World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:2, No:8, 2008

A New Approach to Polynomial Neural Networks based on Genetic Algorithm

S. Farzi

***Abstract*—**Recently, a lot of attention has been devoted to advanced techniques of system modeling. PNN(polynomial neural network) is a GMDH-type algorithm (Group Method of Data Handling) which is one of the useful method for modeling nonlinear systems but PNN performance depends strongly on the number of input variables and the order of polynomial which are determined by trial and error. In this paper, we introduce GPNN (genetic polynomial neural network) to improve the performance of PNN. GPNN determines the number of input variables and the order of all neurons with GA (genetic algorithm). We use GA to search between all possible values for the number of input variables and the order of polynomial. GPNN performance is obtained by two nonlinear systems. the quadratic equation and the time series Dow Jones stock index are two case studies for obtaining the GPNN performance.

***Keywords*—GMDH, GPNN, GA, PNN.**

I. Introduction

RECENTLY, a lot of attention has been directed to advanced techniques of system modeling. The panoply of the existing methodologies and detailed algorithms is confronted with nonlinear systems, high dimensionality of the problems, a quest for high accuracy and generalization capabilities of the ensuing models. Nonlinear models can address some of these issues but they require a vast amount of data. When the complexity of the system to be modeled increases, both experimental data and some prior domain knowledge (conveyed by the model developer) are of paramount importance to complete an efficient design procedure. One of the first approaches along systematic design of nonlinear relationships comes under the name of a Group Method of Data Handling (GMDH). GMDH [1]-[3] was developed in the late 1960s by Ivahnenko as a vehicle for identifying nonlinear relations between input and output variables. The GMDH algorithm generates an optimal structure of the model through successive generations of Partial Descriptions of data (PDs) being regarded as quadratic regression polynomials with two input variables. While providing with a systematic design procedure, GMDH has some drawbacks [2]-[4]. So PNN (polynomial neural network) is introduced.

S.Farzi is with Department of Computer Engineering, Islamic Azad University – branch of Kermanshah, Iran (corresponding author to provide phone: +988318247902; fax: +988317243065; e-mail: saeedfarzi@ gmail.com).

PNN is a GMDH-Type algorithm, which is one of useful approximated techniques. PNN is neural network whose neurons are called PD(partial description).The output of each PD is obtained by using several types of high order polynomials such as linear, quadratic, cubic of the input variables[1][3][7][10]. Although the PNN is structured by a systematic design procedure, it has some drawbacks to be solved. PNN performance depends on the number of input variables and the type or order of each PD. These parameters must be determined by trial and error method, which has a heavy computational load and low efficiency.

In this paper, GPNN (genetic polynomial neural network) has been introduced which uses genetic algorithm (GA) to alleviate the above mentioned drawbacks. The GA is used to determine the number of input variables in each PD and to determine the appropriate type of polynomials in each PD.

This paper is organized as follows. The PNN algorithm and its generic structure is described in *Sec.II* Genetic Partial Description (GPD) is described in *Sec.III*. GPNN (Genetic Polynomial Neural Network) is described in *Sec.IV*. A suite of experimental studies is covered in *Sec.V*. Concluding remarks are included in *Sec.VI*.

II. The PNN Algorithm and its Generic Structure

In this section, we elaborate on algorithmic details of the optimal identification method related to two types of the PNN structures.

A. . PNN algorithm

The PNN algorithm is based on the GMDH method and utilizes a class of polynomials such as linear, modified quadratic, cubic, etc. By choosing the most significant input variables and polynomial order among these various types of forms available, we can obtain the best of the extracted partial descriptions according to both selecting nodes of each layer and generating additional layers until the best performance is reached. Such methodology leads to an optimal PNN structure. Let us recall that the input–output data are given in the form

$$(X_i, y_i) = (x_{1i}, x_{2i}, ..., x_{Ni}, y_i),\ \text{i=1; 2; 3; . . . ; n} \quad (1)$$

The input–output relationship of the above data by PNN algorithm can be described in the following manner:

$$y = f(x_1, x_2, ..., x_N) \quad (2)$$

The estimated output $\hat{y}$ reads as

$$\hat{y} = \hat{f}(x_1, x_2, ..., x_n) = c_0 + \sum_i c_i x_i + \sum_i \sum_j c_{ij} x_i x_j + \sum_i \sum_j \sum_k c_i c_j c_k x_i x_j x_k + ... \quad (3)$$

Where ck's denote the coefficients of the model. The framework of the design procedure of the PNNs comes as a sequence of the following steps[8].

Step1: Determine system's input variables

Here, we define the input variables as x_i; i=1; 2; . . . ; N related to output variable y. If required, the normalization of input data is also completed.

Step 2: Form a training and testing data

The input–output data set $(X_i, y_i) = (x_{1i}, x_{2i}, ..., x_{Ni}, y_i)$, i=1; 2; 3; . . . ; n is divided into two parts, that is a training and testing data set. Denote their sizes by n_{tr} and n_{te} respectively. Obviously we have $n=n_{tr}+n_{te}$. The training data set is used to construct a PNN model (including an estimation of the coefficients of the PD of nodes situated in each layer of the PNN). Next, the testing data set is used to evaluate the estimated PNN model [8].

