Minimum Description Length Quality Measurues for Modular Functional Network Architectures

A. S. Cofiño and J.M. Gutiérrez
Dept. of Applied Mathematics, University of Cantabria, Santander, Spain

gutierjm@unican.es,
http://personales.unican.es/gutierjm

María L. Ivanissevich
Universidad Nacional de la Patagonia Austral, Río Gallegos, Argentina

Abstract

Modular neural networks (MNNs) are increasingly popular models for dealing with complex problems constituted by a number of dependent subtasks. An important problem on MNNs is finding the optimal aggregation of the neural modules, each of them dealing with one of the subproblems. In this paper, we present a functional network approach, based on the minimum description length quality measure, to the problem of finding optimal modular network architectures for specific problems. Examples of function approximation and nonlinear time series prediction are used to illustrate the performance of these models when compared with standard functional and neural networks.

Keywords

Neural networks, functional networks, modular systems, model selection methods, time series prediction.

1 Introduction

Neural networks are simple and efficient computing techniques which have proven their efficiency in many practical problems [1]. One of their most popular applications of these models is approximating a target mapping $y(\mathbf{x}) = y(x_1, \ldots, x_m)$ from a sampled data set, $(\mathbf{x}_1, y_1)$, ..., $(\mathbf{x}_n, y_n)$. To this aim, an specific architecture of processing units (neurons) organized in layers and connected by weights is designed (multilayer feedforward networks, MFNs). Then, the set of input-output data representing the problem of interest, $\mathbf{x}_i - y_i$, $i = 1, \ldots, n$, is used to fit the weights using backpropagation-like algorithms. For instance, Fig. 1(a) shows the structure of a MFN consisting of an input layer with two processing units, a hidden/intermediate layer with four units, and a single output unit. The computation performed by a single neuron unit is schematically shown in Fig. 1(b) (see [2], chapter 1, for a detailed overview of neural networks). Therefore, the network in 1(a) maps a 2D space into a 1D space as follows:

$$y_1 = f(\sum_{j=1}^{4} W_{j1} h_j - \Theta_1) = f(\sum_{j=1}^{4} W_{j1} f(\sum_{i=1}^{2} w_{ij} x_i - \theta_i) - \Theta_1), \tag{1}$$

where W_{j1}, w_{ij}, Θ_1, and θ_i are the parameters of this fully-connected network ($4 + 8$ weights and $1 + 4$ bias) to be adjusted with the sampled data to approximate the original mapping.

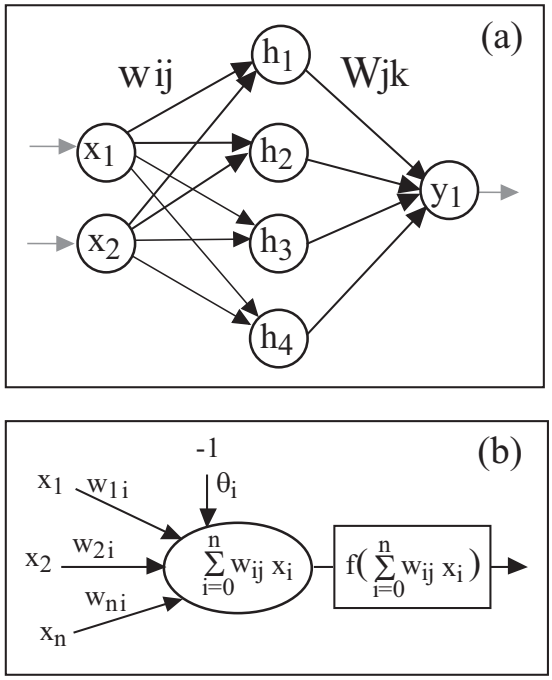

Figure 1: (a) A simple MFN with two inputs and a single output; (b) Schematic neuron computation consisting of a weighted sum of the inputs from other neurons, including a threshold value θ_i, and a processing activity function $f(x)$ (e.g., the sigmoid logistic function $f(x) = 1/(1 + e^{-x})$).

