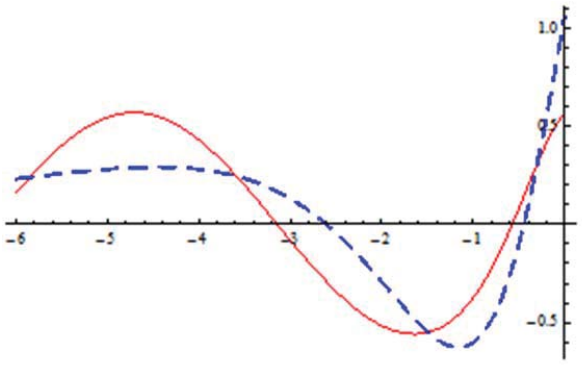

Fig. 2 Solid line: plot of the task $F(x)$ over $[-6, 0]$. Dashed line: the best approximation $F_{ap}(x)$ through the neural model in the case of the first method. The algorithm used is the Gradient Descent, and the number of iterations is 500

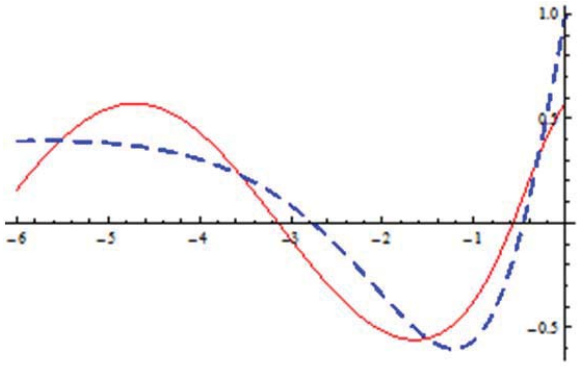

Fig. 3 Solid line: plot of the task $F(x)$ over $[-6, 0]$. Dashed line: the best approximation $F_{ap}(x)$ through the neural model in the case of the first method. The algorithm used is the Gradient Descent, and the number of iterations is 1000

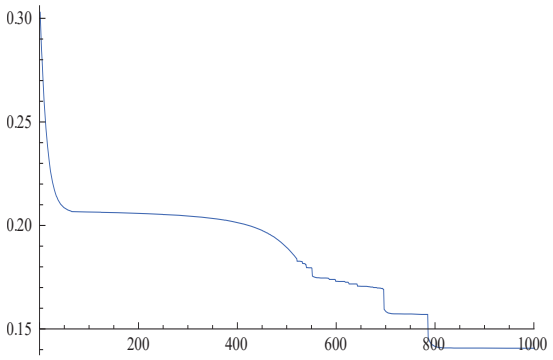

Fig. 4 Plot of the mean square norm of the error versus the number of iterations in the case of the first method

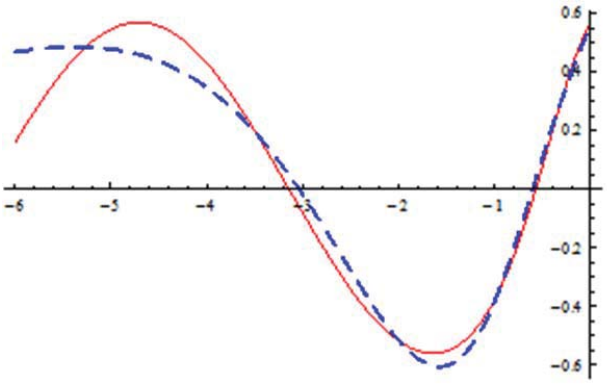

Fig. 5 Solid line: plot of the task $F(x)$ over $[-6, 0]$. Dashed line: the best approximation $F_{ap}(x)$ through the neural model in the case of the first method. The algorithm used is the Levenberg Marquard, and the number of iterations is 45

B. Results through the Second Methodology

In this next methodology which is a proposal one, the number of neurons is increased in the last layer. It is based on an indirect mapping of training input-output data. The neuron number in the last layer is equal to K neurons. In this case, the input-output mapping is organized as

- The training patterns $x(p)$ are uniformly and randomly distributed on the interval I

- The output is generated indirectly by dividing the training data into two categories: (i) we use the K neurons in the output layer and these neurons will be taken as the first category of the training data. (ii) The second category will have the number of training data as Q/K training patterns. And then in total, we will have Q training patterns since $K.(Q/K) = Q$. In this case, we are to choose the number Q and K in such a way that Q/K to be an integer.

