One-variable Interpolation Function Based on Functional Networks[1]

Yong-Quan Zhou [1,2] and Li-Cheng Jiao [2]

[1] College of Computer and Information Science, Guangxi University for Nationalities, Nanning 530006, China
zhou_yongquan@163.com

[2] Institute for Intelligence Information Processing, Xidian University, Xi' an 710071, China
lchjiao@mail.xidian.edu.cn

Abstract

In this paper, the interpolation mechanism of functional networks is discussed. A kind of four-layer (with 1 input and 1 output unit) and a five-layer (with double input and single output unit) functional network are designed. Meanwhile, a learning algorithm based on Minimizing a least squares error function with a unique minimum has been proposed for the purpose of approximating function. Experimental results indicate that the interpolation method is completely correct.

1 Introduction

One of the important problems in digital signal processing is a reconstruction of the continuous time signal $F(t)$ from its samples $F(nT)$ [1]. This problem is known as interpolation described by the following equation

$$F(x) = \sum_{i=1}^{N} w_i \phi_i(x) \qquad (1)$$

Where w_i are coefficients of interpolation and $\{\phi_i(x) \mid i = 1,2,...,N\}$ are basis interpolation function family. If a controlled process is difficult to access, the problem of interpolation becomes complicated. In this case it is difficult to satisfy Nyquist Sampling rate condition and signal is down sampled [1]. In the real time signal process system application of the conventional interpolator based on Lagrange polynomial (parabolic or cubic splines) and orthogonal functions [2] do not satisfy limited time processing requirement related with the generation of functions $\phi_i(x)$, operation of multiplication etc. in accordance with eq. (1).

[1] This work was supported by Grants 60461001 from NSF of China and Grants 0542048 from Guangxi Science Foundation.

Interpolation of discrete time signal $F(nX)$ by polynomial filtering (Butterworth,Tchebyschev filters) cause a shape distortion. Independent of an original signal pattern, the result of interpolation is sharply defined at the nodes and it is exponentially varied between the nodes of interpolation [3]. To satisfy time limiting and precision specifications in this paper operation of interpolation is realized using functional network. In this paper, a kind of four-layer (with 1 input and 1 output unit) and a five-layer (with double input and single output unit) functional networks are designed, meanwhile, a learning algorithm based on minimizing a sum of squares with a unique minimum has been proposed for the purpose of approximating function. Experimental results indicate that the interpolation method is effective and practical.

This paper is organized as follows. In Section 2 the functional network is introduced. In Section 3, one-variable interpolation function model based on functional network is designed. In Section 4, one-variable interpolation function model learning algorithm is presented. Finally, In Section 5, experimental results are discussed.

2 Functional Networks

Castillo et al [4-7] present functional networks as an extension of artificial neural networks. Unlike neural networks, in these networks there are no weights associated with the links connecting neurons, and the internal neuron functions are not fixed but learnable. These functions are not arbitrary, but subject to strong constraints to satisfy the compatibility conditions imposed by the existence of multiple links going from the last input layer to the same output units. In fact, writing the values of the output units in different forms, by considering these different links, a system of functional equation is obtained. When this system is solved, the number of degree of freedom of these initially multidimensional functions is considerably reduced. To learn the resulting functions, a method based on minimizing a least squares error function is used, which, unlike the functions used in neural networks has a single minimum.

Our definition is simple but rigorous: a functional network is a network in which the weights of the neurons are substituted by a set of basis functions. In fact, functional networks are extensions of neural network. For examples, in Fig 1 a neural network and its equivalent functional network are show. Note that weights are subsumed by the neural functions.

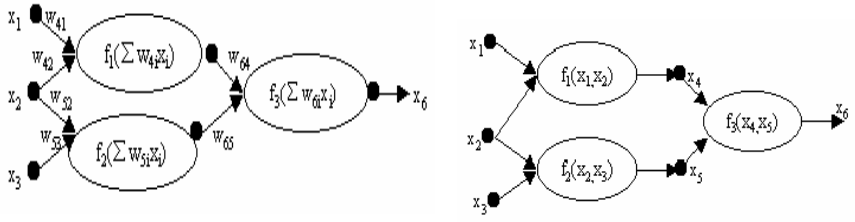

Fig .1. (a) A neural network. (b) A functional network.

