Military Technical College, Kobry El-Kobbah, Cairo, Egypt

9th International Conference On Aerospace Sciences & Aviation Technology

LEARNING USING ERROR BACKPROPAGATION: A NEW VERSION

MOUSSA* H. M., ELARABY**S. M. AND KOUTB*** M. A.

ABSTRACT

Training of multilayered feed-forward neural networks (MLFFNNs) is considered in this work. A procedure to derive high performance learning algorithm for updating the network weights is proposed. The proposed algorithm is based on heuristic technique that is developed from an analysis of the performance of the basic-backpropagation training algorithm. A unified formulation of the conventional learning algorithms including the basic-backpropagation algorithm, the momentum algorithm, and the exponential-smoothing algorithm alongside with the proposed learning algorithm is introduced. Recursive relations for updating the weights of the network are derived which greatly simplifies the application of these rules. Simulation results are presented and comparative studies are carried out to demonstrate the effectiveness of the new learning algorithm. The new algorithm can converge more than hundred times faster than the conventional algorithms.

KEY WORDS

Neural Networks, Learning, Backpropagation and Convergence.

* Electronics Engineer, Atomic Energy Authority, Nuclear research Center, Anshas, Egypt.

** Associate Professor, Atomic Energy Authority, Nuclear research Center, Anshas, Egypt.

*** Professor, Dept. of Industrial Electronics and Control, Faculty of Electronic Engineering, Menoufia, Egypt.

1. Introduction

Multilayered feed-forward neural networks (MLFFNNs) offer an exciting alternative for modeling complex nonlinear systems, Irwin, et. Al. [1]. A properly trained backpropagation network has the capability of approximating a system to any degree of arbitrary accuracy, Narendra, et. Al. [2]. Unfortunately, training of MLFFNNs is a complicated process because of the many factors and parameters that influence the training process, Zurada, J. M. [3]. During training the weights, biases, neurons transfer function, the learning rate, the number of hidden layers and the number of neurons in each layer are iteratively adjusted to optimize the network performance. Several training algorithms have been applied to train FFNNs, Billings and Chen [4]. These algorithms are often too slow for practical problems [5 – 8]. In this paper, we will introduce a high performance algorithm that can converge more than hundred times faster than the conventional algorithms. This faster algorithm is based on heuristic technique, which is developed from an analysis of the performance of the basic-backpropagation training algorithm. A unified formulation of the conventional algorithms including the basic-backpropagation method, the momentum method, the exponential-smoothing method and the proposed method is introduced. The proposed algorithm is well suited for real-time, on line, computer control systems because of its ability in terms of rapid processing of collected plant input–output data.

2. Background

2.1 Feed-forward neural networks

The basic processing element of neural networks is often called a neuron. In this paper, the term neuron will refer to an operator, which maps $R^n \rightarrow R$. The basic model of a neuron is illustrated in Fig.1 and is described by the equation

$$y = F(\sum_{j=1}^{n} w_j u_j + T_o) \qquad (1)$$

Where $\boldsymbol{u}=[u_1\ u_2\ .....\ u_n]^T$ is the input vector, $\boldsymbol{W}=[w_1\ w_2\ .....\ w_n]^T$ is the weight vector of the neuron and T_o is known as its bias. *F(.)* is a monotone continuous function F: $R \rightarrow (-1,1)$, its selection depends on the case under consideration. In this paper, the sigmoidal function is used.

A set of interconnected neurons constitutes a neural network. If the neurons are arranged in layers $\ell = 0,1, \ldots, L$, and a neuron at layer ℓ is allowed to receive its inputs only from neurons in the $(\ell-1)$ layer, then this network is known as feed-forward neural network, Fig. 2.