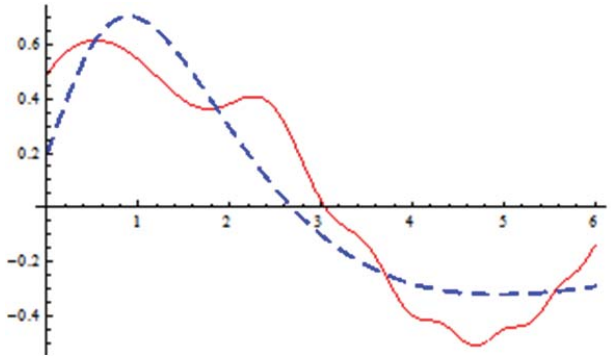

Fig. 6 Solid line: plot of the task $F(x)$ over $[0, 6]$. Dashed line: the best approximation $F_{ap}(x)$ through the neural model in the case of the first method. The algorithm used is the Gradient Descent, and the number of iterations is 500

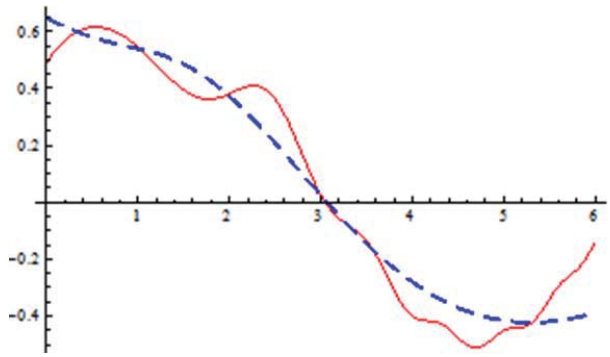

Fig. 7 Solid line: plot of the task $F(x)$ over $[0, 6]$. Dashed line: the best approximation $F_{ap}(x)$ through the neural model in the case of the first method. The algorithm used is the Gradient Descent, and the number of iterations is 1000

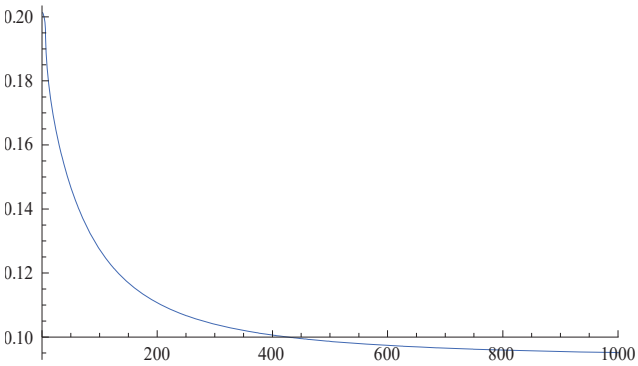

Fig. 8 Plot of the mean square norm of the error versus the number of iterations in the case of the first method

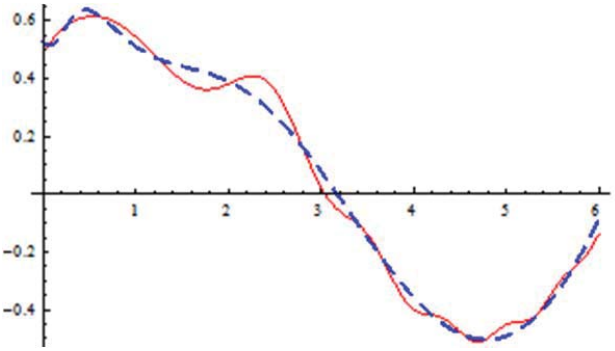

Fig. 9 Solid line: plot of the task $F(x)$ over $[0, 6]$. Dashed line: the best approximation $F_{ap}(x)$ through the neural model in the case of the first method. The algorithm used is the Levenberg Marquand, and the number of iterations is 45

