The Design of Self-Organizing Evolved Polynomial Neural Networks Based on Learnable Evolution Model 3

Saeed Farzi
Faculty of Computer Engineering, Islamic Azad University of Kermanshah, Iran

Abstract: *Nowadays, the development of advanced techniques of system modelling has received much attention. Polynomial Neural Network (PNN) is a GMDH-type algorithm (Group Method of Data Handling), which is one of the useful methods for modelling 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 discuss a new design methodology for polynomial neural networks PNN in the framework of Learnable Evolution Model (LEM3). LEM3 is a new approach to evolutionary computation, which employs machine learning to guide evolutionary processes. LEM3 is obtained better performance in shorter time in comparing with other well-known methods. Also, LEM3 appears to be particularly suitable for solving complex optimization problems in which the fitness evaluation function is time consuming. In this paper, we use LEM3 to search between all possible values for the number of input variables and the order of polynomial. Evolved PNN performance is obtained by two nonlinear systems. The experimental part of the study involves two representative time series such as Box-Jenkins gas furnace process and the Dow Jones stock index.*

Keywords: *GMDH, PNN, LEM3, polynomial.*

Received March 28, 2009; accepted January 3, 2010

1. Introduction

The development of advanced techniques of system modelling has received much attention. Group Method of Data Handling (GMDH) [9, 10], originated by Ivakhnenko [11], is a useful data analysis technique for the identification of nonlinear complex systems. GMDH is self organizing and can automatically select essential input variables without using prior information on the relationship among input-output variables [22].

Polynomial Neural Networks (PNN) [18, 23] is a GMDH-type algorithm, which is one of the useful approximation techniques. PNN has architecture similar to feed forward neural networks, whose neurons are replaced by polynomial nodes. The output of the each node in a PNN structure is obtained by using several types of high-order polynomials such as linear, quadratic, and modified quadratic of the input variables.

These polynomials are called as Partial Descriptions (PDs). PNNs have fewer nodes than Artificial Neural Networks (ANNs), but the nodes are more flexible. The PNN shows better performance than the previous fuzzy modelling methods. Although the PNN is structured by a systematic design producer, it has some drawbacks to be solved. If there are sufficiently large number of input variables and data points, PNN algorithm has a tendency to produce overly complex networks. On the other hand, if a small number of input variables are available, PNN does not maintain good performance. Moreover, PNN performance depends strongly on the number of input variables available to the model, and the type or order in each PD.

These parameters must be chosen in advance before the architecture of PNN is constructed. In most cases, they are determined by the trial and error method, which has a heavy computational load and low efficiency. Moreover, the PNN algorithm is a heuristic method so the trial-and-error method does not guarantee that the obtained PNN will be the best one for nonlinear system modelling. Therefore, these drawbacks must be solved with more consideration.

This paper presents a new design methodology of PNN using Learnable Evolution Model 3 (LEM3) to alleviate the above-mentioned drawbacks of the PNN. LEM3 has been used as search method for optimization problem [5]. LEM3 is a new approach to evolutionary computation, which employs machine learning to guide the process of generating new populations [16].

LEM3 integrates two modes of operation, a darwinian evolution mode, which is based on traditional evolutionary computation methods [16, 29], and machine learning mode, which generates new individuals through a process of theory formation and instantiation. Specifically, machine learning mode generates hypotheses that characterize differences between groups of high performing and low performing individuals, and then instantiates these hypotheses to generate new individuals.

In this paper, we use LEM3 to determine the number of input variables to be optimally chosen among many input variables for each PD and to determine the appropriate type of polynomials for each PD. Using LEM3; we can alleviate several disadvantages of the conventional PNN algorithm.

This paper is organized as follows. The PNN algorithm and its problem description are described in section 2. An overview of the LEM methodology is described in section 3. Design evolved PNN is in section 4. Experimental study is covered in section 5, and section 6 concludes the paper.

2. The PNN Algorithm and Its Problem Description

The PNN algorithm is based on the GMDH method and utilization a class of polynomials such as linear, quadratic, and modified quadratic. The design framework of the PNN is based on the following steps [17, 18]:

- *Step 1: Determine system's input variables*. We define the input variables such as $x_{1i}, x_{2i}...x_{Ni}$ related to output variables y_i, where N and i are the numbers of the entire input variables and input-output data set, respectively. The input data are normalized, if required.
- *Step 2: Form training and testing data*. The input-output data set is separated into training (n_{tr}) data set and testing (n_{te}) data set. Then, $n=n_{tr}+n_{te.}$ The training data set is used to construct the PNN model, and the testing data set is used to evolutes the constructed PNN model.
- *Step 3: Choose a structure of the PNN*. The structure of the PNN is strongly depending on the number of input variables and the order of PD in each layer. Two types of PNN structures, namely, the basic PNN structure and the modified PNN structure, are available. Two cases are specified for each type of PNN structure. Table 1 summarizes the various PNN structure:

  a. *Basic PNN*: Structure-the number of input variables of PDs is the same in every layer.
  Case 1: The polynomial order of the PDs is the same in each PD is the same in each layer of the network.
  Case 2: The polynomial order of PDs in 2nd or higher layer is different from those PDs in the 1st layer.

  b. *Modified PNN*: Structure-the number of input variables of PDs varies from layer to layer.
  Case 1: The polynomial order of the PDs is same in every Layer.
  Case 2: The polynomial order of the PDs in the 2nd layer or higher is different from those PDs in the 1st layer.

