International Journal of Sciences and Techniques of Automatic control & computer engineering
IJ-STA, Volume 2, N° 1, July 2008, pp. 386–399.

On the Use of Neural Network as a Universal Approximator

Amel SIFAOUI[1], Afef ABDELKRIM [1], Mohamed BENREJEB [1]

[1] UR LA.R.A Automatique, Ecole Nationale d'Ingénieurs de Tunis, BP 37
1002 Tunis Belvédère
Amel.Sifaoui@enit.rnu.tn, afef.abdelkrim@esti.rnu.tn,
mohamed.benrejeb@enit.rnu.tn

Abstract. *Neural network process modelling needs the use of experimental design and studies. A new neural network constructive algorithm is proposed. Moreover, the paper deals with the influence of the parameters of radial basis function neural networks and multilayer perceptrons network in process modelling. Particularly, it is shown that the neural modelling, depending on learning approach, cannot be always validated for large classes of dynamic complex processes in comparison with* Kolgomorov *theorem.*

keywords. *Function approximation, Radial Basis Function (RBF), MultiLayer Perceptron (MLP), Chaotic behaviour.*

1. Introduction

The general scheme of approximation supposes the existence of a relationship between several input variables and output dependent variables. For unknown relationship, an approximator is built between them using input-output couples from the learning data. One of the most important aspects of neural network conception is the selection of the architecture. The objective is to find the smallest architecture that accurately fits the true function described by the training data. Research in architecture selection has resulted in various different approaches to solve this problem such as network construction and pruning. Network construction algorithms start training with a small network and incrementally add hidden units during training when the network is trapped in a local minimum [1-3]. Neural network pruning algorithms start with an oversized network and remove unnecessary network parameters, either during training or after convergence to a local minimum. Multilayer Perceptrons networks (MLP networks) and Radial Basis Function networks (RBF networks) are two of the most commonly types of feedforward

This paper was recommended for publication in revised form by the editor Staff.
Edition: CPU of Tunis, Tunisia, ISSN: 1737-7749

neural networks used as neural network approximators. Multilayer perceptrons networks have a nonparametric architecture, with an input layer, one or more hidden layers, and an output layer. The input signal propagates through the network in a forward direction, layer by layer. RBF networks are non-linear parametric approximation models based on combinations of gaussian functions.
The MLP and RBF neural networks have been widely used for function approximation, pattern classification and recognition because to their structural simplicity and fast learning abilities [4-8]. The two types of network differ. The first difference is structure of architecture: RBF network has a simple architecture with a nonlinear hidden layer and a linear output one; MLP network may have one or more hidden layers.
The second difference is the mode of learning: for MLP network synaptic weights are adapted for each pattern but for RBF network they are trained simultaneously.
In this paper, first we propose a constructive algorithm based on a method known as experimental design; secondly, we use Kolgomorov theorem, to prove that a neural network with one hidden layer such as RBF neural network constitutes a good approximator of a multivariable function.
A dynamic system with a static non linearity is considered in order to show the applicability conditions of this theorem.

2. ANN design approach

2.1. Basic idea

A very important aspect of neural network model selection is to find the optimal architecture that accurately fits the true function described by the training data. The main problem in the design is that a lot of parameters need to be determined: number of layers, number of neurons in a layer, type of activation functions, number of patterns for the training process, etc. Therefore, it is not easy to find the optimal values of all these parameters, leading to a neural network satisfying the problem to which we are seeking a solution. A very large architecture may accurately fit the training data, but may have bad generalization due to over fitting of the training data. On the other hand, a too small architecture will save the computational costs but may not have enough processing elements to accurately approximate the true function. Thus architecture selection algorithms have to balance network complexity with the best fit of the function being approximated. One possible solution to this problem is the proposed iterative design approach **to be described in the following.**

2.2. ANN constructive algorithm

Many methods in neural network design, try to define the optimal architecture. In this way we propose a neural network constructive algorithm. Starting with one hidden layer network containing only one neuron, this initial neural network is

trained using one couple of pattern/target. The design process consists of adding successively neurons in the hidden layer, and then increasing the number of pattern/target couples, and finally the number of layers.
The algorithm permits the exploration of one part of the possible solution. The possible variations on the architecture of the network obtained can be done only on the number of neurons of the last layer, i.e. the number of neurons is equal to a predefined maximum for all the hidden layers except the last one, where this number can be lower than this maximum.
The behaviour expected from the neural network is to represent as close as possible a given dynamic system.
In order to go further in the neural network design process, other loops can be added in the proposed algorithm, allowing exploring the effect of initial weights and the nature of activation functions onto the network under construction.