Finally, the training set $\{\{x\,(1 + p.Q/K\ to\ (p + 1).Q/K)\}; y(1 + p.Q/K\ to\ (p + 1).Q/K)\}$ is created, where the set $\{x\,(1 + p.Q/K\ to\ (p + 1).Q/K)\}$ and $\{y(1 + p.Q/K\ to\ (p + 1).Q/K)\}$ are the p-th inputs patterns and their output values over the interval $I = [-6, 6]$, where in this case p goes from 0 to $K - 1$.

The main purpose is to compare the exact function $F(x)$ to the neural network's approximated function $F_{ap}(x)$, using the second methodology.

We sample the function $F(x)$ to yield 150 training data generated uniformly over I. As an application the neuron number k in the last layer is selected to be 10. Furthermore, the initialization of the network weights and bias were generated by using Nguyen and Widrow's initialization algorithm. The results are depicted in Figs. 10-17.

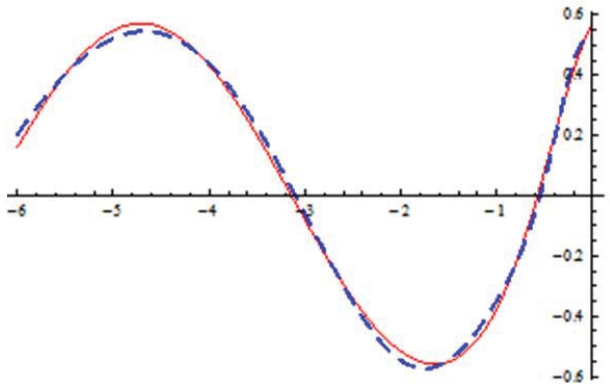

Fig. 10 Solid line: plot of the task $F(x)$ over $[-6, 0]$. Dashed line: the best approximation $F_{ap}(x)$ through the neural model in the case of the second method. The algorithm used is the Gradient Descent, and the number of iterations is 8

- This study confirmed the performance of the neural network while dealing with the second methodology. It has been proven, from Figs. 1-17, that the network recognized better the functions under study in the second approach and this by dealing only with the first order algorithm. However, for the first methodology, the network response was less accurate, by using the first

order algorithm. Consequently, the first methodology requires the use of the second order algorithm to get better results.

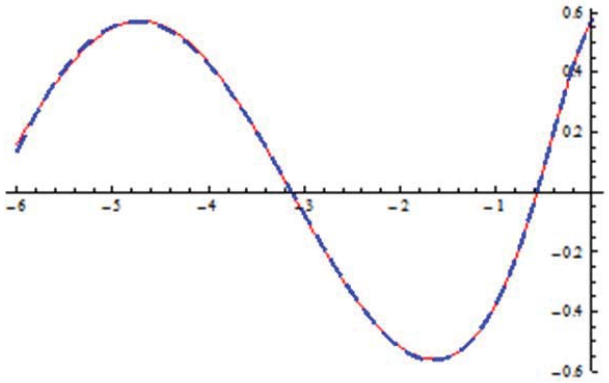

Fig. 11 Solid line: plot of the task $F(x)$ over $[-6, 0]$. Dashed line: the best approximation $F_{ap}(x)$ through the neural model in the case of the second method. The algorithm used is the Gradient Descent, and the number of iterations is 22

