ESANN'1994 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 20-21-22 April 1994, D-Facto public., ISBN 2-9600049-1-4, pp. 13-18

# Capabilities of a Structured Neural Network. Learning and Comparison with Classical Techniques.

Joan Codina, J. Carlos Aguado, Josep M. Fuertes.

Automatic Control and Computer Engineering Department
Universitat Politècnica de Catalunya
Pau Gargallo, 5 E-08028 Barcelona - Spain

**Abstract.** The use of Artificial Neural Networks (ANN) to identify dynamic processes and non linear functions usually provides a black box with the same inputs and outputs as the identified system. Our purpose in this paper is to adapt the ANN architecture to the structure of the mathematical equations that describe the system. Using similar structures for the mathematical study of dynamic systems and for the ANN description model facilitates comparing both type of solutions while we can apply mathematical well known techniques to ANN results. In this paper we present an architecture for ANN based on the state space description of dynamic systems. Such ANN uses lineal combinations of sines and cosines as activation functions. The knowledge of the number of Fourier coefficients used by the network allows finding, by analytical or numerical methods, the maximum learning capacity of the network, which is being compared with the real learning capacity of the system. While in the simplest cases this study can be carried out theoretically and we can state that the maximum can be achieved, in more complicated systems simulations had to be used to compare results.

## 1. Introduction

The use of ANN in the identification of dynamic systems has been carried out mainly in two ways, either using classical dynamic neural networks or using feed-forward neural networks with external delays. The use of standard dynamic neural networks such in Jordan [1] or Elman [2], presents some problems: an example is given when some of the system states are not connected directly to the output signals, for instance in systems with delays. In Jordan nets we also need the number of outputs to be at least equal as the system order. This condition can be sometimes unattainable.
The research in the field of systems identification has developed some models based on feed- forward neural networks. Such models usually take the form given by (1) and use external delay blocks for the inputs and outputs to accomplish the dynamic behavior:

$$x(k)= F [\ u(k), u(k-1), u(k-2)... \quad x(k), x(k-1), x(k-2)...\ ] \qquad (1)$$

In Narendra[3] it is shown a very extensive work with this methodology in which different configurations are studied. The neuron activation function used in such work are of sigmoidal type and it is obtained the input-output simulation of some proposed dynamic systems.

The use of dynamic neurons or the Hopfield network [4] increases the order of the system to the number of neurons, obstructing the study of the network dynamics.
To allow simpler models for MIMO systems (Multiple Inputs, Multiple Outputs), and to allow the application of classical techniques, we have developed a ANN architecture based on the state space representation of dynamic systems, Fig 1. Such structure allows the application of modern control theory to SISO and MIMO systems. Beginning with linear discrete-time systems, we obtained a neural network in Codina[5], able to learn the matrices of the system state representation from pairs of input-output vector signals. The model was a neural network with three layers: input, state and output. The state and the output layers are connected to the input layer, composed by the actual inputs and state of the system. It was used the "back-propagation through time" learning algorithm Werbos[6], where the error is back-propagated from the actual state to the previous state. With this methodology, and using linear neurons, we can obtain from the system, the state space matrices A, B, C and D:

