ELSEVIER

Information Sciences 141 (2002) 237–258

INFORMATION SCIENCES
AN INTERNATIONAL JOURNAL

www.elsevier.com/locate/ins

The design of self-organizing Polynomial Neural Networks

Sung-Kwun Oh [a], Witold Pedrycz [b,c,*]

[a] *School of Electrical and Electronic Engineering, Wonkwang University, 344-2, Shinyong-Dong, Iksan, Chon-Buk 570-749, Republic of Korea*

[b] *Department of Electrical and Computer Engineering, University of Alberta, Edmonton, Alta., Canada AB T6G 2G6*

[c] *Systems Research Institute, Polish Academy of Sciences, Warsaw, Poland*

Received 30 December 2000; received in revised form 30 October 2001; accepted 23 November 2001

Abstract

In this study, we introduce and investigate a class of neural architectures of Polynomial Neural Networks (PNNs), discuss a comprehensive design methodology and carry out a series of numeric experiments. PNN is a flexible neural architecture whose structure (topology) is developed through learning. In particular, the number of layers of the PNN is not fixed in advance but becomes generated on the fly. In this sense, PNN is a self-organizing network. The essence of the design procedure dwells on the Group Method of Data Handling (GMDH). Each node of the PNN exhibits a high level of flexibility and realizes a polynomial type of mapping (linear, quadratic, and cubic) between input and output variables. The experimental part of the study involves two representative time series such as Box–Jenkins gas furnace data and a pH neutralization process.

Keywords: Polynomial Neural Networks; Group Method of Data Handling; Design procedure; High-order polynomial; Multi-variable systems; Time series

1. Introduction

Recently, a lot of attention has been directed to advanced techniques of system modeling. The panoply of the existing methodologies and detailed

* Corresponding author. Tel.: +1-780-492-4661; fax: +1-780-492-1811.
E-mail address: pedrycz@ee.ualberta.ca (W. Pedrycz).

0020-0255/02/$ - see front matter
PII: S0020-0255(02)00175-5

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. The global nonlinear behavior of the model may also cause undesired effects (the well-known is a phenomenon of data approximation by high-order polynomials where such approximation leads to unexpected ripples in the overall nonlinear relationship of the model). 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. It is also worth stressing that the nonlinear form of the model acts as a two-edge sword: while we gain flexibility to cope with experimental data, we are provided with an abundance of nonlinear dependencies that need to be exploited in a systematic manner. 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] 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. First, it tends to generate quite complex polynomial for relatively simple systems (data). Second, owing to its limited generic structure (quadratic two-variable polynomial), GMDH also tends to produce an overly complex network (model) when it comes to highly nonlinear systems.

In this study, in alleviating the problems with the GMDH algorithm, we introduce a new class of Polynomial Neural Networks (PNNs). In a nutshell, these networks come with a high level of flexibility as each node (processing element forming a PD) can have a different number of input variables as well as exploit a different order of the polynomial (say, linear, quadratic, cubic, etc.). In comparison to well-known neural networks whose topologies are commonly prior to all detailed (parametric) learning, the PNN architecture is not fixed in advance but becomes fully optimized (both structurally and parametrically). Especially, the number of layers of the PNN architecture can be modified with new layers added, if required.

In this study, we provide with a general taxonomy of the PNNs, discuss detailed learning schemes and include detailed experimental studies. The material is organized into six sections. First, in Section 2 we discuss the GMDH algorithm that is regarded as an underlying design method of the PNN architecture. Section 3 is devoted to various architectures of the PNN and their development. A suite of experimental studies is covered in Section 4. Concluding remarks are included in Section 5.

2. The GMDH algorithm

The GMDH algorithm uses estimates of the output variable obtained from simple primeval regression equations that include small subsets of input variables [2]. To elaborate on the essence of the approach, we adhere to the following notation. Let the original data set consist of a column of the observed values of the output variable y and N columns of the values of the independent system variables, that is $\mathbf{x} = x_1, x_2, \ldots, x_N$. The primeval equations form a PD which comes in the form of a quadratic regression polynomial

$$z = A + Bu + Cv + Du^2 + Ev^2 + Fuv. \tag{1}$$

In the above expression A, B, C, D, E, and F are parameters of the model, u, v are pairs of variables standing in $\mathbf{x}$ whereas z is the best fit of the dependent variable y.

The generation of each layer is completed within three basic steps:

Step 1. In this step we determine estimates of y using primeval equations. Here, u and v are taken out of all independent system variables $x_1, x_2, \ldots, x_N$. In this way, the total number of polynomials we can construct via (1) is equal to $N(N-1)/2$. The resulting columns z_m of values, $m = 1, 2, \ldots, N(N-1)/2$, contain estimates of y resulting from each polynomial that are interpreted as new "enhanced" variables that may exhibit a higher predictive power than the original variables being just the input variables of the system, $x_1, x_2, \ldots, x_N$.

Step 2. The aim of this step is to identify the best of these new variables and eliminate those that are the weakest ones. There are several specific selection criteria to do this selection. All of them are based on some performance index (mean square, absolute or relative error) that express how the values z_m follow the experimental output y. Quite often the selection criterion includes an auxiliary correction component that "punishes" a network for its excessive complexity. In some versions of the selection method, we retain the columns (z_m) for which the performance index criterion is lower than a certain predefined threshold value. In some other versions of the selection procedure, a prescribed number of the best z_m is retained. Summarizing, this step returns a list of the input variables. In some versions of the method, columns of $x_1, x_2, \ldots, x_N$ are replaced by the retained columns of $z_1, z_2, \ldots, z_k$, where k is the total number of the retained columns. In other versions, the best k retained columns are added to columns $x_1, x_2, \ldots, x_N$ to form a new set of the input variables. Then the total number N of input variables changes to reflect the addition of z_m values or the replacement of old columns x_N with z_m new total number of input variables.

If Step 2 is completed within the generation of the current layer (or the current iteration) of the design procedure, the iteration of the next layer (or the next iteration) begins immediately by repeating step 1 as described above, otherwise we proceed with step 3.

Step 3 consists of testing whether the set of equations of the model can be further improved. The lowest value of the selection criterion obtained during this iteration is compared with the smallest value obtained at the previous one. If an improvement is achieved, one goes back and repeats steps 1 and 2, otherwise the iterations terminate and a realization of the network has been completed. If we were to make the necessary algebraic substitutions, we would have arrived at a very complicated polynomial of the form which is also known as the Ivahnenko polynomial

$$\hat{y} = a + \sum_{i=1}^{m} b_i x_i + \sum_{i=1}^{m} \sum_{j=1}^{m} c_{ij} x_i x_j + \sum_{i=1}^{m} \sum_{j=1}^{m} \sum_{k=1}^{m} d_{ijk} x_i x_j x_k \cdots, \tag{2}$$

where a, b_i, c_{ij}, d_{ijk} and so forth are the coefficients of the polynomial.