Step 3: Choose a structure of the PNN

The structure of PNN is selected based on the number of input variables and the order of PD in each layer. Two kinds of PNN structures, namely a basic PNN and a modified PNN structure are distinguished. Each of them comes with two cases. Table 1 summarizes all the options available [8]. More specifically, the main features of these architectures are as follows:

TABLE 1 A TAXONOMY OF VARIOUS PNN STRUCTURES

Layer	No. of input var	Order of polynomial	PNN
First layer	p	P	p=q Basic PNN Case1 P=Q Case2 P≠Q
Second to fifth layer	q	Q	p≠ q modified PNN Case1 P=Q Case2 P≠Q

p,q=2,3,4 P,Q=1,2,3

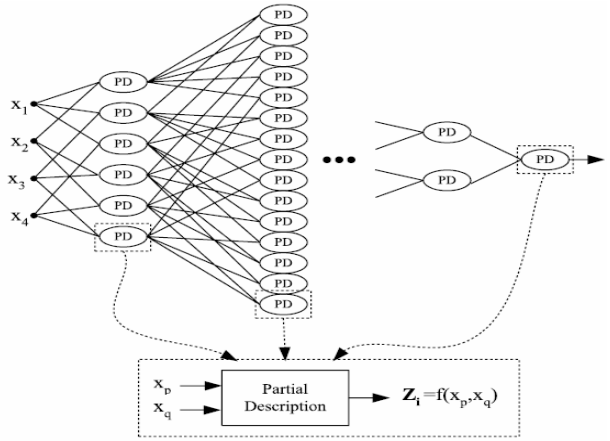

Fig. 1 Basic PNN case 1

Step 4: Determine the number of input variables and the order of the polynomial forming a partial description (PD) of data

We determine the regression polynomial structure of a PD related to PNN structure; for details refer to Table 2. In particular, we select the input variables of a node from N input variables x_1; x_2; . . . ; x_N. The total number of PDs located at the current layer differs according to the number of the selected input variables from the nodes of the preceding layer. This results in $k = N!/(N! – r!)r!$ nodes, where r is the number of the chosen input variables. The choice of the input variables and the order of a PD itself helps select the best model with respect to the characteristics of the data, model design strategy, nonlinearity and predictive capability [8].

Step 5: Estimate the coefficients of the PD

The vector of coefficients C_i is derived by minimizing the mean squared error between y_i and $\hat{y}_i$.

TABLE II REGRESSION POLYNOMIAL STRUCTURE

Order	No. of input:1	No. of input:2	No. of input:3
1	linear	[1]Bilinear	**trilinear**
2	quadratic	BiQuadratic-1[2] BiQuadratic-2[3]	**triQuadratic-1** **triQuadratic-2**
3	cubic	Bicubic-1[4] Bicubic-2	**tricubic-1** **tricubic-2**

$$PI = E = \frac{1}{n_{tr}} \sum_{i}^{n_{tr}} (y_i - \hat{y}_i)^2 \tag{4}$$

Using the training data subset, this gives rise to the set of linear equations

$$Y = X_i C_i \tag{5}$$

Apparently, the coefficients of the PD of the processing nodes in each layer are derived in the form

$$C_i = (X_i^T X_i)^{-1} X_i^T Y, \tag{6}$$

Where

$$\begin{aligned} Y &= [y_1 y_2 y_3 .. y_{n_{tr}}]^T \\ Xi &= [X_{1i} X_{2i} ... X_{ki} ... X_{n_{tr}i}]^T \\ X_{ki}^T &= [x_{ki1} x_{ki2} ... x_{kin} .. x_{ki1}^m x_{ki2}^m .. x_{kin}^m]^T \\ C_i &= [c_{0i} c_{1i} .. c_{n'i}]^T \end{aligned} \tag{7}$$

with the following notations: i the node number, k the data number, n_{tr} the number of the training data subset, n the number of the selected input variables, m the maximum order, and n0 the number of estimated coefficients. This procedure is implemented repeatedly for all nodes of the layer and also for all layers of PNN starting from the input layer and moving to the output layer.

Step 6: Select PDs with the best predictive capability

[1] Bilinear pd $=c_0+c_1x_1+c_2x_2$
[2] BiQuadratic-1 pd= $c_0+c_1x_1+c_2x_2+c_3x_1^2+c_4x_2^2+c_5x_1x_2$
[3] Biquadratic-2 pd = $c_0+c_1x_1+c_2x_2+ c_3x_1x_2$
[4] Bicubic-1 pd= $c_0+c_1x_1+c_2x_2+c_3x_3+c_4x_1^2+c_5x_2^2+c_6x_3^2+c_7x_1x_2+c_8x_1x_3+c_9x_2x_3+c_{10}x_1^3+c_{11}x_2^3+c_{12}x_3^3+c_{13}x_1^2x_2+c_{14}x_1x_2^2+c_{15}x_1^2x_3+c_{16}x_1x_3^2+c_{17}x_2^2x_3+c_{18}x_2x_3^2$

Each PD is estimated and evaluated using both the training and testing data sets. Then we compare these values and choose several PDs, which give the best predictive performance for the output variable. Usually we use a predetermined number W of PDs.