On the one hand, the main advantage of neural networks is their simple application to real-world problems, which allow us obtaining valuable approximations of complex input-output mappings (universal approximation property [3]). On the other hand, the key disadvantage of these models is their rigid structure of fully connected layers with many degrees of freedom that may overfit the data, train slowly, or converge to local minima.

In recent years, several attempts for obtaining more flexible neural structures beyond the fully connected rigid topology of neural networks have been developed using the idea of modularity (hybrid neural systems [4], Modular Neural Networks (MNNs) [5], mixtures of experts [6], etc.). The concept of modularity is linked to the notion of local computation, in the sense that each module is an independent system and interacts with others in a whole architecture, in order to perform a given task.

For instance, the concept of module in MNNs is defined as a group of unconnected neurons, each connected to the same set of nodes. The local structure and the complexity reduction sustained by modularity in MNNs have shown to overcome some of the problems of fully connected MFNs [7]. However, in order to have meaningful and efficient models, each module has to perform an interpretable and relevant function according to the mathematical or physical properties of the system. Unfortunately, it is unclear how to best design such a modular topology based on the data. For example, given the trivial modular network shown in Fig. 2(a) (a MFN consisting of four modules: an input layer, two hidden layers and an output layer, respectively), several nontrivial modular networks can be easily obtained by splitting up some layers into sub-layers, hereby reducing the number of weights (two of such modular networks are shown in Fig. 2(b) and (c)). However, there is no general procedure to design optimal modular structures according to some available domain knowledge.

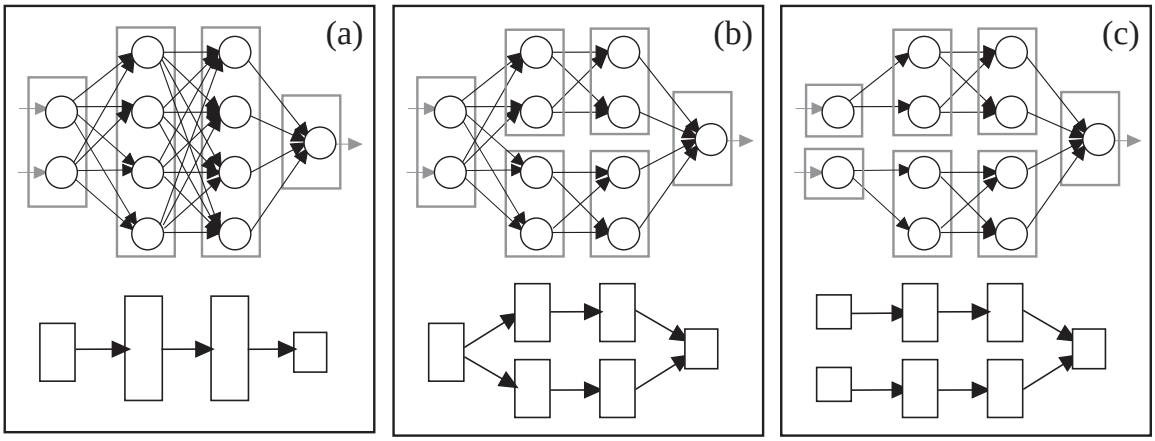

Figure 2: (a) A fully connected feedforward 2:4:4:1 network; (b) and (c) modular neural networks with different aggregation configurations.

This problem has been partially addressed with functional networks (see [2] for an introduction to functional networks). In a recent work, Cofiño and Gutiérrez [8] presented an hybrid model combining both functional and neural networks for obtaining efficient modular architectures from qualitative and quantitative knowledge. Domain knowledge

was used for obtaining an appropriate functional network for the given problem; afterwards, the resulting processing units were trained with neural networks using the available data. However, this technique lacks of an efficient adaptive method for determining the computational requirements needed for each of the processing units according to a problem of interest. In this paper, we present a quality measure derived from the Minium Description Length (MDL) criterion for obtaining the optimal modular functional architecture for a given problem, both in terms of network topology and modules training.