3. 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.

3.1. 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

$$(\mathbf{X}_i, y_i) = (x_{1i}, x_{2i}, \ldots, x_{Ni}, y_i), \quad i = 1, 2, 3, \ldots, n. \tag{3}$$

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

$$y = f(x_1, x_2, \ldots, x_N). \tag{4}$$

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

$$\begin{aligned} \hat{y} &= \hat{f}(x_1, x_2, \ldots, x_N) \\ &= c_0 + \sum_{k1} c_{k1} x_{k1} + \sum_{k1k2} c_{k1k2} x_{k1} x_{k2} + \sum_{k1k2k3} c_{k1k2k3} x_{k1} x_{k2} x_{k3} + \cdots, \end{aligned} \tag{5}$$

where c_k's denote the coefficients of the model.

To determine the estimated output $\hat{y}$, we construct a PD form for each pair of independent variables in the first iteration according to the number of the input variables. Here one determines the parameters of PD by the least square method by using given training data. Furthermore we choose the optimal model forming the first layer. In the sequel, we construct new PDs using intermediate variables (for example z_m) being generated in the current iteration. Afterwards, we take another pair of new input variables, and repeat operation until the stopping criterion has been satisfied. Once the final layer has been constructed, the node characterized by the best performance is selected as the output node. The remaining nodes in that layer are discarded. Furthermore, all the nodes of previous layers that do not have influence on the estimated output node are also removed by tracing the data flow path of each iteration. Overall, the framework of the design procedure of the PNNs comes as a sequence of the following steps.

Step 1: Determine system's input variables

Here, we define the input variables as $x_i,\ i = 1, 2, \ldots, N$ 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 $(\mathbf{X}_i, y_i) = (x_{1i}, x_{2i}, \ldots, x_{Ni}, y_i),\ i = 1, 2, 3, \ldots, n$ 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 $n = n_{\mathrm{tr}} + n_{\mathrm{te}}$. 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.

Step 3: Choose a structure of the PNN

The structure of PNN is selected on the basis of 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. More specifically, the main features of these architectures are as follows:

Table 1
A taxonomy of various PNN structures

Layer	PD Type		PNN structure
	No. of input variables	Order of polynomial	
First layer	p	P	(1) $p = q$: Basic PNN (a) $P = Q$: Case 1 (b) $P \neq Q$: Case 2
Second to fifth layer	q	Q	(2) $p \neq q$: Modified PNN (a) $P = Q$: Case 1 (b) $P \neq Q$: Case 2

($p = 2, 3, 4$, $q = 2, 3, 4$; $P = 1, 2, 3$, $Q = 1, 2, 3$).

(a) Basic PNN structure – The number of input variables of PDs is the same in every layer.

Case 1. The polynomial order of PDs is the same in each layer of the network.

Case 2. The polynomial order of PDs in the second layer or higher has a different or modified type in comparison with the one of PDs in the first layer.

(b) Modified PNN structure – The number of input variables of PDs varies from layer to layer.

Case 1. The polynomial order of PDs is same in every layer.

Case 2. The polynomial order of PDs in the second layer or higher has a different or modified type in comparison with the one of PDs in the first layer.

The outstanding feature of the modified PNN structure is its high flexibility. Not only the order but the number of independent input variables may vary between PDs located at each layer. Therefore the complex PDs as well as the simple PDs can be utilized effectively according to the various kinds of modified PNN structures by taking into consideration both compactness and mutual input–output relationships encountered at each layer.

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 $x_1, x_2, \ldots, x_N$. 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.

Step 5: Estimate the coefficients of the PD

The vector of coefficients $\mathbf{C}_i$ is derived by minimizing the mean squared error between y_i and z_{mi}

Table 2
Regression polynomial structure

Order	No. of inputs		
	1	2	3
1	Linear	Bilinear	Trilinear
2	Quadratic	Biquadratic-1 Biquadratic-2	Triquadratic-1 Triquadratic-2
3	Cubic	Bicubic-1 Bicubic-1	Tricubic-1 Tricubic-1

1: Basic type; 2: Modified type.

$$E = \frac{1}{N_{\mathrm{tr}}} \sum_{i=0}^{N_{\mathrm{tr}}} (y_i - z_{mi})^2. \tag{6}$$

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

$$\mathbf{Y} = \mathbf{X}_i \mathbf{C}_i. \tag{7}$$

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

$$\mathbf{C}_i = (\mathbf{X}_i^{\mathrm{T}} \mathbf{X}_i)^{-1} \mathbf{X}_i^{\mathrm{T}} \mathbf{Y}, \tag{8}$$

where

$$\mathbf{Y} = [y_1 y_2 \cdots y_{n_{\mathrm{tr}}}]^{\mathrm{T}}, \quad \mathbf{X}_i = [\mathbf{X}_{1i} \mathbf{X}_{2i} \cdots \mathbf{X}_{ki} \cdots \mathbf{X}_{n_{\mathrm{tr}} i}]^{\mathrm{T}},$$
$$\mathbf{X}_{ki}^{\mathrm{T}} = [1 x_{ki1} x_{ki2} \cdots x_{kin} \cdots x_{ki1}^m x_{ki2}^m \cdots x_{kin}^m],$$
$$\mathbf{C}_i = [c_{0i} c_{1i} c_{2i} \cdots c_{n'i}]^{\mathrm{T}}$$

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 n' 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

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 (i) a predetermined number W of PDs or (ii) the prespecified cutoff value of the performance index the PD has to exhibit in order to be retained at the next generation of the PNN algorithm. Especially the method of (ii) uses the threshold criterion θ_m to select the node with the best performance in each layer. A new PD is preserved (retained) if the following condition holds:

$$E_j < \theta_m = E_* + \delta, \tag{9}$$

where E_j is a minimal identification error of the current layer, θ_m stands for a threshold value while E_* is a minimal identification error of the previous layer. Furthermore δ is a positive constant whose value is specified by the model developer.

The second method has some practical drawbacks. It cannot effectively reduce a large number of nodes and avoid a large amount of time-consuming iterations of PNN layers. The first method is better with this regard as it confines the computing to the predetermined value of W.