3. Nonlinear function approximation

In conventional non–artificial intelligence-based systems, the function approximation is usually performed using a mathematical model of the considered system. However, sometimes it is not possible to have an accurate mathematical model [9], [10], or there may not exist any conventional type of model at all. In such cases, artificial-intelligence-based function approximators can be used [5], [6], [11], [12]. For nonlinear multivariable functions, Kolgomorov theory [10] defines a class of multilayer or RBF neural approximators.

Kolgomorov's theorem

A continuous nonlinear real function $y = f(x)$ with *n* variables, $x = (x_1, x_2, ..., x_n)$, can be approximated by the sums and superpositions of $(2n+1)$ continuous functions with single variables z_i :

$$y(x) = \sum_{i=1}^{2n+1} g_i(z_i) \qquad \textbf{(1)}$$

with:

$$z_i = \sum_{j=1}^{n} h_{ji}(x_i) \qquad \textbf{(2)}$$

and h_{ji} are nonlinear monovariable functions.

Kolgomorov's theorem states that the nonlinear function approximator can be mathematically described by:

$$y(x_1, x_2, ..., x_n) = \sum_{i=1}^{2n+1} g_i \left(\sum_{j=1}^{n} h_{ji}(x_i) \right) \quad \textbf{(3)}$$

It is also possible to modify Kolgomorov's theorem as follows. The h_{ji} are replaced by $\lambda_i h_j$, where λ_i are constants and h_j are strictly monotonic functions. The nonlinear function approximator can thus be defined as:

$$y(x_1, x_2, ..., x_n) = \sum_{i=1}^{2n+1} g_i \left(\lambda_i \sum_{j=1}^{n} h_j(x_j) \right) \quad \textbf{(4)}$$

The theorem does not indicate the forms of $h_j(x_j)$ and $g_i\left(\lambda_i \sum_{j=1}^{n} h_j(x_i)\right)$. Furthermore there is no known constructive technique to obtain them. Kolgomorov's theorem has been employed in [11], [12] [5], to demonstrate the nonlinear function aproximation capabilities of artificial neural networks.

Let g be non-constant, bounded and monotonic increasing continuous function, defined by:

$$g(S_i) = \frac{1}{\left[1 + \exp(-S_i)\right]} \quad (5)$$

Given $y = f(x_1, x_2, \ldots, x_n)$ and by using equation (4), the goal is to obtain an approximation of y, if $h_j(x_i) = S_j = w_{ij} - b_i$, where w_{ij} and b_i are real constants $(i = 1, 2, \ldots, M = 2n+1,\ j = 1, 2, \ldots, n)$.

It follows that, [13] :

$$y = f(x_1, x_2, ..., x_n) = \sum_{i=1}^{M} \lambda_i g_i \left(\sum_{j=1}^{M} w_{ij} x_j \right) \quad \textbf{(6)}$$

If for simplicity $b_i = 0$ and $g_1 = g_2 = \ldots = g$, then

$$y = f(x_1, x_2, ..., x_n) = \sum_{i=1}^{M} \lambda_i g \left(\sum_{j=1}^{M} w_{ij} x_j \right) \quad \textbf{(7)}$$

The last equation can represent the nonlinear function aproximator $f(x)$ by the neural network shown in figure 1.

It should be noted that there are n input nodes to the network and these can be considered as nodes of the so-called input layer. This layer is followed by a so-called hidden layer, which contains *M* hidden nodes. All input nodes are connected to all hidden nodes and an ith hidden node is connected to the $1,2,..., n^{th}$, input nodes,

via the $w_{i1}, w_{i2}, ..., w_{in}$ weigths, which are constant. All the hidden nodes are then connected to a so-called output node via the weights λ_j, $j = 1,2,..., M$. The output y, on the output node, is the linear combination of the outputs of the hidden nodes.
The weigths can be obtained by using input/output training data and applying a training method such as the backpropagation training algorithm [8].