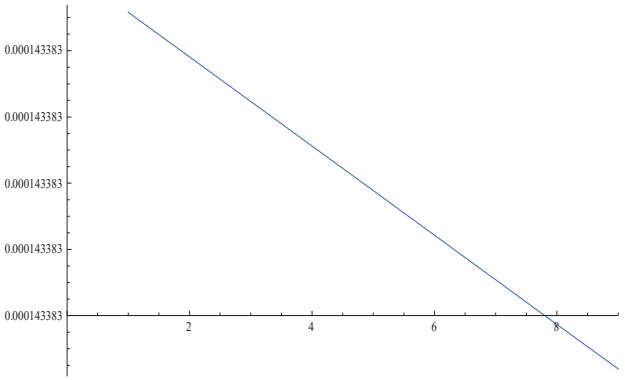

Fig. 12 Plot of the mean square norm of the error versus the number of iterations in the case of the second method

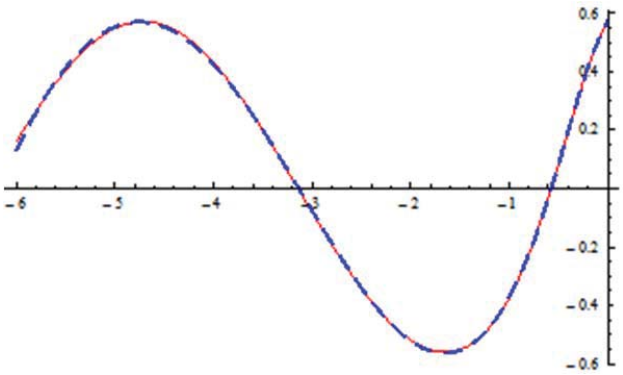

Fig. 13 Solid line: plot of the task $F(x)$ over $[-6, 0]$. Dashed line: the best approximation $F_{ap}(x)$ through the neural model in the case of the second method. The algorithm used is the Levenberg Marquand, and the number of iterations is 10

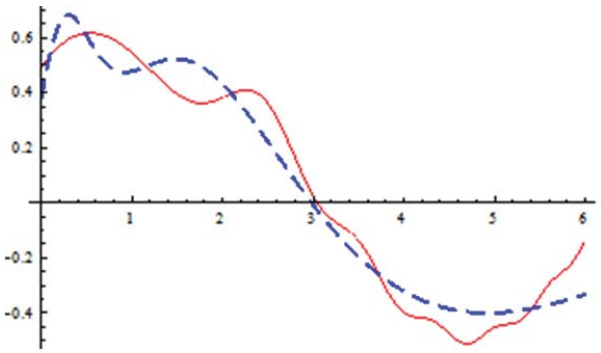

Fig. 14 Solid line: plot of the task $F(x)$ over $[0,\ 6]$. Dashed line: the best approximation $F_{ap}(x)$ through the neural model in the case of the second method. The algorithm used is the Gradient Descent, and the number of iterations is 8

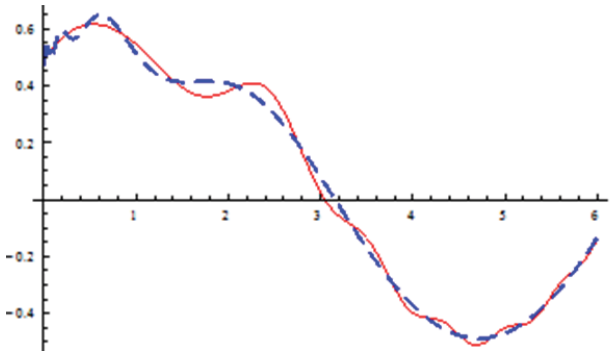

Fig. 15 Solid line: plot of the task $F(x)$ over $[0, 6]$. Dashed line: the best approximation $F_{ap}(x)$ through the neural model in the case of the second method. The algorithm used is the Gradient Descent, and the number of iterations is 22