In section 2 we briefly introduce the functional network paradigm and describe hybrid modular functional-neural networks. In Sec. 3 the MDL quality measure is analyzed and applied to our problem; some comparative examples are presented to illustrate this new technique. Finally, in Sec. 4 the resulting models are applied to a nontrivial problem in nonlinear time series prediction.

2 Modular Functional Networks

Functional networks are a generalization of neural networks which allow combining both domain and data knowledge to develop optimal functional structures for several problems. The topology of the network (functional units and connections) is obtained using qualitative knowledge of the problem at hand (symmetries or other functional properties) which are analyzed and simplified using functional equations [9]. Then, the resulting functional units are fitted to data as linear combinations of a predetermined set of appropriate functions such as polynomials, trigonometric Fourier functions, etc. (see [10] for an overview of functional network applications).

Apart from the input, output and hidden layers of processing units, a functional network also includes a layer of intermediate units, which do not perform any computation, but only store intermediate information and may force the outputs of different processing units to be equal. For instance, in Fig. 3(a) the output unit u is connected to the processing units I and F_3, so their outputs must be coincident. Therefore, this network is the graphical representation of the functional equation:

$$F_4(x,y) = F_3\left(F_1(x,z), F_2(y,z)\right), \ \forall x,y,z. \tag{2}$$

This property characterizes the so-called general associativity models, which generalize the class of models which combine the separate contribution of independent variables.

The above functional equation imposes some constraints in the neuron functions $F_1,\ldots,F_4$, which may lead to a simplification of the functional structure. In this case, the solution of the functional equation is:

$$\begin{aligned} F_1(x,y) &= f_4^{-1}\left(f_1(x) + f_6(y)\right), \\ F_2(x,y) &= f_5^{-1}\left(f_2(x) - f_6(y)\right), \\ F_3(x,y) &= f_3^{-1}\left(f_4(x) + f_5(y)\right), \end{aligned} \tag{3}$$

$$F_4(x,y) = f_3^{-1}\left(f_1(x) + f_2(y)\right). \tag{4}$$

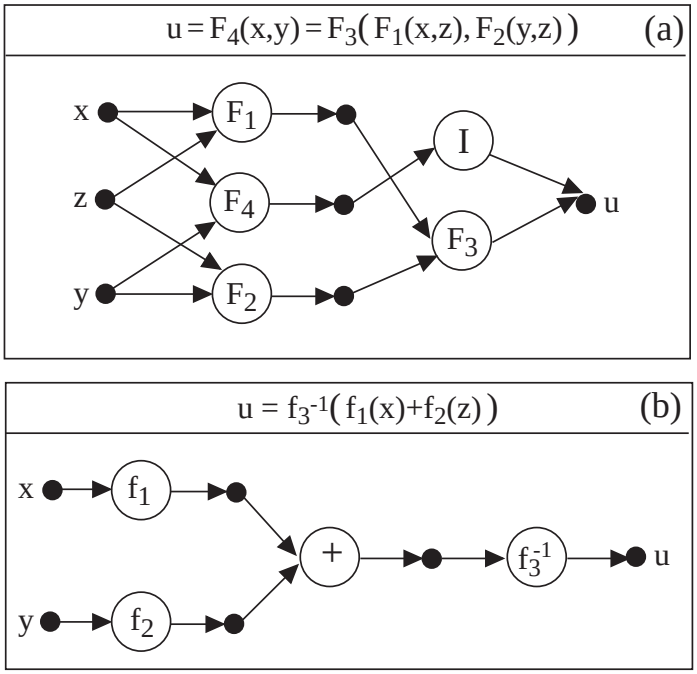

Figure 3: (a) Functional network (dots represent intermediate units, circles represent the processing units, and I denotes the identity function); (b) simplified functional network.

where $f_1, \ldots, f_6$ are arbitrary continuous and strictly monotonic functions.
Replacing (3) in (2) we obtain the simplest functional expression of general associativity models models

$$u = F_4(x, y) = F_3\left(F_1(x, z), F_2(y, z)\right) = f_3^{-1}\left(f_1(x) + f_2(y)\right), \tag{5}$$

defining the functional network in Figure 3(b).