Step 7: Check the stopping criterion

The stopping condition indicates that an optimal PNN model has been accomplished at the previous layer, and the modeling can be terminated. This condition reads as $PI_j > PI_*$ where PI_j is a minimal identification error of the current layer whereas PI_* denotes a minimal identification error that occurred at the previous layer.

Step 8: Determine new input variables for the next layer

If PI_j (the minimum value in the current layer) has not been satisfied (so the stopping criterion is not satisfied), the model has to be expanded. The outputs of the preserved PDs serve as new inputs to the next layer.

III. *GENETIC PARTIAL DESCRIPTION (GPD)*

As mentioned in Sec.2, there are two parameters (the order of polynomial and the number of input variables in each PD) which affect the performance of network. PNN performance depends strongly on the number of input variables and type or order in each PD. These parameters must be chosen in advance before the architecture of PNN is constructed. In most cases, those parameters are determined by the trial and error method, which has a heavy computational load and low efficiency. If we can determine those parameters properly, the performance of network will be increased.

In this paper, GPNN (genetic polynomial neural network) has been introduced. GPNN has been included by nodes that called GPD. Each GPD uses genetic algorithm to make these determinations. In our method, each GPD is represented by binary chromosome that is illustrated in Fig.2.

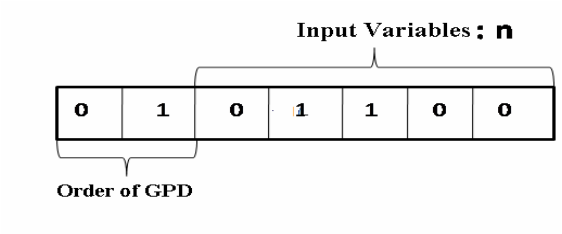

Fig. 2 Structure of binary chromosome

TABLE III ORDER OF GPD

The order of	Code
1	00
2	01
2	10
3	11

The chromosome is made of n+2 bits. Two bits are used to show the order of GPD according to Tables 3 and n bits are the number of entire input variables candidates in the current layer. The input candidate is represented by a 1 bit if it is chosen as an input variable to the GPD and by a 0 bit it is not chosen. For example, chromosome 010110 presents a GPD with order=2 (quadratic) and X_2, X_3 are two input variables. In addition, type of GPD is determinate by Table 2.

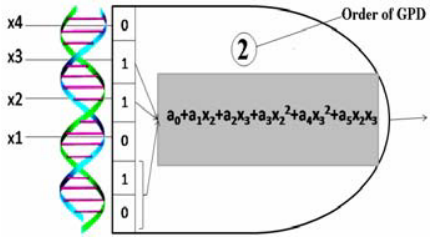

Fig. 3 GPD (genetic partial description)

A. Glearning algorithm

The Glearing algorithm determines the order and the number of input variables in each GPD. It uses GA to search between possible values to determine design parameters. First, we are going to explain fitness function and selection operator, which are important factors of GA, then we explain Glearning algorithm.

1) *Fitness function*

The important consideration following the representation is choice of fitness function. The genotype representation encodes the problem into a string, whereas the fitness function measures the system performance. For predication and estimation of our system, we define a fitness function as

$$fitness\_function = \frac{1}{EPI} \tag{8}$$

Where EPI is mean squared errors calculated by testing data set.

$$\text{EPI} = \frac{1}{N_{test}} \sum_{i \in test} (y_i - \hat{y}_i)^2 \tag{9}$$

2) *Selection operator*

Simple reproduction allocates offspring strings using a Roulette wheel with slots sized according to fitness. This is a way of choosing members from the population of chromosome in a way that is proportional to their fitness. Parents are selected according to their fitness. The better the fitness of the chromosome, the greater the chance it will be selected.

3) *Other parameters*

We use two points crossover operator with cross over rate (pc^5) =.85 and mutation rate (p^6_m) =1/ (n+2) where n+2 is the number of genes.

4) *Glearning algorithm*

Glearning algorithm uses GA to learn and select w GPDs for current layer. The Glearning algorithm is organized in nine steps. *Step1*: Determine the number of members of initial population (N_{pop}). *Step2*: Determine the number of generation (N_{gen}). *Step3*: create a population of random genes with N_{pop} chromosomes and set number_of_generation=0. *Step4*: evaluate population (Eq.3). *Step 5*: select two chromosomes as parents by Roulette wheel selection operator. *Step6*: Childs (new members) are reproduced by two points cross over

[5] Cross Over rate
[6] Mutation rate

operator. *Step 7*: apply mutation and add new Childs to new generation. *Step 8*: if the number of members is less than N_{pop}, you will go to the *Step 5*. *Step 9*: if number_of_generation $=N_{gen}$, the algorithm will be finished and w chromosomes will be selected from current generation. Otherwise, you go *Step 4* and set number_of_generation= number_of_generation+1. The Glearning algorithm is shown in Fig.4.