Where f_1, f_2 and f_3 are functional neuron functions in Fig.1 (b), and in Fig.1 (a), f_1, f_2 and f_3 are some activation functions, i.e. sigmoid function, triangle function etc [8-11].

In Fig.1 (b), our objective is to calculate the functions $\{f_i \mid i = 1,2,3\}$ that allow us to obtain an acceptable prediction of our output variable. In order to obtain a unique solution, we only need to fix the functions $\{f_i \mid i = 1,2,3\}$ in one point, as explained by Castillo et al [12]. The only criterion for our choice of function bases was simplicity: there is no model or mathematical equation that provides us with preliminary information. If a mathematical function with a physical meaning existed in our problem, it would belong to the family of approximating functions. Also we would be giving our processing units a physical interpretation that does not in exist in artificial neural networks.

As can be seen in Fig.1 (b), a functional network [4] consists of: a) several layers of storing units, one layer for containing the input data $(x_i; i = 1...3)$, another for containing the output data (x_6) and none, one or several layers to store intermediate information $(x_4$ and $x_5)$; b) one or several layer of processing units that evaluate a set of input values and delivers a set of output values (f_i) and c) a set of directed links. Functional networks extend neural networks by allowing neural functions f_i to be not only true multi-argument and multivariate functions, but to be different and learnable, instead of fixed functions. In functional networks, the activation functions are unknown functions from a given family, i.e. polynomial, to be estimated during the learning process. In addition, functional networks allow connecting neuron output, forcing them to be coincident. Some differences between a neural network and a functional network are shown in Fig. 1.

3 One-variable Interpolation Function Model Based on Functional Network

The one-variable interpolation function model of functional network consists of the following elements (seen Fig.3): (1) a layer of input units. This first layer accepts input signals. (2) Three layers of processing units that evaluate a set of input signals and delivers a set of output signals (f_i). (3) A set of directed links, indicating the signal flow direction. (4) A layers of output units. This is the last layers, and contains the output signals.

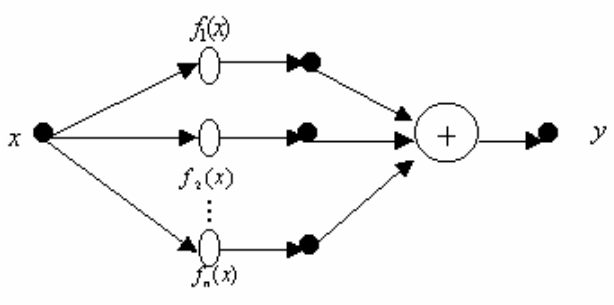

Fig.3. A four- layer functional network for approximating one-variable function $\hat{F}(x)$

In Fig.3, the neuron functions are not fixed and learnable, instead of fixed functions. In functional networks, the activation function functions are unknown functions from a give family, i.e. polynomial, Fourier, etc. to be estimated during the learning process. In addition, functional networks allow connecting neuron outputs, forcing them to be coincident. Some differences between a neural network and a functional network are show in Fig.1 (a), (b). Fig.3 show a functional network corresponding to the functional equation

$$y = \hat{F}(x) = \sum_{i=1}^{N} f_i(x) \tag{2}$$

Learning interpolation function $F(x)$ is equivalent to learning function $\hat{F}(x)$, and learning function $\hat{F}(x)$ is equivalent to learning the functions $f_i(x)$. To this end, we can approximate $f_i(x) : i = 1,2,...,N$ by

$$\hat{f}_i(x) = \sum_{j=1}^{M} w_{ij}\phi_{ij}(x) \tag{3}$$

Replace this in (2) we get

$$y = \hat{F}(x) = \sum_{i=1}^{N}\sum_{j=1}^{M} w_{ij}\phi_{ij}(x) \tag{4}$$

Where the functions family $\{\phi_{ij}(x); j = 1,2,...,M\}$ is a set of given one-variable interpolation basis functions family, capable of approximation of one variable interpolation $\hat{F}(x)$ to the desired accuracy.