Step 7: Check the stopping criterion

Two termination methods are exploited here:

(i) The stopping condition shown in (10) indicates that an optimal PNN model has been accomplished at the previous layer, and the modeling can be terminated. This condition reads as

$$E_j \geqslant E_*, \tag{10}$$

where E_j is a minimal identification error of the current layer whereas E_* denotes a minimal identification error that occurred at the previous layer.

(ii) The PNN algorithm terminates when the number of iterations predetermined by the designer is reached.

It is prudent to take into consideration a stopping condition for better performance and the number of iterations predetermined by the designer. This criterion helps achieve a balance between model accuracy and its complexity.

Step 8: Determine new input variables for the next layer

If E_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. This is captured by the expression

$$x_{1i} = z_{1i}, \quad x_{2i} = z_{2i}, \quad \ldots, \quad x_{wi} = z_{wi}. \tag{11}$$

The PNN algorithm is carried out by repeating steps 4–8 of the algorithm.

3.2. The PNN structure

We introduce two kinds of PNN structures, namely the basic and the modified PNN. While their structure has been captured in Table 1, here we discuss their architectural details.

3.2.1. Basic PNN structure

The design of the PNN structure continues and involves a generation of some additional layers. These layers consist of PDs for which the number of input variables is the same in every layer. Two cases (*Case* 1 and *Case* 2) for the regression polynomial in each layer are considered.

Case 1. As stated, in this case the order of the polynomial of PDs is the same across the entire network. The resulting network is visualized in Fig. 1.

It becomes apparent that all PDs are the same and the design of the network repeats (that is we use the same technique as applied to the first layer).

Case 2. The order of the polynomial of PDs in the second layer or higher is different in comparison with the units located in the first layer, see Fig. 2.

In this figure, the notations Z_i' related to the second layer and more point out that the order of the polynomial is different in comparison to the one (Z_i) encountered at the first layer.

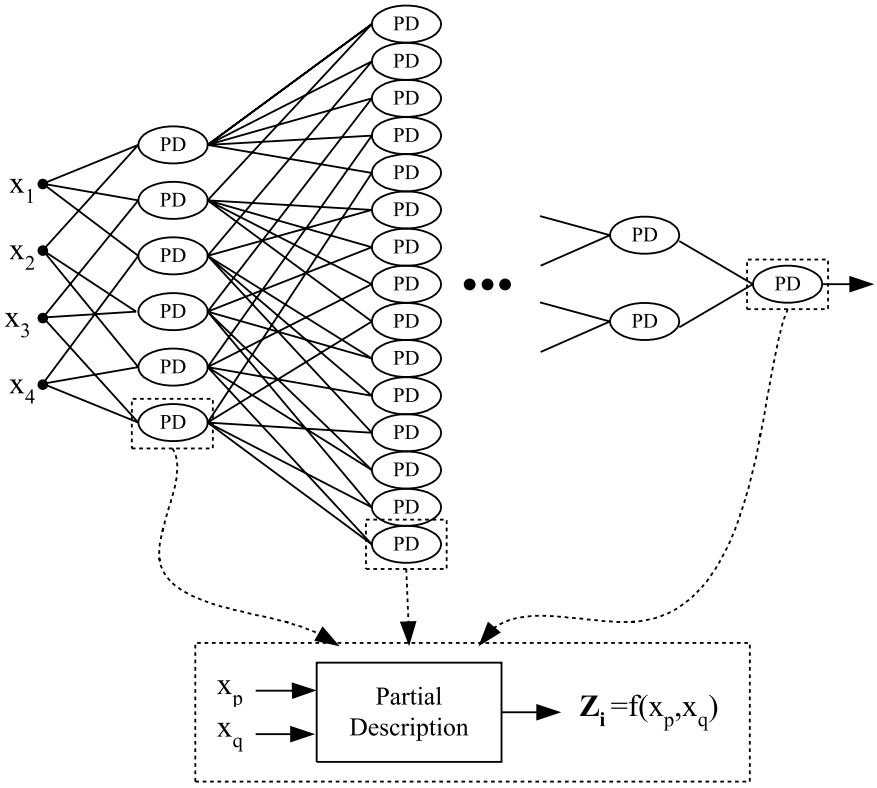

Fig. 1. Configuration of the basic PNN structure – Case 1.

3.2.2. Modified PNN structure

The outstanding feature of the modified PNN structure resides in its increased variability. Not only an order of the regression polynomial varies but the number of the input variables of each PDs can be changed. Therefore, the simplex PDs as well as the complex PDs can be utilized effectively by taking into consideration a structural form of input–output relationships between the nodes of each layer. Two cases for the regression polynomial in each layer can be sought as well.

Case 1 – The order of PDs is the same in every layer, see Fig. 3. For example, consider that the PDs of the first layer are the form of the 2nd order (quadratic) regression polynomial:

$$z = c_0 + c_1 x_p + c_2 x_p^2. \tag{12}$$

We estimate the parameters of the PDs and determine the best group of the PDs. In the second layer, the PDs are the second-order polynomials with two

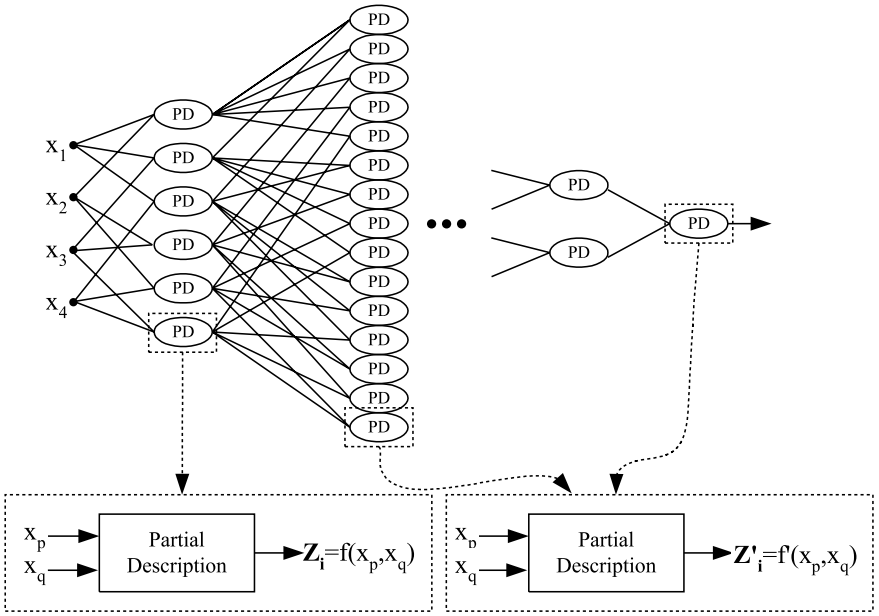

Fig. 2. Configuration of the basic PNN structure – Case 2.

