Neural Modules: Networks with Constrained Architectures for Nonlinear Function Identification

Bernardo Morcego⁽¹⁾, Josep M. Fuertes⁽¹⁾ and Gabriela Cembrano⁽²⁾

⁽¹⁾ Automatic Control and Computer Engineering Department

Universitat Politècnica de Catalunya

c/ Pau Gargallo 5, 08028 Barcelona, Spain

⁽²⁾ Institute of Cybernetics

Universitat Politècnica de Catalunya - Consejo Superior de Investigaciones Científicas

c/ Diagonal 647, planta 2, 08028 Barcelona, Spain

{bernardo, pepf}@esaii.upc.es, cembrano@ic.upc.es

Abstract: *The aim of this work is the design of a class of neural networks for nonlinear function identification: the so-called neural modules. A neural module is a neural network with an internal structure specially designed to be able to learn and mimic the behaviour of a certain class of dynamic systems. Neural networks are abstract models well suited for approximating nonlinear functions. The training cost and the structural complexity of neural networks can be drastically reduced if a-priori knowledge of the function to be learned is internally incorporated in the form of structural constraints. The resulting neural network has less parameters than a conventional one, much faster learning convergence and it can provide meaningful information about the learned nonlinear function. This paper describes the design of a useful set of neural modules for system identification and gives general guidelines for the design of neural modules. The resulting networks are evaluated and their use on general systems identification is pointed out.*

1. Introduction

The identification of linear systems has a well established theory and a wide range of efficient mathematical tools which provide a good basis for system analysis and controller design [15]. This is not the case, however, for nonlinear systems. Since no generally applicable techniques exist for the analysis of these systems, the identification and controller design are usually performed on a case-by-case basis. The main available techniques for nonlinear modelling can be classified into three categories ([1], [5]): functional series, block oriented and black box methods.

Neural network models are used as black boxes for the identification and control of nonlinear systems (e.g. [8] and [13]) due to their ability to approximate any continuous or discontinuous analytical function [7]. Different Artificial Neural Networks (ANN) structures have been used for system identification. Most of the research in this area has been carried out for discrete-time systems. In this context, some authors have chosen to

use an input-output representation of their system (e.g. [9]). Others have preferred the state-space representation (e.g. [2], [3], [12] and [14]). In both cases, the most commonly used neural architectures are feedforward networks rule and recurrent networks. One of the most common problems in using neural networks as black boxes is that the number of neurons increases rapidly with the order of the system and the learning process needs more time for each training pattern, and more training patterns, to be able to learn.

In order to circumvent that problem a new approach for systems modelling with neural networks has been developed. This method attempts to combine the ability of ANN to approximate any nonlinear function, with the clarity of information in block-oriented methodologies for system identification. This is the Modular Artificial Neural Network (MANN) approach. The heart of this framework, the so-called Neural Module (NM), is presented in this paper and some rules for its design are given.

This paper is organized as follows. Firstly, the concept of neural module is presented in section 2. Some guidelines for the design of NM's and one example are given in section 3. Section 4 presents the results of performance comparisons between NM's and plain ANN's. Finally, section 5 summarizes the conclusions of this paper and outlines further research directions.

2. Neural Modules

2.1 Definition

Most of the popular available ANN topologies (3-layer feedforward, fully recurrent, etc.) are flexible models that can adapt to a wide range of input-output mappings. This is their main advantage, but it can also be their main drawback because:

- when a network does not learn properly, it is not obvious whether this happens because the error minimum is difficult to reach or because the network configuration in use is not the correct one, and
- similar behaviours can be observed in radically different network structures and topologies.

The probability that a certain behaviour is attained by an ANN is intrinsically dependent on the architecture and the dynamics of the network, as Solla pointed out, in [16]. The procedure to increase the probability of the correct input-output mapping usually relies primarily on the learning algorithm (the dynamics of the network) rather than on topology selection. It is argued by Happel and Murre, in [6], that "a potentially much more versatile method of changing the probability distribution of network configurations is to constrain the network topology". This is the approach followed in this work. A sensible way to constrain network topologies is to design specific configurations for each family of input-output mapping or function to be learned, which is useful for the particular purpose of dynamic system identification.