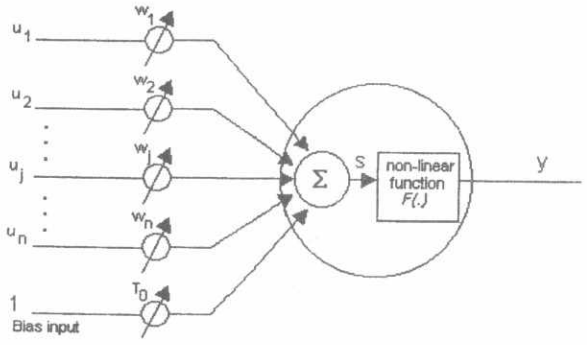


Fig.1 Basic model of neuron

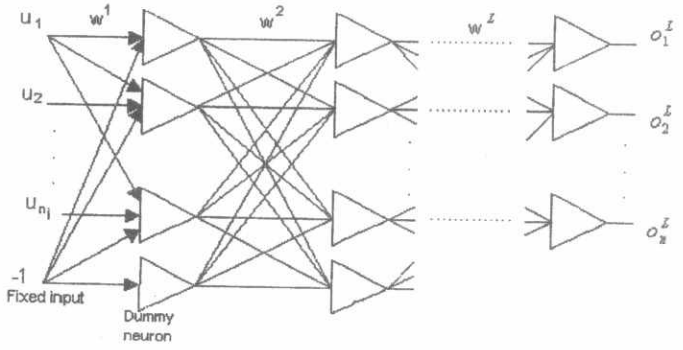


Fig. 2 Multilayer feed-forward neural network.

The output of the i-th neuron in the ℓ-th layer is given by

$$o_i^{\ell} = F\left(\sum_{j=1}^{n_{(\ell-1)}} w_{ij}^{\ell} o_j^{\ell-1} + T_i^{\ell}\right) \qquad i = 1,2,\ldots,n_{\ell} \tag{2}$$

Where $w_i^{\ell} = [w_{i1}^{\ell}\ w_{i2}^{\ell} \ldots\ldots w_{in_{\ell}}^{\ell}]^T$ is the weight vector associated with the i-th neuron in the ℓ-th layer, and n_{ℓ} is the number of neurons in that layer. It is common practice to refer to the ℓ=0 layer as the input layer, to the ℓ=L as the output layer and to all other layers as hidden layers. For simplicity, it is assumed that the threshold values T_i^{ℓ} are adjustable along with the other weights, and no distinction is made between the weights and thresholds during learning. The thresholds T_i^{ℓ} are learning exactly in the

same manner as the remaining weights. This implies that

$$w_{in_\ell}^{\ell} = T_i^{\ell}$$

and the fixed input is of value

$$o_{n_\ell}^{\ell} = -1 \qquad \ell = 0,1, \ldots, L$$

Therefore, Equation (1) can be written as

$$o_i^{\ell} = F\left(\sum_{j=1}^{n_{(\ell-1)}} w_{ij}^{\ell} o_j^{\ell-1}\right) \tag{3}$$

To describe the architecture of the MLFFNNs, the notation introduced by Levin and Narendra [9] will be adopted. A family of networks with n_ℓ neurons at the ℓ-th layer will be denoted by

$$N_{n_0, n_1, \ldots, n_L}^{L}$$

In the following section, the basic-backpropagation method and some other modifications of this method are briefly described.

2.2 The backpropagation algorithm

This algorithm which performs stochastic gradient descent, provides an effective method to train a feed-forward neural networks to approximate a given continuous function over a compact domain $\mathcal{D}$, Rumelhart et. Al. [10]. Let $u \in \mathcal{D}$ be a given input, the network approximation error for this input is given by

$$e(u) = d(u) - o(u) \tag{4}$$

Training the neural network to minimize this error is equivalent to minimizing

$$E = \int_{\mathcal{D}} \|e(u)\| du \tag{5}$$

The learning procedure for the network is carried out as follows.
The training pattern vectors ***u*** should be arranged in pairs with the desired response vectors ***d***. Let ***W*** denote the weighting matrix of the network. Following each training pair, the weights of the network are adjusted according to

$$w_{ij}(k+1) = w_{ij}(k) - \eta(k) \frac{\partial E}{\partial w_{ij}(k)} \Big|_{w=w(k)} \tag{6}$$

where η(k) is a positive constant known as the learning constant, $\eta : R \rightarrow (0,1)$.
The weight changes as given in this equation are chosen in such a way to minimize the output error by an approximation to gradient descent until an acceptable minimum error is achieved.
There is a number of learning rules for updating the weights in the network according to Equation (6). The learning procedure can be divided into two steps. In the first

step, the weights are adjusted in the output layer, then weights adjustment in the hidden layers are determined.
The methods used for weights updating are :

i- The basic-backpropagation method.
ii- The momentum method.
iii- The exponential-smoothing method.