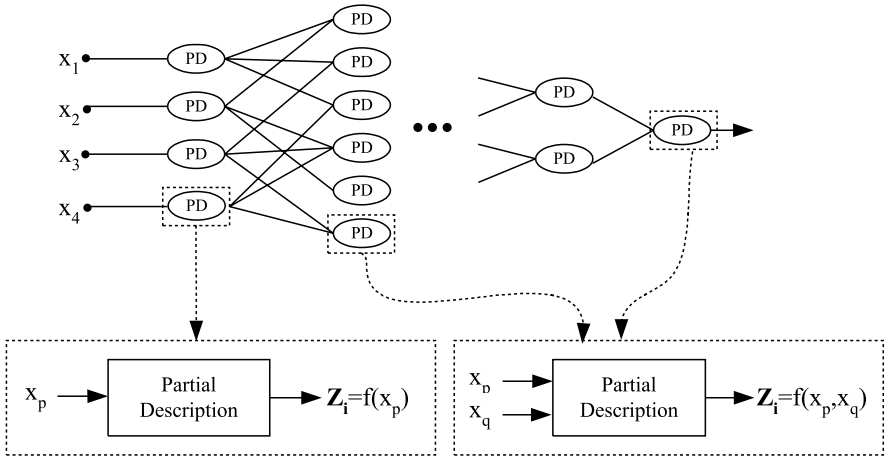

Fig. 3. Configuration of the modified PNN structure – Case 1.

variables. Even though the polynomial order of PDs is the same as that of the first layer, the number of input variables can be different from that of the first layer.

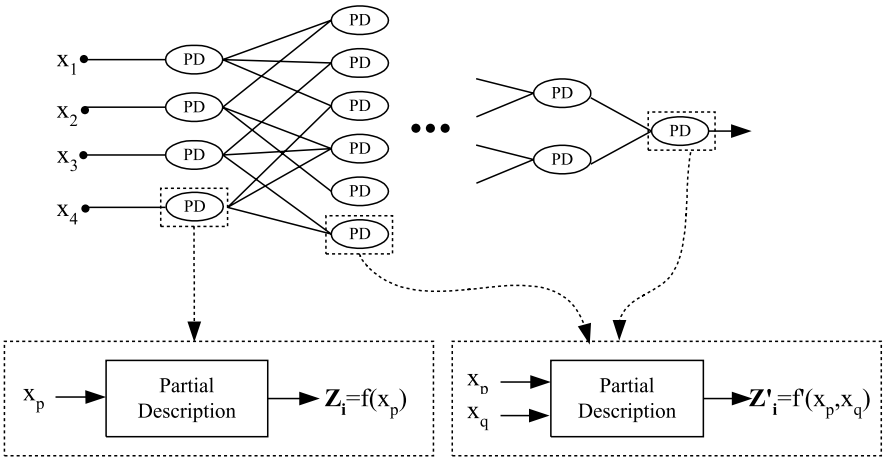

Fig. 4. Configuration of the modified PNN structure – Case 2.

Case 2 – The order of the polynomials of PDs in the second and higher layers differs in comparison with the PDs existing in the first layer, Fig. 4.

4. 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 [3] and pH neutralization process [18]. These two are representative examples of well-documented data sets used in the realm of fuzzy modeling. We also contrast the performance of the model introduced here with those existing in the literature.

4.1. Gas furnace process

The time series data resulting from the gas furnace process have been intensively studied in the previous literatures [3–15]. For easy reference, we highlight the main design steps discussed in the previous section.

Step 1: Determine system's input variables

The delayed terms of methane gas flow rate, $u(t)$ and carbon dioxide density, $y(t)$ are used as system input variables such as $u(t-3)$, $u(t-2)$, $u(t-1)$, $y(t-3)$, $y(t-2)$, and $y(t-1)$. $y(t)$ is a single output variable. We choose the input variables of nodes in the first layer of PNN structure from these system input variables. We use two types of system input variables of PNN structure, Type I and Type II to design an optimal model from gas furnace process data.

Type I utilizes four system input variables such as $u(t-2)$, $u(t-1)$, $y(t-2)$, and $y(t-1)$ and Type II utilizes six system input variables explained above.

Step 2: Form a training and testing data set

The total data set includes 296 input–output pairs for the proposed PNN modeling. The total data set is divided into two parts, one is used for training purposes (148 input–output data) and the remaining serves for testing purposes.

Step 3: Choose a structure of the PNN

We consider two kinds of PNN structures – the basic and modified one.

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

We determine the number of the input variables and the order of PD from N system input variables obtained in step 1. Step 3 concerns the decision as to the structure of the PNN. The PDs differ according to the number of input variables and the polynomial order of a node. Here Type 1, Type 2, and Type 3 stand for a linear, quadratic, and modified quadratic regression polynomial, respectively.

Step 5: Estimate the coefficients of a PD

Using the training data subset obtained in step 2, the coefficients (c_i) of a PD are estimated by the standard least squares method.

Step 6: Select PDs with the best predictive capability

Using both the training and testing data subset obtained in step 2, each PD of the current layer is evaluated by computing the performance index defined as the mean squared error

$$PI(EPI) = \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2, \tag{13}$$

where y_i is the actual output, $\hat{y}_i$ is the estimated one of each PD, and m stands for the total number of data.

Then we compare these values and choose several PDs by a predefined number, 30, which give better predictive performance than remaining PDs of the current layer. Such selected PDs of the current layer are retained as the inputs to the successive layer.

Step 7: Check the stopping criterion (condition)

Because of a large amount of computing, we follow a practical guideline to confine the depth of the PNN to a maximum of five layers. It will be shown that this selection is well supported by experimental evidence gained through intensive experimentation.

Step 8: Determine new input variables for the next layer

If the stopping condition of step 7 has not been not satisfied, the output values estimated in the first layer serve at the second layer as input variables. The algorithm goes through steps 4–8 and generates PDs at the next layer.

In the sequel, we discuss the results produced by various PNNs.