IV. GPNN (GENETIC POLYNOMIAL NEURAL NETWORK)

GPNN is made by GPNN algorithm that is shown in Fig.5. GPNN algorithm is organized in five steps. Step1: Determine the number of members of initial population(N_{pop}). Step2: Determine the number of generationes(N_{gen}). Step3: Form train set and test set(splite data into the parts (test set and train set). Step3: determine the number of GPDs for new layer(w).Step4: run Glearing algorithms with N_{pop} and N_{gen}that are determined in step1 and step2(W GPDs have been learned by Glearning algorithm and new layer is made of these GPDs). Step 5: if the best EPI of new layer is less than the best EPI of pervious layer(if (newlayer.best_GPD.EPI<previouselayer_best_GPD.EPI), the new layer will be added to the network and their outpus are selected as inputs to next layer and you will go to the step 4. Otherwise, the algorithm will be finished and output of the best GPD of pervious layer is selected as output of network.

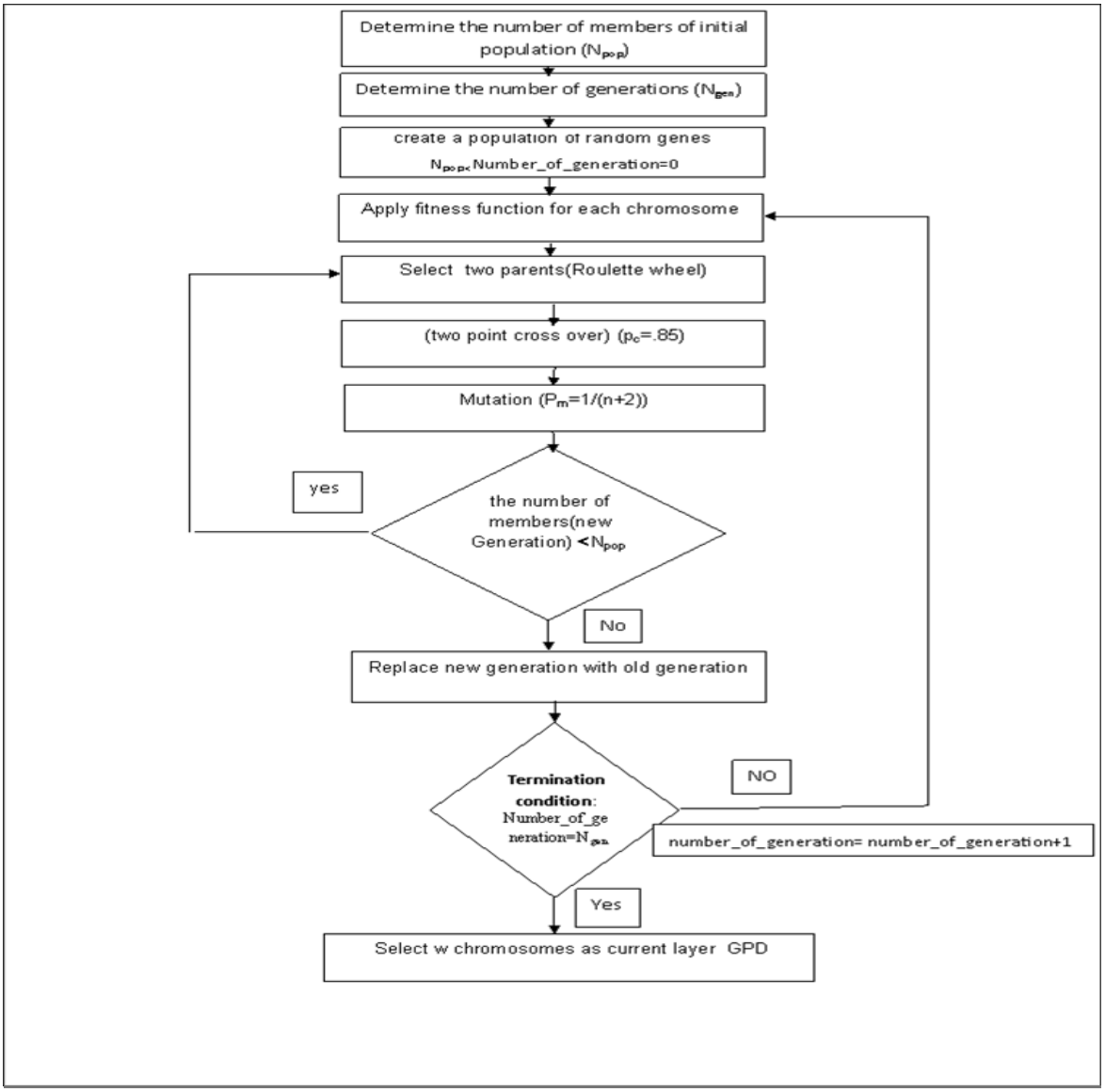

Fig. 4 The Glearning Algorithm

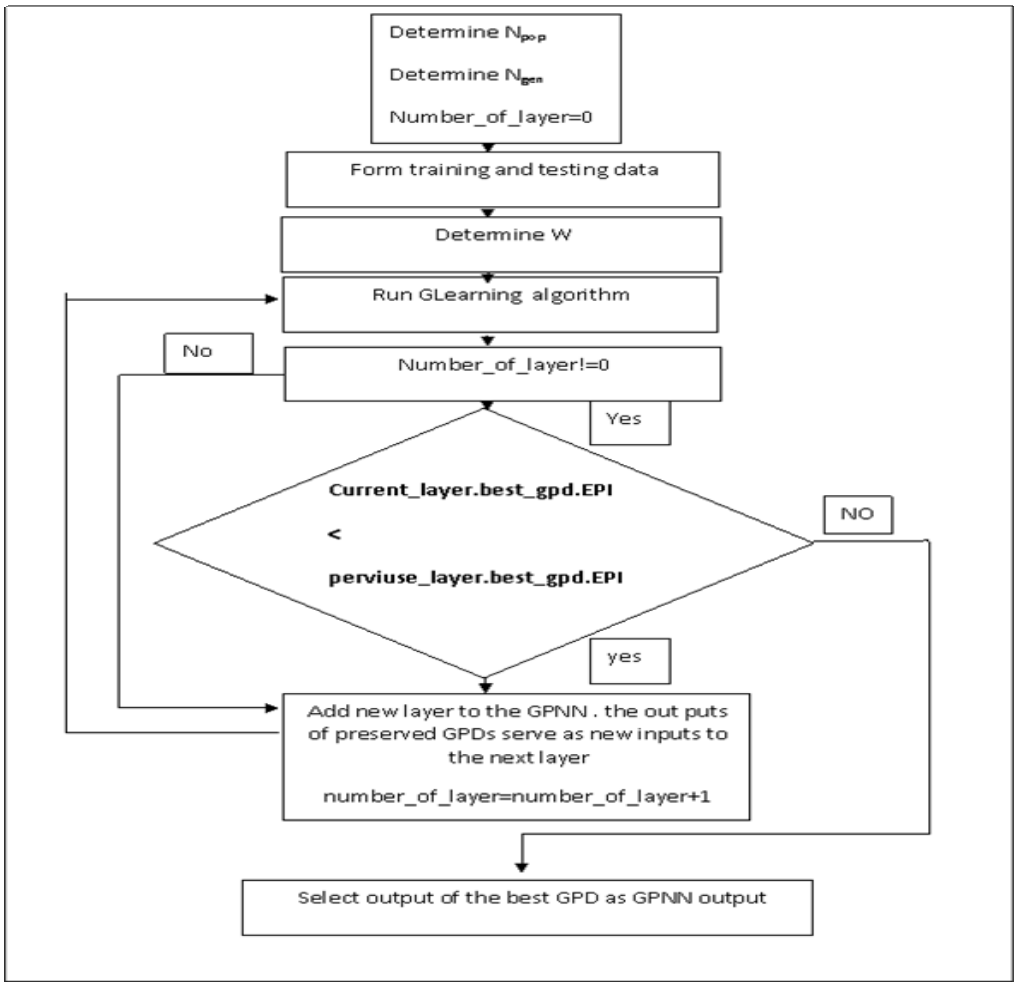

Fig. 5:The GPNN algorithm

V.EXPREMENTAL STUDIES

In this section, we illustrate the performance of the network and elaborate on it is development by experimenting with data coming from quadratic equation and daily values of the Dow Jones industrial index.