4 One-variable Interpolation Function Learning Algorithm

Let us consider one-variable interpolation functional network (seen Fig.3) learning algorithm. To estimate the coefficients $\{w_{ij}; i = 12,..., N, j = 1,2,..., M\}$, we use the collected data pairs (x_{1p}, x_{2p}). where $p = 1,2,..., P$ training patterns. The error can be measured as follows

$$e_p = F(x_p) - \hat{F}(x_p) = F(x_p) - \sum_{i=1}^{N}\sum_{j=1}^{M} w_{ij}\phi_{ij}(x_p) \tag{5}$$

The formula of the sum of the squares of the errors is

$$Q = \sum_{p=1}^{P} e_p^2 = \sum_{p=1}^{P}[F(x_p) - \sum_{i=1}^{N}\sum_{j=1}^{M} w_{ij}\phi_{ij}(x_p)]^2 \tag{6}$$

Subject to

$$\hat{f}_i(x_0) = \sum_{j=1}^{M} w_{ij}\phi_{ij}(x_0) = \alpha_i; i = 1,2,..., N. \tag{7}$$

Where α_i are arbitrary but given real constants, in order to guarantee the equation (4) have unique solution.

Our objective is to calculate the coefficients w_{ij}, in such a way that the estimated error becomes as small as possible. This implies that we have to minimize the error function while taking into account the auxiliary conditions. To this effect, we use the Lagrange multipliers and construct the auxiliary function Q_λ :

$$\begin{aligned} Q_\lambda = & \sum_{p=1}^{P}[F(x_p) - \sum_{i=1}^{N}\sum_{j=1}^{M} w_{ij}\phi_{ij}(x_p)]^2 \\ & + \sum_{i=1}^{N} \lambda_i (\sum_{j=1}^{M} w_{ij}\phi_{ij}(x_0) - \alpha_i) \end{aligned} \tag{8}$$

We obtain the minimum with the following system of linear equations, where the unknown quantities are coefficients w_{ij}, and multipliers λ_i.

$$\begin{cases} \dfrac{\partial Q_\lambda}{\partial w_{tr}} = -2\sum_{p=1}^{P} e_p \phi_{tr}(x_p) + \lambda_t \phi_{tr}(x_0) = 0; r = 1,2,...,N. \\ \dfrac{\partial Q_\lambda}{\partial \lambda_t} = \sum_{i=1}^{N} w_{tj} \phi_{tj}(x_o) - \alpha_t = 0; \end{cases} \tag{9}$$

Which is valid for $t = 1,2,...,M$.

5 Experimental Results

To verify the effectiveness and efficiency of the proposed functional networks interpolation method, we take one-variable interpolation examples to conduct the experiments. To estimate the performance of approximation, the Root Mean Square Error (RMSE) is defined as

$$RMSE = \sqrt{\frac{1}{P}\sum_{p=1}^{P} \| b_p - \hat{b}_p \|^2} \tag{10}$$

Where $\hat{b}_p$ is the network output, and the norm function $\| \cdot \|$ reduces to the usual absolute value function $| \cdot |$.

Example 1. Consider one-variable interpolation function

$$F : [0, \frac{\pi}{2}] \to R \quad y = F(x) = \sin(x) \tag{11}$$

Suppose we are given the data set consisting of the triplets shown in Table 1 and we are interested in estimating a representative functional network model (seen Fig.3) of the form (4).

Table.1 (x_t, y_t) data used to fit the approximate interpolation function $\hat{F}(x)$

x_t	y_t	x_t	y_t
0	0	0.9425	0.8090
0.1571	0.1564	1.0996	0.8910
0.3142	0.3090	1.2566	0.9511
0.4712	0.4540	1.4137	0.9877
0.6283	0.5878	1.5708	1.0000
0.7854	0.7071		

By using a structure of single-input and single output functional network with the learning algorithm, we can discuss three interpolation cases as follows:

Case I. Consider the polynomial interpolation basis functions $\phi(x) = \{1, x, x^2, x^3, x^4, x^5\}$ for the neuron functions f_i, we can obtain the one-variable interpolation function $F(x)$ of approximation function $\hat{F}(x)$.