(a) *The basic PNN structure*

Case 1 – The values of the performance index vis-à-vis number of layers of the PNN with Type 3 in Type II architecture are shown in Fig. 5. Considering the training and testing data sets, the best results for the network of Type I are obtained when using three inputs of Type 1, (that are quantified as $PI = 0.0175$, $EPI = 0.1486$). The best results for the network of Type II coming with $PI = 0.0124$ and $EPI = 0.0849$ have been reported when using four inputs and Type 3.

Case 2 – Fig. 6 visualizes the performance index of the PNN structure. The notation used here, namely "Type 1 → Type 2" states that the polynomial order of the PDs changes from Type 1 (those are PDs in the first layer) to Type 2 (when dealing with PDs in the second layer or higher). When the polynomial order of PDs changes from Type 3 to Type 1, the best results for the Type I network are quantified by $PI = 0.0175$ and $EPI = 0.1476$. These values are obtained for the PNN structure with three node inputs. When the order of the polynomial of the PDs changes from Type 1 to Type 2 (Type 1 → Type 2), this gives better results for the Type II network both for the training and testing sets. Especially in this case, the PNN structure with three node inputs is characterized by the best results ($PI = 0.021$, $EPI = 0.0849$, respectively).

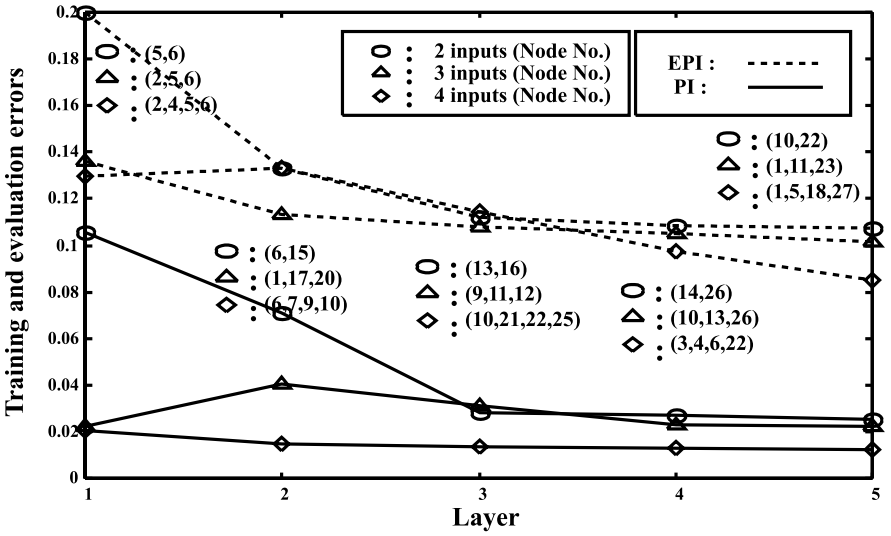

Fig. 5. Performance index for training and evaluation in Type II (each layer includes neurons of Type 3).

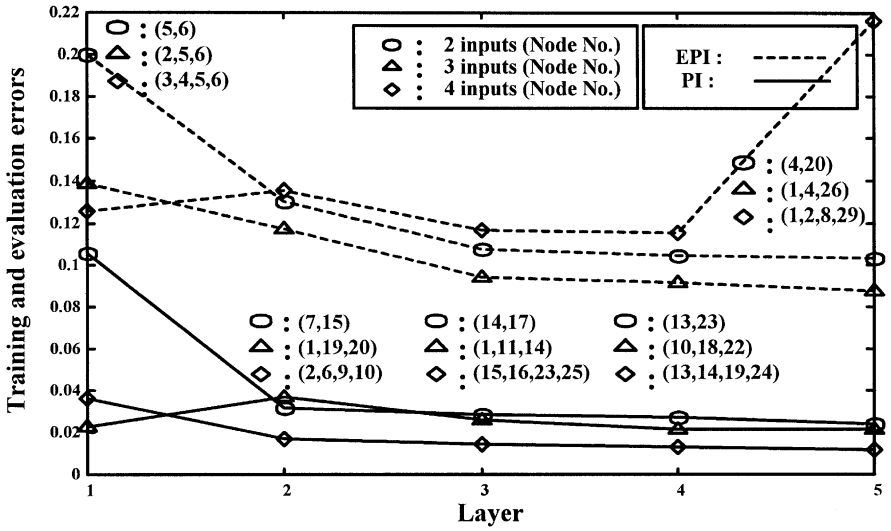

Fig. 6. Performance index for training and testing in Type II network (Type 1 → Type 2).

Fig. 7 illustrates the detailed topology of the network. The shadowed nodes in Fig. 7 identify optimal nodes in each layer, namely those with the best predictive performance.

(b) *The modified PNN structure*

Case 1 – The values of the performance index of the PNN structure with Type 2 in Type II is shown in Fig. 8.

Case 2 – As before, the performance index summarizes the behavior of the network, Fig. 9.

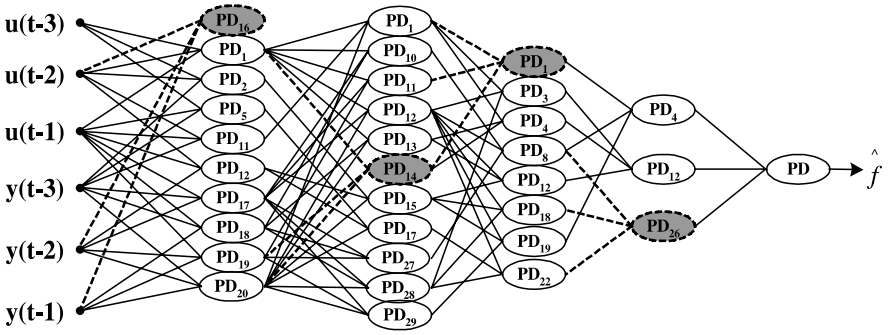

Fig. 7. Optimal PNN structure of Type II (three inputs and Type 1 → Type 2); see description in text.

