Self-Organizing Multi-layer Fuzzy Polynomial Neural Networks Based on Genetic Optimization

Sung-Kwun Oh[1], Witold Pedrycz[2], Hyun-Ki Kim[3], and Jong-Beom Lee[1]

[1] Department of Electrical Electronic and Information Engineering, Wonkwang University, 344-2, Shinyong-Dong, Iksan, Chon-Buk, 570-749, South Korea
{ohsk, ipower}@wonkwang.ac.kr
http://autosys.wonkwang.ac.kr

[2] Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB T6G 2G6, Canada
and Systems Research Institute, Polish Academy of Sciences, Warsaw, Poland
pedrycz@ee.ualberta.ca

[3] Department of Electrical Engineering, University of Suwon, South Korea
hkkim@suwon.ac.kr

Abstract. In this paper, we introduce a new topology of Fuzzy Polynomial Neural Networks (FPNN) that is based on a genetically optimized multilayer perceptron with fuzzy polynomial neurons (FPNs) and discuss its comprehensive design methodology involving mechanisms of genetic optimization, especially genetic algorithms (GAs). The proposed FPNN gives rise to a structurally optimized structure and comes with a substantial level of flexibility in comparison to the one we encounter in conventional FPNNs. The structural optimization is realized via GAs whereas in case of the parametric optimization we proceed with a standard least square method-based learning. Through the consecutive process of such structural and parametric optimization, an optimized and flexible fuzzy neural network is generated in a dynamic fashion. The performance of the proposed gFPNN is quantified through experimentation that exploits standard data already used in fuzzy modeling. These results reveal superiority of the proposed networks over the existing fuzzy and neural models.

1 Introduction

Recently, a lots of attention has been directed towards advanced techniques of complex system modeling. While neural networks, fuzzy sets and evolutionary computing as the technologies of Computational Intelligence (CI) have expanded and enriched a field of modeling quite immensely, they have also gave rise to a number of new methodological issues and increased our awareness about tradeoffs one has to make in system modeling [1-4]. The most successful approaches to hybridize fuzzy systems with learning and adaptation have been made in the realm of CI. Especially neural fuzzy systems and genetic fuzzy systems hybridize the approximate inference method of fuzzy systems with the learning capabilities of neural networks and evolutionary algorithms [5]. As one of the representative design approaches which are advanced tools, a family of fuzzy polynomial neuron (FPN)-based SOPNN(called “FPNN” as a

M. Bubak et al. (Eds.): ICCS 2004, LNCS 3037, pp. 179–187, 2004.

new category of neuro-fuzzy networks)[6] were introduced to build predictive models for such highly nonlinear systems. The FPNN algorithm exhibits some tendency to produce overly complex networks as well as a repetitive computation load by the trial and error method and/or the repetitive parameter adjustment by designer like in case of the original GMDH algorithm. In this study, in addressing the above problems with the conventional SOPNN (especially, FPN-based SOPNN called "FPNN" [6, 9]) as well as the GMDH algorithm, we introduce a new genetic design approach; as a consequence we will be referring to these networks as GA-based FPNN (to be called "gFPNN"). The determination of the optimal values of the parameters available within an individual FPN (viz. the number of input variables, the order of the polynomial, and input variables) leads to a structurally and parametrically optimized network.

2 The Architecture and Development of Fuzzy Polynomial Neural Networks (FPNN)

2.1 FPNN Based on Fuzzy Polynomial Neurons (FPNs)

The FPN consists of two basic functional modules. The first one, labeled by **F**, is a collection of fuzzy sets that form an interface between the input numeric variables and the processing part realized by the neuron. The second module (denoted here by **P**) is about the function – based nonlinear (polynomial) processing. This nonlinear processing involves some input variables. In other words, FPN realizes a family of multiple-input single-output rules. Each rule reads in the form

$$\text{If } x_p \text{ is } \boldsymbol{A}_l \text{ and } x_q \text{ is } \boldsymbol{B}_k \text{ then } z \text{ is } P_{lk}(x_i, x_j, \boldsymbol{a}_{lk}) \tag{1}$$

where $\boldsymbol{a}_{lk}$ is a vector of the parameters of the conclusion part of the rule while $P_{lk}(x_i, x_j, \boldsymbol{a}_{lk})$ denotes the regression polynomial forming the consequence part of the fuzzy rule which uses several types of high-order polynomials besides the constant function forming the simplest version of the consequence; refer to Table 1. The activation levels of the rules contribute to the output of the FPN being computed as a weighted average of the individual condition parts (functional transformations) P_K (note that the index of the rule, namely "K" is a shorthand notation for the two indexes of fuzzy sets used in the rule (1), that is K = (l, k)).