A neural module is a neural network that, due to structural constraints, behaves inherently like a specified function (or set of functions). The procedure for tuning one specific behaviour within the family of functions structurally represented by the NM is the learning mechanism, which only concerns a subset of weights while the others are forced to remain fixed. For example, an NM can be designed to represent the input-output mapping of a typical nonlinearity in systems theory, like a saturation, a threshold, etc.

2.2 Types of neural modules

The classical literature on nonlinear control and dynamic systems usually describes nonlinearities distinguishing simple analytic functions from other piecewise-linear functions that arise naturally in physical systems. The former are usually smooth enough to be linearized by means of Taylor expansions in the neighborhood of an operating point and, consequently, to be treated as linear. The latter are grouped together to become the kernel of most studies on nonlinear dynamic systems. This set of basic nonlinear functions varies slightly from author to author and, in order to choose a working set of functions, three of them were considered: Cook ([4]), Netushil *et al.* ([11]) and Slotine and Li ([14]). The common set of functions obtained includes: the ideal relay (sign function), the dead zone, the saturation, the relay with hysteresis and the backlash.

A detailed study of this set of functions reveals that they share two behavioural traits, namely the threshold behaviour and the piecewise-linear behaviour. Hysteresis, for instance, is a function that requires the detection of two points in order to decide if the output is high, low or dependent on the previous input. However, it is often necessary to obtain different linear outputs according to the interval the input lies in. For instance, the dead zone function considers three input intervals $((-\infty, \alpha_1), (\alpha_1, \alpha_2)$ and $(\alpha_2, \infty))$ for which there is a different output linear equation for each one.

Consequently, it will be interesting for design purposes to classify the set of basic nonlinear functions into two groups, namely, threshold and piecewise-linear functions, and to model their behaviours with the aid of the neural networks that perform the basic threshold function and piecewise-linear function, explained in section 3.

3. Design of Neural Modules

The following conventions will be used:

- $x(k)$: input to a network or neuron at time step k , usually referred to as x or x_1, x_2 , etc.
- $x1, x2, x3, \dots$ and $n1, n2, n3, \dots$: activation of neuron 1, 2, 3, ... and a reference to neuron 1, 2, 3, ...
- $sig(x)$: sigmoidal function, defined as $sig(x) = \frac{1}{1 + \exp(-x)}$
- $sigs(x)$: symmetric sigmoidal function, defined as $sigs(x) = \frac{1 - \exp(-x)}{1 + \exp(-x)}$
- Symbols for sigmoidal and symmetric sigmoidal neurons:

3.1 Threshold behaviour

Sigmoidal functions provide a natural way to perform thresholding because their output is fundamentally different for inputs larger or smaller than zero. The basic thresholding scheme is:

$$detect(x) = sig(\omega_d(x - bias))$$

where $bias$ is the threshold value beyond which the output of $detect(x)$ changes from 0 to 1 and ω_d determines the abruptness of the change. That factor is interesting because it greatly influences the performance of the designs including thresholds.

The change abruptness is defined as the minimum distance between inputs corresponding to outputs zero and one. It is well known that sigmoidal functions only reach zero and one asymptotically, consequently, the change abruptness (noted γ) is redefined as the distance between inputs corresponding to a certain percentage (p) of outputs zero and one, i.e. to $0+p$ and $1-p$. This distance can be computed for $detect(x)$ from the definition of $sig(x)$:

$$\gamma(\omega_d, p) = 2 \cdot \frac{k}{-\omega_d}, \text{ where } k = \ln\left(\frac{1-p}{p}\right) \quad (3.2)$$

It may be noticed that the distance γ is inversely proportional to the weight ω_d , i.e. the larger the weight, the more abrupt the change. In order to design a detection neuron, it is only necessary to fix p and γ and then, to isolate ω_d for the values given in (3.2). For instance, let $p = 0.9$ (90%) and $\gamma = 0.1$; the resulting ω_d is 43.94. This means that $detect(x)$ has an uncertainty region smaller than 0.1 with $\omega_d \geq 44$.