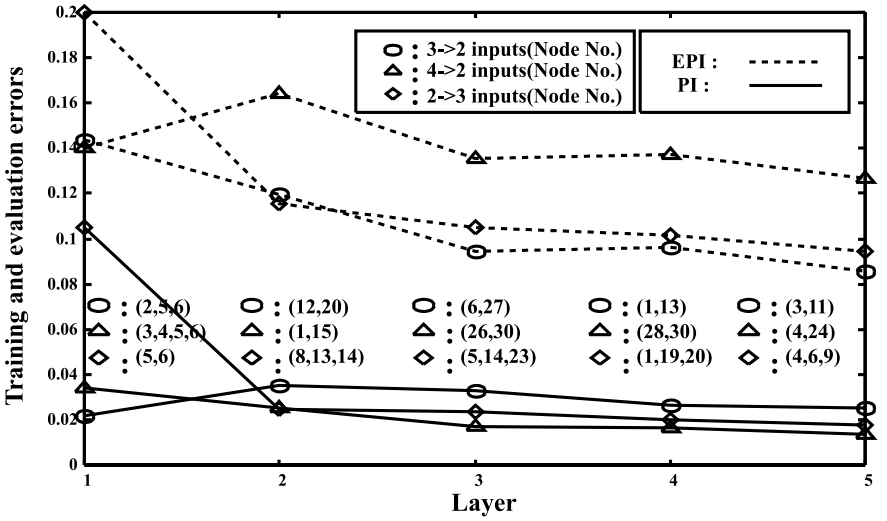

Fig. 8. Performance index for training and evaluation in Type II (every layer: Type 2).

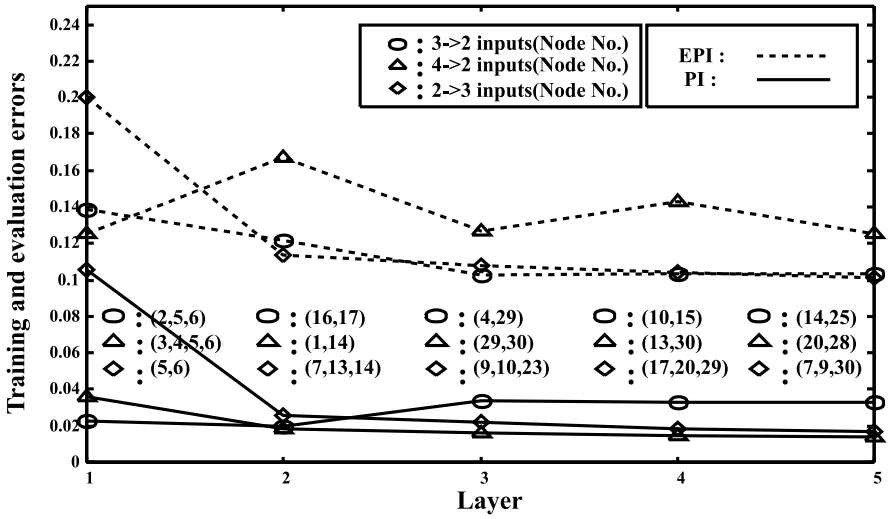

Fig. 9. Performance index for training and evaluation in Type II (first layer: Type 1, second layer or higher: Type 2).

Table 3 contrasts the performance of the PNN network with other fuzzy models studied in the literature. The experimental results clearly reveal that the PNN outperforms the existing models both in terms of better approximation

Table 3
Comparison of identification error with previous fuzzy models

Model				Mean squared error		
				PI	PI_s	EPI_s
Box and Jenkins' model [3]				0.710		
Tong's model [4]				0.469		
Sugeno and Yasukawa's model [5]				0.355		
Sugeno and Yasukawa's model [6]				0.190		
Xu and Zailu's model [7]				0.328		
Pedrycz's model [8]				0.320		
Chen's model [12]				0.268		
Gomez-Skarmeta's model [14]				0.157		
Oh and Pedrycz' model [9]				0.123	0.020	0.271
Kim et al.'s model [10]				0.055		
Kim et al.'s model [11]					0.034	0.244
Leski and Czogala's model [13]				0.047		
Lin and Cunningham's model [15]					0.071	0.261
Our model	Type I	Basic	Case 1	0.057	0.017	0.148
		PNN	Case 2	0.057	0.017	0.147
		Modified	Case 1	0.046	0.015	0.103
		PNN	Case 2	0.045	0.016	0.111
	Type II	Basic	Case 1	0.029	0.012	0.085
		PNN	Case 2	0.027	0.021	0.085
		Modified	Case 1	0.035	0.017	0.095
		PNN	Case 2	0.039	0.017	0.101

PI – performance index over the entire data set, PI_s – performance index on the training data, EPI_s – performance index on the testing data.

capabilities (lower values of the performance index on the training data, PI_s) as well as superb generalization abilities (expressed by the performance index on the testing data EPI_s).

4.2. pH neutralization process

To demonstrate the high modeling accuracy of the PNN, we apply it to a highly nonlinear of pH neutralization of a weak acid and a strong base. This model can be found in a variety of practical areas including wastewater treatment, biotechnology processing, and chemical processing [16,17,19,22,23]. pH is the measurement of the acidity or alkalinity of a solution containing a proportion of water. It is mathematically defined, for dilute solution, as the negative decimal logarithm of the hydrogen ion concentration $[H^+]$ in the solution, that is,

$$\mathrm{pH} = -\log_{10}[\mathrm{H}^+]. \tag{14}$$

In the continuously stirred tank reactor (CSTR) [18,21] investigated acetic acid (HAC) of concentration C_a flows into the tank at flow rate F_a, and is neutralized by sodium hydroxide (NaOH) of concentration C_b which flows into the tank at rate F_b. The equations of the CSTR can be described as follows (here we assume that the tank is perfectly mixed and isothermal, cf. [18]). The process equations for the CSTR is given by

$$\frac{V\,\mathrm{d}w_a}{\mathrm{d}t} = F_a C_a - (F_a + F_b)W_a, \tag{15a}$$

$$\frac{V\,\mathrm{d}w_b}{\mathrm{d}t} = F_b C_b - (F_a + F_b)W_b, \tag{15b}$$

where the constant V is the volume of the content in the reactor, w_a and w_b are the concentrations of the acid and base, respectively.

The above equation describes how the concentration of w_a and w_b changes dynamically with time subject to the input streams F_a and F_b. To obtain the pH in the effluent, we need to find a relation between instantaneous concentrations w_a and w_b and pH values. This relationship can be described by a nonlinear algebra equation known as the titration or characteristic curve. Depending on the chemical species used, the titration curve varies. Here we consider the case that a weak influent neutralized by a strong reagent. The words strong and weak are used to characterize the degree of ionic dissociation in an aqueous solution. Strong reagents completely dissociate into their hydrogen or hydroxyl ions whereas weak reagents are only partially ionized.