The problem of learning the above functional network now reduces to estimating the neuron functions f_1, f_2 and f_3 from the available data of the form (x_i, y_i, u_i), $i = 1, \ldots, n$. To this aim each of the functions f_s in (5) is supposed to be a linear combination of known functions from given families $\phi_s = \{\phi_{s1}, \ldots, \phi_{sm_s}\}$, $s = 1, 2, 3$, i.e.,

$$\hat{f}_s(x) = \sum_{i=1}^{m_s} a_{si}\phi_{si}(x);\ s = 1, 2, 3, \tag{6}$$

where the coefficients a_{si} are the parameters of the functional network (see [11] for implementation details).

The efficiency of functional networks depends on the appropriate choice of the family of functions in (6). However, in general it may not be possible to specify a convenient family of functions. In these cases, it would be desirable to learn the neuron functions using some standard non-parametric technique, such as neural networks. This idea was presented in [8] with the so-called hybrid functional-neural modular networks. Fig. 4 illustrates the architecture resulting from the functional network in Fig. 3(b) when using MFNs for approximating each of the processing units f_1, f_2, and f_3. The resulting

models can be considered particular types of modular neural networks, where some of the processing units may be known (such as the sum operator in module 3) and other may be unknown (modules 1, 2 and 4). Therefore, the parameters can be estimated from data using modular backpropagation [12]. The resulting network is an optimal modular neural network architecture for this problem.

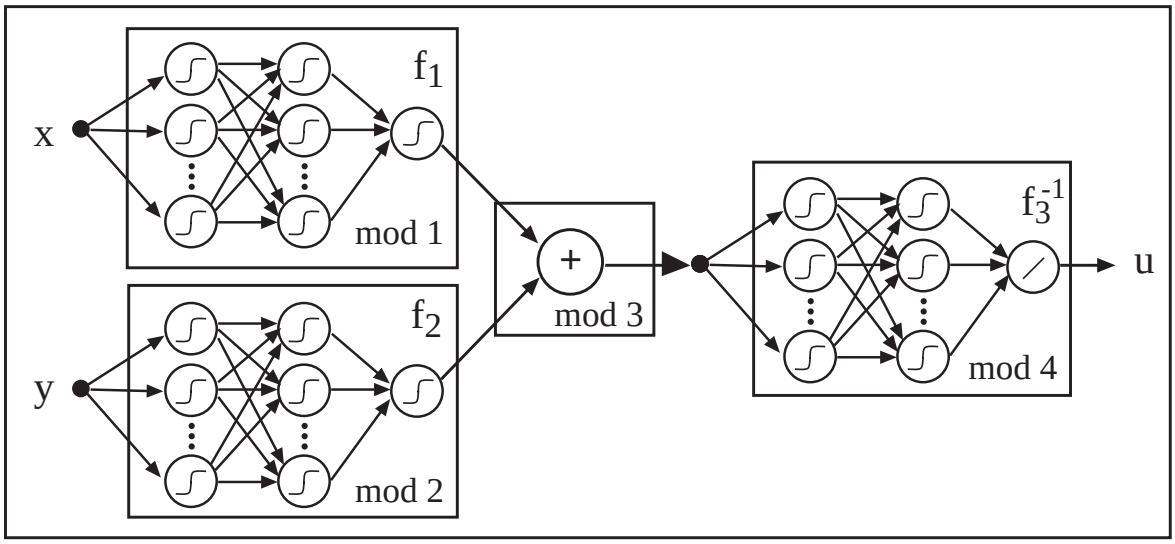

Figure 4: Modular neural network resulting form the functional network in Fig. 3(b).