$$z = \sum_{K=1}^{all\ rules} \mu_K P_K(x_i, x_j, \boldsymbol{a}_K) \Big/ \sum_{K=1}^{\text{all rules}} \mu_K = \sum_{K=1}^{all\ rules} \tilde{\mu}_K P_K(x_i, x_j, \boldsymbol{a}_K) \tag{2}$$

$$\tilde{\mu}_K = \frac{\mu_K}{\sum_{L=1}^{all\ rules} \mu_L} \tag{3}$$

Table 1. Different forms of the regression polynomials standing in the consequence part of the fuzzy

No. of inputs / Order of the polynomial	1	2	3
0 (Type 1)	Constant	Constant	Constant
1 (Type 2)	Linear	Bilinear	Trilinear
2 (Type 3)	Quadratic	Biquadratic-1	Triquadratic-1
2 (Type 4)		Biquadratic-2	Triquadratic-2

1: Basic type, 2: Modified type

2.2 Genetic Optimization of FPNN

GAs is a stochastic search technique based on the principles of evolution, natural selection, and genetic recombination by simulating "survival of the fittest" in a population of potential solutions(individuals) to the problem at hand [7]. For the optimization of the FPNN model, GA uses the serial method of binary type, roulette-wheel used in the selection process, one-point crossover in the crossover operation, and a binary inversion (complementation) operation in the mutation operator. To retain the best individual and carry it over to the next generation, we use elitist strategy [8].

3 The Algorithms and Design Procedure of Genetically Optimized FPNN

The framework of the design procedure of the Fuzzy Polynomial Neural Networks (FPNN) based on genetically optimized multi-layer perceptron architecture comprises the following steps.

[Step 1] *Determine system's input variables*

[Step 2] *Form training and testing data*

The input-output data set $(\boldsymbol{x}_i, y_i)=(x_{1i}, x_{2i}, \ldots, x_{ni}, y_i)$, $i=1, 2, \ldots, N$ is divided into two parts, that is, a training and testing dataset.

[Step 3] *Decide initial information for constructing the FPNN structure*

[Step 4] *Decide FPN structure using genetic design*

When it comes to the organization of the chromosome representing (mapping) the structure of the FPNN, we divide the chromosome to be used for genetic optimization into three sub-chromosomes. The 1st sub-chromosome contains the number of input variables, the 2nd sub-chromosome involves the order of the polynomial of the node, and the 3rd sub-chromosome (remaining bits) contains input variables coming to the corresponding node (FPN). All these elements are optimized when running the GA.

[Step 5] *Carry out fuzzy inference and coefficient parameters estimation for fuzzy identification in the selected node (FPN)*

Regression polynomials (polynomial and in the specific case, a constant value) standing in the conclusion part of fuzzy rules are given as different types of Type 1, 2, 3, or 4, see Table 1. In each fuzzy inference, we consider two types of membership

functions, namely triangular and Gaussian-like membership functions. The consequence parameters are produced by the standard least squares method

[Step 6] *Select nodes (FPNs) with the best predictive capability and construct their corresponding layer*

The generation process can be organized as the following sequence of steps

Sub-step 1) We set up initial genetic information necessary for generation of the FPNN architecture.

Sub-step 2) The nodes (FPNs) are generated through the genetic design.

Sub-step 3) We calculate the fitness function. The fitness function reads as

$$F(\text{fitness function}) = 1/(1+EPI) \tag{4}$$

where *EPI* denotes the performance index for the testing data (or validation data).

Sub-step 4) To move on to the next generation, we carry out selection, crossover, and mutation operation using genetic initial information and the fitness values obtained via ***sub-step 3***.

Sub-step 5) We choose several FPNs characterized by the best fitness values. Here, we use the pre-defined number W of FPNs with better predictive capability that need to be preserved to assure an optimal operation at the next iteration of the FPNN algorithm. The outputs of the retained nodes (FPNs) serve as inputs to the next layer of the network. There are two cases as to the number of the retained FPNs, that is

(i) If $W^* < W$, then the number of the nodes retained for the next layer is equal to z.

Here, W^*denotes the number of the retained nodes in each layer that nodes with the duplicated fitness values are moved.

(ii) If $W^* \geq W$, then for the next layer, the number of the retained nodes is equal to W.