$$x(k+1) = A\,x(k) + B\,u(k)$$
$$y(k) = C\,x(k) + D\,u(k) \qquad (2)$$

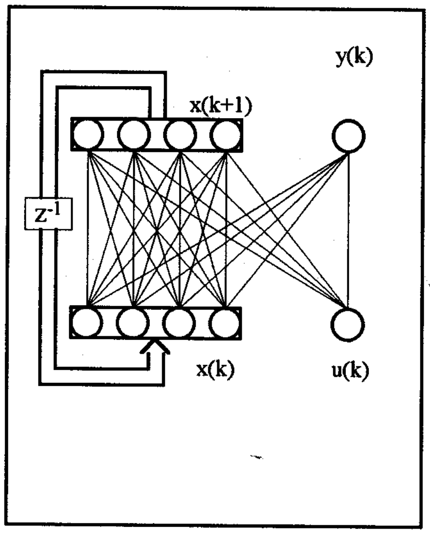

**Fig. 1** Proposed linear NN structure

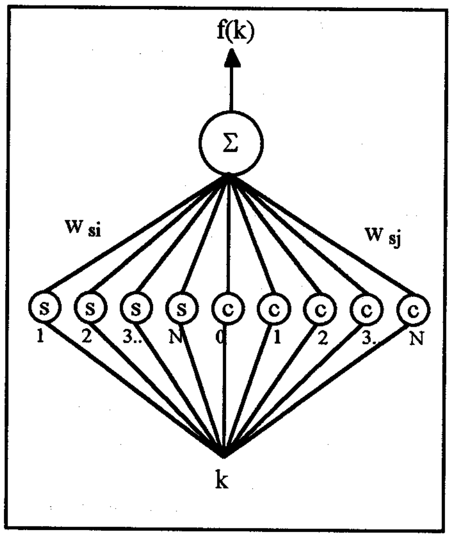

**Fig 2.** Fourier Series based model

## 2. Fourier Series Model

The main interest of neural networks applications to dynamical systems arises from their ability to learn and map non linear functions. We have expanded our previous model to deal with non linear systems. But expanding the linear model using sigmoids impedes the study of the previous model. To solve this drawback we have used a structured neural network based on the Fourier theory in order to approximate any non linear function, within a bounded interval, by means of a weighted sum of sines and cosines.

A non linear discrete-time system can be expressed by the following difference equations:

$$x(k+1) = F(x(k), u(k)) \qquad (4)$$
$$y(k) = G(x(k), u(k))$$

where F and G can be approximated in a bounded interval by a Fourier series, if we assume that both functions accomplish the Dirichlet conditions (which is a usual restriction).
For one input and one state:

$$x(k+1) = \sum_{n=0}^{\infty} \sum_{m=-\infty}^{\infty} A_{Fn,m} \cos(nw_n x(k) + mw_m u(k)) + B_{Fn,m} \sin(nw_n x(k) + mw_m u(k))$$

$$y(k) = \sum_{n=0}^{\infty} \sum_{m=-\infty}^{\infty} A_{Gn,m} \cos(nw_n x(k) + mw_m u(k)) + B_{Gn,m} \sin(nw_n x(k) + mw_m u(k)) \qquad (5)$$

Using sines and cosines as activation functions, the network model is expressed by:

$$\tilde{x}(k+1) = \sum_{n=0}^{N} \sum_{m=-M}^{M} \tilde{A}_{Fn,m} \cos(nw_n \tilde{x}(k) + mw_m u(k)) + \tilde{B}_{Fn,m} \sin(nw_n \tilde{x}(k) + mw_m u(k))$$

$$\tilde{y}(k) = \sum_{n=0}^{N} \sum_{m=-M}^{M} \tilde{A}_{Gn,m} \cos(nw_n x(k) + mw_m u(k)) + \tilde{B}_{Gn,m} \sin(nw_n x(k) + mw_m u(k)) \qquad (6)$$

which is an approximation of the function with a fixed number of terms. As the model is learned from input-output data another state space model can be learned. This does not mean that the model is bad, but means the state don't correspond to physical system variables. Thus we obtain:

$$\tilde{z}(k+1) = \sum_{n=0}^{N} \sum_{m=-M}^{M} \tilde{A}'_{Fn,m} \cos(nw_n \tilde{z}(k) + mw_m u(k)) + \tilde{B}'_{Fn,m} \sin(nw_n \tilde{z}(k) + mw_m u(k))$$

$$\tilde{y}(k) = \sum_{n=0}^{N} \sum_{m=-M}^{M} \tilde{A}'_{Gn,m} \cos(nw_n \tilde{z}(k) + mw_m u(k)) + \tilde{B}'_{Gn,m} \sin(nw_n \tilde{z}(k) + mw_m u(k)) \qquad (7)$$