Whenever large weights are inconvenient, the easiest way to enhance the detection and contain weight values is to cascade detection schemes. With the definition $detect(x) = sig(\omega_{d1} \cdot sigs(\omega_{d2}(x - bias)))$ a "second order" detection is obtained. With an uncertainty region as before ($\gamma < 0.1$), much smaller values of the weights ($\omega_{d1} = \omega_{d2} = 9.5$) are obtained. In order to allow different output patterns four types of detections are defined:

type 1: $detect(x) = sig(\omega_d(x - bias))$ type 2: $detect(x) = sig(-\omega_d(x - bias))$

type 3: $detect(x) = sigs(\omega_d(x - bias))$ type 4: $detect(x) = sigs(-\omega_d(x - bias))$

where ω_d is a positive value and the output of each threshold type is shown in Fig. 3.1.

Fig. 3.1: Output samples of detection types 1, 2, 3 and 4

3.2 Piecewise-linear behaviour

It is known that the response of symmetric sigmoidal neurons is linear in a neighbourhood of $x = 0$. In order to force the linear response it is necessary to reduce the input interval to a suitable one. For instance,

$$linear(x) = \beta \omega_r \cdot sigs\left(\frac{x}{\omega_r}\right)$$

behaves linearly for any symmetric interval if ω_r is chosen appropriately. The output slope is controlled by β (a unitary slope requires $\beta = 2$ because the derivative of $sigs(0) = 0.5$). The error produced by this approximation depends on the input interval and on ω_r . A

sample input interval $(-10, 10)$ would lead to a maximum error of 0.066 with $\omega_s = 50$ and if $\omega_s = 20$, the error would rise to 0.406.

In order to make use of the linear behaviour it is necessary to be able to control the input interval for which it should be exhibited. Since it is possible to detect whether x belongs to a certain interval or not, that information will be used to alter the response of $linear(x)$ when the input value falls outside the interval of interest.

Let the interval of interest be $(-\alpha, \alpha)$. It is easy to see that this symmetry does not imply a loss of generality. Let $n1$ and $n2$ be two neural configurations performing the $detect(x)$ function on $-\alpha$ and α . The ideal activations $x1$ and $x2$ of $n1$ and $n2$ are given in Table 3.1. Now consider the following modified linear behaviour:

	$(-\infty, -\alpha)$	$(-\alpha, \alpha)$	(α, ∞)
$x1$	1	0	0
$x2$	0	0	1

Table 3.1

$$modLinear(x) = \beta\omega_r \cdot sigs\left(\frac{x}{\omega_r} - \omega_s x1 + \omega_s x2\right)$$

where ω_s is chosen appropriately (usually a large value, which must be empirically found). When either $x1$ or $x2$ is active, $modLinear(x)$ is fully saturated to its minimum or maximum value without being affected by the term $\frac{x}{\omega_r}$. The output of $modLinear(x)$ is

shown in the first row of Table 3.2 and it is obviously not continuous. In order to eliminate the jumps between intervals compensation neurons will be used:

$$compens1(x) = \beta\omega_r \cdot sigs\left(\omega_s x1 + \frac{\alpha}{\omega_r}\right) \quad \text{and} \quad compens2(x) = \beta\omega_r \cdot sigs\left(-\omega_s x2 - \frac{\alpha}{\omega_r}\right)$$

Note that neither $compens1(x)$ nor $compens2(x)$ are directly dependent on the network input x . They depend on x only through $n1$ and $n2$. Let us consider again the outputs of those three functions for each interval in Table 3.2.