A. The quadratic Equation

In this section, the performance of GPNN for Eq.10 is obtained. First, the design parameters of the GPNN are examined. Next, the GPNN will be compared with a conventional PNN.

$$y = 2.368x_1 + 5.538x_1x_2 + 2.10125x_1x_3 + 2.253x_4 \quad (10)$$

Our model is shown in Fig.6.

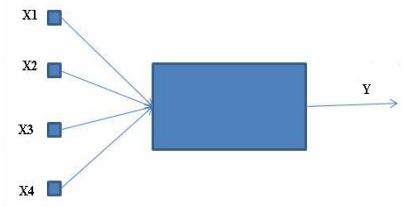

Fig. 6: Proposed Model for Eq.10

a) The number of members of initial population (N_{pop}) and the number of generations (N_{gen}) are two important design parameters, which affect the GPNN performance. The performance index (PI) and extended performance index (EPI) are used in the computer simulation will be the same as given by Eq.4 and Eq.9, respectively. The design parameters of The GPNN are shown in Table.4 and Table 5.

TABLE IV THE NUMBER OF MEMBERS OF INITIAL POPULATION AND PI AND EPI AND TIME

Npop	10	20	30	40	50
Ngen	6	6	6	6	6
W	10	20	30	40	50
EPI	7.99	**4.48E-24**	**4.03E-24**	**1.46E-23**	**1.36E-23**
PI	7.99	**3.76E-25**	**2.49E-25**	**7.74E-26**	**7.74E-26**
No. of layers	1	3	3	1	1
Time(min)	0.5	1.8	3.36	2.05	2.67

Table.4 shows the GPNN with 20 or more than 20 chromosomes (N_{pop}) where the values of PI and EPI is better than others. In addition, Table.5 shows the GPNN

with N_{pop} =20 and N_{gen} >6 where the values of PI and EPI is better than others. Fig.7 depicted structure of the best GPNN.