$$\hat{F}(x) = 0.9987x + 0.0076x^2 - 0.1818x^3 + 0.0129x^4 + 0.0040x^5$$

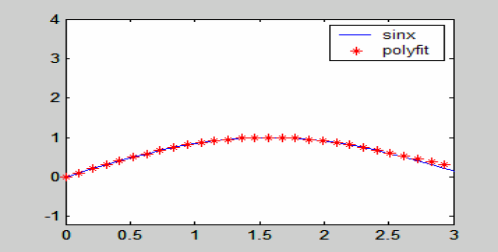

Fig. 4. Exact function $F(x)$ (solid line) and interpolation function $\hat{F}(x)$ (dashed line)

Fig 4 shows the exact function $F(x)$ (solid line) and interpolation function $\hat{F}(x)$ (dashed line). Thus, we get $RMSE = 0$

Case II. Consider the polynomial interpolation basis functions $\phi(x) = \{1, x, x^2, x^3, x^4\}$ for the neuron functions f_i, we can obtain the one-variable interpolation function $F(x)$ of approximation function $\hat{F}(x)$.

$$\hat{F}(x) = 0.9964x + 0.0197x^2 - 0.2033x^3 + 0.0286x^4.$$

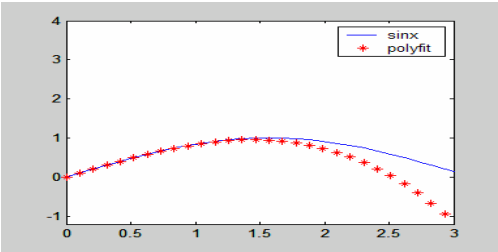

Fig 5. Exact function $F(x)$ (solid line) and interpolation function $\hat{F}(x)$ (dashed line)

Fig.5 shows the exact function $F(x)$ (solid line) and interpolation function $\hat{F}(x)$ (dashed line). Thus, we get $RMSE = 0.8512$.

Example 2. The Hènon series

The Hènon map [6] is one of the most popular dynamical systems exhibiting chaos, and can be defined in the following way:

$$x_n = 1 - 1.4x_{n-1}^2 + 0.3x_{n-2} \tag{12}$$

A time series plot of the Hènon map for the initial conditions $x_0 = 0.5, x_1 = 0.5$ is given in Fig 6.

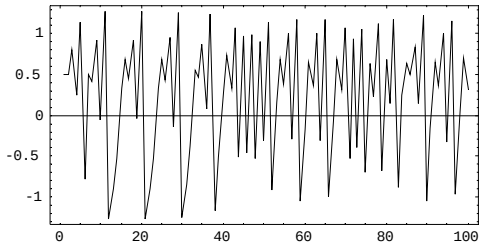

Fig .6. Time series for the Hènon map

Suppose that we have the time series consisting of 100 points shown in Fig.6 and we can use the one-variable interpolation functional network model in Fig 7:

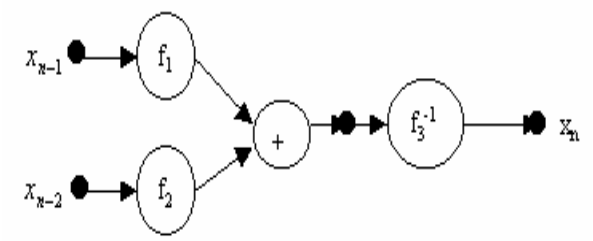

Fig.7. A five-layer functional network associated with the Hènon map

Where, $f_i, i = 1,2,3.$ is one-variable function, we obtain models of the form:

$$x_n = f_3^{-1}[f_1(x_{n-1}) + f_2(x_{n-2})] \tag{13}$$

if we consider any polynomial family containing the basis functions $\phi_i = \{1, x, x^2\}, i = 1,2,3.$ for the neuron functions f_1, f_2 and f_3 ,then we obtain the exact Hènon map given in (12), after resolving the system of equations (9), the following approximation provides us with the global minimum of the error function:

$$\begin{aligned} f_1(x_{n-1}) &= 1.818 - 0.818x_{n-1}, \\ f_2(x_{n-2}) &= -2.818 + 3.818x_{n-2}^2, \\ f_3(x_n) &= -1.727 + 2.727x_n . \end{aligned} \tag{14}$$

And RMSE=0. Then, the resulting model

$$\begin{aligned} x_n &= f_3^{-1}[f_1(x_{n-1}) + f_2(x_{n-2})] \\ &= \frac{f_1(x_{n-1}) + f_2(x_{n-2})}{2.727} + 1.727 = 1 + 0.3x_{n-1}^2 - 1.4x_{n-2} \end{aligned} \tag{15}$$

However, if we use a different family for the neuron functions approximate Hènon map, Table 2 illustrates the performance of the approximate model by giving the RMSE.