x	$(-\infty, -\alpha)$	$(-\alpha, \alpha)$	(α, ∞)
$modLinear(x)$	$-\beta\omega_r$	$\frac{\beta x}{2}$	$\beta\omega_r$
$compens1(x)$	$\beta\omega_r$	$\frac{\beta\alpha}{2}$	$\frac{\beta\alpha}{2}$
$compens2(x)$	$-\frac{\beta\alpha}{2}$	$-\frac{\beta\alpha}{2}$	$-\beta\omega_r$
Total	$-\frac{\beta\alpha}{2}$	$\frac{\beta x}{2}$	$\frac{\beta\alpha}{2}$

Table 3.2

The function $pieceLin(x) = modLinear(x) + compens1(x) + compens2(x)$ is the desired function that responds linearly for a specified input interval and constant on the rest of the input range. The design of $pieceLin(x)$ uses three sigmoidal and one linear neurons: a total of 4 neurons and 10 weights (provided the detection signals are given).

The accuracy of $pieceLin(x)$ depends on the accuracy of $modLinear(x)$, as studied previously, and on the transition intervals of detection signals. The smaller those intervals, the better the performance. The maximum error of a typical $pieceLin(x)$ function ($\omega_t = 25$, $\omega_s = 18$ and $\alpha = 1$) is 0.026 with first order detection schemes ($\omega_{d1} = 50$) and 0.0021 with second order detection schemes ($\omega_{d1} = \omega_{d2} = 35$).

3.3 Guidelines for the design of neural modules

The design of neural modules to model non-analytic nonlinear functions is not an obvious task. There exists no formal methodology to assist in it and it will always be necessary to carefully analyze the function in question. Nevertheless, the development of the library of neural modules referred to in [10] offers a broad set of examples and some general guidelines can be extracted from it:

- Detection schemes are helpful to classify the inputs in different intervals.
- Any piecewise-linear function can be modelled using an adequate addition of the already designed $pieceLin(x)$ function.
- Neural module interconnection is an endless source of function generation. For instance, the dead zone and the rate limiter functions can be modelled from the saturation NM, the backlash can be modelled from the dead zone NM, etc.

In order to illustrate how NM's may be designed using the threshold function, the design of the hysteresis neural module is described in more detail.

The hysteresis is a multivalued nonlinearity defined by:

$$\text{hysteresis}(x(k)) = \begin{cases} -1 & \text{if } x \leq \alpha_1 \\ \text{hysteresis}(x(k-1)) & \text{if } \alpha_1 < x < \alpha_2 \\ +1 & \text{if } x \geq \alpha_2 \end{cases}$$

A neural network that performs like a relay with hysteresis is shown in Fig. 3.2. The two input neurons perform threshold functions of type 3 to determine the interval $((-\infty, \alpha_1)$, (α_1, α_2) or (α_2, ∞)) where the inputs lie. The output neuron uses that information to decide if the corresponding output should be high, low or the previous output.

Fig. 3.2: Hysteresis neural module

Three neurons (with symmetric sigmoidal activation functions) and seven weights are shown in Fig. 3.2. These weights labelled ω_t and ω_s are fixed, i.e. not learnable, and their values provide a means to control the interval discrimination sensitivity and output activation steepness, respectively. In this case $\omega_t \geq 30$ and $\omega_s \geq 10$ would suffice. The weights labelled $-\alpha_1 \cdot \omega_t$ and $-\alpha_2 \cdot \omega_t$ are learnable, i.e. the learning algorithm will determine the parameters of the specific hysteresis function being learned using input-output examples.

4. Evaluation of the Neural Modules

The collection of 7 NM's mentioned next in this section belongs to a library of nonlinear modules (described in full detail in [10]) based on the aforementioned set of basic nonlinear functions, also inspired by the Simulink library of nonlinear functional blocks. Only those modules directly related to the threshold and the piecewise-linear behaviours are referred to in this paper.

4.1 Experiments

This section presents a collection of experiments designed to compare the capabilities of NM function approximation versus those of more popular neural topologies, like the three and four-layered ANN. There is a common framework for all the experiments that will be outlined next.

An experiment comprises different learning runs and each run is composed of 45 tests. Given a neural topology and a function to be learned, every test consists on training that topology until a prefixed degree of accuracy (the error minimum) is attained.