In order to illustrate the performance of the above methodologies, we have implemented some modular architectures using the Matlab Neural Network Toolbox [13]. We generated some data by randomly simulating a hundred of points from the model

$$u_i^3 = x_i + x_i^2 + y_i^2 + y_i^3 + \epsilon_i;\ x_i, y_i \in (0,1);\ i = 1, \ldots, 100, \tag{7}$$

where ϵ is a noise term given by a uniform random variable in $(-0.01, 0.01)$.

As a first experiment, the MFN in Fig. 2(a) was used as a black-box method for this problem, fitting the weights to data using the backpropagation algorithm; we performed ten experiments with networks consisting of two neurons in each of the hidden layers (2:2:2:1 MFN with 15 parameters) and random initial weight configurations. The average Root Mean Squared Error (RMSE) obtained in these experiments was 0.0074. The same experiment was repeated considering three neurons in each of the hidden layers (2:3:3:1 with 25 parameters) obtaining a significant improvement (RMSE=0.0031).

Now, assuming that the model satisfies the generalized associativity property, the hybrid model shown in Fig. 4 was applied, considering MFNs with a single hidden layer of two neurons for each of the neural modules ($7 \times 3 = 21$ parameters); in this case, the average RMSE after ten initializations and training of the model was 0.0027. Therefore,

hybrid neural-functional networks outperforms standard neural nets due to the optimal network structure applied to the problem.

Finally, the same experiment was performed assuming also that the functions involved in the problem are polynomials (domain knowledge). Then we can use the functional network shown in Fig. 3(b) selecting an appropriate functional family, say $\{1, x, x^2, x^3\}$, obtaining the model:

$$\begin{aligned} f_1(x_1) &= 0.3350 + 0.322x_1 + 0.352x_1^2 - 0.0095x_1^3, \\ f_2(x_2) &= 0.334 - 0.00372x_2 + 0.331x_2^2 + 0.339x_2^3, \\ f_3(y) &= 0.659 + 0.0322y - 0.037y^2 + 0.346y^3, \end{aligned}$$

with a RMSE = 0.0024. Note how the error is reduced, even though the functional model consists only of 12 parameters. In this case, the efficiency of the functional network hinges on the knowledge of an appropriate functional family for the neuron functions.

This example illustrates how increasing degrees of knowledge allow obtaining more efficient modular architectures for approximating the problem of interest. However, in the above examples the number of parameters chosen for estimating each of the processing units was selected by a trial and error procedure. In this paper we are interested in automatic procedures for selecting the most efficient configuration of parameters for learning each of the modules. In the next section we illustrate how a technique from information theory can be applied to this problem.

3 Minimun Description Length for Model Selection

The problem of model selection has been extensively analyzed from several points of view [14]. Among several proposed methods, the Minimum Description Length (MDL) algorithm has proven to be simple and efficient in several problems. The description length measure rewards the quality of the fit, but penalizes the information required to store the data using the model. In our case, the model is given by the weights of the different neural networks used for approximating each of the processing units. According to Rissanen [14], if the parameter θ_k (a real number) is estimated using a set of data x of size n, then it can be encoded using $log_2(\theta_k) + 1/2 log_2(n)$ bits. Therefore, the information required to store the data using a model with parameters $\theta = \{\theta_1, \ldots, \theta_k\}$ would be

$$-\sum_{i=1}^{k} log_2(\theta_i) + \frac{k\, log_2(n)}{2}. \tag{8}$$

On the other hand, the quality of the model is taken into account by considering the log-likelihood of the data given the model, $logf(x|\theta)$. If the errors $e_i = x_i - \hat{x_i}$ are supposed to be normally distributed, then it can be shown that a measure of this quality is

$$\frac{n}{2} \log \left(\frac{1}{n} \sum_{j=1}^{n} e_j(\theta)^2 \right). \tag{9}$$

From (8) and (9) the description length of a model results

$$DL(\theta) = -\sum_{i=1}^{k} \log \pi(\theta_i) + \frac{k\, logn}{2} + \frac{n}{2} \log \left(\frac{1}{n} \sum_{j=1}^{n} e_j(\theta)^2 \right). \tag{10}$$

where $\pi(\theta_i)$ is the "prior" probability given for some human expert. The first term in (10), that can be removed, allows including the feeling of a human expert about the models qualities. The second term penalizes the complexity of the model, and the third term rewards the quality of the fitting.