Sub-step 6) For the elitist strategy, we select the node that has the highest fitness value among the selected nodes (W).

Sub-step 7) We generate new populations of the next generation using operators of GAs obtained from ***Sub-step 4***. We use the elitist strategy. This sub-step carries out by repeating ***sub-step 2-6***.

Sub-step 8) We combine the nodes (W populations) obtained in the previous generation with the nodes (W populations) obtained in the current generation.

Sub-step 9) Until the last generation, this sub-step carries out by repeating ***sub-step 7-8***.

[Step 7] *Check the termination criterion*

As far as the performance index is concerned (that reflects a numeric accuracy of the layers), a termination is straightforward and comes in the form,

$$F_1 \leq F_* \tag{5}$$

Where, F_1 denotes a maximal fitness value occurring at the current layer whereas F_* stands for a maximal fitness value that occurred at the previous layer. In this study, we use a measure (performance index) that is the Root Mean Squared Error (RMSE).

$$E(PI \ \ or \ \ EPI) = \sqrt{\frac{1}{N}\sum_{p=1}^{N}(y_p - \hat{y}_p)^2} \tag{6}$$

[Step 8] *Determine new input variables for the next layer*
If (5) has not been met, the model is expanded. The outputs of the preserved nodes (z_{1i}, z_{2i}, …, z_{Wi}) serves as new inputs to the next layer (x_{1j}, x_{2j}, …, x_{Wj})(j=i+1). This is captured by the expression

$$x_{1j} = z_{1i}, x_{2j} = z_{2i}, \ldots, x_{wj} = z_{wi} \tag{7}$$

The FPNN algorithm is carried out by repeating steps 4-8.

4 Experimental Studies

The performance of the GA-based FPNN is illustrated with the aid of well-known and widely used dataset of the chaotic Mackey-Glass time series [10-17].
The time series is generated by the chaotic Mackey-Glass differential delay equation comes in the form

$$\dot{x}(t) = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t) \tag{8}$$

To obtain the time series value at each integer point, we applied the fourth-order Runge-Kutta method to find the numerical solution to (8). From the Mackey-Glass time series $x(t)$, we extracted 1000 input-output data pairs in the following format:

$$[x(t\text{-}18), x(t\text{-}12), x(t\text{-}6), x(t)\ ;\ x(t\text{+}6)]$$

where, t=118 to 1117. The first 500 pairs were used as the training data set while the remaining 500 pairs formed the testing data set. To come up with a quantitative evaluation of the network, we use the standard RMSE performance index as given by (6). Table 2 summarizes the list of parameters used in the genetic optimization of the network.

Table 2. Summary of the parameters of the genetic optimization

	Parameters	1st layer	2nd to 5th layer
GA	Maximum generation	100	100
	Total population size	60	60
	Selected population size (W)	30	30
	Crossover rate	0.65	0.65
	Mutation rate	0.1	0.1
	String length	3+3+30	3+3+30
FPNN	Maximal no.(Max) of inputs to be selected	$1 \le l \le$ Max(2~5)	$1 \le l \le$ Max(2~5)
	Polynomial type (Type T) of the consequent part of fuzzy rules	$1 \le$ T ≤ 4	$1 \le$ T ≤ 4
	Consequent input type to be used for Type T (*)	Type T*	Type T
	Membership Function (MF) type	Triangular	Triangular
		Gaussian	Gaussian
	No. of MFs per input	2	2

l, P, Max: integers, T* means that entire system inputs are used for the polynomial in the conclusion part of the rules.

Table 3 summarizes the performance of the 1st and the 2nd layer of the network when changing the maximal number of inputs to be selected; here Max was set up to 2 through 5.

Table 3. Performance index of the network of each layer versus the increase of maximal number of inputs to be selected

(a) Triangular MF

Max	1st layer					2nd layer					
	Node		T	PI	EPI	Node		T	PI	EPI	
2	1	2	3	0.0056	0.0054	7	29	4	0.0050	0.0049	
3	1	2	3	3	0.0040	0.0040	2	16	25	2	0.0033	0.0034
4	1	2	3	4	3	0.0017	0.0016	9	12	22	28	3	0.0015	0.0014
5	1	2	3	4	0	3	0.0017	0.0016	8	24	25	27	28	2	0.0014	0.0013

(b) Gaussian-like MF

1st layer					2nd layer					
Node		T	PI	EPI	Node		T	PI	EPI	
1	2	3	0.0028	0.0027	19	26	3	0.0026	0.0025	
1	2	4	3	0.0014	0.0013	15	25	27	4	0.0012	0.0011
1	2	3	4	3	0.0005	0.0006	10	23	24	29	2	0.0004	0.0006
1	2	3	4	0	3	0.0005	0.0006	7	12	18	22	29	4	0.0004	0.0006

Fig. 1 depicts the performance index of each layer of gFPNN according to the increase of maximal number of inputs to be selected. Fig. 2(a) illustrates the detailed optimal topologies of gFPNN for 1 layer when using Max=4 and triangular MF. And also Fig. 2(b) illustrates the detailed optimal topology of gFPNN for 1 layer in case of using Max= 3 and Gaussian-like MF.