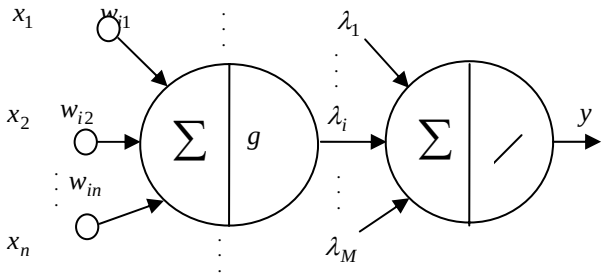


Fig.1. Neural network structure

4. RBF and MLP neural networks limitations

In this section, different RBF and MLP neural networks parameters are varied to approximate a nonlinear function and the result of each test is discussed and interpreted.

4.1 System description

In order to illustrate the influence of the neural network parameters, a nonlinear system, given by the following differential equations is simulated with a nonlinearity *f(x)* represented by a neural network.

$$\begin{aligned} \frac{dx}{dt} &= 10\sin(110\pi t) - 18 f(x) - y \\ \frac{dy}{dt} &= 10^6 f(x) \\ f(x) &= -0.01x + 0.015x^3, x(0) = 0.4, y(0) = 20 \end{aligned} \quad \textbf{(8)}$$

When the neural network parameters vary, several system responses can be obtained. The system's response with the considered real nonlinearity is presented in figure 2.
By varying the parameters, different results can be obtained when using a neural network structure. In fact, they depend on the number of the layers, the hidden number of nodes, the activation functions, the initial weights, the bias, the number of

patterns, the learning algorithm, the centers c_i and the widths σ_i of the Gaussian functions.

4.2 RBF neural network influence on the system's response

The first part of this section describes the RBF neural network used. The second part deals with the influence of the number of nodes and patterns, and the initial weights on a unique response of the studied dynamic system. The result of each test is interpreted.

The RBF neural network architecture consists of one input layer with one input, one hidden layer with N nodes and one output layer with one node. The input neuron is fully connected to the N hidden layer neurons except the bias neuron.

Again, each of the hidden layer neurons and the bias neuron also fully connected to the output neurons. The output of a hidden layer neuron i is usually generated by a gaussian function ϕ_i as follows:

$$\phi_i(x) = \begin{cases} \exp\left(-\dfrac{\|x - c_i\|}{2\sigma_i^2}\right) \\ 1 \end{cases} \quad \textbf{(9)}$$

where x is an input vector, c_i and σ_i are the center and the width of the receptive field of the ith neuron of the hidden layer respectively and $\|.\|$ is the euclidean distance between the input vector and the center vector.

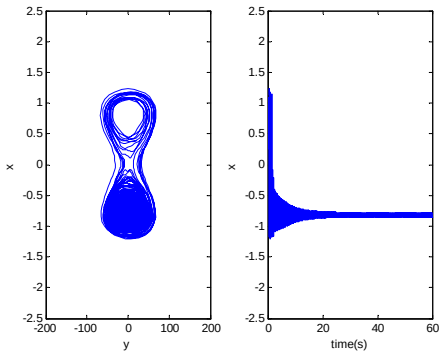


Fig.2. Simulation of the differential system (8) using the real non linearity

The output layer neuron computes a linear weighed sum of the outputs of the hidden layer neurons as follows:

$$y = \sum_{i=1}^{N} w_i \phi_i(x) \quad \textbf{(10)}$$

where w_i is the weight between the i[th] hidden layer neuron and the output layer neuron. In this paper, the hybrid learning method is used for training the RBF neural network. The learning stage is divided into two successive steps.
In the first step, the centers c_i of the hidden layer neurons are selected by using k-means clustering algorithm [14-16], then the width σ_i [17] of the radial basis functions are determined by P-nearest neighbour heuristics:

$$\sigma_i = \frac{1}{p} \sum_{k=1}^{p} \left(\| c_i - c_k \|^2 \right)^{\frac{1}{2}} \quad \textbf{(11)}$$

where c_k is the P-nearest neighbour of σ_i and p is determined heuristically.