The learning algorithm, and its related parameters, remain constant for each test on a learning run. The learning speed, measured with the number of training cycles to attain a prefixed error minimum, is used as an index to compare the different topologies.

Problem	Net Type	Converged Nets	Mean	St. dev.	Max. cycles
Equality	NM	45 / 45 / 45	43.6	13.4	1000
Equality	1-12-8-1 ANN	0 / 41 / 45	3318.2	1140.6	10000
Saturation	NM	45 / 45 / 45 / 45 / 45	2.7	0.3	1000
Saturation	1-3-1-1 ANN	31 / 41 / 32 / 35 / 28	624.2	299.5	2000
Dead Zone	NM	45 / 45 / 45 / 45	39.6	1.8	1000
Dead Zone	1-5-1 ANN	28 / 42 / 22 / 2	1789.9	833.0	4000
Hysteresis	NM	45 / 45 / 43	276.3	113.9	1200
Hysteresis	1-3-1 delayed out. ANN	21 / 13 / 1	12947.1	994.7	15000
Rate limit	NM	45 / 45	3.4	0.8	1000
Rate limit	1-8-10-1 delayed inp. ANN	24 / 0	645.1	650.0	2000
Backlash	NM	45 / 45 / 44 / 45	9.3	2.0	1000
Backlash	1-6-1 delayed out. ANN	45 / 16 / 45 / 0	577.9	109.1	2000

Table 4.1: Summary of the NM tests

Several experiments were designed to evaluate the learning performance of the following neural modules: equality, saturation, dead zone, hysteresis, rate limiter and backlash. The results of those experiments are summarized in Table 4.1.

Each neural module was tested against a large number of neural networks and a total of 3420 neural networks were trained, but the following results only compare the best straightforward ANN with the corresponding NM. Besides, a number of different experiments was carried out to cover a reasonable parameter range of the function to be

learned, and Table 4.1 reflects only the averages over those experiments for the sake of clarity. More detailed results of these tests may be found in [10].

The best conventional ANN modelling the equality, saturation, dead zone, hysteresis, rate limiter and backlash functions were respectively: a four-layered ANN type 1-12-8-1 (meaning the number of neurons in each layer, beginning from the input layer), a four-layered ANN type 1-3-1-1, a three-layered ANN type 1-5-1, a three-layered ANN type 1-3-1 with one tapped-delay output, a three-layered ANN type 1-8-10-1 with one tapped-delay input, and finally a three-layered ANN type 1-6-1 with one tapped-delay output.

Each row in the table contains the results of one network for the complete set of experiments dealing with a given problem. The first two columns show the name of the problem and network type. The column entitled *Converged Nets* gives the number of networks that attained the error minimum within the maximum number of cycles permitted in each experiment. Accordingly, '0 / 45' means that in the first experiment no network achieved convergence and in the second one all of them converged.

The following two columns (*Mean*, and *St. dev.*) give the mean value and the standard deviation of the number of cycles that were required for convergence in the corresponding set of experiments. For instance: three experiments were carried out with the function equality(x); the NM attained the error minimum in a mean of 43.8 cycles on the first experiment, 33.6 cycles on the second one and 53.5 cycles on the third one; the resulting *Mean* value is the average of those three quantities: 43.6.

4.2 Analysis of the results

The most obvious conclusion inferred from the results on Table 4.1 is that the tested neural modules learn faster than conventional neural networks. The *Mean* results make the good performance of NM's apparent: in general, they are at least one order of magnitude better than conventional ANN's. It is, therefore, legitimate to state that equalityNM, saturationNM, deadZoneNM, hysteresisNM, rateLimiterNM and backlashNM learn the functions faster than conventional ANN's.

Another important conclusion derived from the figures in Table 4.1 is that NM's are able to represent generic functions, i.e. the different instances obtained with all possible values of the parametric range of a given function. Conversely, conventional networks may learn acceptably a particular instance and show problems on another (like in the dead zone problem, for instance).