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			

- *Step 4: Determine the number of input variables and the order of the polynomial forming a PD*. The number of input variables and the type of the polynomial in the PDs are determined arbitrarily. The polynomials are different according to the number of input variables and the polynomial order. Several types of polynomials are shown in the Table 2. The total number of PDs located at the current layer is determined by the number of the selected input variables (r) from the nodes of the preceding layer become the input variables to the current layer. The total number of PDs in the current layer is equal to the combination $_{N}C_{r}$, that is $\frac{N!}{r!(N-r)!}$, where N is the number of nodes in the preceding layer.
- *Step 5: Estimate the coefficients of the PD*. The vector of the coefficients of the PDs, as shown in Table 2, is determined by using the standard Mean Squared Error (MSE) obtained by minimizing the following index.

Table 2. 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

$$E_k = \frac{1}{n_{tr}} \sum_{i}^{n_{tr}} (y_i - Z_{ki})^2, k = 1,2,..., \frac{N!}{r!(N-r)!} \tag{1}$$

[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= $c_0+c_1x_1+c_2x_2+c_3x_3+c_4x_1^2+c_5x_2^2+c_6x_3^2+c_7x_1x_2+ c_8x_1x_3+c_9x_2x_3+c_{10}x_1^3+c_{11}x_2^3+c_{12}x_3^3+c_{13}x_1^2x_2+c_{14}x_1x_2^2+c_{15}x_1^2x_3+c_{16} x_1x_3^2+c_{17}x_2^2x_3+c_{18}x_2x_3^2$

Where, z_{ki} denotes the output of the k-th node with respect to the i-th data and n_{tr} is the number of training data subset.

This step is completed repeatedly for all the nodes in the current layer and, in the sequel, all layers of the PNN starting from the input to the output layer.

- *Step 6: Select PDs with the good predictive capability*. The predictive capability of each PD is evaluated by the performance index using the testing data set. Then, we choose w PDs among ${}_{N}C_{r}$ PDs in due order from the best predictive capability (the lowest value of the performance index). Here, w is the pre-defined number of PDs that must be preserved to the next layer. The outputs of the chosen PDs serve as inputs to the next layer. There are two cases as to the number of the preserved PDs in each layer. If $\frac{N!}{r!(N-r)!}<w$ then the number of the chosen PDs retained for the next layer is equal to $\frac{N!}{r!(N-r)!}$. If $\frac{N!}{r!(N-r)!}>=w$ then the number of the chosen PDs retained for the next layer is equal to *w*.
- *Step 7: Check the stopping criterion*. The PNN algorithm terminates when the number of layers predetermined by the designer is reached.
- *Step 8: Determine new input variables for the nest layer*. If the stopping criterion is not satisfied, the next layer is constructed by repeating Step 4 through Step 8.

Figure 1 shows PNN architecture. In Figure 4 input variables ($x_1 \ldots x_4$), 3 layers, and a PD processing example are considered, where , Z^{j-1}_{i} means the output of the i_{th} node in the $j\text{-}1_{th}$ layer, which is employed as the new input of the j_{th} layer. Black nodes have influence on the best node (output node), and these networks represent the ultimate PNN model. Meanwhile, solid line nodes have no influence over the output node. In addition, the dotted line nodes are excluded in choosing PDs with the best predictive performance in the corresponding layer owing to poor performance.

Therefore, the solid line nodes and dotted line nodes should not be present in the final PNN model. As a result, if the final layer has been constructed, the node with the best predictive capability is selected as the output node. All remaining nodes except the output node in the final layer are discarded. Furthermore, all the nodes in the previous layers that do not have influence on the output node are also removed by tracing the data flow path of each layer.

The PNN is a flexible neural architecture, whose structure is developed by modeling. In particular, the number of the layers and the number of nodes in each layer of the PNN are not fixed in advance (it usually happens in the case of multilayer perceptron) but generated in a dynamic way. Each node exhibits a high level of flexibility and realizes a polynomial type mapping between input and output variables. PNN provides a systematic design procedure, but its performance depends strongly on a few factors stated in section 1. In this paper, we propose a new design procedure using LEM3 for systemic design of PNN with optimum performance.