Consider an acetic acid (weak acid) denoted by HAC being neutralized by a strong base NaOH (sodium hydroxide) in water. The reactions are

$$\mathrm{H_2O} \iff \mathrm{H^+ + OH^-} \tag{16a}$$

$$\mathrm{HAC} \iff \mathrm{H^+ + AC^-} \tag{16b}$$

$$\mathrm{NaOH} \Rightarrow \mathrm{Na^+ + OH^-} \tag{16c}$$

According to the electroneutrality condition, the sum of the charges of all ions in the solution must be zero, i.e.,

$$[\mathrm{Na^+}] + [\mathrm{H^+}] = [\mathrm{OH^-}] + [\mathrm{AC^-}] \tag{17}$$

where the symbol [X] denotes the concentration of the ion X.

On the other hand, the following equilibrium relationships hold for water and acetic acid:

$$K_a = [\mathrm{AC^-}][\mathrm{H^+}]/[\mathrm{HAC}], \tag{18a}$$

$$K_w = [\mathrm{H^+}][\mathrm{OH^-}], \tag{18b}$$

where K_a and K_w are the dissociation constants of the acetic acid and water with $K_a = 1.76 \times 10^{-5}$ and $K_w = 10^{-14}$.

Defining $w_a = [HAC] + [AC^-]$ as the total acetate and $w_b = [Na^+]$ and inserting Eqs. (18a) and (18b) into Eq. (17), we have

$$[H^+]^3 + [H^+]^2\{K_a + w_b\} + [H^+]\{K_a(w_b - w_a) - K_w\} - K_aK_w = 0. \quad (19)$$

Using Eq. (14), Eq. (19) becomes

$$W_b + 10^{-pH} - 10^{pH-pK_w} - \frac{W_a}{1 + 10^{pK_a - pH}} = 0, \quad (20)$$

where $pKa = -\log_{10} k_a$.

We consider the weak acid–strong base neutralization process described by Eqs. (15a), (15b) and (20). By fixing the acid flow-rate F_a (81 cc/min) at a specific value, the process is regarded as a single variable system with base flow-rate F_b and the pH in the effluent being the input and output, respectively. The (F_b, y_{pH}) data pairs were produced by using the process physical model with the parameter values given in Table 4.

The base flow rate F_b was given by

$$F_b = 515 + 51.5\sin(2\pi t/25) \quad \text{for } t \leqslant 150, \quad (21a)$$

$$F_b = 515 + 25.75\sin(2\pi t/25) + 25.75\sin(2\pi t/10) \quad \text{for } t > 150. \quad (21b)$$

For obtaining such a data pairs, we applied Newton–Raphson method that is given by Eq. (22):

$$pH_{i+1} = pH_i - \frac{f(pH_i)}{f'(pH_i)}. \quad (22)$$

The system inputs of the PNN structure consist of the delayed terms of $F_b(t)$ and $y_{pH}(t)$ which are input and output of the process, i.e.,

$$\hat{y}_{pH}(t) = \varphi(F_b(t-3), F_b(t-2), F_b(t-1), y_{pH}(t-3), y_{pH}(t-2), y_{pH}(t-1)), \quad (23)$$

Table 4
Parameters and initial values for pH process

Variables	Meaning	Initial setting
V	Volume of tank	1000 cc
F_a	Flow rate of acid	81 cc/min
F_b	Flow rate of base	515 cc/min
C_a	Concentration of acid in F_a	0.32 mole/l
C_b	Concentration of base in F_b	0.05 mole/l
K_a	Acid equilibrium constant	1.76×10^{-5}
K_w	Water equilibrium constant	1.0×10^{-14}
$W_a(0)$	Concentration of acid	0.0435 mole/l
$W_b(0)$	Concentration of base	0.0432 mole/l

where $\hat{y}_{\mathrm{pH}}$ and y_{pH} denote the PNN model output and the actual process output, respectively. Five hundred data pairs are generated from Eqs. (21a), (21b) and (22) where total data are used for training.

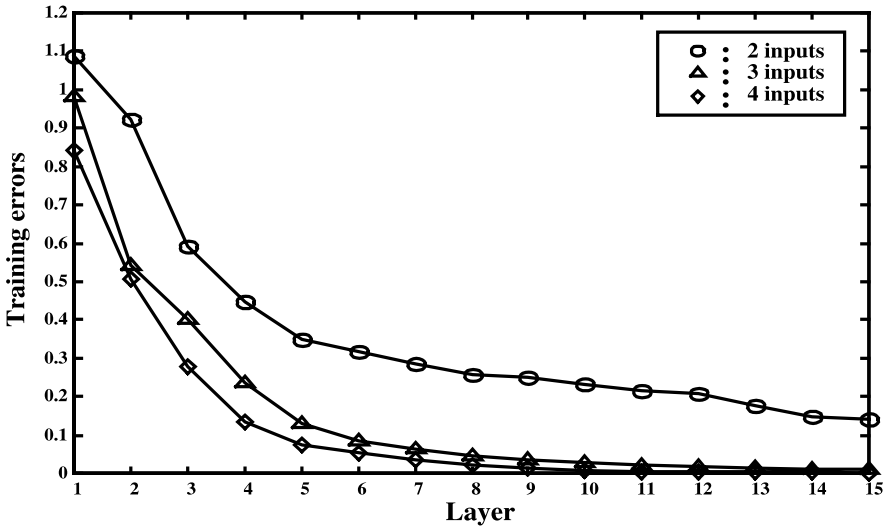

Fig. 10. Performance index for the training data (each layer is of Type 2).

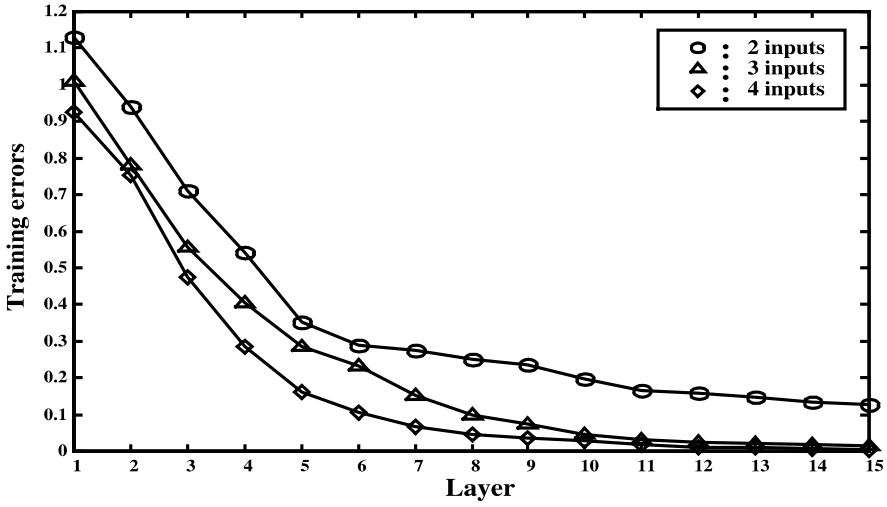

Fig. 11. Performance index for the training data (Type 1 → Type 2).