## 3. Backpropagation Versus Fourier Coefficients.

This section demonstrates that, in a structured network, the use of the back-propagation algorithm can give the same result as the DFT. Some restrictions will be applied and the single case of one dimensional functions will be treated.
The Discrete Fourier Transform, expressed in sines and cosines terms, is calculated by the following formulas (for a set on 2N points):

$$A_n = \frac{1}{N}\sum_{k=0}^{2N-1} f(k)\cos(\tfrac{2\pi}{2N}kn) \;; \qquad B_n = \frac{1}{N}\sum_{k=0}^{2N-1} f(k)\sin(\tfrac{2\pi}{2N}kn) \tag{8}$$

$$f(k) = \frac{A_0}{2} + \sum_{n=1}^{n=N-1} A_n\cos(\tfrac{2\pi}{2N}kn) + B_n\sin(\tfrac{2\pi}{2N}kn) \tag{9}$$

To allow the calculation of the Fourier Series of a function, we need a neural network structure as shown in fig 2. Where S and C stands for Sin(μ2π/2N) and Cos(μ2π/2N), with μ=kn., being n the value of the fixed weights from the input. Such structure is similar to the functional-link net Pao[7] but including the link net as part of the network. This difference is essential to apply backpropagation.
If we now compare the algorithm to update weights in backpropagation we observe that the increment of the weight connecting the j neuron to the output neuron s, at time k, with desired output $t_s(k)$, is:

$$\Delta W_{sj}(k) = \eta\delta_k(k)o_j(k) \tag{9}$$

$$\delta_s(k) = (t_s(k) - o_s(k))f_s'(net_s(k)) \tag{10}$$

where net(k) is the weighted sum of the inputs to the neuron and $O_j$, $O_i$ and $O_s$ are the outputs of the a cosine, sine and an output neuron.
If $w_{sj}$ and $w_{si}$ are 0 for all j then $o_s(k)$ is 0 for all k. Thus the weight increments become:

$$\delta_s(k) = t_s(k) \;; \tag{11}$$

$$\Delta w_{sj}(k) = \eta\delta_s(k)o_j(k) = \eta t_s(k)\cos(j\tfrac{2\pi}{N}k) \tag{12}$$

$$\Delta w_{si}(k) = \eta\delta_s(k)o_i(k) = \eta t_s(k)\sin(i\tfrac{2\pi}{N}k) \tag{13}$$

As the weights at the beginning were zero, after the first update it will be equal to the increment, and if we update the weights every 2N points, then choosing η properly (η=2).

$$w_{sj} = \Delta w_{sj} = \frac{1}{2N}\sum_{k=0}^{k=2N-1}\Delta w_{sj}(k) = \frac{1}{N}\sum_{k=0}^{k=2N-1} t_s(k)\cos(j\tfrac{2\pi}{2N}k) \tag{14}$$

$$w_{si} = \Delta w_{si} = \frac{1}{2N}\sum_{k=0}^{k=2N-1}\Delta w_{si}(k) = \frac{1}{N}\sum_{k=0}^{k=2N-1} t_s(k)\sin(i\tfrac{2\pi}{2N}k) \tag{15}$$

Being $w_{sj} = A_n$ with $n = j$ and $w_{si} = B_n$ with $n = i$. Even that $A_o = w_{so}$, in the series the coefficient is divided by two, and that makes the only difference and can be solved with special learning rate for this neuron.
To reduce the number of terms in the Fourier series a linear term can be added to significally decrease the error in functions that could be lienarized.

$$f(k) = \alpha k + \frac{A_0}{2} + \sum_{n=1}^{n=N-1} A_n\cos(\tfrac{2\pi}{2N}kn) + B_n\sin(\tfrac{2\pi}{2N}kn) \tag{16}$$

Adding a linear function to the Fourier series involves loosing the orthogonality but with the advantage that a linear approximation is achieved. The NN will find the α that, together with the Fourier coefficients, will minimize the RMS.

## 4. The application to non linear dynamic systems

In the identification of dynamic systems a more elaborated structure is used, as shown (fig. 2). In this case the weight increments depend on many factors, as the state space trajectory or the initial values. It is difficult to compare, for a general case, the differences between the use of backpropagation through time and the way the Fourier coefficients should be used. The change in the state representation that the network does is also optimized in order to reduce the mean square error, for the training data.
The method used to train the network is as follows:

1- A linear network with initial random weights is used to learn a linear approximation of the system.
2- Some Sine and Cosine neurons with fixed input weighs and zero output weights are added to the previous network.
3- If the RMS error is too large, new Sine and Cosines can be added in order to improve the results, this will not change in a meaningful way the previous weights. Also some non-significant Sines and Cosines can be removed.