These methods propose different ways and modification to proceed in the change of weights. The modification introduced by these methods aims to speed up the training time and to complete the learning process as fast as possible.
The updating rules for each method are summarized below.

i- The basic-backpropagation method

a) The updating rule for the output layer can be derived as follows.
Obtain the gradient of E with respect to w_{ij}^{ℓ} as follows

$$\frac{\partial E}{\partial w_{ij}^{\ell}} = \frac{\partial E}{\partial o_i^{\ell}} \frac{\partial o_i^{\ell}}{\partial net_i^{\ell}} \frac{\partial net_i^{\ell}}{\partial w_{ij}^{\ell}}$$

$$= \frac{\partial E}{\partial o_i^{\ell}} f'(net_i^{\ell})\, o_j^{\ell-1}$$

$$\stackrel{\Delta}{=} \delta_{ij}^{\ell} o_j^{\ell-1}$$

where $$\delta_{ij}^{\ell} \stackrel{\Delta}{=} -\frac{\partial E}{\partial o_i^{\ell}} f'(net_i^{\ell}) \qquad (7)$$

then substitute in Equation (6) to determine the weight adjustments as

$$w_{ij}^{L}(k+1) = w_{ij}^{L}(k) + \eta \delta_{ij}^{L} o_j^{\ell-1} \qquad (8a)$$

b) The updating rules for the hidden layers can be derived as follows.
The gradient of E with respect to w_{jk}^{ℓ} is given by

$$\frac{\partial E}{\partial w_{jk}^{\ell-1}} = \sum_i \frac{\partial E}{\partial o_i^{\ell}} \frac{\partial o_i^{\ell}}{\partial o_j^{\ell-1}} \frac{\partial o_j^{\ell-1}}{\partial net_j^{\ell-1}} \frac{\partial net_j^{\ell-1}}{\partial w_{jk}^{\ell-1}}$$

$$= \sum_i \frac{\partial E}{\partial o_i^{\ell}} f'(net_i^{\ell})\, w_{ij}^{\ell}\, f'(net_j^{\ell-1})\, o_k^{\ell-2}$$

$$= f'(net_j^{\ell-1})\, o_k^{\ell-2} \sum_i \delta_{ij}^{\ell}\, w_{ij}^{\ell}$$

where δ_{ij}^{ℓ} was already defined by Equation (7). Substitute in Equation (6) to determine the weight adjustments as

$$w_{jk}^{\ell-1}(k+1) = w_{jk}^{\ell-1}(k) + \eta\, f'(net_j^{\ell-1})\, o_k^{\ell-2} \sum_i \delta_{ij}^{\ell}\, w_{ij}^{\ell} \qquad (8b)$$

On the basis of Equations (8a) and (8b), the different learning rules will formulated as explained below.

ii- The momentum method

In Schalkoff R. J. [11] a momentum term $\alpha \in (0,1)$ is suggested to be included in Equations (8a) and (8b), so that these equations become.

a) The output layer

$$w_{ij}^{L}(k+1) = w_{ij}^{L}(k) + \eta \delta_{ij}^{L} o_{j}^{\ell-1} + \alpha \, \Delta w_{ij}^{L}(k) \tag{9a}$$

where $\Delta w_{ij}^{L}(k)$ is the previous weight change.

b) The hidden layers

$$w_{jk}^{\ell-1}(k+1) = w_{jk}^{\ell-1}(k) + \eta \, f'(net_{j}^{\ell-1}) \, o_{k}^{\ell-2} \sum_{i} \delta_{ij}^{\ell} \, w_{ij}^{\ell} + \alpha \, \Delta w_{jk}^{\ell-1}(k) \tag{9b}$$

iii- Exponential-smoothing method

An alternative modification to the backpropagation learning rules was introduced by Wassermann, P. D. [12]. This modification is known as exponential-smoothing as explained below.