TABLE V THE NUMBER OF GENERATIONS AND PI AND EPI, TIME

Npop	20	20	20	20
Ngen	6	10	15	20
W	**20**	**30**	**40**	**50**
EPI	**3.487E-24**	**1.369E-23**	**1.369E-23**	**3.216E-24**
PI	**3.764E-25**	**7.746E-26**	**7.746E-26**	**1.309E-25**
No. of layers	3	1	1	3
Time(min)	1.8	1.06	1.9	2.83

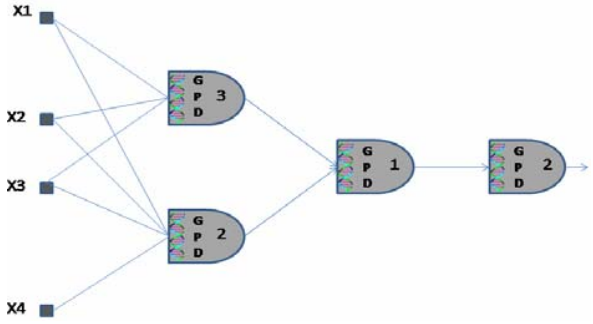

Fig. 7 structure of the best GPNN

We use all possible values for two parameters (N_{pop}, N_{gen}). Eventually, if N_{pop} is between 20, 50, and N_{gen} is under 10, we can construct the best GPNN with less time complexity and high performance (Fig. 8).

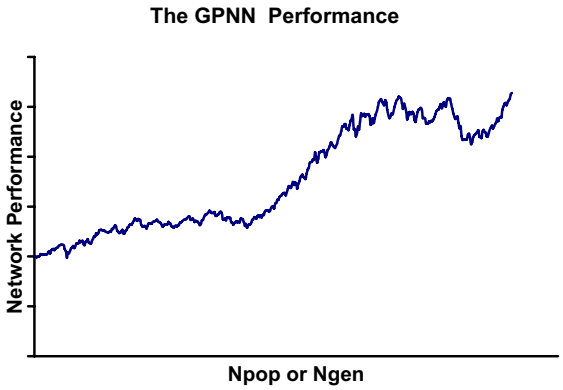

Fig. 8 The GPNN performance

b) Comparison of the GPNN with the PNN.

Table.6 provides a comparison of the GPNN with the PNN. The comparison based on the same performance index for training (PI) and testing (EPI) datasets and the number of layers(network complexity) and time complexity. We use the brute force method to determine design parameters of the PNN such as the number of input variables and the order of polynomial PD. On other hands, we design all PNNs by all available values for the design parameters then we choose the best of them (lowest EPI).

Table.6 shows the GPNN with EPI=3.216E-24, time complexity=2.83(min) is better than the best PNN with EPI=1.369E-23 and time complexity=13.1(min). The structure of the best PNN is shown in Fig.9.

TABLE VI COMPARISON OF THE GPNN WITH THE PNN

Method	EPI	PI	No. of layers	Time(min)
Brute Force PNN	1.369E-23	7.746E-26	1	13.1
GPNN	3.216E-24	1.309E-25	3	2.83

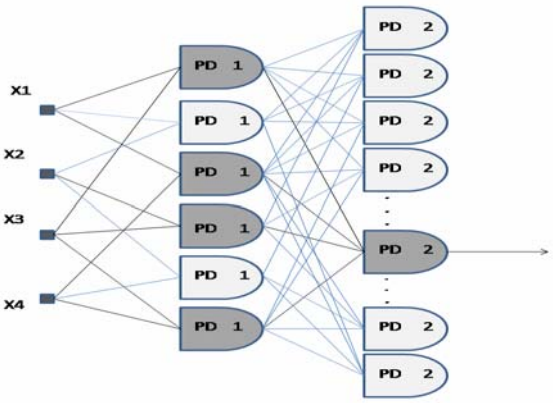

Fig. 9 Structure of the best PNN

B. The Dow Jones series

This time series comprises 2050 Daily closing values of the Dow Jones industrial index (www.finance.yahoo.com) from Jan 1, 1900 to Jan 1, 1960. The dataset is split into the two parts. The first part is used as the training set (from Jan 1, 1900 to Jan 1, 1950) and the remaining part of the dataset is used as the testing set (from Jan 1, 1950 to Jan 1, 1960). It is time series system, which predicates the Dow Jones values by values of four days ago. This system is shown in Fig.10.

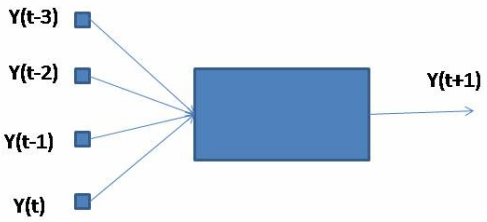

Fig. 10 The forecasting system

Where *Y(t)* equals the value of the Dow Jones index of today. *Y(t-1)* equals the value of the Dow Jones index of yesterday. *Y(t+1)* equals the value of the Dow Jones index of tomorrow. The GPNN and The brute force PNN model this system. Fig.11 shows the output of GPNN follows the actual output very well. Fig.12 shows the error for testing and training data daily. Table.7 shows the GPNN with EPI=5.216E—20 time=9.13 (min) is better than the best PNN with EPI=6.8323 time=50.1(min).