A computer implementation of this selection method can be done in several ways. In this paper we consider a simple forward search method, which starts with networks with a single neuron in each hidden layer. Then, it incrementally analyzes the DL of the models resulting of incorporating a neuron to the previous model, selecting the one leading to the smallest DL value. The process continues until no further improvement in the DL is obtained.

As an illustrative application of this algorithm, we consider the example in (7). We start with three networks with a single hidden layer with only one neuron (one for each of the modules in Fig. 4); then, we apply the forward iterative method, looking for improvements in the DL by considering all the models resulting of adding a single hidden neuron to some of the neural networks. In the table 1 we show details of the first ten iterations of the algorithm (see also Fig. 4).

Step	Network	Parameters	RMSE	$log_2(MSE)$	DL
0	$1;1;1$	17	0.0156	-12.01	-788.3
1	$1;1;2$	20	0.0067	-14.43	-947.5
2	$1;1;3$	**23**	**0.0038**	**-16.04**	**-1051.4**
3	$1;1;4$	26	0.0037	-16.14	-1050.2
4	$2;1;4$	29	0.0037	-16.15	-1042.9
5	$2;2;4$	32	0.0037	-16.16	-1035.7
6	$2;2;5$	**35**	**0.0028**	**-16.93**	**-1081.1**
7	$3;2;5$	38	0.0028	-16.93	-1073.5
8	$3;2;6$	41	0.0028	-16.93	-1065.2
9	$3;3;6$	44	0.0028	-16.94	-1058.2
10	$3;4;6$	47	0.0028	-16.94	-1050.5

Table 1: Forward iterative method for selecting the optimal hybrid neural model. *Step* denotes the iteration step; *network* shows the best model found at a given step $-a;b;c$ stands for the number of neurons in the hidden layer of each of the three networks 1:a:1, 1:b:1, 1:c:1, used for approximating the modules, respectively$-$; the four last columns indicate the number of *parameters*, the $RMSE$, the error term associated with the DL and the total description length DL of the successive models.

From the initial trivial model $1;1;1$ (one hidden neuron in the networks of the three modules), the models $2;1;1$, $1;2;1$, and $1;1;2$ are checked in Step 1, resulting the last

one as the best model at this step. The process then continues by incrementally adding hidden neurons to the best model found in the previous step. From this table, we can see how local minima in the DL are attained at the second and sixth steps, whereas the training error is a non-decreasing function (see also Fig. 5). Therefore, the forward method will stop at the second step, leading to a sub-optimal solution. In this paper we use this simple search procedure for the sake of simplicity, since we are only interested in illustrating the possibilities of the method (we leave a more detailed analysis of a convenient search technique for a future paper).