Finally, the weights w_i between the hidden and output layers are estimated by using the Least Means Square (LMS) algorithm. The following equation is used to update the weight:

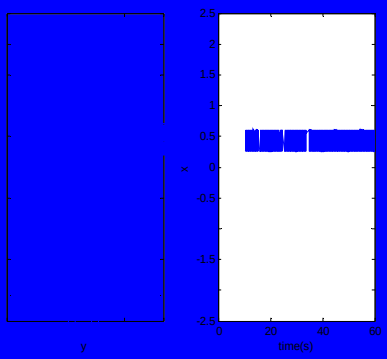
$$w_i^{t+1} = w_i^t + \eta e^t \phi_i^t \quad \textbf{(12)}$$

with :

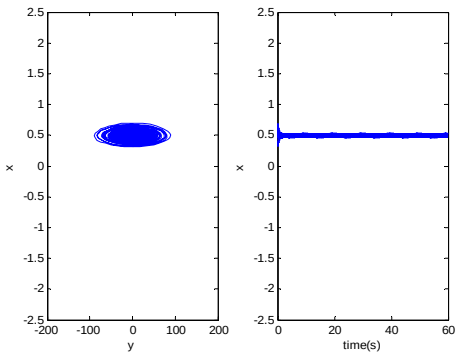
w_i^t	: weight vector at iteration t ,	
ϕ_i^t	: hidden layer output vector,	
η	: training rate,	
y^d	: desired output,	
y	: network output,	
e^t	: error	$y_i^d - y_i$

(13)

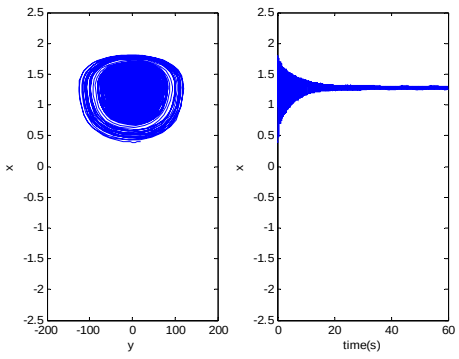
The purpose of this section is to analyse the influence of the RBF neural network's parameters on the uniqueness of the second order dynamic system's response, described above. The nonlinearity $f(x)$ is approximated by an RBF neural network.
First, the number of hidden nodes is changed for the same initial weights. For 10 hidden nodes, figure 3.a, 15 hidden nodes, figure 3.b, the neural network response show the existence of a limit cycle in contrast with the real response system, figure 2, which show the existence of a jump to another limit cycle.
The second varied parameter is the initial weights for 10 hidden nodes, figure 4.a and figure 4.b. Finally, the number of patterns is changed. In fact, for 10 hidden


2.5
2
1.5
1
0.5
0
-0.5
-1.5
-2.5
x
0
20
40
60
time(s)
y

algorithm [8]. Usually, initial weights are randomly selected, but in these tests, the weights, the bias and the patterns remain the same from one test to another.
The number of neurons of the second hidden layer N_2 has been fixed. Then many tests have been carried out where the number of the neurons of the first hidden layer N_1 is increased. It is noted that the activation functions are the same (tansig) for the two layers. The different illustrations of figure 6 show that the system gives different responses. The second set of tests serial consists of varying the activation functions of the hidden layers. The activation functions are (tansig) and (logsig) respectively for the first and second hidden layers, figure 7.a, then they are interchanged, figure 7.c. The results consequently achieved are different. In fact, a permanent flow is obtained when x drifts towards 0.8 in the first case, and when x drifts towards -0.8 in the second case.
In the case where the two hidden layers have the same activation function, figure 7.b, the system reaction is different from the two previous ones. Consequently, they illustrate the incapacity of the artificial neural network design approach to represent chaotic behaviour.