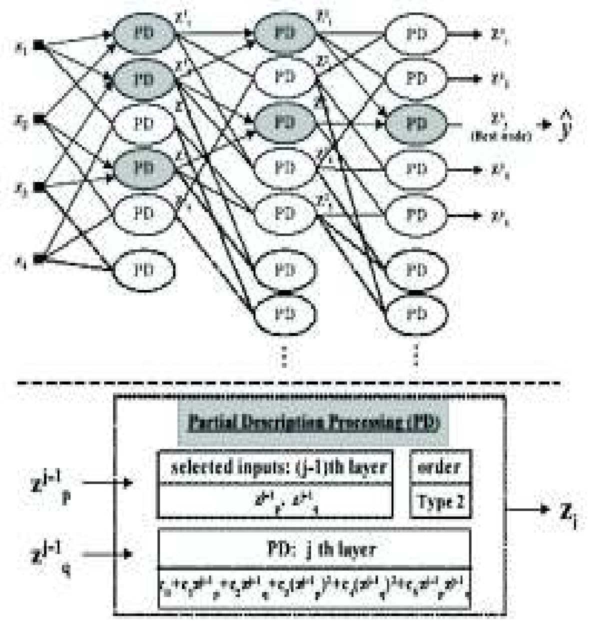


Figure 1. Overall architecture of the conventional PNN.

3. An Overview of the LEM Methodology

The LEM is fundamentally different from the darwinian-type model that underlies most of the current methods of evolutionary computation [6, 9, 15, 16]. The central engine of evolution in LEM is machine Learning mode, which creates new individuals by processes of generalization and instantiation rather than mutation and/or recombination as in the Darwinian-type evolutionary computation methods [2, 12].

Machine learning mode consists of two processes: a hypothesis characterizing differences hypothesis generation, which determines between high fitness "H-group" and low fitness "L-group" individuals in one or more past populations, and hypothesis instantiation, which generates new individuals by instantiating the hypothesis in various ways.

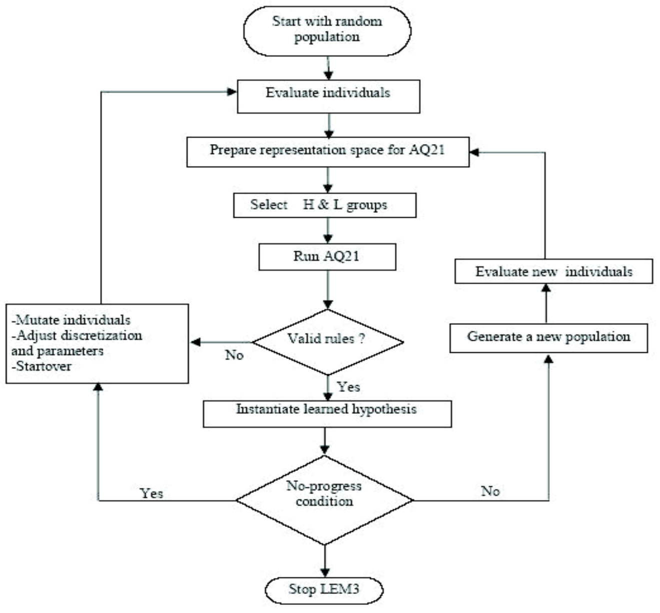
Machine learning mode thus produces new individuals not through semi-random darwinian-type operations, but rather through a deliberate reasoning process involving generation and instantiation of hypotheses about populations of individuals. Thus, in LEM, new individuals are genetically engineered, in the sense that they are determined according to descriptions learned from the analysis of the current and possibly past generations. LEM may alternate between machine learning mode and darwinian evolution mode (executing one of the conventional evolutionary computation methods as in LEM2 implementation) [3], or may rely entirely on machine learning mode. The main parameters of LEM are those that control the way the H-group and the L-group are selected and the number of new individuals that ought to be instantiated from each rule found. Other parameters control the persistence of executing each mode, the start-over operation, and termination conditions.

Selecting H- and L-groups can be done according to a fitness-based method, a population-based method, or a combination of the two. The fitness-based method partitions the population using two fitness thresholds, High Fitness Threshold (HFT) and Low Fitness Threshold (LFT), which specify portions of the total fitness value range in the population that are used to determine the H- and L-groups. The population based method partitions the population using parameters; the High Population Threshold (HPT) and the Low Population Threshold (LPT) that specify the portions of the population to be used as H-group and L-group group, respectively.

The H-group and L-group are then passed as positive and negative training examples to the AQ attributional learning program. AQ learning was selected because it has many features particularly useful for LEM, such as internal disjunction and conjunction in the representation language, the ability to generate rules at different levels of generalization, and others [5].

AQ determines rule sets that differentiate between the H-group and L-group. The search mechanism conducted in machine learning mode can be interpreted as a progressive partitioning of the search space. An H-group description hypothesizes a region or regions that likely contain the optimal individual. Each subsequent H-group description hypothesizes a new, typically more specialized, partition of the search space. Due to this effect, the LEM evolution process may converge to the optimum (local or global) much more rapidly than Darwinian-type evolutionary algorithms. Since partitioning is guided by inductive inference, this process may miss the area with the global optimum. In such cases, LEM executes a start-over operation or temporarily switches to the Darwinian evolution mode [5].