a) *The output layer*

$$w_{ij}^{L}(k+1) = w_{ij}^{L}(k) + (1-\eta) \, \delta_{ij}^{L} o_{j}^{\ell-1} + \eta \; \Delta w_{ij}^{L}(k) \tag{10a}$$

b) The hidden layers

$$w_{jk}^{\ell-1}(k+1) = w_{jk}^{\ell-1}(k) + (1-\eta) \, f'(net_{j}^{\ell-1}) \, o_{k}^{\ell-2} \sum_{i} \delta_{ij}^{\ell} \, w_{ij}^{\ell} + \eta \; \Delta w_{jk}^{\ell-1}(k) \tag{10b}$$

3- Error Sensitivity Method (The New Method)

The main drawback of learning MLFFNNs using the above mentioned methods is the long time required to perform the training process in terms of number of iterations and the number of epochs.

In this section, we propose a new learning method that overcomes this drawback. The weight adjustments of Equation (6) are instead modified according to the following rule

$$w_{ij}(k+1) = w_{ij}(k) - (1-\eta) \frac{\partial E}{\partial w_{ij}} - \mu \frac{\partial E}{\partial net_{i}} + \eta \Delta w_{ij}(k) \tag{11}$$

where μ is a learning constant $\mu \in (0,1)$.

Using Equation (11), the updating rules for the MLFFNNs can be derived as follows.

a) The output layer

The weight adjustments for this layer is given by

$$w_{ij}^{L}(k+1) = w_{ij}^{L}(k) + \mu\delta_{ij}^{L} + (1-\eta)\,\delta_{ij}^{L} o_{j}^{\ell-1} + \eta\ \Delta w_{ij}^{L}(k) \qquad (11a)$$

b) The hidden layers

The weight adjustments for the hidden layers are given by

$$w_{jk}^{\ell-1}(k+1) = w_{jk}^{\ell-1}(k) + \mu f'(net_{j}^{\ell-1})\sum_{i}\delta_{ij}^{\ell} w_{ij}^{\ell}(k) + (1-\eta) f'(net_{j}^{\ell-1}) o_{k}^{\ell-2}\sum_{i}\delta_{ij}^{\ell} w_{ij}^{\ell} + \eta\Delta w_{jk}^{\ell-1}(k)$$

$$\ell = 0,1, \ldots\ldots, L\text{-}1 \qquad (11b)$$

Fig. 3 depicts the block diagram of the new training method and explains both the flow of signals and the flow of errors within the network. The shaded portions of the diagram refer to feed-forward phase. The blank portion of the diagram refers to the training phase of the network. The backpropagation of the error from each output using the negative gradient descent technique is illustrated.

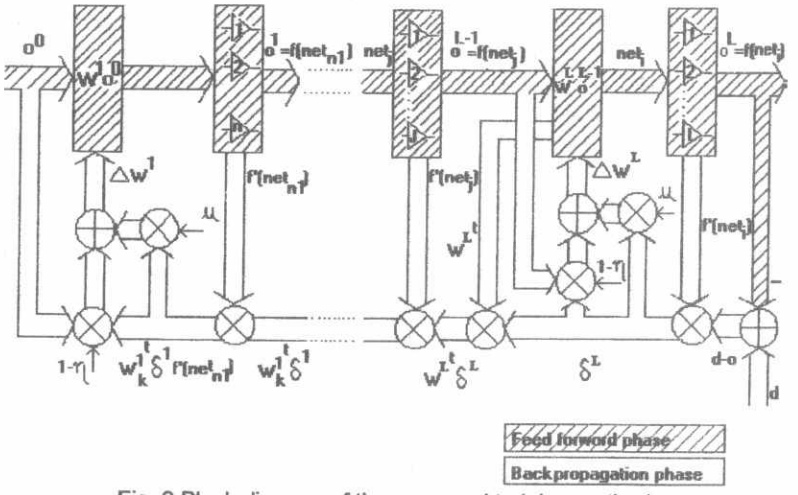


Fig. 3 Block diagram of the proposed training method

4- Simulation Results

The training is carried out by observing the input-output behavior of the system with the neural network which receives the same input as the system and a fixed number of delayed inputs and outputs. The system output is the desired output of the network. The system and the network outputs are compared to allow the weight update in such a way to reduce the error until the required precision is achieved.