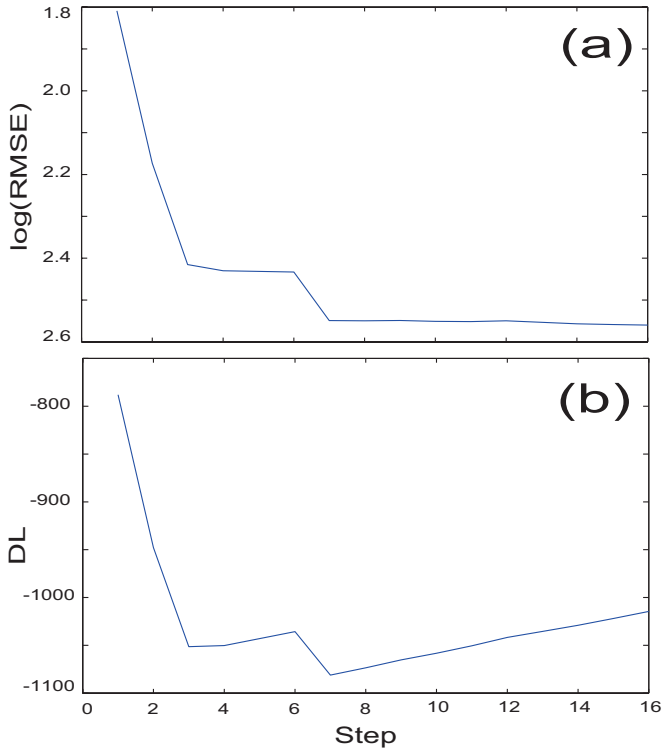

Figure 5: Search steps of the model selection algorithm. (a) error term associated with the DL; (b) DL measure of the models.

In the following section, the present model is applied to a non-trivial problem of time series forecasting. We illustrate how valuable information can be obtained from the analysis of the final model resulting from the optimization process.

4 Nonlinear Time Series Prediction

A challenging problem for time series analysis techniques is modeling nonlinear chaotic systems from observed data. Neural networks [15] and functional networks [10] have been efficiently applied to this problem inferring deterministic nonlinear models from data. In this section we show that the method introduced in the previos section allow us obtaining optimal MNNs for this task.

We analyze the special case of 2D delayed maps of the form $x_n = F(x_{n-1}, x_{n-2})$; one of the most illustrative and widely studied of these models is the non-differentiable Lozi map [16]:

$$x_n = 1 - 1.7\,|x_{n-1}| + 0.5\,x_{n-2}. \tag{11}$$

We generated a time series $\{x_n\}$, $n = 1, \ldots, 200$, corresponding to the initial conditions $x_0 = 0.5$ and $x_1 = 0.7$. From (11) we can see how the model satisfies the general associativity property (in this particular case the function $f_3^{-1}(x)$ is simply the identity function). Note also that there is no obvious choice for the functional families to be used in this problem; therefore, hybrid functional-neural networks are necessary in this case. In the following we apply the above model selection technique to this problem, analyzing the resulting model for discovering particular features underlying the data. We start with the hybrid network given in Fig. 4, and consider three neural networks with two hidden layers for approximating each of the functional units (a_1:a_2; b_1:b_2; c_1:c_2 denote the number of neurons in each of the hidden layers for the neural networks: 1:a_1:a_2:1 for the first module, and so on).

Table 2 shows the results of the search process. From this table we can see how the only module growing in the search process is the second one, associated with the non-linear term of the Lozi map (11). On the other hand, the networks associated with f_1 and f_3 remain unchanged, since they are associated with the linear terms in (11). The optimal model, found after six iteration steps, contains 44 parameters and has a RMSE=7.04×10^{-5}. If we compare these results with the ones reported in [17], where Fourier functional families were used for approximating f_1, f_2, and f_3, we can see how the model selection method presented in this paper clearly outperforms standard functional models for this problem.

Step	Network	Parameters	$RMSE$	DL
0	1:1; 1:1; 1:1	17	0.171	-64.758
1	1:1; 1:2; 1:1	20	0.016	-192.70
2	1:1; 2:2; 1:1	24	4.8×10^{-3}	-255.05
3	1:1; 3:2; 1:1	28	7.14×10^{-4}	-356.75
4	1:1; 3:3; 1:1	33	4.65×10^{-4}	-370.47
5	1:1; 4:3; 1:1	38	2.52×10^{-4}	-394.94
6	**1:1; 4:4; 1:1**	**44**	7.04×10^{-5}	**-455.32**
7	1:1; 5:4; 1:1	50	8.12×10^{-5}	-433.94
8	1:1; 5:5; 1:1	57	6.84×10^{-5}	-428.53
9	1:1; 6:5; 1:1	64	8.09×10^{-5}	-403.49
10	1:1; 7:5; 1:1	71	6.16×10^{-5}	-403.88

Table 2: Forward iterative method for selecting the optimal hybrid neural model for the Lozi map.