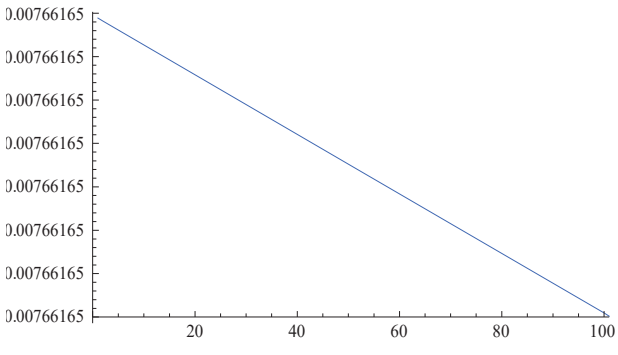

Fig. 16 Plot of the mean square norm of the error versus the number of iteratios in the case of the second method

- It is important to remember that the second order algorithms such as Levenberg Marquand are more efficient, but they require a huge amount of calculation which means the need of computers that are equipped with high memory. For this reason, the proposed method is very useful since it avoids the use of second order algorithms and decrease the training time by using only the first order algorithm.
- From the results, it is clear that the training time is considerably reduced when dealing with the proposed method. However, for the first approach, we can notice

that it needs more time to reach the required mean square error and sometimes we stuck in local optima. Accordingly, we have to go over the second order algorithm to overcome the issue of local optima.

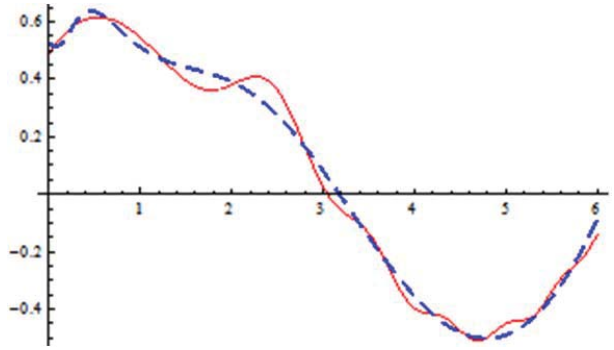

Fig. 17 Solid line: plot of the task $F(x)$ over $[0, 6]$. Dashed line: the best approximation $F_{ap}(x)$ through the neural model in the case of the second method. The algorithm used is the Levenberg Marquand, and the number of iterations is 10

The results can be improved by acting on several parameters: (i) the quality of the training data (examples that constitute the training set) needs to broaden and more diversified. (ii) For having an optimal model, we have to increase the neuron number in the hidden layer. The latter helps to get the optimal neural network parameters that would be adequate to ensure a higher accuracy of the proposed tasks. (iii) Neural network training is essentially an optimal weight determination problem for which the second order algorithms have been employed. The objective and the important factor of the second order algorithm are to expedite the learning process and minimize the mean square error function.

V. Conclusion

The gradient descent and Levenberg Marquardt backpropagation algorithm were taken into consideration to train the neural network. The performance of the neural network for the training data is given by using only the topology in which the number of neurons was powered by 3 neurons in the first hidden layer. Furthermore, the initialization of the network weights and bias were generated by using Nguyen and Widrow's initialization algorithm. The results advocated in favor of the second method. In fact, we have found that the proposed method is very useful since it allows using the first order algorithm without being stuck in local optima. Moreover, it provides the best network's parameters that fit with the tasks under study. We conclude that using neural networks with a new method of generating the input-output mapping would be very effective in term of accuracy and performance, as this study showed. Also, the study confirmed that neural techniques are typically considered as a strong tool to deal with in information processing.