To have correct results the set of input-output data used during learning must be rich in values for any of the system signals: inputs, outputs and states. The learning coefficient, , as seen in the previous state must be smaller than 2, to avoid system instabilities.

### 4.1 Example

A first order non-linear system has been used to test the NN structure:

$$x(k+1) = x(k)u(k) - .5u(k) \tag{17}$$

$$y(k) = e^x + hist(u(k)); \quad hist(x) = x \text{ if } |x| > .4 \tag{18}$$

A signal composed of random steps with noise has been used to train the network. To test the learning capabilities three input test signals have been used: A signal similar to the training one, white noise and a chirp signal. Six Fourier coefficients where used and where indicated the linear term was added. The results obtained by the NN are compared with: The output obtained by using for the simulation the first six real Fourier Coefficients, and with the NN obtained from the direct learning of F(k) and G(k). In table 1 can be seen how the NN learn always the best state representation in order to minimize the error for the training signal.

|  | Calculated | Direct | Learned | direct with | Learned with |
|---|---|---|---|---|---|
| RMS |  |  |  | linear term | linear term |
| random steps | 5.537 E-2 | 5.543 E-2 | 2.308 E-3 | 4.722 E-3 | 1.662 E-3 |
| random steps 2 | 3.252 E-2 | 3.255 E-2 | 8.213 E-3 | 5.436 E-3 | 2.623 E-3 |
| white noise | 2.893 E-2 | 2.895 E-2 | 4.682 E-2 | 5.281 E-3 | 4.458 E-3 |
| Chirp | 7.423 E-2 | 7.530 E-2 | 6.647 E-2 | 4.928 E-3 | 2.081 E-3 |

**Table 1**

## Conclusions

The use of structured ANN with sine, cosine, and linear activation functions allows us to extract information from the network about the system. Input-output models with sigmoids as activation functions allow the use of neural networks for identification purposes, but no information can be obtained from the resulting structure. In addition, the use of a network structure related with state space description of systems allows us to obtain the equations. With the equations we can apply many of the classical methodologies dealing with non linear systems, usually related with the state space description of systems. The use of state space representation has as a drawback the difficulty to know the initial state, and that the neural network learns its own state representation, that perhaps has no match with the real systems states. On the other hand, if we use input-output models for systems with multiple inputs and multiple outputs we will obtain different equations for every output.

We use structured neural networks with sine and cosine as activation functions in order to obtain a network architecture which is equivalent to Fourier Series. This ANN allows us to compare the learning capabilities of the model and compare it with the theoretical results obtained from an analytical study of the system. Is also remarkable that the use structured ANN based on state space description of system will allow the use of ANN to extract information from unknown systems and not only the obtention of a system simulator.

## References

1. Jordan, M.I. Serial Order: A Parallel Distributed Processing approach. Institute for Cognitive Science Report 8604. University of California, Sand Diego. 1986
2. Elman, J.L. Finding Structure in Time. Center For Research in Language. Report 8801. University of California. San Diego 1988.
3. Narendra K.S., Parthasarathy K. Identification and Control of Dynamical Systems Using Neural Networks. IEEE Transactions on Neural Networks. Vol 1. N 1. March 1990.
4. Hopfield, J.J. Neural netwoks and physical systems with emergent collective computational abilities. Procc. National Academy of Science. 79:2554-2558 (1984)
5. Codina J., Morcego B., Fuertes J.M., Català A. A Novel Neural Network Structure for Control. IEEE Int. Conf. on Systems, Man and Cybernetics. 1339-1344. Chicago (1992)
6. Werbos P.J. Backpropagation Through Time: What it Does and How to Do it. Proceedings of the IEEE. Vol. 78 N. 10 pp. 1550-1560 October 1990.
7 Pao Y-H. Adaptive pattern recognition and Neural Networks. Addison-Wesley 1989