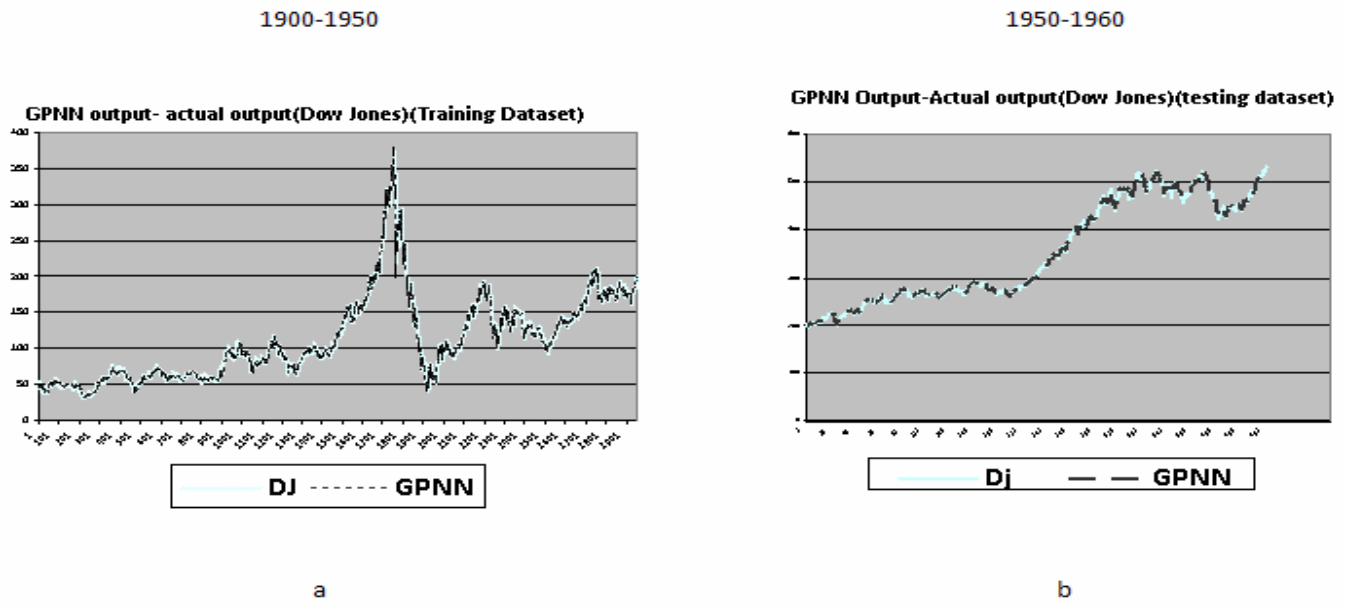

Fig. 11 The GPNN output-actual output

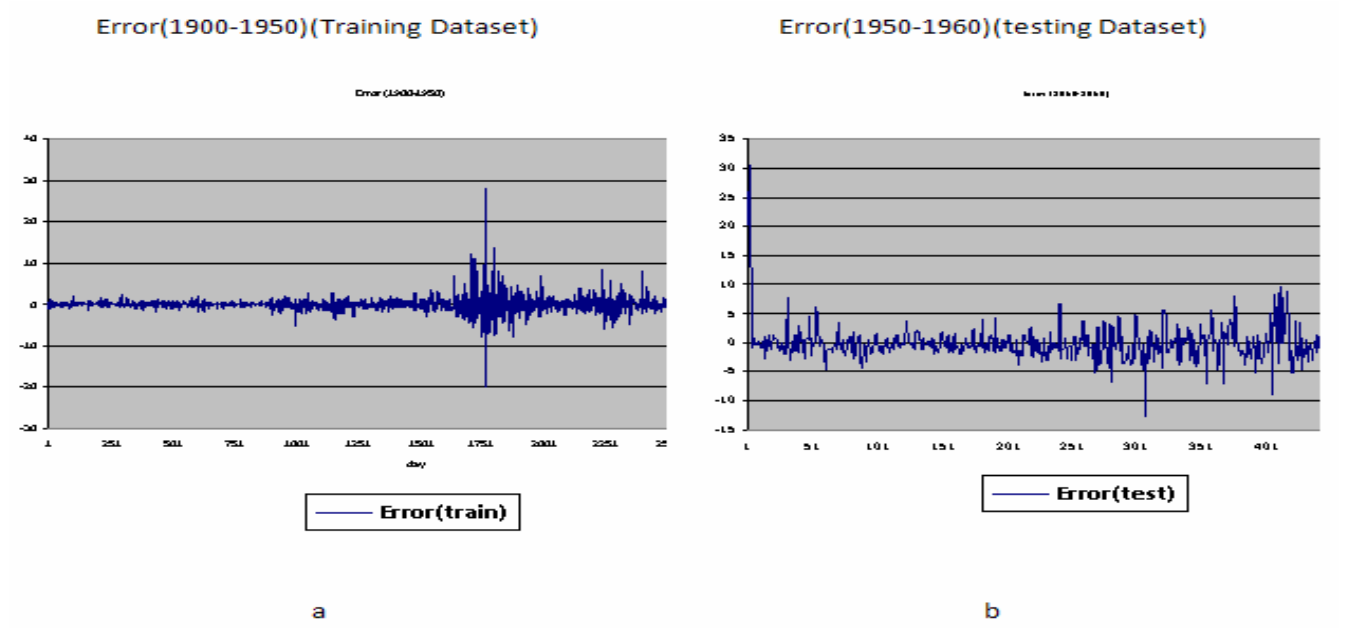

Fig. 12 Error of GPNN

TABLE VII COMPARISON OF THE GPNN WITH THE PNN

Method	EPI	PI	No. of layers	Time(min)
Brute Force PNN	6.8323	5.123	2	50.1
GPNN	5.216E--20	9.309E-21	3	9.13

Eventually, these results indicate that the time complexity and approximation of GPNN is better than the PNN.

VI. CONCLUSION

In this study, we have introduced a new design methodology of polynomial neural network, which calls GPNN. The experimental method was superior to the conventional PNN model in term of modeling performance and time complexity. And the architecture of the model was not fully predetermined, but can be generated during the identification process. GA achieves the flexibility of the model for the optimal choice of inputs and order of the polynomial. In this study, we used testing data for the selection of the most predictive GPDs. This means that, somewhat, the testing data is used in the course of evolved PNN model architecture building. To get more valid generalization ability, determination of number of members of initial population and determination of number of generation and data separating such as training, testing are needed.

REFERENCES

[1] A.G. Ivahnenko, Polynomial theory of complex systems, IEEE Trans. Syst., Man Cybern.SMC-1,1971,pp.364–378.
[2] S.J. Farlow, The GMDH algorithm, in: S.J. Farlow (Ed.), Self-organizing Methods in Modeling: GMDH Type Algorithms, Marcel Dekker, New York, 1984, pp. 1–24.
[3] S.-K. Oh, D.-W. Kim, and B.-J. Park, "A study on the optimal design of polynomial neural networks structure," The Trans. of the Korean Institute of Electrical Engineers,2001, vol. 49d, no. 3, pp.365-396.
[4] G. Ivahnenko, "The group method of data handling: a rival of method of stochastic approximation," Soviet Automatic Control, 1968, vol.13, no. 3, pp. 43-55.