5. Conclusions

A new approach to neural network modelling of nonlinear systems has been presented in this paper. It is an attempt to combine the capabilities of neural networks for function approximation with classical block oriented techniques for system modelling.

The concept of Neural Module (a neural network with structural constraints designed to represent a certain class of dynamic behaviours) is introduced. General guidelines for their design are given and it is shown that the convergence capabilities and the learning time of NM's are significantly better than those of straightforward neural networks, for a variety of nonlinear systems.

The advantages of the NM approach to systems modelling are, primarily, the ability to use a-priori information about the physical process being modelled and the amenability of the resulting models to the extraction of meaningful information about the system, as well as the improved convergence and learning times.

Further work following this research line is centered on a thorough validation of the NM concept as a general functional block, carrying out experiments with more complex systems and introducing the uncertainties usually found in physical systems, like noise or non-ideal behaviour of components.

Acknowledgements: Research on neural networks for nonlinear control at the Institute of Cybernetics is in part supported by the research grant CICYT - TAP94 - 0552 - C03 - 01 of the Spanish Science and Technology Council. The authors B. Morcego and J.M. Fuertes are members of CERCA (Collective for the Study and Research of Control and Automation).

6. Bibliography

- [1] Billings, S.A., "Identification of Nonlinear Systems - a survey", *IEEE proceedings* Vol 127, N. 6, pp 272-285, 1980.
- [2] Cembrano, G., Wells, G. "Neural Networks for Control", *Application of Artificial Intelligence in Process Control*, L. Boullart, A. Krijgsman, R.A. Vingerhoeds (eds.), Pergamon Press, 1992.
- [3] Codina, J., Aguado, J.C., Fuertes, J.M., "Neural Network Identification of Nonlinear Systems Using State Space Techniques", Sprann'94, L'ille, 1994.
- [4] Cook, P.A. , *Nonlinear Dynamical Systems*, Prentice Hall, 1986.
- [5] Haber, R., Unbenhauen, H., "Structure Identification of Non Linear Dynamic Systems - A survey on input/output Approaches", *Automatica* Vol 26, N. 4, pp651-677, 1990.
- [6] Happel, B.L.M., & J.M.J. Murre (1994). The design and evolution of modular neural network architectures. *Neural Networks*, 7: 985-1004.
- [7] Hornik, K.M., Stinchcombe, M., White, H., "Multilayer Feedforward Networks are Universal Approximators", *Neural Networks*, 2:359-366, 1989.
- [8] Hunt, K.J., Sbarbaro, D., Zbikowski, R., Gawthrop, P.J., "Neural Networks for Control Systems - A Survey", *Automatica*, vol. 28, N. 26, 1993.
- [9] Miller, T., Sutton, R.S., Werbos, P.J. (eds.) *Neural Networks for Control*, MIT press, 1990.
- [10] Morcego, B., Fuertes, J.M., Cembrano, G. "Neural Modules. Design of a Nonlinear Dynamic Systems Modelling Library of Neural Networks", Technical Report, Computer Science School, Polytechnical University of Catalonia, 1996.
- [11] Netushil, A., Baltrushevich, A., Burliaev, V., Kuzin, P., Alexandrovski, N., *Automatic Control Theory* (Spanish version), Mir, 1987.
- [12] Nguyen, D.H., Widrow, B., "Neural Networks for Self-learning Control Systems", *IEEE Control Systems Magazine* pp18-23, April, 1990.
- [13] Pearlmutter, B., "Gradient Calculations for Dynamic Recurrent Neural Networks: a Survey", *IEEE Transactions on Neural Networks*, vol. 6, N. 5, 1995.
- [14] Slotine, J.E., Li, W. *Applied Nonlinear Control*, Prentice-Hall, 1991.
- [15] Söderström, T., Stoica, P., *System Identification*. Prentice-Hall, 1989.
- [16] Solla, S.A. (1989). Learning and Generalization in Layered Neural Networks: the Contiguity Problem. In L. Personnas and G. Dreyfus (eds.), *Neural Networks: from Models to Applications*. Paris: I.D.S.E.T., 1989.