- *The LEM3 Algorithm*
  The LEM3 algorithm that is the most recent implementation of the general LEM methodology will be presented here. The LEM3 algorithm contains several components also found in traditional evolutionary algorithms, including generation of an initial population, generation of a new population, and evaluation of individuals. Other LEM components were designed in order to guide evolution through machine learning, including selection of the H- and L-groups (positive and negative examples for learning), the AQ21 rule learning program, and the instantiation of learned rules into new individuals. Figure 2 shows a flowchart of LEM3 algorithm [5]. In the evaluate individuals phase the program has to determine the value of the fitness function for each individual. It may be the case that this is a very time-consuming operation, especially when fitness is not defined as a mathematical formula, but rather by a different approach, such as through the application of a simulation. The LEM3 has a number of predefined fitness function and possibility of definition of the fitness function by user. Preparation of representation space for AQ21 learning program consists of discretization of individuals before passing them to the AQ21 learning module [5].


Start with random population
Evaluate individuals
Prepare representation space for AQ21
Select H & L groups
Run AQ21
Evaluate new individuals
-Mutate individuals
-Adjust discretization and parameters
-Startover
Valid rules ?
No
Yes
Generate a new population
Instantiate learned hypothesis
Yes
No-progress condition
No
Stop LEM3
Figure 2. A flowchart of LEM3 algorithm.

4. Design PNN Using LEM3

The PNN algorithm must determine the optimal number of input variables and the order of the polynomial (design parameters of PDs) forming a PD in each node. In this paper, a new design method for PNN uses LEM3 to make these determinations. All of initial populations are randomized so that minimum heuristic knowledge is used. The appropriate inputs and order are evolved accordingly and are tuned gradually throughout the LEM3 iterations.

In the evolutionary design procedure, key issues are the encodings of the order of the polynomial, and the optimum input variables as a chromosome and the defining of a criterion to compute the fitness of each chromosome. The detailed representation of the coding strategy and the choice of a fitness function are given.

4.1. Design of PDs

The most important consideration is the representation strategy, that is, how to encode the key factors of the PNN into the chromosome. Binary coding is applied for the available design specification.

The order and the inputs of each node are coded as a finite-length string. Our chromosome is made of two sub-chromosomes. The first one consists of 2 bits for the order of the polynomial (PD); the second one consists of the n bits, which are equal to the number of entire input candidates in the current layer. These inputs candidates are the node outputs of the previous layer. The representation of the binary chromosomes is illustrated in Figure 3.

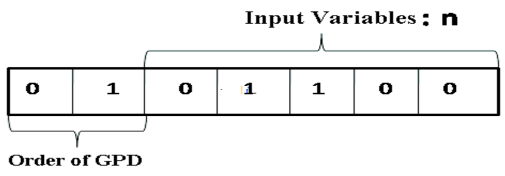


Figure 3. Structure of binary chromosome.

The 1st sub-chromosome is made of 2 bits, and represents several types of the orders of PD. The relationship between bits in the 1st sub-chromosome and the orders of PD is shown in Table 3. Thus, each node can exploit different orders of a polynomial.

The 2nd sub-chromosome has *n* bits, which concatenate the bits of 0s and 1s coding. The input candidate is represented by 1 bit if it is chosen as an input variable to the PD and by 0 bit if it is not chosen. For example, chromosome 010110 presents a evolved PD with order=2 (quadratic) and x2, x3 are two input variables.

Figure 4 shows this node. In addition, the output of this node can be expressed as equation 2. Moreover, it is shown in Figure 4:

$$\hat{y}=f(x_2,x_3)=c_0+c_1x_2+c_2x_3+c_3x_2^2+c_4x_3^2+c_5x_2x_3 \tag{2}$$

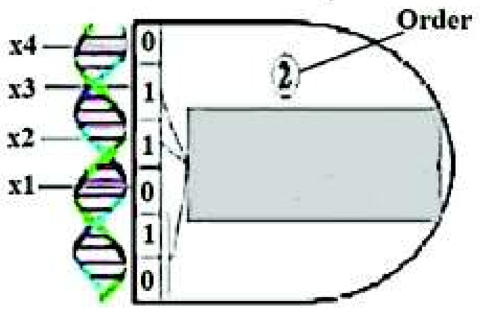


Figure 4. The evolved PD.

Table 3. The order and the code.