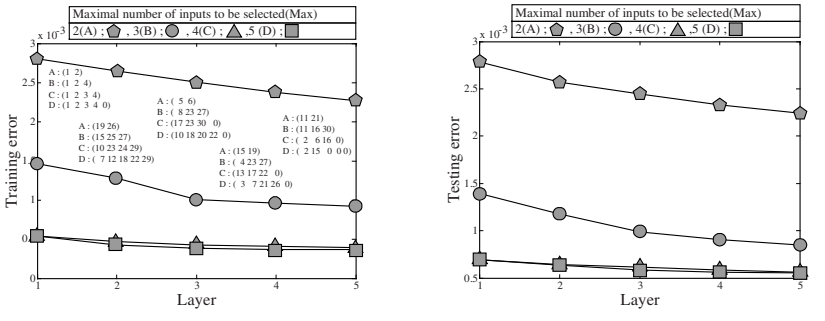
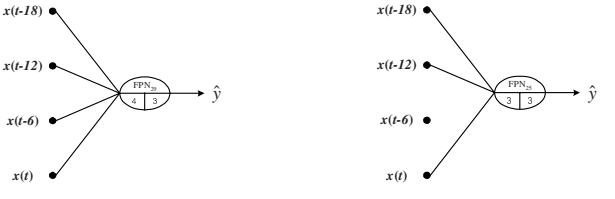


Fig. 1. Performance index of gFPNN with respect to the increase of number of layers (Gaussian-like MF)



(a) Triangular MF (1 layer) (b) Gaussian-like MF

Fig. 2. Genetically optimized FPNN(gFPNN) architecture

Table 4 gives a comparative summary of the network with other models.

Table 4. Comparative analysis of the performance of the network; considered are models reported in the literature

Model			Performance index			
			PI	PI_s	EPI_s	NDEI*
Wang's model[10]			0.044			
			0.013			
			0.010			
Cascaded-correlation NN[11]						0.06
Backpropagation MLP[11]						0.02
6th-order polynomial[11]						0.04
ANFIS[12]				0.0016	0.0015	0.007
FNN model[13]				0.014	0.009	
Recurrent neural network[14]			0.0138			
SONN**[15]	Basic (5^{th} layer)	Case 1		0.0011	0.0011	0.005
		Case 2		0.0027	0.0028	0.011
	Modified (5^{th} layer)	Case 1		0.0012	0.0011	0.005
		Case 2		0.0038	0.0038	0.016
Proposed gFPNN	Triangular	2^{nd} layer(Max=4)	0.0014	0.0015	0.0014	0.0061
		5^{th} layer(Max=4)	0.0010	0.0011	0.0011	0.0045
		2^{nd} layer(Max=5)	0.0011	0.0014	0.0013	0.0051
		5^{th} layer(Max=5)	0.0007	0.0009	0.0010	0.0031
	Gaussian	1^{st} layer(Max=3)	0.0014	0.0014	0.0013	0.0064
		1^{st} layer(Max=4)	0.0005	0.0005	0.0006	0.0023
		1^{st} layer(Max=5)	0.0005	0.0005	0.0006	0.0023

*Non-dimensional error index (NDEI) as used in [16] is defined as the root mean square errors divided by the standard deviation of the target series. ** is called "conventional optimized FPNN".

5 Concluding Remarks

In this study, the GA-based design procedure of Fuzzy Polynomial Neural Networks (FPNN) along with their architectural considerations has been investigated. The design methodology comes as a hybrid structural optimization and parametric learning being viewed as two fundamental phases of the design process. The GMDH method is now comprised of both a structural phase such as a self-organizing and an evolutionary algorithm (rooted in natural law of survival of the fittest), and the ensuing parametric phase of the Least Square Estimation (LSE)-based learning. The comprehensive experimental studies involving well-known datasets quantify a superb performance of the network in comparison to the existing fuzzy and neuro-fuzzy models. Most importantly, through the proposed framework of genetic optimization we can efficiently search for the optimal network architecture (structurally and parametrically optimized network) and this becomes crucial in improving the performance of the resulting model.