[5] D. E. Goldberg, "Genetic Algorithms in Search, Optimization & Machine Learning", Addison Wesley, 1989.
[6] B.-J. Park, S.-K. Oh, and W. Pedrycz, "The hybrid multi-layer inference architecture and algorithm of FPNN based on FNN and PNN," Joint 9th IFSA World Congress, 2001, pp. 1361-1366.
[7] S.-K. Oh, T.-C. Ahn, and W. Pedrycz, "A study on the self-organizing polynomial neural net works," Joint 9th IFSA World Congress, , 2001, pp.1690-1695.
[8] S.K. Oh, W.pedrycz, and B.J. Park, "polynomial neural networks architecture: Analysis and design," comput.Electr. Eng., ,2003, vol.29, no.6, pp.703-725.
[9] Oh S-K, Pedrycz W, Ahn T-C. "Self-organizing neural networks with fuzzy polynomial neurons". Appl Soft Comput 2002.
[10] Oh S-K, Pedrycz W. "The design of self-organizing polynomial neural networks". Inf Sci 2002, pp.237–258.
[11] Oh S-K, Pedrycz W. "Fuzzy polynomial neuron-based self-organizing neural networks". Int J Gen Syst 2003, pp.237–250.
[12] Oh S-K, Pedrycz W. "Self-organizing polynomial neural networks based on PNs or FPNs: analysis and design. Fuzzy Sets" Syst, 2004, pp.:163–198.
[13] Hayashi, H. Tanaka, "The Fuzzy GMDH algorithm by possibility models and its application," Fuzzy Sets and Systems 36, 1990, pp.245–258.
[14] S.-K. Oh, D.-W. Kim and B.-J. Park, "A study on the optimal design of polynomial neural networks structure," The Trans. of the Korean Institute of Electrical Engineers, 2000 (in Korean),vol. 49d, no. 3, pp. 145-156.
[15] G. Ivahnenko, "The group method of data handling: a rival of method of stochastic approximation," Soviet Automatic Control, 1968, vol.13, no. 3, pp. 43-55.
[16] D. E. Goldberg, Genetic Algorithms in Search, Optimization & Machine Learning, Addison -Wesley, 1989.
[17] M. Bishop, Neural Networks for Pattern Recognition, Oxford Univ. Press, 1995.
[18] B.-J. Park, W. Pedrycz, and S.-K. Oh "Fuzzy polynomial neural networks: hybrid architectures of fuzzy modeling," IEEE Trans. on Fuzzy Systems, October 2002, vol. 10, no. 5, pp. 607-621.