Fig. 4 illustrates the principle of modeling a nonlinear single-input single-output system by learning a neural network.

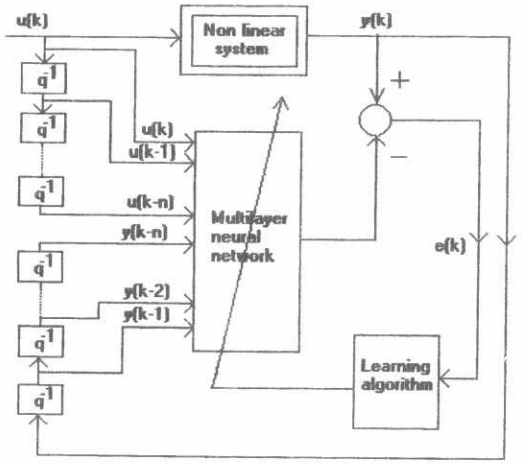


Fig. 4 Multilayer neural network learning scheme

Several examples that illustrate the performance of the four algorithms have been carried out. To illustrate the typical results that are obtained, consider the following examples.

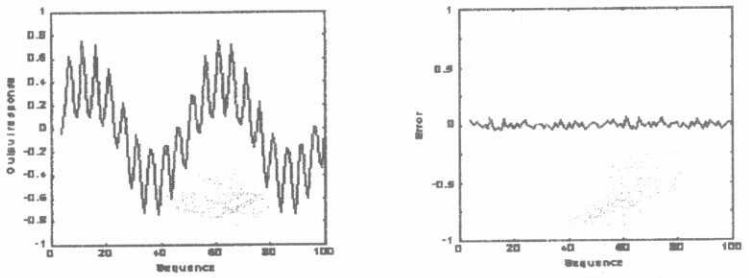
Example 1

This is a simulated system, 101 pattern of input-output data were generated by

$$y(k+1)=\frac{y(k)y(k-1)y(k-2)u(k-1)(y(k-2)-1)+u(k)}{1+y^2(k-1)+y^2(k-2)}$$

$$u(k)=0.5\sin(2\pi k/50)+0.3\sin(2\pi k/5)+0.2\sin(2\pi k/2)$$

A neural network model with eight neurons in the hidden layer was fitted to the identification data set. The learning constants were chosen as η=0.05 , α=0.05 and μ=0.75. The task was to learn 101 input-output pairs. Each inputs was a pattern of the values of the input variables and each associated output was a single scalar. The response of this neural network model is illustrated in Fig. 5, where it is seen that the neural network response closely matches the system response. Typical behavior of the evaluation of learning error with time for the proposed learning algorithm is also shown in this figure



(a) (b)

Fig. 5 Time-domain depiction of the neural network output compared with the true nonlinear output (Example 1).
a) System response. b) Learning error.

Comparison of the family of curves in Fig. 6 indicates that in the case of the proposed learning rule a system error of about 0.01 can be reached some where after 1152 epochs with 101 patterns being processed in each epoch. Whilst in the case of the other learning rules the same system error value of 0.01 is attained some where after 4099994 iterations. The decrease in processing times was correspondingly larger, the new learning algorithm being faster by factor of 100 or more. This example was taken from a realistic system identification task.