The Order GPDThe	Code
1	00
2	01
2	10
3	11

4.2. 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:

$$fitness\_function = \frac{1}{EPI} \tag{3}$$

Where EPI[5] is mean squared errors calculated by testing data set.

$$EPI = \frac{1}{N_{test}} \sum_{i \in test} (y_i - \hat{y}_i)^2 \tag{4}$$

4.3. Design the Evolved PNN

The design procedure of the evolved PNN is made of two main algorithms. First, algorithm1 is to learn nodes and make a layer of learned nodes. It uses LEM3 to find the best design parameters for each node and then it estimates the coefficients of the PDs by using the standard MSE (section 2 and Step 5).

Finally, it selects *w* nodes with good predicative capability. (*w* is determined by user) and it builds a layer from the best nodes (*w*). Second, algorithm2 is used to build the evolved PNN. It uses algorithm1 to build a new layer and then it adds new layer to the network. While the stopping criterion is not satisfied, algorithm 2 adds new layer to the network. We explain algorithm 1 and algorithm 2 in details as follows:

- *Algorithm 1*
  Algorithm 1 uses LEM3 to learn and select w PDs for current layer:
  Step 1: Determine the number of members of initial population (N_{pop}).
  Step 2: Determine the number of generation (N_{gen}).
  Step 3: Run LEM3.
  Step 4: Select w chromosomes as current layer PDs.
- *Algorithm 2*
  The evolved PNN is made by algorithm 2. Algorithm 2 is organized in six steps:
  Step 1: Determine the number of members of initial population (N_{pop}).
  Step 2: Determine the number of generationes (N_{gen}).
  Step 3: Form train set and test set (split data into the two parts (test set and train set).
  Step 4: Determine the number of PDs for new layer(w).
  Step 5: Run algorithm 1 with N_{pop} and N_{gen}that are determined in step1 and step2 (w PDs have been learned by algorithm 1 and new layer is made of these PDs).
  Step 6: If the best EPI of new layer is less than the best EPI of pervious layer (if (newlayer.best_PD.EPI<previouselayer_best_PD.EPI), the new layer will be added to the network and their

[5]Note that EPI (Extended Performance Index) is mean squared errors calculated by testing data set and PI (Performance Index) is mean squared errors calculated by training data set

outputs 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 PD of pervious layer is selected as output of network.

5. Experimental Studies

In this section we illustrate the performance of the network and elaborate on its development by experimenting with data coming from the gas furnace process [17] and time series Dow Jones stock index. These two are representative examples of well-documented data sets used in the realm of nonlinear modelling [1, 17, 18, 19].

5.1. Gas Furnace Process

The delayed terms of methane gas flow rate u(t) and carbon dioxide density y(t) such as u(t-1), u(t-2), u(t-3), y(t-1), y(t-2), y(t-1) are used as input variables to the evolved PNN. The actual system output y(t) is used as the output variable for this model. This model is shown in Figure 5. The total data set (296 input-output pairs) is split into two parts. The first part (150 pairs) is used as the training set, and the remaining part of the data is used as the testing set. Using the training data set, the coefficients of the polynomial are estimated by using the standard LSE. The performance index is defined as MSE.

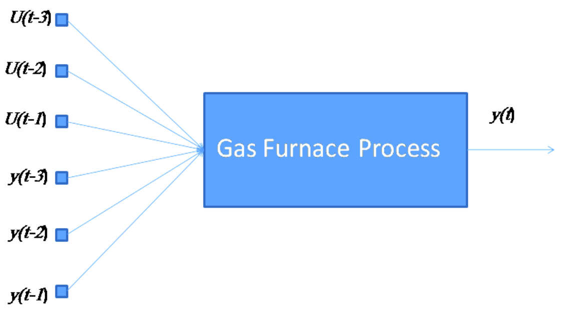


Figure 5. Gas furnace process.

The performance of the evolved PNN for the gas furnace process is obtained. First, the design parameters of the evolved PNN are examined. Next, the evolved PNN will be compared with a conventional PNN and other well-known models.

The number of members of initial population (N_{pop}) and the number of generations (N_{gen}) are two important factors, which affect the evolved PNN performance. The Performance Index (PI) and Extended Performance Index (EPI) are used in the computer simulation will be the same as given by equitation 1, and equitation 4, respectively. The design parameters of the evolved PNN are shown in Tables 4 and 5.

Table 4. The number of members of initial population and PI and EPI and time.

Npop	10	20	30	40	50
Ngen	5	5	5	5	5
w	50	50	50	50	50
EPI	.1301	.125	.1091	.1228	.1278
PI	.0299	.0145	.0130	.0282	.0272
No. of Layers	2	3	3	2	2
Time	.5	1.8	3.36	2.05	2.67

Table 4 shows the evolved PNN with 30 chromosomes (N_{pop}) whereas the values of PI and EPI are better than others. In addition, Table 5 shows the the evolved PNN with N_{pop}=30 and N_{gen}=30 whereas the values of PI and EPI is better than others (PI=0.0125, EPI=0.1011). Figure 6 depicts structure of the best evolved PNN.

Table 5 .The number of generations and PI and EPI, time.