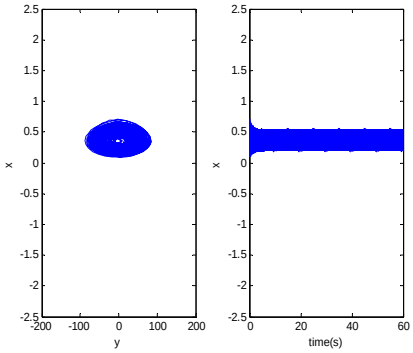


(a) First run

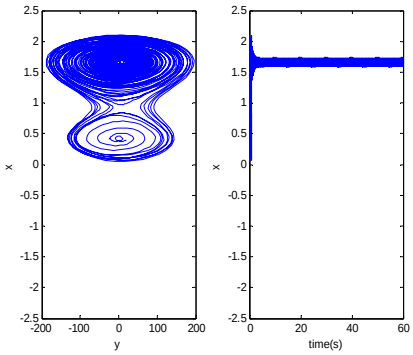


(b) Second run

Fig.4. RBF neural network: Initial weights influence for the same hidden number of neurons, case of 10 neurons

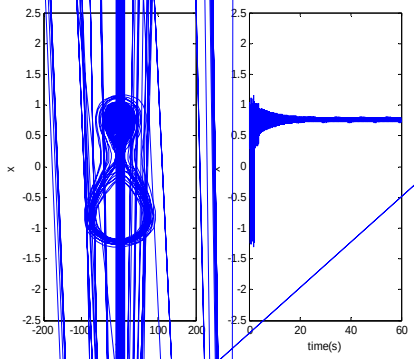


(a) Case of 67 patterns

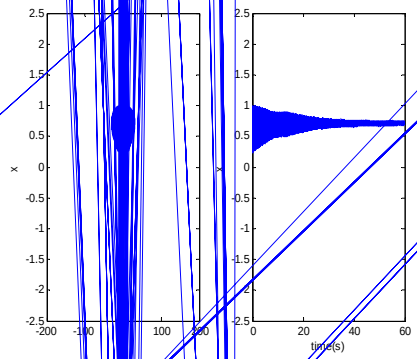


(b) Case of 101 patterns

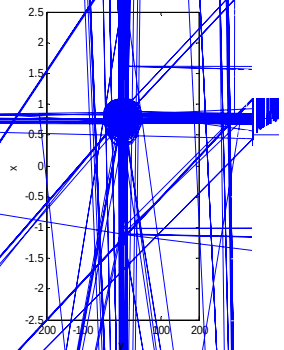
Fig.5. RBF neural network: Number of patterns influence, for 10 hidden neurons



(a) N_1 = 2 / tansig, N_2 = 5 / tansig
Goal = 0.0001

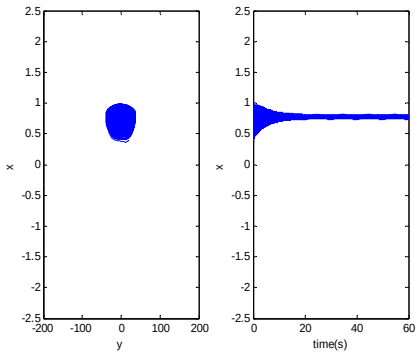


(b) N_1 = 5 / tansig, N_2 = 5 / tansig
Goal = 0.0001

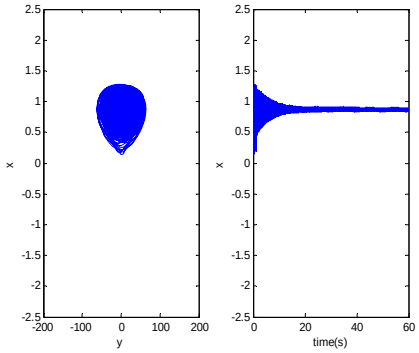


(c) N_1 = 10 / tansig, N_2 = 1 / tansig
Goal = 0.0001

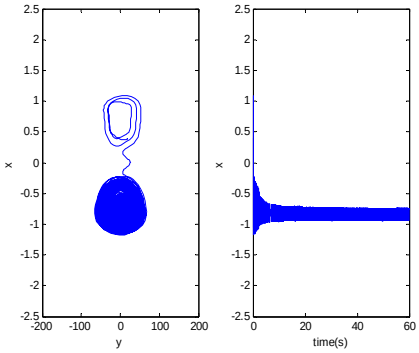
Fig.6. Hidden neurons numbers N_1 and N_2 influence for the same initial weights



(a) tansig and logsig neurons of first and second hidden layer respectively



(b) tansig and tansig neurons of first and second hidden layer respectively



(c) logsig and tansig neurons of first and second hidden layer respectively

Fig.7. Activation functions influence $N_1 = 5$, $N_2 = 10$, goal = 0.0001 for the same initial weights