References

[1] A. Alessandri, L. Cassettari, R. Mosca, "Nonparametric nonlinear regression using polynomial, and neural approximators," *A numerical comparison. Comput Manag. Sci*., 2009, vol. 6, pp. 5–24.
[2] H.K. Lam, U. Ekong, H. Liu, B. Xiao, H. Araujo, S.H. Ling, and K.Y. Chan, "A study of neural-network-based classifiers for material classification," *Neurocomputing.*, 2014, vol. 144, pp. 367–377.
[3] A. Nazemi, M. Dehghan, "A neural network method for solving support vector classification problems," *Neurocomputing*., 2015, vol. 152, pp. 369–376.
[4] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks.,* 1989, vol. 2, pp. 359.
[5] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals and Systems,* 1989, vol. 2, pp. 303.
[6] V. Kurkova, "Kolmogorov's theorem and multilayer neural networks," *Neural Networks.,* 1992, vol. 5, pp. 501-506.
[7] R.S. Scalero, N. Tepedelenlioglu, "A fast new algorithm for training feedforward neural networks," *IEEE Trans. Signal Processing., 1992,* vol. 40, pp. 202–210.
[8] M.T. Hagan, M.B. Menhaj, "Training feedforward networks with the Marquardt algorithm," *IEEE Trans. Neural Netw.*, 1994, vol. 5, pp. 989–993.
[9] Y. Liu, J.A. Starzyk, Z. Zhu, "Optimized approximation algorithm in neural networks without overfitting," *IEEE Trans. Neural Netws.*, 2008, vol. 19, pp. 983–995.
[10] M. Bogdan, M. Wilamowski, H. Yu, " Improved computation for Levenberg–Marquardt training," *IEEE Trans. Neural Netw., 2010, vol.* 21, pp. 930-937.
[11] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradientbased learning applied to document recognition," *Intelligent Signal Processing, IEEE Press.*, 2001, pp. 306–351, 200.
[12] M. Ventresca, H.R. Tizhoosh, "Improving gradient based learning algorithms or large scale feedforward networks," *IJCNN'09: Proceedings of the International Joint Conference on Neural Networks. IEEE Press, Piscataway, NJ, USA*, 2009, pp. 1529–1536.
[13] J.M. Wu, "Multilayer Potts perceptrons with Levenberg-Marquardt learning," *IEEE Trans. NeuralNetw.*, 2008, vol. 19, pp. 2032–2043.
[14] P. Chandra, Y. Singh, "An activation function adapting training algorithm for sigmoidal feedforward networks," *Neurocomputing*., 2004, vol. 61, pp. 429–437.
[15] F.M. Ham, I. Kostanic, "Principles of neurocomputing for science and engineering," *2nd edn. McGraw Hill,* 2001, *New York.*
[16] L. Ait-Gougam, M. Tribeche, F. Mekideche-Chafa, "A systematic investigation of a neural network for function approximation" *Neural Networks.,* 2008, vol. 21, pp. 1311_1317.
[17] M. Bogdan, F. Wilamowski, Y. Hao, "Improved computation for Levenberg Marquardt training" IEEE Trans. *Neural Netw*., 2010, vol. 21, pp. 930-93.
[18] S. Haykin, "Neural Networks: A Comprehensive Foundation," *Macmillan College Publishing,* 1994, *New-York.*
[19] Y. Liu, J.A Starzyk, and Z. Zhu, "Optimized approximation algorithm in neural networks without overfitting," *IEEE Trans. Neural Netws.*, 2008, vol. 19, pp. 983–995.
[20] M. Bogdan, M. Wilamowski, and Yu. Hao, "Improved Computation for Levenberg–Marquardt Training," *IEEE Trans.*, 2010.
[21] N. Ampazis, and S.J. Perantonis, "Two highly efficient second-order algorithms for training feedforward networks," *IEEE Trans. Neural Netw*., 2002, vol. 13, pp.1064–1074.
[22] D. Nguyen, B. Widrow, "Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights," *Proc. of the Int. Joint Conference on Neural Networks*., 1990, vol. 3, pp. 21–26.

A. Belayadi is an assistant professor at Ecole Superieur des Sciences des Aliments et Industries Alimentaires (ESSAIA), Algiers, Algeria. At present, he is working in the domain of artificial neural network computing and their application in different area, namely image processing, signal treatments and elementary excitation of atoms (electronic and magnetic). He is integrated in the Laboratory of Coating, Materials and Environment (LRME) at the University M'hamed Bougara, Boumerdes, Algeria.