Npop	30	30	30	30	30	30
Ngen	5	10	30	50	80	100
w	50	50	50	50	50	50
EPI	.1091	.1090	.1011	.1087	.1170	.1201
PI	.0130	.0128	.0125	0.0131	.0131	.0221
No. of Layers	3	3	3	3	3	2
Time	3.36	3.40	3.55	3.83	3.95	4.01

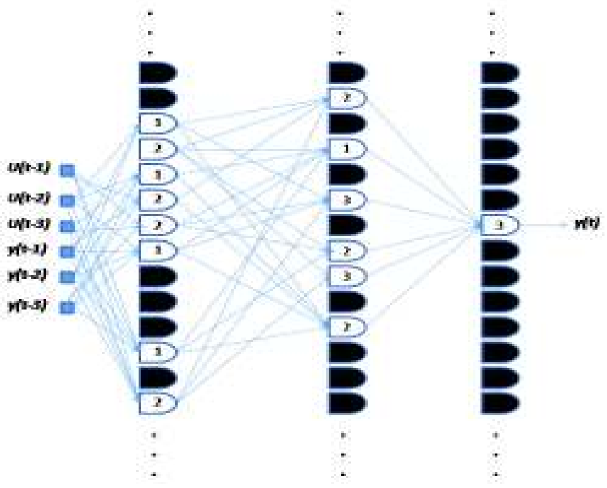


Figure 6. The evolved PNN.

Figure 7 depicts performance index of the evolved PNN with three layers.

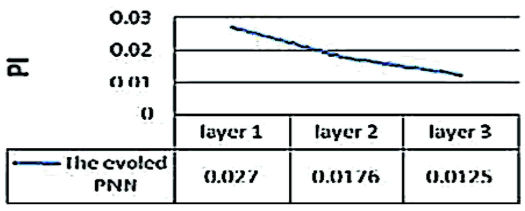


Figure 7. The evolved PNN- performance index.

Figure 8 depicts extended performance index of the evolved PNN with three layers.

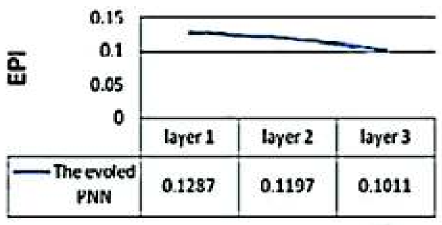


Figure 8. The evolved PNN-extended performance index.

Table 6. The evolved PNN- conventional PNN.

Method	EPI	PI	No. of Layers	Time(min)
Conventional PNN	3.8E-19	9.8E-21	2	80
Evolved PNN	5.2E-20	9.3E-21	3	3.25

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 evolved PNN and The conventional PNN model this system Table 6 shows the evolved PNN with EPI=5.2E-20 time=3.25 (min) is better than the best PNN with EPI=3.8E-19 time=80(min).

Table 7 provides a comparison of our model with other techniques. The comparison is based on the same performance index for the training and the testing data set. Our model outperforms other models both in terms of their accuracy and higher generalization capabilities. The structure of the optimal model is selected and proper inputs and order for each node are harmonized gradually throughout the LEM3 operations without requiring specific prior knowledge of the target system or its components. It should be noted that some good values of conventional PNN in Table 7 are obtained based on 4th or 5th layer. The PNN has a tendency to produce overly complex networks as it tries to stretch for the last bit of accuracy. Although the network size is a simple, the evolved PNN has a comparable performance.

Table 7. Comparison of identification error with previous models.

Model				Mean Squared Error		
				PI*	PI	EPI
Box and Jenkins` model [2]				.710		
Tong` s model [27]				0.469		
Sugeno`s model [25]				0.355		
Sugeno `s model [26]				0.190		
Xu and Zailu`s model [30]				0.328		
Pedrycz `s model [24]				0.320		
Chen`s model [7]				0.268		
Oh and Pedryczs` model [20]				0.123	0.020	0.271
Kim *et al.*`s model [13]					0.034	0.244
Lin *et al.*`s model [14]					0.071	0.261
Conventional PNN [21]	Type 1	Basic	Case1	0.057	0.017	0.148
		Basic	Case2	0.057	0.017	0.147
		Modified	Case1	0.046	0.015	0.103
		Modified	Case2	0.045	0.016	0.111
	Type 2	Basic	Case1	0.029	0.012	0.085
		Basic	Case2	0.027	0.021	0.085
		Modified	Case1	0.035	0.017	0.095
		Modified	Case2	0.039	0.017	0.101
Our Model					0.0125	0.1011
PI*-- performance index over the entire data set.						

5.2. 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 Figure 9.

Figure 10 shows the output of the evolved PNN follows the actual output very well. Figure 11 shows the error for testing and training data daily.