5. Conclusion

The selection of neural networks parameters is of a great importance in modelling either the whole or a part of a process. Many algorithms have been proposed in the literature, in order to select the best neural architecture combining the network complexity and its capacity to approximate a given function. The first part of this paper proposes a network constructive algorithm. The second part deals with neural approximation of nonlinear function presenting chaotic behaviour. Two types of neural network are used: MLP and RBF.
The obtained results show that the approximators of nonlinear functions or part of them is not a proper approximation. Besides, they do not lead to a unique behaviour of the system as it must have been expected by Kolgomorov theorem.

References

1. Moody, J., Antsaklis, P.J.: The dependence identification neural network construction algorithm. IEEE Trans. Neural Networks, Vol.7. (1996) 3–15.
2. Fahlman, S.E., Lebiere, C.: The cascade-correlation learning architecture. School Comput, Sci., Carnegie Mellon Univ., Pittsburgh, PA., Tech. Rep. CMU-CS,(1991) 90–100.
3. Zhang, J., Morris, A.: A sequential learning approach for single hidden layer neural networks. Neural Networks, Vol.11. (1997) 65–80.
4. Gori, M., F. S.: A Multilayer Perceptron adequate for pattern recogition and verification. IEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 20. (1998) 1121–1132.
5. Hornik, K., Stinchcombe, M., White, H.: Multilayer feedforward networks are universal approximators. Neural Networks, Vol. 2. (1989) 359–66.
6. Funahashi, K.: On the approximate realization of continuous mappings by neural networks. Neural Networks, Vol. 2. (1989) 1831–92.
7 Borne, P., Benrejeb, M., Haggege, J.: Les réseaux de neurones. Présentation et applications. Ed. Technip, Paris (2007).
8. Borne, P.: Les réseaux de neurones. Apprentissage. REE, No. 9, (2006) 37-42.
9. Watson, G.A.: Approximation theory and numerical methods. Wiley New York (1989).
10. Kolgomorov, A. N.: On the representation of continuous functions of several variables by the superposition of continuous functions of one variable and addition. Doklady Akademiia Nauk SSR, 114 (1957) 953–6.
11. Girosi, F., Poggio T.: Representation properties of networks Kolgomorov's theorem is irrelevant. Neural Computation, 1 (1989) 465–9.
12. Kurkova, V.: Kolgomorov's theorem is relevant. Neural Computation 3 (1991) 617–22.
13. Vas, P.: Artificial-Intelligence-Based Electrical Machines and Drives. Application of Fuzzy, Neural. Fuzzy-Neural, and Genetic-Algorithm-Based Techniques - Oxford University Press, New York (1999).
14. Hwang, Y. S., Bang, S. Y.: An efficient method to construct a radial basis function neural network classifier. Neural Networks,Vol. 10. (1997) 1495–1503.
15. Orr, M. J.: Optimising the Widths of Radial Basis Functions. in Proc. of 5th Brazilian Symposium on Neural Networks Belo Horizonte (1998).

16. Leonard, J. A., Kramer M. A.: Radial basis function networks for classifying process faults. IEEE Control Systems (1991) 31–38.
17. Benoudjit, N., Archambeau, C., Lendasse, A., Lee, J., Verleysen, M.: Width optimization of the Gaussian kernels in Radial Basis Function Networks. In Proc. of 10th European Symposium on Artificial Neural Networks, d-side, Brussels (2002).
18. Zarouan, M., Dieulot, J.Y., Benrejeb, M.: Sur l'unicité de la réponse d'un réseau d'énergie électrique en régime de défauts. Revue des Sciences et Technologies de l'Automatique, Revue e-STA, Vol. 3. (2006).
19. Robert, B., IU, H.H.C., Feki, M.: Adaptative time-delayed feedback for chaos control in a PWM single phase inverter. Journal of Circuits, Systems, and Computers, Vol. 13, No. 3 (2006) 1-16.
20. Feki, M., Fourati, A.: Sliding mode-based synchronization and control of chaotic systems with parametric uncertainties. 4th International Multi-Conference on Systems, Signals and Devices, (2007).