Table. 2. Performance of several basis function for the Hènon map

Model	$\phi_k, k = 1,2$ sets	ϕ_3 set	RMSE
1	$\{1, x, x^2\}$		0
2	$\{Sinx, Sin2x, Cosx, Cos2x\}$	$\{1, x\}$	0.00593219
3	$\{Sinx, Sin2x, Cosx, Cos2x\}$		0.00337892
4	$\{1, \log(2+x), \log(3+x), \log(4+x), \log(5+x)\}$		0.00001598

Table 2 shows the root mean squared error (RMSE) over the 100 samples points. In fact, the values of the Hènon series range from –1.5 to 1.5, which implies that the above errors are very small. Thus, it seems that over-fitting is not a problem.

6 Conclusions

This paper discussed how to use functional networks to realize the interpolation of arbitrary one-variable functions, which is an important research topic in neural computation and digital signal processing. The interpolation mechanism of functional networks is discussed, a kind of four-layer (with 1 input and 1 output unit) and a five-layer (with double input and single output unit) functional networks are designed. Meanwhile, a learning algorithm based on minimizing a sum of squares with a unique minimum has been proposed for the purpose of approximating function. Experimental results indicate that the interpolation method is effective and practical.

References

1. Fred J Taylor. Principles of Signals and Systems. New York: McGraw-Hill,1994.
2. John H. Mathews. Numerical Methods. NJ: Prentice-Hall,1992.
3. Fakhreddin M . Communication System for the Plant with Difficult of Access. Intentional Conference on Components and Electronics Systems, Algeria, 1991.
4. Enrique Castillo. Functional Networks. Neural Processing Letters 7:151~159,1998
5. Enrique Castillo, Josè Manuel Gutièrrez.A Comparison between Functional Networks and Artificial Neural Networks. In: Proceedings of the IASTED International Conference on Artificial Intelligence and Soft Computing.1998: 439-442.
6. Enrique Castillo, Angel Cobo, Josè Manuel Gutièrrez and Rose Eva Pruneda, Functional Networks with Applications. Kluwer Academic Publishers,1999.
7. Alfonso Iglesias, Bernardino Arcay, J.M. cotos, et.al. A Comparison between Functional Networks and Artificial Neural Networks for the Prediction of Fishing Catches.Neural Comput & Applie.Vol.13 (2004) 24~31.
8. M.S Chen and M. T. Manry. Conventional Modeling of the Multiplayer Perceptron Using Polynomial Basis Functions. IEEE Trans Neural Networks, Vol.4. (1993) 64-166.
9. Y. H. Pao. Adaptive Pattern Recognition and Neural Networks. Reading, MA: Addison-Wesley, 1989.
10. De-Shuang Huang. A Constructive Approach for Finding Arbitrary Roots of Polynomials By Neural Networks. IEEE Trans Neural Networks, Vol.15. (2004) 477- 491.
11. De-Shuang Huang. On the Comparisons between RLSA and CLA for Solving Arbitrary Linear Simultaneous Equation. LNCS 2690, (2003) 169-176.
12. Castillo E, Gutierrez J M, Cobo A, Castillo C. A Minimax Method for Learning Functional Networks. Neural Processing Letters 11 (1). (2000) 39-49

Yong-Quan Zhou received the B.S. degree in mathematics from Xianyang Normal university, Xianyang, China. in 1983, the M.S. degree in computer from Lanzhou University, Lanzhou, China., in 1993. and professor, Ph.D. candidate in computation intelligent from Xidian University. His current research interests include neural networks, computation intelligent.

Li-Cheng Jiao, Ph.D. professor and Ph.D. supervisor. His current research interests include neural networks, data mining, and artificial immune system.