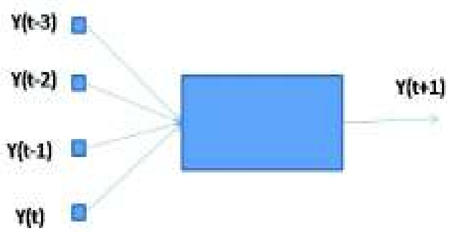
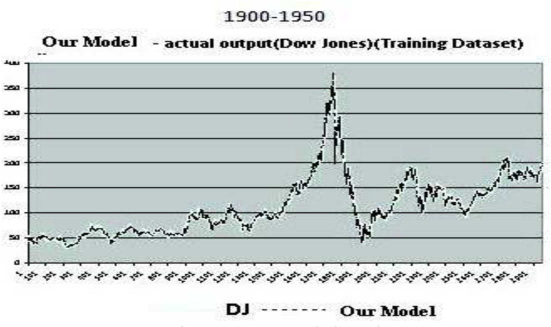
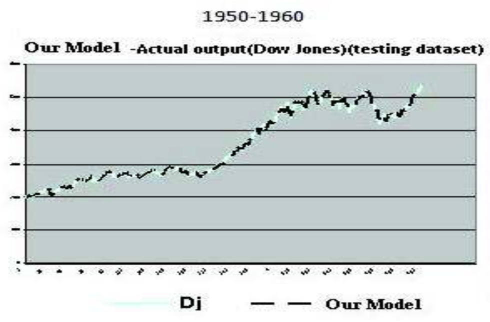


Figure 9. The forecasting system.

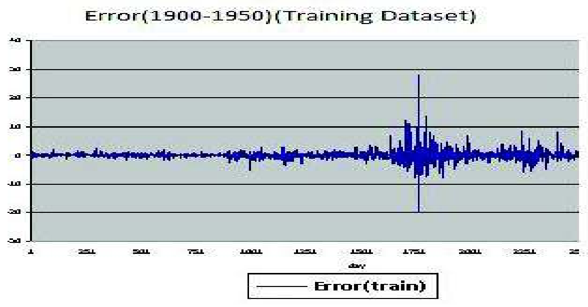


a) Actual output on training data.

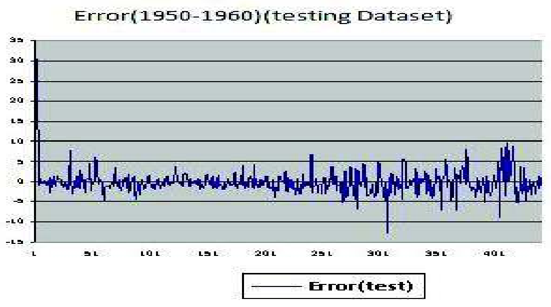


b) Actual output on testing data.

Figure 10. The evolved PNN output.



a) Error on training data.



b) Error on testing data.

Figure 11. Error of the evolved PNN.

6. Conclusions

In this study, we have introduced a new design methodology of polynomial neural network, which calls evolved PNN. The experimental method was superior to the conventional PNN model in term of modelling performance and time complexity. The architecture of the model was not fully predetermined, but can be generated during the identification process. We use LEM3 to search between all possible values for the number of input variables and the order of polynomial.

References

[1] Bishop M., *Neural Networks for Pattern Recognition*, Oxford University Press, 1995.

[2] Box P. and Jenkins M., *Time Series Analysis: Forecasting and Control*, Holden-Day, 1976.

[3] Cervone G., "LEM2: Theory and Implementation of the Learnable Evolution Model," *Reports of the Machine Learning and Inference Laboratory, MLI,* George Mason University, Virginia, pp. 99-6, 1999.

[4] Cervone G., Kaufman K., and Michalski S., "Recent Results from the Experimental Evaluation of the Learnable Evolution Model," *in Proceeding of the Genetic and Evolutionary Computation Conference, GECCO-2002*, Poland, pp. 1-2, 2002.

[5] Cervone G., Kaufman K., and Michalski S., "Experimental Validations of the Learnable Evolution Model," *in Proceedings of 2000 Congress on Evolutionary Computation*, USA, pp. 1064-1071, 2000.

[6] Cervone G., Panait, A., and Michalski S., "The Development of the AQ20 Learning System and Initial Experiments," *in Proceedings of 10th International Symposium on Intelligent Information Systems*, Poland, pp. 1-3, 2001.

[7] Chen Q., Xi G., and Zhang J., "A clustering Algorithm for Fuzzy Model Identification, Fuzzy," *Journal of Sets System*, vol. 98, no. 5, pp. 319-329, 1998.

[8] Farlow J., *Self-Organizing Methods in Modelling: Gmdh Type Algorithms Book Description*, CRC Press, 1984.

[9] Goldberg D., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Longman Publishing Co., 1989.

[10] Ivahnenko G., "The Group Method of Data Handling: A Rival of Method of Stochastic Approximation," *Journal of Soviet Automatic Control*, vol. 13, no. 3, pp. 43-55, 1968.