Moreover, the final distribution of neurons among the different modules gives us a valuable information about the particular structure of the model underlying the data. In this case,

if we start with empty neural networks the associative model (5) could be simplified to $F(x,y) = lin(x) + f_2(y)$, since both f_1 and f_3 would result to be linear in the final model ($lin(x)$ stands for a linear function). Note how this structure corresponds to the Lozi map $x_n = F(x_{n-2}, x_{n-1}) = 1 - 1.7\,|x_{n-1}| + 0.5\,x_{n-2} = lin(x_{n-2}) + f_2(x_{n-1})$. Therefore, besides of approximating the model associated with a given data set, the method presented in this paper allows obtaining some qualitative properties about the structure of the model.

Acknowledgments

The authors are grateful to the University of Cantabria, the Instituto Nacional de Meteorología and the Comisión Interministerial de Ciencia y Tecnología (CICYT) (Project REN2000-1572) for partial support of this work.

References

[1] Hertz, J., Krogh, A. and Palmer, R.G.: Introduction to the Theory of Neural Computation. Addison Wesley, 1991.

[2] Castillo, E., Cobo, A., Gutiérrez, J.M. and Pruneda, E.: An Introduction to Functional Networks with Applications. Kluwer Academic Publishers, 1999.

[3] Cybenko, G.: Approximation by supperpositions of a sigmoidal function. Mathematics of Control, Signals, and Systems **2**, 303-314, 1989.

[4] Wermter, S. and Sun, R. editors: Hybrid Neural Systems. Springer-Verlag (Lecture Notes in Computer Science, Vol. 1778), 2000.

[5] Happel, B. and Murre, J.: Design and evolution of modular neural network architectures. Neural Networks **7**, 985-1004, 1994.

[6] Jacobs, R.A., Jordan, M.I., Nowlan, S.J., and Hinton, G.E. (1991) Adaptive mixtures of local experts. Neural Computation **3**, 79-87.

[7] Boers, E.J.W., Kuiper, H., Happel, B.L.M., and Sprinkhuizen-Kuyper, I.G.: Designing modular artificial neural networks. In H.A. Wijshoff, editor: Proceedings of Computing Science in The Netherlands, 87-96, 1993.

[8] Cofiño, A.S. and Gutiérrez, J.M. (2001) Optimal Modular FeedForward Neural Networks based on Functional Networks. Lecture Notes in Artificial Intelligence, **2083**, 308-315.

[9] Aczél, J.: Lectures on Functional Equations and Their Applications. Academic Press, 1966.

[10] Castillo, E., Gutiérrez, J.M., Hadi, A.S. and Lacruz, B.: Some applications of functional networks in statistics and engineering. Technometrics **43**, 10-24, 2001.

[11] Castillo, E. and Gutiérrez, J.M.: A minimax method for learning functional networks. Neural Processing Letters **11**, 39-49, 2000.

[12] Ballard, D.: Modular learning in neural networks. Sixth National Conference on AI **1**, 13-17, 1987.

[13] `http://www.mathworks.com`

[14] J. Rissanen, Stochastic complexity in statistical inquiry (World Scientific, 1989).

[15] Stern, H.S.: Neural networks in applied statistics. Technometrics **38**, 205-214, 1993.

[16] Misiurewicz, M.: The Lozi map has a strange attractor. In R. Hellemann, editor: Nonlinear Dynamics, Annals of the NY Academy of Science **357**, 348-358, 1980.

[17] Castillo, E. and Gutiérrez, J.M.: Nonlinear time series modeling and prediction using functional networks. Physics Letters A **244**, 71-84, 1998.