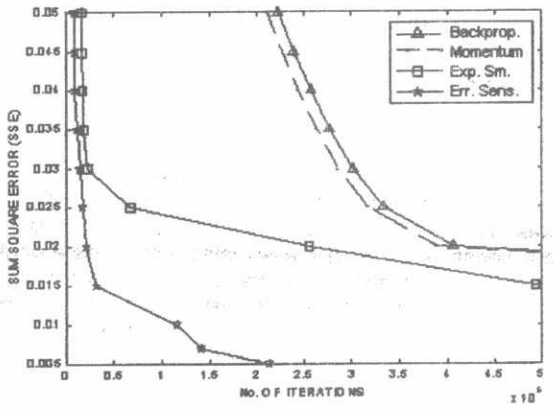
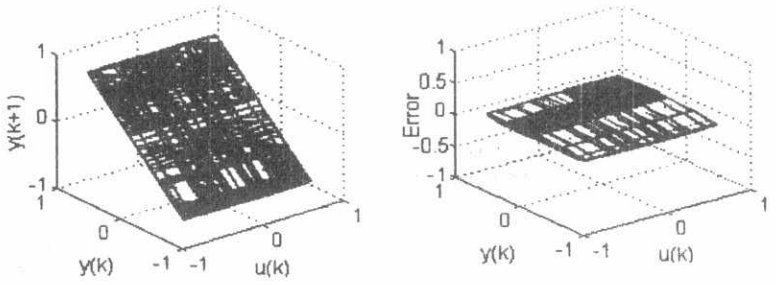


Fig.6 Variation of the learning error with number of iterations for the different methods of learning.

In Fig. 7 it is possible to observe the response surface of the neural network achieved by the proposed learning algorithm. Furthermore, the error surface computed as the difference between the network output and the system output is shown. Of course the flatter this surface the more the neural network has learned and generalized about the problem.



(a) Response surface (b) Error surface

Fig.7 The response surface of the neural network and the error surface
(a) Response surface of neural network. (b) Error surface.

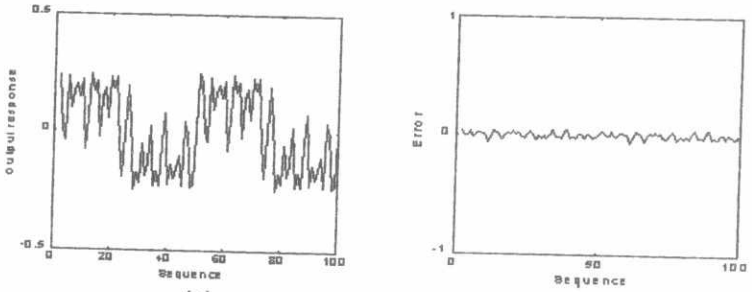
Example 2

This is a simulated system, 101 pattern of input-output data were generated by

$$y(k+1) = -0.5u(k)\sin\left[\pi\, e^{\left(-u^2(k)-y^2(k)\right)}\right] + \left(u(k) - u^3(k)\right)$$

$$u(k) = 0.5\sin(2\pi k/50) + 0.3\sin(2\pi k/5) + 0.2\sin(2\pi k/2)$$

A neural network model with six neurons in the hidden layer was fitted to the identification data set. The learning constants were chosen as η=0.05 , α=0.05 and μ=0.8. The task was to learn 101 input-output pairs. Each inputs was a pattern of the values of the input variables and each associated output was a single scalar. The response of this neural network model is illustrated in Fig. 8, where it is seen that the neural network response closely matches the system response. Typical behavior of the evaluation of learning error with time for the proposed learning algorithm is also shown in this figure.



(a) (b)

Fig. 8 Time-domain depiction of the neural network output compared with the true nonlinear output (Example 2).
a) System response b)Learning error.

Comparison of the family of curves in Fig. 9 indicates that in the case of the proposed learning rule a system error of about 0.01 can be reached some where after 73 epochs with 101 patterns being processed in each epoch. Whilst in the case of the other learning rules the same system error value of 0.01 is attained some where after 5501 epochs. The decrease in processing times was correspondingly larger, the new learning algorithm being faster by factor of 100 or more. This example was taken from a realistic system identification task.