[11] Ivahnenko G., "Polynomial Theory of Complex Systems," *in Proceeding of IEEE Transaction System, Man Cybern, SMC-1*, South Korea, pp. 364-378, 1971.

[12] Kaufman K. and Michalski S., "The AQ18 System for Machine Learning and Data Mining System: An Implementation and User's Guide," *Reports of the Machine Learning and Inference Laboratory, MLI 00-3*, George Mason University, USA, pp. 1-3, 2000.

[13] Kim T., Park K., Ji H., and Park M., "A new Approach to Fuzzy Modelling," *IEEE*

Transaction Fuzzy System, vol. 5, no. 3, pp. 328-337, 1997.

[14] Lin Y. and Cunningham A., "A new Approach to Fuzzy-Neural Modelling," *IEEE Transaction on Fuzzy System*, vol. 3, no. 2, pp. 190-197, 1995.

[15] Michalski S., "Learnable Evolution Model Evolutionary Processes Guided by Machine Learning," *Journal of Machine Learning*, vol. 38, no. 2, pp. 9-40, 2000.

[16] Michalski S., "Learning and Evolution: An Introduction to Non-Darwinian Evolutionary Computation," *in Proceedings of 12th International Symposium on Methodologies for Intelligent Systems*, USA, pp. 21-30, 2000.

[17] Oh K., Ahn C., and Pedrycz W., "A Study on the Self-Organizing Polynomial Neural Net Works," *in Proceedings of Joint 9th IFSA World Congress*, Australia, pp. 1690-1695, 2001.

[18] Oh K., Kim W., and Park J., "A Study on the Optimal Design of Polynomial Neural Networks Structure," *The Transaction of the Korean Institute of Electrical Engineers*, vol. 49, no. 3, pp. 365-396, 2001.

[19] Oh K., Kim W., and Park J., "A Study on the Optimal Design of Polynomial Neural Networks Structure," *The Transaction of the Korean Institute of Electrical Engineers*, vol. 49, no. 3, pp. 145-156, 2000.

[20] Oh K. and Pedrycz W., "Identification of Fuzzy Systems by Means of an Auto-Tuning Algorithm and its Application to Nonlinear Systems," *Journal of Fuzzy Sets System*, vol. 115, no. 2, pp. 205-230, 2000.

[21] Oh K. and Pedrycz W., "The Design Self-Organizing Polynomial Neural Networks," *Journal of Information Sciences*, vol. 14, no.1, pp. 237-258, 1998.

[22] Park J., Oh K., and Pedrycz W., "Polynomial Neural Networks Architecture: Analysis and Design," *Journal of Computers and Electrical Engineering*, vol. 29, no. 6, pp. 703-725, 2003.

[23] Park J., Oh K., and Pedrycz W., "The Hybrid Multi-layer Inference Architecture and Algorithm of FPNN Based on FNN and PNN," *in Proceeding of Joint 9th IFSA World Congress,* Heidelberg, pp. 1361-1366, 2001.

[24] Pedrycz W., "An Identification Algorithm in Fuzzy Relational System," *Journal of Fuzzy Sets System*, vol. 13, no. 2, pp. 153-167, 1984.

[25] Sugeno M. and Yasukawa T., "A Fuzzy-Logic-Based Approach to Qualitative Modelling," *IEEE Transaction on Fuzzy System*, vol. 1, no. 1, pp. 7-31, 1993.

[26] Sugeno M. and Yasukawa T., "Linguistic Modelling Based on Numerical Data, in: IFSA'91, Brussels," *in Proceedings of Computer, Management and Systems Science*, China, pp. 264-267, 1991.

[27] Tong M., "The Evaluation of Fuzzy Models Derived from Experimental Data," *Journal of Fuzzy Sets System*, vol. 4, no. 1, pp. 1-12, 1998.

[28] Wetter M., Wright J., "Comparison of a Generalized Pattern Search and a Genetic Algorithm Optimization Method," *in Proceedings of 8th International IBPSA Conference*, Netherlands, pp. 11-14, 2003.

[29] Wnek J. and Michalski R., "Learning Hybrid Descriptions," *in Proceedings of the on Intelligent Information Systems*, Poland, pp. 13-95, 1995.

[30] Wojtusiak J., "The LEM3 Implementation of Learnable Evolution Model User's Guide," *in Proceedings of Reports of the Machine Learning and Inference Laboratory, MLI 04-5,* George Mason University, USA, pp. 4-7, 2004.

[31] Xu W. and Zailu Y., "Fuzzy Model Identification Self-learning for Dynamic System," *IEEE Transaction System Man Cybern. SMC-17*, vol. 4, no.1, pp. 683-689, 1987.

Saeed Farzi is a faculty member at the Department of Computer Engineering, Islamic Azad University-beranch of Kermanshah, Iran. He received his BS in computer engineering from Razi University, Iran in 2004, and his MS in artificial intelligence from Isfehan University, Iran in 2006. His current research interests include artificial intelligence.