Acknowledgements. This work has been supported by EESRI(R-2003-B-274), which is funded by MOCIE (Ministry of Commerce, Industry and Energy).

References

1. V. Cherkassky, D. Gehring, F. Mulier.: Comparison of adaptive methods for function estimation from samples. IEEE Trans. Neural Networks, 7 (1996) 969-984
2. J. A. Dickerson, B. Kosko.: Fuzzy function approximation with ellipsoidal rules. IEEE Trans. Syst., Man, Cybernetics. Part B. 26 (1996) 542-560
3. V. Sommer, P. Tobias, D. Kohl, H. Sundgren, L. Lundstrom.: Neural networks and abductive networks for chemical sensor signals: A case comparison. Sensors and Actuators B. 28 (1995) 217-222
4. S. Kleinsteuber, N. Sepehri.: A polynomial network modeling approach to a class of large-scale hydraulic systems. Computers Elect. Eng. 22 (1996) 151-168
5. O. Cordon, et al.: Ten years of genetic fuzzy systems: current framework and new trends. Fuzzy Sets and Systems. 2003(in press)
6. S.-K. Oh, W. Pedrycz.: Self-organizing Polynomial Neural Networks Based on Polynomial and Fuzzy Polynomial Neurons: Analysis and Design. Fuzzy Sets and Systems. 142(2) (2003) 163-198
7. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. 3rd edn. Springer-Verlag, Berlin Heidelberg New York (1996)
8. D. Jong, K. A.: Are Genetic Algorithms Function Optimizers?. Parallel Problem Solving from Nature 2, Manner, R. and Manderick, B. eds., North-Holland, Amsterdam
9. S.-K. Oh, W. Pedrycz.: Fuzzy Polynomial Neuron-Based Self-Organizing Neural Networks. Int. J. of General Systems, 32(2003) 237-250
10. L. X. Wang, J. M. Mendel.: Generating fuzzy rules from numerical data with applications. IEEE Trans. Systems, Man, Cybern. 22 (1992) 1414-1427
11. R. S. Crowder III.: Predicting the Mackey-Glass time series with cascade-correlation learning. In: D. Touretzky, G. Hinton, and T. Sejnowski, editors, Proceedings of the 1990 Connectionist Models Summer School, Carnegic Mellon University, (1990) 117-123
12. J. S. R. Jang.: ANFIS: Adaptive-Network-Based Fuzzy Inference System. IEEE Trans. System, Man, and Cybern. 23 (1993) 665-685
13. L. P. Maguire, B. Roche, T. M. McGinnity, L. J. McDaid.: Predicting a chaotic time series using a fuzzy neural network. Information Sciences. 112(1998) 125-136
14. C. James Li, T. -Y. Huang.: Automatic structure and parameter training methods for modeling of mechanical systems by recurrent neural networks. Applied Mathematical Modeling. 23 (1999) 933-944
15. S.-K. Oh, W. Pedrycz, T.-C. Ahn.: Self-organizing neural networks with fuzzy polynomial neurons. Applied Soft Computing. 2 (2002) 1-10
16. A. S. Lapedes, R. Farber.: Non-linear Signal Processing Using Neural Networks: Prediction and System Modeling. Technical Report LA-UR-87-2662, Los Alamos National Laboratory, Los Alamos, New Mexico 87545, (1987)
17. M. C. Mackey, L. Glass.: Oscillation and chaos in physiological control systems. Science, 197 (1977) 287-289
18. S.-K. Oh, W. Pedrycz.: The design of self-organizing Polynomial Neural Networks. Information Science. 141 (2002) 237-258

19. S.-K. Oh, W. Pedrycz, B.-J. Park.: Polynomial Neural Networks Architecture: Analysis and Design. Computers and Electrical Engineering. 29 (2003) 703-725
20. B.-J. Park, D.-Y. Lee, S.-K. Oh: Rule-Based Fuzzy Polynomial Neural Networks in Modeling Software Process Data. International Journal of Control, Automation and Systems, 1(3) (2003) 321-331
21. H.-S. Park, S.-K. Oh: Rule-based Fuzzy-Neural Networks Using the Identification Algorithm of GA hybrid Scheme. International Journal of Control, Automation and Systems, 1(1) (2003) 101-110