Table 5
Comparison of identification errors with previous modeling methods

Model			Performance index
Nie's model [20]	USOCPN		0.230
	SSOCPN		0.012
Our model	Basic PNN	Case 1	0.0015
		Case 2	0.0052
	Modified PNN	Case 1	0.0039
		Case 2	0.0124

We conducted a series of comprehensive experiments for all four main architecture of the PNNs, refer to Figs. 10 and 11 in case of the basic PNN. The generation procedure of the PNN is carried out until the 15th layer in the basic PNN structure, see Figs. 10 and 11 and the 10th layer in the modified PNN structure.

Table 5 provides with a comparative analysis of various fuzzy models. The two models proposed in [20] (that is an unsupervised self-organizing counterpropagation network algorithm (USOCPN) and unsupervised self-organizing counterpropagation network algorithm (SSOCPN)) are characterized by higher values of the MSE values. The number of rules used there is equal to 31 (USOCPN) and 34 (SSOCPN). As becomes apparent from Table 5, in these two architectures the resulting performance index assumes far higher values than reported for the PNN architectures.

5. Concluding remarks

In this study, we introduced a class of self-organizing polynomial neural networks, discussed a diversity of their topologies, came up with a detailed procedure, and used these networks to nonlinear system modeling. The key features of this approach can be enumerated as follows:

- The proposed design methodology helps reach a compromise between approximation and generalization capabilities of the constructed PNN model.
- The PNN comes with a diversity of local characteristics (PDs) that are useful in coping with various nonlinear characteristics of the nonlinear systems. Based on these, one can proceed with polynomials of different order as well as vary the number of the input variables associated with the individual processing units.
- The depth of the PNN can be selected as a result of a tradeoff between accuracy and complexity of the overall model.

- The structure of the network is not predetermined (as in most of the existing neural networks) but becomes dynamically adjusted during the development process.

The comprehensive experimental studies involving well-known data sets show a superb performance of the network in comparison to the existing fuzzy models.

Acknowledgements

Support from the KOSEF (a grant no. R02-2000-00284) and the Natural Sciences and Engineering Council of Canada (NSERC) is gratefully acknowledged.

References

[1] A.G. Ivahnenko, Polynomial theory of complex systems, IEEE Trans. Syst., Man Cybern. SMC-1 (1971) 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] G.E.P. Box, F.M. Jenkins, Time Series Analysis: Forecasting and Control, second ed., Holden-Day, San Francisco, CA, 1976.
[4] R.M. Tong, The evaluation of fuzzy models derived from experimental data, Fuzzy Sets Syst. 13 (1980) 1–12.
[5] M. Sugeno, T. Yasukawa, Linguistic modeling based on numerical data, in: IFSA'91, Brussels, Computer, Management & Systems Science, 1991, pp. 264–267.
[6] M. Sugeno, T. Yasukawa, A fuzzy-logic-based approach to qualitative modeling, IEEE Trans. Fuzzy Syst. 1 (1) (1993) 7–31.
[7] C.W. Xu, Y. Zailu, Fuzzy model identification self-learning for dynamic system, IEEE Trans Syst., Man Cybern. SMC-17 (4) (1987) 683–689.
[8] W. Pedrycz, An identification algorithm in fuzzy relational system, Fuzzy Sets Syst. 13 (1984) 153–167.
[9] S.K. Oh, W. Pedrycz, Identification of fuzzy systems by means of an auto-tuning algorithm and its application to nonlinear systems, Fuzzy Sets Syst. 115 (2) (2000) 205–230.
[10] E.T. Kim, M.K. Park, S.H. Ji, M. Park, A new approach to fuzzy modeling, IEEE Trans. Fuzzy Syst. 5 (3) (1997) 328–337.
[11] E. Kim, H. Lee, M. Park, M. Park, A simple identified Sugeno-type fuzzy model via double clustering, Inf. Sci. 110 (1998) 25–39.
[12] J.Q. Chen, Y.G. Xi, Z.J. Zhang, A clustering algorithm for fuzzy model identification, Fuzzy Sets Syst. 98 (1998) 319–329.
[13] J. Leski, E. Czogala, A new artificial neural networks based fuzzy inference system with moving consequents in if–then rules and selected applications, Fuzzy Sets Syst. 108 (1999) 289–297.
[14] A.F. Gomez-Skarmeta, M. Delgado, M.A. Vila, About the use of fuzzy clustering techniques for fuzzy model identification, Fuzzy Sets Syst. 106 (1999) 179–188.
[15] Y. Lin, G.A. Cunningham, A new approach to fuzzy-neural modeling, IEEE Trans. Fuzzy Syst. 3 (2) (1995) 190–197.

[16] F.G. Shinskey, pH and pION Control in Process and Waste Streams, Wiley, New York, 1973.
[17] R.C. Hall, D.E. Seberg, Modeling and self-tuning control of a multivariable pH neutralization process, Proc. ACC (1989) 1822–1827.
[18] T.J. McAvoy, E. Hsu, S. Lowenthal, Dynamics of pH in controlled stirred tank reactor, Ind. Eng. Chem. Process Des. Develop. 11 (1972) 68–70.
[19] T.J. McAvoy, Time optimal and Ziegler–Nichols control, Ind. Eng. Chem. Process Des. Develop. 11 (1972) 71–78.
[20] J. Nie, A.P. Loh, C.C. Hang, Modeling pH neutralization processes using fuzzy-neural approaches, Fuzzy Sets Syst. 78 (1996) 5–22.
[21] T.K. Gustafsson, K.V. Waller, Dynamic modeling and reaction invariant control of pH, Chem. Eng. Sci. 38 (1983) 389–398.
[22] G.A. Pajunen, Comparison of linear and nonlinear adaptive control of a pH-process, IEEE Control Syst. Mag. (1987) 39–44.
[23] C.L. Karr, E.J. Gentry, Fuzzy control of pH using genetic algorithms, IEEE Trans. Fuzzy Syst. 1 (1993) 46–53.