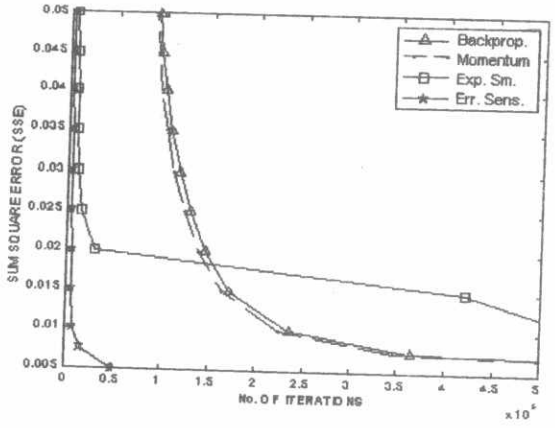
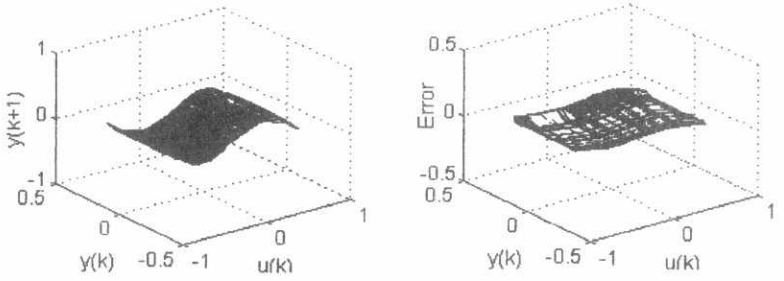


Fig. 9 Variation of the learning error with number of iterations for the different methods of learning.

In Fig. 10 it is possible to observe the response surface of the neural network achieved by the proposed learning algorithm. Furthermore, the error surface computed as the difference between the network output and the system output is shown. Of course the flatter this surface the more the neural network has learned and generalized about the problem.



(a) Response surface (b) Error surface

Fig.10 The response surface of the neural network and the error surface
(a) Response surface of neural network. (b) Error surface.

5- Conclusions

A new version of the error backpropagation learning algorithm has been developed to adapt multilayer neural network weights during the learning process. A unified formulation of the different rules underlying the different methods of learning has been adopted. Recursive relations of the learning algorithms have also been derived. Application to some simulated nonlinear time series has been demonstrated. The results obtained suggest that, learning MLFFNNs using the proposed algorithm is by far most efficient than the conventional learning algorithms, leading to reduced training requirements and more reliable training.

References

[1] Irwin, G. W., Warwick, K. and Hunt, K. J., " Neural Network Applications in Control " The Institution of Electrical Engineers, London, United Kingdom, (1995).
[2] Narendra, K., and Pathasarathy, K., "Identification and Control of Dynamical Systems Using Neural Networks " IEEE Transactions on Neural Networks, Vol. 1, pp. 4-27, (1990).
[3] Zurada, J. M., " Introduction to Artificial Neural Systems " West Publishing Company, St. Paul, New York, Los Angeles, San Francisco, (1992).
[4] Billings, S. A. and Chen, S., " Neural Networks and System Identification " The Institution of Electrical Engineers, London, United Kingdom, (1995).

[5] Schenker, B. and Agarwal, M., " Dynamic Modeling Using Neural Networks " Int. Journal of Systems Science, Vol. 28, No. 12, pp. 1285-1298, (1997).
[6] Billings, S. A., Jamaluddin, H. B. and Chen, S., " Properties of Neural Networks With Applications to Modeling Nonlinear Dynamical Systems " Int. Journal Control, Vol. 55, pp. 143-224, (1992).
[7] Al-Duwaish, H., Karim, M. N. and Chandrasekar, V., " Hammerstein Model Identification by Multilayer Feed-Forward Neural Networks " Int. Journal of Systems Science, Vol. 28, No. 1, pp. 49 -54, (1997).
[8] Yaw-Terng Su and Yuh-Tay Sheen, " Neural Network for System Identification" Int. Journal of Systems Science, Vol. 23, No. 12, pp. 2171-2186, (1992).
[9] Levin, A. U. and Narendra, K. S., " Recursive Identification Using Feed-Forward Neural Networks " Int. Journal Control, Vol. 61, No. 3, pp. 533 -547, (1995).
[10] Rumelhart, D. E., and McLelland, J. L., " Parallel Distributed Processing " I-II, MIT Press. (1986).
[11] Schalkoff R. J., " Artificial Neural Networks " McGraw-Hill Co., Inc., St. Louis, New York. (1997)
[12] Wassermann, P. D., "Neural Computer Theory and Practice " Van Nostrand Reihnold, New York. (1989).